# Αλγόριθμοι

**Sanjoy Dasgupta**
*University of California, San Diego*

**Christos Papadimitriou**
*University of California at Berkeley*

**Umesh Vazirani**
*University of California at Berkeley*

*Κεφάλαιο 8*

# NP-πλήρη προγράμματα

## 8.1 Προβλήματα αναζήτησης

Στα προηγούμενα επτά κεφάλαια αναπτύξαμε αλγορίθμους για να βρίσκουμε συντομότερες διαδρομές και ελάχιστα επικαλυπτικά δένδρα, αντιστοιχίσεις σε διμερή γραφήματα, μέγιστες αύξουσες υπακολουθίες, μέγιστες ροές σε δίκτυα, και ούτω καθεξής. Όλοι αυτοί οι αλγόριθμοι είναι *αποδοτικοί*, επειδή σε κάθε περίπτωση οι χρονικές απαιτήσεις τους αυξάνουν ως μια πολυωνυμική συνάρτηση του μεγέθους της εισόδου (όπως $n$, $n^2$, ή $n^3$).

Για να μπορέσουμε να έχουμε μια καλύτερη εκτίμηση τέτοιων αποδοτικών αλγορίθμων, θεωρήστε την εναλλακτική λύση: Σε όλα αυτά τα προβλήματα αναζητούμε μια λύση (διαδρομή, δένδρο, αντιστοίχιση, κ.λπ) ανάμεσα σε έναν *εκθετικό* πληθυσμό δυνατοτήτων. Πράγματι, $n$ αγόρια μπορούν να αντιστοιχίζονται με $n$ κορίτσια με $n!$ διαφορετικούς τρόπους, ένα γράφημα με $n$ κορυφές έχει $n^{n-2}$ επικαλυπτικά δένδρα, και ένα τυπικό γράφημα έχει εκθετικό πλήθος διαδρομών από τον κόμβο $s$ στον κόμβο $t$. Όλα αυτά τα προβλήματα θα μπορούσαν κατ' αρχήν να λυθούν σε εκθετικό χρόνο με έλεγχο όλων των υποψήφιων λύσεων, μία προς μία. Όμως ένας αλγόριθμος του οποίου ο χρόνος εκτέλεσης είναι $2^n$, ή χειρότερα, είναι στην πράξη άχρηστος (δείτε το πλαίσιο που ακολουθεί). Η έρευνα για αποδοτικούς αλγορίθμους αφορά την εύρεση έξυπνων τρόπων για να παρακάμψουμε αυτή τη διεργασία της εξαντλητικής αναζήτησης, χρησιμοποιώντας ενδείξεις από τα στοιχεία της εισόδου έτσι ώστε να περιορίσουμε δραστικά το χώρο αναζήτησης.

Μέχρι τώρα στο βιβλίο αυτό έχουμε δει τις πιο λαμπρές επιτυχίες σε αυτόν το στόχο, αλγοριθμικές τεχνικές που κατατροπώνουν το φάντασμα της εκθετικότητας: άπληστους αλγορίθμους, δυναμικό προγραμματισμό, γραμμικό προγραμματισμό (ενώ η τεχνική διαίρει και βασίλευε συνήθως δίνει ταχύτερους αλγορίθμους για προβλήματα τα οποία μπορούν να λυθούν σε πολυωνυμικό χρόνο). Τώρα ήρθε η ώρα να συναντήσετε τις πιο δυσάρεστες και επίμονες αποτυχίες αυτής της πορείας. Θα δούμε κάποια άλλα «προβλήματα αναζήτησης» στα οποία και πάλι ζητάμε μια λύση με ιδιαίτερες ιδιότητες μέσα σε ένα εκθετικό χάος εναλλακτικών λύσεων. Αλλά γι' αυτά τα νέα προβλήματα δε φαίνεται δυνατός κάποιος σύντομος δρόμος. Οι ταχύτεροι αλγόριθμοι που γνωρίζουμε

για αυτά είναι όλοι εκθετικοί — ουσιαστικά δεν είναι καλύτεροι από μια εξαντλητική αναζήτηση. Τώρα θα παρουσιάσουμε μερικά σημαντικά παραδείγματα.

## Η ιστορία των Sissa και Moore

Σύμφωνα με τη μυθολογία, το σκάκι εφευρέθηκε από το Βραχμάνο Sissa για να διασκεδάσει και να διδάξει το βασιλιά του. Ο ευγνώμων μονάρχης του ζήτησε να πει τι αντάλλαγμα θα ήθελε, και ο σοφός άνδρας ζήτησε να τοποθετήσει ο βασιλιάς έναν κόκκο ρύζι στο πρώτο τετράγωνο της σκακιέρας, δύο στο δεύτερο, τέσσερις στο τρίτο, και ούτω καθεξής, διπλασιάζοντας την ποσότητα του ρυζιού μέχρι το 64ο τετράγωνο. Ο βασιλιάς συμφώνησε αμέσως, και έτσι ήταν ο πρώτος άνθρωπος ο οποίος διδάχθηκε το πολύτιμο — αν και ταπεινωτικό — μάθημα της *εκθετικής αύξησης*. Το αίτημα του Sissa ισοδυναμούσε με $2^{64} - 1 = 18\,446\,744\,073\,709\,551\,615$ κόκκους ρυζιού, αρκετούς για να καλυφθεί όλη η Ινδία αρκετές φορές!

Παντού στη φύση, από τις αποικίες βακτηριδίων μέχρι τα κύτταρα των εμβρύων, βλέπουμε συστήματα τα οποία αυξάνονται εκθετικά — για λίγο. Το 1798, ο Βρετανός φιλόσοφος T. Robert Malthus δημοσίευσε ένα δοκίμιο στο οποίο προέβλεπε ότι η εκθετική αύξηση (την οποία αποκαλούσε «γεωμετρική αύξηση») του ανθρώπινου πληθυσμού σύντομα θα εξαντλούσε πόρους με γραμμική αύξηση, ένα επιχείρημα το οποίο επηρέασε βαθιά τον Charles Darwin. Ο Malthus γνώριζε το θεμελιώδες γεγονός ότι μια εκθετική αύξηση αργά ή γρήγορα υπερκεράζει οποιαδήποτε πολυωνυμική.

Το 1965, ο πρωτοπόρος των υπολογιστών Gordon E.Moore παρατήρησε ότι η πυκνότητα των τρανζίστορ στα τσιπ διπλασιαζόταν κάθε χρόνο κατά τη δεκαετία του 1960, και προέβλεψε ότι αυτή η τάση θα συνεχιζόταν. Αυτή η πρόβλεψη, η οποία μετριάστηκε σε διπλασιασμό κάθε 18 μήνες και επεκτάθηκε στην ταχύτητα των υπολογιστών, είναι γνωστή ως *νόμος του Moore*. Διατηρεί την ισχύ του αξιοσημείωτα καλά επί 40 χρόνια. Και αυτές είναι οι δύο βασικές αιτίες για την έκρηξη της τεχνολογίας των πληροφοριών τις περασμένες δεκαετίες: *ο νόμος του Moore και οι αποδοτικοί αλγόριθμοι*.

Θα φαινόταν ότι ο νόμος του Moore δίνει αντικίνητρα για την ανάπτυξη πολυωνυμικών αλγορίθμων. Σο κάτω-κάτω, αν ένας αλγόριθμος είναι εκθετικός, γιατί να μην περιμένουμε μέχρι να τον κάνει εφικτό ο νόμος του Moore; Στην πραγματικότητα όμως συμβαίνει το ακριβώς αντίθετο: ο νόμος του Moore αποτελεί ένα τεράστιο κίνητρο για την ανάπτυξη αποδοτικών αλγορίθμων, επειδή τέτοιοι αλγόριθμοι είναι απαραίτητοι προκειμένου να εκμεταλλευτούμε την εκθετική αύξηση της ταχύτητας των υπολογιστών.

Ορίστε γιατί. Αν, για παράδειγμα, ένας αλγόριθμος $O(2^n)$ για τη λογική (Boolean) ικανοποιησιμότητα (SAT) είχε στη διάθεσή του μία ώρα για εκτέλεση, θα μπορούσε να λύσει στιγμιότυπα με 25 μεταβλητές το 1975, με 31 μεταβλητές στους ταχύτερους υπολογιστές του 1985, με 38 μεταβλητές το 1995, και περίπου με 45 μεταβλητές με τα σημερινά μηχανήματα. Αρκετή πρόοδος — με τη διαφορά ότι κάθε επιπλέον μεταβλητή απαιτεί ενάμιση χρόνο αναμονής, ενώ η όρεξη για εφαρμογές (πολλές από τις οποίες, κατά ειρωνικό τρόπο, αφορούν σχεδίαση υπολογιστών) αυξάνεται πολύ ταχύτερα. Αντιθέτως, το μέγεθος των στιγμιοτύπων που επιλύονται από έναν αλγόριθμο $O(n)$ ή έναν αλγόριθμο $O(n \log n)$ θα *εκατονταπλασιαζόταν* κάθε



## Η ιστορία των Sissa και Moore (*Συνέχεια*)

δεκαετία. Στην περίπτωση ενός αλγορίθμου $O(n^2)$, το μέγεθος του στιγμιοτύπου το οποίο θα μπορούσε να λυθεί σε συγκεκριμένο χρόνο θα δεκαπλασιαζόταν κάθε δεκαετία. Ακόμη και για έναν αλγόριθμο $O(n^6)$, που είναι πολυωνυμικός αλλά μη ελκυστικός, το μέγεθος των στιγμιοτύπων που θα μπορούσαν να λυθούν θα γινόταν περισσότερο από διπλάσιο κάθε δεκαετία. Όσον αφορά την αύξηση του μεγέθους των προβλημάτων τα οποία μπορούμε να αντιμετωπίσουμε με έναν αλγόριθμο, έχουμε μια αντιστροφή: οι εκθετικοί αλγόριθμοι παρουσιάζουν μια πολυωνυμικά αργή πρόοδο, ενώ οι πολυωνυμικοί αλγόριθμοι προοδεύουν εκθετικά γρήγορα! Για να αντανακλάται ο νόμος του Moore στον κόσμο *χρειαζόμαστε* αποδοτικούς αλγορίθμους.

Όπως γνώριζαν πολύ καλά ο Sissa και ο Malthus, η εκθετική αύξηση δεν μπορεί να διατηρείται επ' άπειρον στον πεπερασμένο κόσμο μας. Εξαντλείται η τροφή των βακτηριδιακών αποικιών· το μέγεθος των τσιπ φθάνει στο επίπεδο των ατόμων. Ο νόμος του Moore θα σταματήσει να διπλασιάζει την ταχύτητα των υπολογιστών μας μέσα σε μία ή δύο δεκαετίες. Και τότε η πρόοδος θα βασίζεται στην αλγοριθμική επινοητικότητα — ή διαφορετικά σε καινοτόμες ιδέες όπως οι *κβαντικοί υπολογιστές*, που εξετάζονται στο Κεφάλαιο 10.

## Ικανοποιησιμότητα

Η ΙΚΑΝΟΠΟΙΗΣΙΜΟΤΗΤΑ (satisfiability) ή SAT (θυμηθείτε την Άσκηση 3.28 και την Ενότητα 5.3), είναι ένα πρόβλημα μεγάλης πρακτικής σημασίας, με εφαρμογές που εκτείνονται από τον έλεγχο των τσιπ και τη σχεδίαση υπολογιστών μέχρι την ανάλυση εικόνων και την τεχνολογία λογισμικού. Είναι επίσης ένα κανονικό δύσκολο πρόβλημα. Ένα στιγμιότυπο SAT έχει την εξής μορφή:

$$(x \vee y \vee z)(x \vee \overline{y})(y \vee \overline{z})(z \vee \overline{x})(\overline{x} \vee \overline{y} \vee \overline{z}).$$

Πρόκειται για ένα *λογικό τύπο (Boolean formula) σε συζευκτική κανονική μορφή* (conjunctive normal form – CNF). Είναι μια συλλογή *όρων* (clauses) — οι παρενθέσεις — ο καθένας από τους οποίους αποτελείται από τη διάζευξη (λογικό *or*, που συμβολίζεται με $\vee$) διαφόρων *στοιχείων* (literals), όπου κάθε στοιχείο είναι μια λογική μεταβλητή (όπως η $x$) είτε η άρνησή της (όπως η $\overline{x}$). Μια *ικανοποιούσα ανάθεση τιμών αληθείας* (satisfying truth assignment) είναι μια ανάθεση τιμών `false` ή `true` σε κάθε μεταβλητή έτσι ώστε κάθε όρος να περιέχει ένα στοιχείο του οποίου η τιμή να είναι `true`. Το πρόβλημα SAT είναι το ακόλουθο: με δεδομένο ένα λογικό τύπο σε συζευκτική κανονική μορφή, είτε βρείτε μια ικανοποιούσα ανάθεση τιμών αληθείας, ή αλλιώς αναφέρετε ότι δεν υπάρχει καμία.

Στο στιγμιότυπο που παρουσιάσαμε προηγουμένως, αν θέσουμε, για παράδειγμα, σε όλες τις μεταβλητές την τιμή `true` ικανοποιείται κάθε όρος πλην του τελευταίου. Υπάρχει ανάθεση τιμών αληθείας η οποία να ικανοποιεί *όλους* τους όρους;

Με λίγη σκέψη, δεν είναι δύσκολο να ισχυριστούμε ότι στην προκειμένη περίπτωση δεν υπάρχει τέτοια ανάθεση τιμών αληθείας. (*Υπόδειξη:* Οι τρεις ενδιάμεσοι όροι αναγκάζουν και τις τρεις μεταβλητές να έχουν την ίδια τιμή.) Αλλά πώς μπορούμε να το αποφασίσουμε αυτό γενικά; Φυσικά, μπορούμε πάντοτε να διερευνήσουμε όλες τις αναθέσεις τιμών αληθείας μία προς μία, αλλά σε τύπους με $n$ μεταβλητές το πλήθος των δυνατών αναθέσεων τιμών είναι εκθετικό, $2^n$.

Το πρόβλημα SAT είναι ένα τυπικό *πρόβλημα αναζήτησης*. Μας δίνεται ένα *στιγμιότυπο* $I$ (δηλαδή κάποια δεδομένα εισόδου τα οποία καθορίζουν το συγκεκριμένο πρόβλημα, σε αυτή την περίπτωση ένας λογικός (Boolean) τύπος σε συζευκτική κανονική μορφή), και μας ζητείται να βρούμε μια *λύση* $S$ (ένα αντικείμενο το οποίο ικανοποιεί μια συγκεκριμένη προδιαγραφή, στην προκειμένη περίπτωση μια ανάθεση τιμών η οποία ικανοποιεί όλους τους όρους). Αν δεν υπάρχει τέτοια λύση, πρέπει να το αναφέρουμε.

Πιο συγκεκριμένα, ένα πρόβλημα αναζήτησης πρέπει να έχει την ιδιότητα ότι κάθε προτεινόμενη λύση $S$ ενός στιγμιοτύπου $I$ μπορεί να *ελεγχθεί γρήγορα* για την ορθότητά της. Τι συνεπάγεται αυτό; Εν πρώτοις, η $S$ πρέπει τουλάχιστον να είναι συνοπτική (για να διαβάζεται γρήγορα), με μήκος πολυωνυμικά φραγμένο από το μήκος του $I$. Αυτό προφανώς ισχύει στην περίπτωση του προβλήματος SAT, για το οποίο η $S$ είναι μια ανάθεση τιμών στις μεταβλητές. Για να διατυπώσουμε τυπικά την έννοια του γρήγορου ελέγχου, θα πούμε ότι υπάρχει ένας αλγόριθμος πολυωνυμικού χρόνου ο οποίος δέχεται ως είσοδο τα $I$ και $S$ και αποφασίζει αν η $S$ είναι λύση του $I$ ή όχι. Για το πρόβλημα SAT, αυτό είναι εύκολο καθώς ο έλεγχος αφορά απλώς το εάν η ανάθεση τιμών η οποία καθορίζεται από τη λύση $S$ ικανοποιεί πράγματι κάθε όρο του στιγμιοτύπου $I$.

Αργότερα σε αυτό το κεφάλαιο θα είναι χρήσιμο να αλλάξουμε τον τρόπο εξέτασης, και να θεωρήσουμε αυτόν τον αποδοτικό αλγόριθμο για τον έλεγχο των προτεινόμενων λύσεων ως ένα μέσο *ορισμού* των προβλημάτων αναζήτησης. Επομένως:

> *Ένα* πρόβλημα αναζήτησης *καθορίζεται από έναν αλγόριθμο $C$ ο οποίος δέχεται δύο εισόδους, ένα στιγμιότυπο $I$ και μια προτεινόμενη λύση $S$, και εκτελείται σε πολυωνυμικό χρόνο ως προς $|I|$. Λέμε ότι η $S$ είναι μια λύση του $I$, αν και μόνο αν $C(I, S) =$ true.*

Με δεδομένη τη σπουδαιότητα του προβλήματος αναζήτησης SAT, τα τελευταία 50 χρόνια οι ερευνητές έχουν προσπαθήσει σκληρά να βρουν αποδοτικούς τρόπους επίλυσής του, αλλά χωρίς επιτυχία. Οι ταχύτεροι αλγόριθμοι που έχουμε εξακολουθούν να είναι εκθετικοί ως προς τη χειρότερη περίπτωση εισόδου.

Παρόλα αυτά, ενδιαφέρον παρουσιάζει το γεγονός ότι υπάρχουν δύο φυσικές παραλλαγές του προβλήματος SAT για τις οποίες έχουμε καλούς αλγορίθμους. Αν όλοι οι όροι περιέχουν το πολύ ένα θετικό στοιχείο, τότε ο λογικός (Boolean) τύπος ονομάζεται *τύπος Horn*, και στην περίπτωση αυτή μπορεί να βρεθεί μια ικανοποιούσα ανάθεση τιμών αλ-



ληθείας, εφόσον υπάρχει, με τον άπληστο αλγόριθμο της Ενότητας 5.3. Εναλλακτικά, αν όλοι οι όροι έχουν μόνο *δύο* στοιχεία τότε μπαίνει στο παιχνίδι η θεωρία των γραφημάτων, και το πρόβλημα SAT μπορεί να λυθεί σε γραμμικό χρόνο με την εύρεση των ισχυρά συνεκτικών συνιστωσών ενός συγκεκριμένου γραφήματος που κατασκευάζεται από το στιγμιότυπο (θυμηθείτε την Άσκηση 3.28). Στην πραγματικότητα, στο Κεφάλαιο 9 θα δούμε ένα διαφορετικό πολυωνυμικό αλγόριθμο για αυτή την ίδια ειδική περίπτωση, ο οποίος ονομάζεται 2SAT.

Αντιθέτως, αν γίνουμε λίγο περισσότερο ανεκτικοί και επιτρέψουμε οι προτάσεις να περιέχουν *τρία* στοιχεία, τότε το πρόβλημα που δημιουργείται, γνωστό ως 3SAT (ένα παράδειγμα του οποίου είδαμε νωρίτερα), γίνεται και πάλι δύσκολο στην επίλυση!

### Το πρόβλημα του περιοδεύοντος πωλητή

Στο πρόβλημα του περιοδεύοντος πωλητή (TSP) μας δίνονται $n$ κορυφές $1, \ldots, n$ και όλες οι $n(n-1)/2$ αποστάσεις μεταξύ τους, καθώς και ένας *προϋπολογισμός* $b$. Μας ζητείται να βρούμε μια *περιήγηση* (tour), δηλαδή έναν κύκλο ο οποίος διέρχεται από κάθε κορυφή μόνο μία φορά, με συνολικό κόστος $b$ ή λιγότερο — ή να αναφέρουμε ότι δεν υπάρχει τέτοια περιήγηση. Δηλαδή αναζητούμε μια μετάθεση $\tau(1), \ldots, \tau(n)$ των κορυφών, τέτοια ώστε, όταν τις διασχίσουμε με αυτή τη σειρά, η συνολική απόσταση που θα καλυφθεί να είναι το πολύ $b$:

$$d_{\tau(1),\tau(2)} + d_{\tau(2),\tau(3)} + \cdots + d_{\tau(n),\tau(1)} \leq b.$$

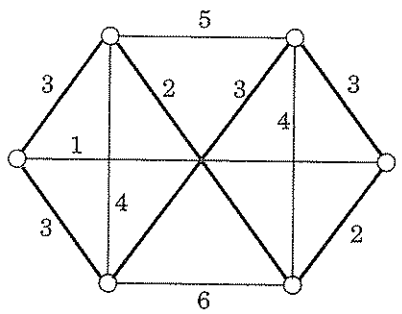
Στην Εικόνα 8.1 μπορείτε να δείτε ένα παράδειγμα (φαίνονται μόνο μερικές από τις αποστάσεις· υποθέστε ότι οι υπόλοιπες είναι πολύ μεγάλες).

Παρατηρήστε ότι έχουμε ορίσει το TSP ως ένα *πρόβλημα αναζήτησης*: με δεδομένο ένα στιγμιότυπο, βρείτε μια περιήγηση μέσα στα όρια του προϋπολογισμού (ή αναφέρατε ότι δεν υπάρχει καμία τέτοια). Γιατί όμως να εκφράσουμε το πρόβλημα του περιοδεύοντος πωλητή με αυτόν τον τρόπο, όταν στην πραγματικότητα είναι ένα *πρόβλημα βελτιστοποίησης* στο οποίο αναζητείται η *συντομότερη* δυνατή περιήγηση; Γιατί να το μεταμφιέσουμε σε κάτι άλλο;

Για καλό λόγο. Το σχέδιό μας σε αυτό το κεφάλαιο είναι να συγκρίνουμε και να συσχετίσουμε προβλήματα. Από αυτή την άποψη βοηθά το πλαίσιο εργασίας των προβλημάτων αναζήτησης, επειδή περιλαμβάνει προβλήματα βελτιστοποίησης όπως το TSP καθώς και πραγματικά προβλήματα αναζήτησης όπως το SAT.

Η μετατροπή ενός προβλήματος βελτιστοποίησης σε πρόβλημα αναζήτησης δεν αλλάζει καθόλου τη δυσκολία του, επειδή οι δύο εκδοχές *ανάγονται η μία στην άλλη*. Οποιοσδήποτε αλγόριθμος που λύνει τη βελτιστοποίηση TSP λύνει εύκολα το πρόβλημα αναζήτησης: βρείτε τη βέλτιστη περιήγηση και, αν είναι μέσα στα όρια του προϋπολογισμού, επιστρέψτε την· αν δεν είναι, δεν υπάρχει λύση.

**Εικόνα 8.1** Η βέλτιστη περιήγηση του περιοδεύοντος πωλητή, η οποία παρουσιάζεται με έντονη γραμμή, έχει μήκος 18.




Αντιστρόφως, μπορεί επίσης να χρησιμοποιηθεί ένας αλγόριθμος του προβλήματος αναζήτησης για τη λύση του προβλήματος βελτιστοποίησης. Για να δείτε γιατί, υποθέστε καταρχήν ότι με κάποιον τρόπο γνωρίζουμε το *κόστος* της βέλτιστης περιήγησης· τότε θα μπορούσαμε να βρούμε αυτή τη περιήγηση καλώντας τον αλγόριθμο για το πρόβλημα αναζήτησης και χρησιμοποιώντας το βέλτιστο κόστος ως προϋπολογισμό. Ωραία, αλλά πώς θα βρούμε το βέλτιστο κόστος; Εύκολο: Με δυαδική αναζήτηση! (Δείτε την Άσκηση 8.1).

Παρεμπιπτόντως, εδώ υπάρχει ένα λεπτό σημείο: Γιατί χρειάζεται να εισαγάγουμε έναν προϋπολογισμό; Κάθε πρόβλημα βελτιστοποίησης δεν είναι επίσης ένα πρόβλημα αναζήτησης, με την έννοια ότι αναζητούμε μια λύση η οποία έχει την ιδιότητα να είναι βέλτιστη; Η παγίδα είναι ότι η λύση σε ένα πρόβλημα αναζήτησης θα πρέπει να είναι εύκολο να αναγνωριστεί, ή όπως τα θέσαμε προηγουμένως να μπορεί να ελεγχθεί σε πολυωνυμικό χρόνο. Με δεδομένη μια πιθανή λύση του προβλήματος TSP, είναι εύκολο να ελέγξουμε τις ιδιότητες «είναι περιήγηση» (απλώς ελέγχουμε ότι επισκέπτεται κάθε κορυφή μία μόνο φορά) και την ιδιότητα «έχει συνολικό μήκος ≤ $b$». Αλλά πώς θα μπορούσαμε να ελέγξουμε την ιδιότητα «είναι βέλτιστη»;

Όπως και με το SAT, δεν υπάρχουν γνωστοί αλγόριθμοι πολυωνυμικού χρόνου για το TSP, παρά τη μεγάλη προσπάθεια που έχουν καταβάλει οι ερευνητές σχεδόν επί έναν αιώνα. Φυσικά, υπάρχει ένας εκθετικός αλγόριθμος για την επίλυσή του, ο οποίος δοκιμάζει όλες τις $(n-1)!$ περιηγήσεις, και στην Ενότητα 6.6 είδαμε έναν ταχύτερο, αλλά παρόλα αυτά εκθετικό, αλγόριθμο δυναμικού προγραμματισμού.

Το πρόβλημα του ελάχιστου επικαλυπτικού δένδρου (MST), για το οποίο *έχουμε* αποδοτικούς αλγορίθμους, δημιουργεί εδώ μια θλιβερή αντίθεση. Για να το διατυπώσουμε ως πρόβλημα αναζήτησης, και πάλι μας δίνεται ένας πίνακας αποστάσεων και ένα φράγμα

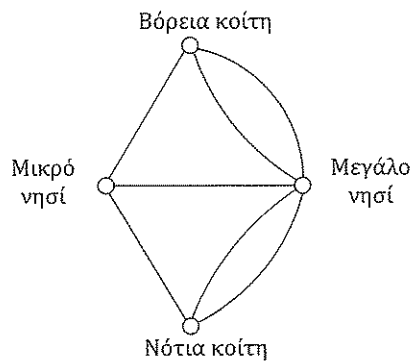


$b$, και μας ζητείται να βρούμε ένα δένδρο $T$ με συνολικό βάρος $\sum_{(i,j)\in T} d_{ij} \le b$. Το TSP μπορεί να θεωρηθεί ως δύσκολος συγγενής του προβλήματος MST, όπου το δένδρο δεν επιτρέπεται να διακλαδώνεται και επομένως είναι μια διαδρομή.[1] Αυτός ο επιπλέον περιορισμός στη δομή του δένδρου οδηγεί σε ένα πολύ δυσκολότερο πρόβλημα.

### Euler και Rudrata

Το καλοκαίρι του 1735 ο Leonhard Euler (προφέρεται «Όιλερ»), ο διάσημος Ελβετός μαθηματικός, περπατούσε στις γέφυρες του Königsberg, μιας πόλης της Ανατολικής Πρωσίας. Μετά από λίγο παρατήρησε με εκνευρισμό ότι, ανεξάρτητα από το σημείο από το οποίο ξεκινούσε τον περίπατό του, και ανεξάρτητα από το πόσο έξυπνα συνέχιζε, ήταν αδύνατο να διασχίσει κάθε γέφυρα μόνο μία φορά. Και από αυτή την αφελή φιλοδοξία γεννήθηκε η θεωρία των γραφημάτων.

Ο Euler αναγνώρισε αμέσως τις ρίζες της ανεπάρκειας του πάρκου. Καταρχήν μετατρέπουμε το χάρτη του πάρκου σε γράφημα του οποίου οι κορυφές είναι οι τέσσερις περιοχές ξηράς (δύο νησιά, δύο κοίτες) και οι ακμές είναι οι επτά γέφυρες:




Αυτό το γράφημα έχει πολλές ακμές μεταξύ δύο κορυφών — μια δυνατότητα που δεν είχαμε επιτρέψει μέχρι τώρα στο βιβλίο, η οποία όμως έχει νόημα γι' αυτό το συγκεκριμένο πρόβλημα καθώς η κάθε γέφυρα πρέπει να ληφθεί υπόψη ξεχωριστά. Αναζητούμε μια διαδρομή η οποία να διέρχεται από κάθε ακμή μόνο μία φορά (η διαδρομή επιτρέπεται να επαναλαμβάνει κορυφές). Με άλλα λόγια, θέτουμε την εξής ερώτηση: *Πότε μπορούμε να σχεδιάσουμε ένα γράφημα χωρίς να σηκώσουμε το μολύβι από το χαρτί;*

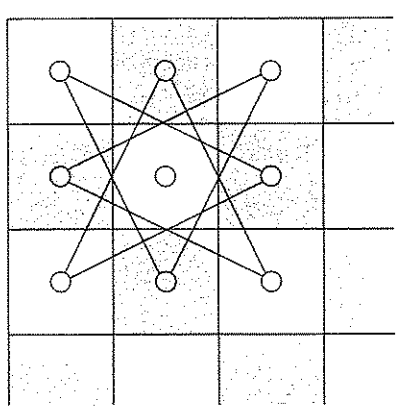
[1] Στην πραγματικότητα το TSP απαιτεί έναν κύκλο, αλλά μπορούμε να ορίσουμε μια εναλλακτική εκδοχή η οποία αναζητά μια διαδρομή, και δεν είναι δύσκολο να δείτε ότι αυτό είναι εξίσου δύσκολο όπως και το ίδιο το TSP.

Η απάντηση την οποία ανακάλυψε ο Euler είναι απλή, κομψή, και διαισθητική: *Αυτό συμβαίνει, αν και μόνο αν (α) το γράφημα είναι συνεκτικό (συνδεδεμένο), και (β) κάθε κορυφή, με πιθανή εξαίρεση δύο κορυφές (τις κορυφές που αντιστοιχούν στην αρχή και το τέλος του περιπάτου), έχει άρτιο βαθμό* (Άσκηση 3.26). Αυτός είναι ο λόγος για τον οποίο είναι αδύνατο να διασχίσουμε το πάρκο του Königsberg: και οι τέσσερις κορυφές έχουν περιττό βαθμό.

Για να το θέσουμε σύμφωνα με τα τρέχοντα ενδιαφέροντά μας, ας ορίσουμε ένα πρόβλημα αναζήτησης με το όνομα ΔΙΑΔΡΟΜΗ EULER (Euler path): Με δεδομένο ένα γράφημα, βρείτε μια διαδρομή η οποία να περιέχει την κάθε ακμή μόνο μία φορά. Έπεται από την παρατήρηση του Euler, και με λίγη περισσότερη σκέψη, ότι αυτό το πρόβλημα αναζήτησης μπορεί να λυθεί σε πολυωνυμικό χρόνο.

Σχεδόν μια χιλιετία πριν από το μοιραίο καλοκαίρι του Euler στην Ανατολική Πρωσία, ένας ποιητής από το Κασμίρ με το όνομα Rudrata είχε θέσει την εξής ερώτηση: Μπορούμε να επισκεφθούμε όλα τα τετράγωνα της σκακιέρας, χωρίς να επαναλάβουμε κανένα τετράγωνο, σε ένα μεγάλο περίπατο ο οποίος τελειώνει στο τετράγωνο από το οποίο ξεκινήσαμε και όπου σε κάθε βήμα κάνουμε μια έγκυρη κίνηση ίππου; Και αυτό είναι ένα πρόβλημα γραφημάτων: τώρα το γράφημα έχει 64 κορυφές, και δύο τετράγωνα ενώνονται με ακμή αν ένας ίππος μπορεί να πάει από το ένα τετράγωνο στο άλλο με μία κίνηση (δηλαδή αν οι συντεταγμένες τους διαφέρουν κατά 2 ως προς τη μία διάσταση και κατά 1 ως προς την άλλη). Στην Εικόνα 8.2 μπορείτε να δείτε ένα τμήμα του γραφήματος που αντιστοιχεί στην επάνω αριστερή γωνία της σκακιέρας. Μπορείτε να βρείτε μια περιήγηση ίππου στη σκακιέρα σας;

**Εικόνα 8.2** Οι κινήσεις ενός ίππου στη γωνία μιας σκακιέρας.





Αυτό είναι ένα διαφορετικό είδος προβλήματος αναζήτησης σε γραφήματα: θέλουμε έναν κύκλο ο οποίος να διέρχεται από όλες τις *κορυφές* (σε αντιδιαστολή με το πρόβλημα του Euler, το οποίο αφορούσε όλες τις ακμές), χωρίς να επαναλαμβάνεται καμία κορυφή. Και δεν υπάρχει κάποιος λόγος να παραμείνουμε στις σκακιέρες· αυτό το ερώτημα μπορεί να τεθεί για οποιοδήποτε γράφημα. Ας ορίσουμε το πρόβλημα αναζήτησης ΚΥΚΛΟΣ RUDRATA (Rudrata cycle) ως εξής: με δεδομένο ένα γράφημα, βρείτε έναν κύκλο ο οποίος να επισκέπτεται κάθε κορυφή μόνο μία φορά — ή αναφέρετε ότι δεν υπάρχει τέτοιος κύκλος.[2] Αυτό το πρόβλημα θυμίζει δυσοίωνα το TSP, και πράγματι δεν είναι γνωστοί πολυωνυμικοί αλγόριθμοι γι' αυτό.

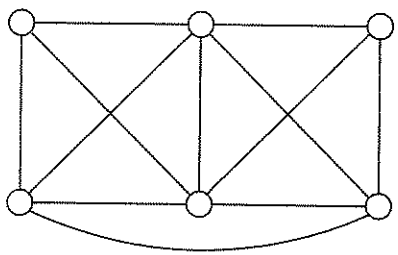
Υπάρχουν δύο διαφορές ανάμεσα στους ορισμούς των προβλημάτων του Euler και του Rudrata. Η πρώτη είναι ότι το πρόβλημα του Euler επισκέπτεται όλες τις *ακμές* ενώ το πρόβλημα του Rudrata επισκέπτεται όλες τις *κορυφές*. Αλλά υπάρχει και το ζήτημα ότι ο ένας απαιτεί διαδρομή ενώ ο άλλος χρειάζεται κύκλο. Ποια από αυτές τις διαφορές αιτιολογεί την τεράστια ανομοιότητα όσον αφορά την υπολογιστική πολυπλοκότητα των δύο προβλημάτων; Πρέπει να είναι η πρώτη, επειδή η δεύτερη διαφορά μπορεί να αποδειχθεί ότι είναι καθαρά διακοσμητική. Πράγματι, ορίζουμε το πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA (Rudrata path) να είναι ίδιο με το πρόβλημα ΚΥΚΛΟΣ RUDRATA, με τη διαφορά ότι ο στόχος είναι τώρα να βρούμε μια *διαδρομή* η οποία να διέρχεται από κάθε κορυφή μόνο μία φορά. Όπως θα δούμε σε λίγο, υπάρχει μια ακριβής ισοδυναμία ανάμεσα στις δύο εκδοχές του προβλήματος του Rudrata.

## Τομές και διχοτομήσεις

*Τομή* (cut) είναι ένα σύνολο ακμών των οποίων η αφαίρεση καθιστά ένα γράφημα μη συνεκτικό. Συχνά παρουσιάζει ενδιαφέρον η εύρεση μικρών τομών, και το πρόβλημα ΕΛΑΧΙΣΤΗ ΤΟΜΗ (Minimum cut ) είναι, με δεδομένο ένα γράφημα και έναν προϋπολογισμό $b$, η εύρεση μιας τομής με $b$ ακμές το πολύ. Για παράδειγμα, η μικρότερη τομή στην Εικόνα 8.3 έχει μέγεθος 3. Αυτό το πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο με $n - 1$ υπολογισμούς μέγιστης ροής: δίνουμε σε κάθε ακμή χωρητικότητα 1, και βρίσκουμε τη μέγιστη ροή ανάμεσα σε κάποιο σταθεροποιημένο κόμβο και κάθε άλλο μεμονωμένο κόμβο. Η μικρότερη από αυτές τις ροές θα αντιστοιχεί (μέσω του θεωρήματος μέγιστης ροής – ελάχιστης τομής) στη μικρότερη τομή. Μπορείτε να δείτε γιατί; Έχουμε επίσης δει έναν πολύ διαφορετικό, τυχαιοκρατικό αλγόριθμο γι' αυτό το πρόβλημα (σελίδα 184).

[2] Στη βιβλιογραφία αυτό το πρόβλημα είναι γνωστό ως πρόβλημα του *κύκλου Hamilton*, από το όνομα του μεγάλου Ιρλανδού μαθηματικού ο οποίος το ανακάλυψε εκ νέου το 19ο αιώνα.

**Εικόνα 8.3** Ποια είναι η μικρότερη τομή σε αυτό το γράφημα;



Σε πολλά γραφήματα, όπως αυτό της Εικόνας 8.3, η μικρότερη τομή αφήνει απλώς μια μονομελή (singleton) κορυφή στη μία πλευρά — αποτελείται από όλες τις ακμές που πρόσκεινται σε αυτή την κορυφή. Ακόμη πιο ενδιαφέρουσες είναι οι μικρές τομές που διαμερίζουν τις κορυφές του γραφήματος σε σχεδόν εξισορροπημένα σύνολα. Ακριβέστερα, το πρόβλημα ΙΣΟΡΡΟΠΗΜΕΝΗ ΤΟΜΗ (Balanced cut) είναι το εξής: με δεδομένο ένα γράφημα με $n$ κορυφές και έναν προϋπολογισμό $b$, διαμερίστε τις κορυφές σε δύο σύνολα $S$ και $T$ τέτοια, ώστε $|S|, |T| \geq n/3$ και να υπάρχουν το πολύ $b$ ακμές μεταξύ των $S$ και $T$. Αυτό είναι ένα άλλο δύσκολο πρόβλημα.

Οι ισορροπημένες τομές ανακύπτουν σε μια ποικιλία σημαντικών εφαρμογών, όπως είναι οι *συστάδες* (clustering). Θεωρήστε για παράδειγμα το πρόβλημα της κατάτμησης (segmenting) μιας εικόνας στα συστατικά της στοιχεία (όπως ένας ελέφαντας που στέκεται σε ένα λιβάδι με έναν καθαρό, γαλάζιο ουρανό από πάνω του). Ένας καλός τρόπος για να το πετύχουμε αυτό είναι να δημιουργήσουμε ένα γράφημα με έναν κόμβο για κάθε εικονοστοιχείο (πίξελ) της εικόνας και να τοποθετήσουμε μια ακμή ανάμεσα στους κόμβους των οποίων τα αντίστοιχα εικονοστοιχεία είναι κοντά ως προς το χώρο και παρόμοια ως προς το χρώμα. Ένα αντικείμενο της εικόνας (όπως ο ελέφαντας, για παράδειγμα) θα αντιστοιχεί τότε σε ένα σύνολο κορυφών του γραφήματος που είναι σε μεγάλο βαθμό συνδεδεμένες. Μια ισόρροπημένη τομή είναι επομένως πιθανό να διαιρεί τα εικονοστοιχεία σε δύο συστάδες χωρίς να διαχωρίζει κανένα από τα κύρια συστατικά της εικόνας. Η πρώτη τομή θα μπορούσε, για παράδειγμα, να διαχωρίζει τον ελέφαντα από τη μια μεριά από τον ουρανό, και από την άλλη από το λιβάδι. Έτσι θα ήταν απαραίτητη μια περαιτέρω τομή για να ξεχωρίσουμε τον ουρανό από το λιβάδι.

### Ακέραιος γραμμικός προγραμματισμός

Αν και ο αλγόριθμος simplex δεν είναι πολυωνυμικού χρόνου, αναφέραμε στο Κεφάλαιο 7 ότι *υπάρχει* ένας διαφορετικός, πολυωνυμικός αλγόριθμος για το γραμμικό προγραμματισμό. Επομένως, το πρόβλημα ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ μπορεί να λυθεί αποδοτικά και στην πράξη και στη θεωρία. Όμως η κατάσταση αλλάζει τελείως αν, εκτός από τον καθορισμό μιας γραμμικής αντικειμενικής συνάρτησης και γραμμικών ανισώσεων, περιορίσουμε και τη λύση (τις τιμές των μεταβλητών) ώστε να είναι *ακέραιη*. Αυτό το τελευταίο πρόβλημα ονομάζεται ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ (Integer Linear Programming — ILP). Ας δούμε πώς θα μπορούσαμε να το διατυπώσουμε ως πρόβλημα αναζήτησης. Μας δίνεται ένα σύνολο γραμμικών ανισώσεων $\mathbf{Ax} \leq \mathbf{b}$, όπου $\mathbf{A}$ είναι ένας πίνακας $m \times n$ και $\mathbf{b}$ ένα $m$-διάνυσμα· μια αντικειμενική συνάρτηση η οποία καθορίζεται από ένα $n$-διάνυσμα $\mathbf{c}$· και τέλος, ένας *στόχος* $g$ (το αντίστοιχο ενός προϋπολογισμού στα προβλήματα μεγιστοποίησης). Θέλουμε να βρούμε ένα μη αρνητικό *ακέραιο* $n$-διάνυσμα $\mathbf{x}$, τέτοιο ώστε $\mathbf{Ax} \leq \mathbf{b}$ και $\mathbf{c} \cdot \mathbf{x} \geq g$.

Όμως εδώ υπάρχει ένας πλεονασμός: ο τελευταίος περιορισμός $\mathbf{c} \cdot \mathbf{x} \geq g$ είναι και ο ίδιος μια γραμμική ανίσωση και μπορεί να απορροφηθεί στην ανίσωση $\mathbf{Ax} \leq \mathbf{b}$. Κατά συνέπεια, ορίζουμε το ILP ως το ακόλουθο πρόβλημα αναζήτησης: με δεδομένα τα $\mathbf{A}$ και $\mathbf{b}$, βρείτε ένα μη αρνητικό ακέραιο διάνυσμα $\mathbf{x}$ το οποίο να ικανοποιεί τις ανισώσεις $\mathbf{Ax} \leq \mathbf{b}$, ή αναφέρετε ότι δεν υπάρχει τέτοια λύση. Παρά τις πολλές κρίσιμες εφαρμογές αυτού του προβλήματος και το έντονο ενδιαφέρον των ερευνητών, δεν υπάρχουν γνωστοί αποδοτικοί αλγόριθμοι για το πρόβλημα αυτό.

Υπάρχει μια ιδιαίτερα ξεκάθαρη ειδική περίπτωση του ILP η οποία είναι δύσκολη αυτή καθ' αυτή: ο στόχος είναι να βρεθεί ένα διάνυσμα $\mathbf{x}$ από μηδενικά και άσους το οποίο να ικανοποιεί τη σχέση $\mathbf{Ax} = \mathbf{1}$, όπου $\mathbf{A}$ είναι ένας πίνακας $m \times n$ με καταχωρίσεις 0 – 1 και $\mathbf{1}$ είναι το $m$-διάνυσμα με όλες τις συνιστώσες 1. Θα πρέπει να είναι προφανές από τις αναγωγές της Ενότητας 7.1.4 ότι πράγματι πρόκειται για ειδική περίπτωση του ILP. Θα το ονομάσουμε ΕΞΙΣΩΣΕΙΣ ΜΗΔΕΝ-ΕΝΑ (Zero-One Equations — ZOE).

Έχουμε πλέον εισαγάγει μια σειρά από σημαντικά προβλήματα αναζήτησης, μερικά από τα οποία είναι γνωστά από προηγούμενα κεφάλαια και για τα οποία υπάρχουν αποδοτικοί αλγόριθμοι, και άλλα που διαφέρουν σε μικρό αλλά κρίσιμο βαθμό με αποτέλεσμα να γίνονται υπολογιστικά δύσκολα προβλήματα. Για να ολοκληρώσουμε την ιστορία μας θα παρουσιάσουμε μερικά ακόμα δύσκολα προβλήματα τα οποία θα παίξουν κάποιο ρόλο στη συνέχεια του κεφαλαίου, όταν θα συσχετίσουμε την υπολογιστική δυσκολία όλων αυτών των προβλημάτων. Αν θέλετε μπορείτε να προχωρήσετε στην Ενότητα 8.2 και μετά να επιστρέψετε στους ορισμούς αυτών των προβλημάτων, ανάλογα με τις απαιτήσεις σας.

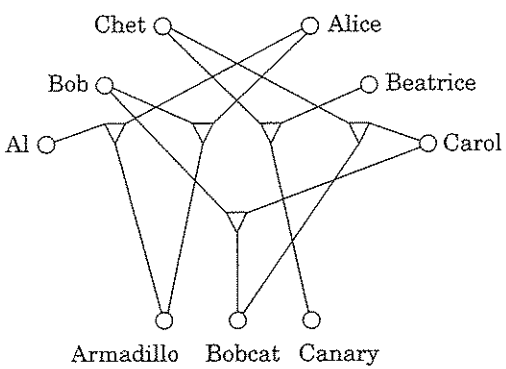
### Τριδιάστατη αντιστοίχιση

Θυμηθείτε το πρόβλημα ΔΙΜΕΡΗΣ ΑΝΤΙΣΤΟΙΧΙΣΗ (Bipartite matching): με δεδομένο ένα διμερές γράφημα με $n$ κόμβους σε κάθε πλευρά (τα *αγόρια* και τα *κορίτσια*), βρείτε ένα σύνολο $n$ ξένων ακμών, ή αποφασίστε ότι δεν υπάρχει τέτοιο σύνολο. Στην Ενότητα 7.3 είδαμε πώς μπορούμε να λύσουμε αποδοτικά αυτό το πρόβλημα με αναγωγή του σε μέ-

γιστη ροή. Ωστόσο, υπάρχει μια ενδιαφέρουσα γενίκευση, που ονομάζεται ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ (3D Matching), για την οποία δεν είναι γνωστός κάποιος πολυωνυμικός αλγόριθμος. Σε αυτή τη νέα μορφή υπάρχουν $n$ αγόρια και $n$ κορίτσια αλλά και $n$ κατοικίδια, και οι συμβατότητες μεταξύ τους καθορίζονται από ένα σύνολο *τριάδων* (triples), όπου κάθε μία περιέχει ένα αγόρι, ένα κορίτσι, και ένα κατοικίδιο. Διαισθητικά, μια τριάδα $(b, g, p)$ σημαίνει ότι το αγόρι $b$, το κορίτσι $g$, και το κατοικίδιο $p$ τα πάνε καλά μεταξύ τους. Θέλουμε να βρούμε $n$ ξένες τριάδες, και επομένως να δημιουργήσουμε $n$ αρμονικά σπιτικά.

Μπορείτε να εντοπίσετε μια λύση στην Εικόνα 8.4;

**Εικόνα 8.4** Ένα πιο περίπλοκο σενάριο δημιουργίας αντιστοιχίσεων. Κάθε τριάδα παρουσιάζεται ως κόμβος τριγωνικού σχήματος ο οποίος συνδέει ένα αγόρι, ένα κορίτσι και ένα κατοικίδιο.



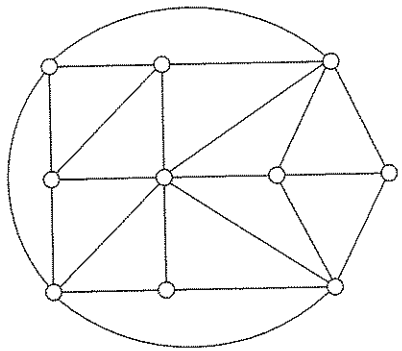

### Ανεξάρτητο σύνολο, κάλυψη κορυφών, και κλίκα

Στο πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ (Independent set) (θυμηθείτε την Ενότητα 6.7) μας δίνεται ένα γράφημα και ένας ακέραιος $g$, και ο σκοπός είναι να βρούμε $g$ κορυφές οι οποίες είναι ανεξάρτητες, δηλαδή ανά δύο δεν έχουν καμία ακμή μεταξύ τους. Μπορείτε να βρείτε ένα ανεξάρτητο σύνολο τριών κορυφών στην Εικόνα 8.5; Τεσσάρων; Είδαμε στην Ενότητα 6.7 ότι αυτό το πρόβλημα μπορεί να λυθεί αποδοτικά σε δένδρα, αλλά για γενικά γραφήματα δεν είναι γνωστός κάποιος πολυωνυμικός αλγόριθμος.

Υπάρχουν πολλά προβλήματα αναζήτησης τα οποία αφορούν γραφήματα. Στο πρόβλημα ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ (Vertex Cover), για παράδειγμα, η είσοδος είναι ένα γράφημα και ένας προϋπολογισμός $b$, και η ιδέα είναι να βρούμε $b$ κορυφές οι οποίες καλύπτουν (αγγίζουν) κάθε ακμή. Μπορείτε να καλύψετε όλες τις ακμές της Εικόνας 8.5 με επτά κορυφές; Με έξι; (Και βλέπετε τη στενή σύνδεση με το πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ ;)



**Εικόνα 8.5** Ποιο είναι το μέγεθος του μεγαλύτερου ανεξάρτητου συνόλου σε αυτό το γράφημα;



Το πρόβλημα κάλυψης κορυφών είναι μια ειδική περίπτωση του προβλήματος κάλυψης συνόλου, το οποίο συναντήσαμε στο Κεφάλαιο 5. Σε εκείνο το πρόβλημα μας δίνεται ένα σύνολο $E$ και αρκετά υποσύνολά του, $S_1, \ldots, S_m$, μαζί με έναν προϋπολογισμό $b$. Μας ζητείται να επιλέξουμε $b$ από αυτά τα υποσύνολα έτσι ώστε η ένωσή τους να είναι το $E$. Το πρόβλημα κάλυψης κορυφών είναι η ειδική περίπτωση όπου το σύνολο $E$ αποτελείται από τις κορυφές ενός γραφήματος, και υπάρχει ένα υποσύνολο $S_i$ για κάθε κορυφή, το οποίο περιέχει τις ακμές που άπτονται σε αυτή την κορυφή. Μπορείτε να δείτε γιατί το ΤΡΙΔΙΑΣΤΑΤΟ ΤΑΙΡΙΑΣΜΑ είναι επίσης μια ειδική περίπτωση του προβλήματος ΚΑΛΥΨΗ ΣΥΝΟΛΟΥ;

Και τέλος υπάρχει το πρόβλημα ΚΛΙΚΑ (Clique): με δεδομένο ένα γράφημα και ένα στόχο $g$, βρείτε ένα σύνολο $g$ κορυφών τέτοιο ώστε να υπάρχουν ανάμεσά τους όλες οι δυνατές ακμές. Ποια είναι η μεγαλύτερη κλίκα στην Εικόνα 8.5;

### Μεγαλύτερη διαδρομή

Γνωρίζουμε ότι το πρόβλημα ΣΥΝΤΟΜΟΤΕΡΗ ΔΙΑΔΡΟΜΗ μπορεί να λυθεί πολύ αποδοτικά, αλλά τι συμβαίνει με το πρόβλημα ΜΕΓΑΛΥΤΕΡΗ ΔΙΑΔΡΟΜΗ; Εδώ μας δίνεται ένα γράφημα $G$ με ακμές χωρίς αρνητικά βάρη και δύο διακριτές κορυφές $s$ και $t$, μαζί με ένα στόχο $g$. Μας ζητείται να βρούμε μια διαδρομή από την $s$ στην $t$ με συνολικό βάρος τουλάχιστον $g$. Φυσικά, για να αποφύγουμε τετριμμένες λύσεις απαιτούμε η διαδρομή να είναι *απλή*, δηλαδή να μην περιέχει επαναλαμβανόμενες κορυφές.

Δεν υπάρχει γνωστός αποδοτικός αλγόριθμος γι' αυτό το πρόβλημα (το οποίο μερικές φέρει το όνομα TAXICAB RIP-OFF).

### Σακίδιο και άθροισμα υποσυνόλου

Θυμηθείτε το πρόβλημα ΣΑΚΙΔΙΟ (Knapsack) της Ενότητας 6.4: μας δίνονται ακέραια βάρη $w_1, ..., w_n$ και ακέραιες τιμές $v_1, ..., v_n$ για $n$ στοιχεία. Μας δίνεται επίσης μια χωρητικότητα βάρους $W$ και ένας στόχος $g$ (η πρώτη υπάρχει στο αρχικό πρόβλημα βελτιστοποίησης, ενώ ο δεύτερος προστίθεται για να το κάνει πρόβλημα αναζήτησης). Αναζητούμε ένα σύνολο στοιχείων των οποίων των συνολικό βάρος είναι το πολύ $W$ και του οποίου η συνολική τιμή είναι τουλάχιστον $g$. Όπως πάντοτε, αν δεν υπάρχει τέτοιο σύνολο θα πρέπει να το αναφέρουμε.

Στην Ενότητα 6.4 αναπτύξαμε ένα σχήμα δυναμικού προγραμματισμού για το πρόβλημα ΣΑΚΙΔΙΟ με χρόνο εκτέλεσης $O(nW)$, το οποίο σημειώσαμε ότι είναι εκθετικό ως προς το μέγεθος της εισόδου, καθώς περιλαμβάνει το $W$ αντί του $\log W$. Και έχουμε επίσης το συνήθη εξαντλητικό αλγόριθμο, ο οποίος εξετάζει όλα τα υποσύνολα των στοιχείων — και τα $2^n$. Υπάρχει πολυωνυμικός αλγόριθμος για το πρόβλημα ΣΑΚΙΔΙΟ; Κανείς δε γνωρίζει.

Αλλά υποθέστε ότι ενδιαφερόμαστε για μια παραλλαγή του προβλήματος του σακιδίου στην οποία οι ακέραιοι κωδικοποιούνται σε *μοναδιαίο* (unary) σύστημα — για παράδειγμα, το 12 γράφεται ως *IIIIIIIIIIII*. Ομολογουμένως, αυτός ο τρόπος αναπαράστασης ακεραίων είναι εκθετικά σπάταλος, αλλά ορίζει ένα έγκυρο πρόβλημα το οποίο θα μπορούσαμε να ονομάσουμε ΜΟΝΑΔΙΑΙΟ ΣΑΚΙΔΙΟ (Unary knapsack). Έπεται από την περιγραφή μας ότι αυτό το κάπως τεχνητό πρόβλημα έχει έναν πολυωνυμικό αλγόριθμο.

Μια διαφορετική παραλλαγή: υποθέστε τώρα ότι η τιμή κάθε στοιχείου είναι ίση με το βάρος του (όλες οι τιμές στο δυαδικό σύστημα), και επιπλέον ότι ο στόχος $g$ είναι ίδιος με τη χωρητικότητα $W$. (Για να προσαρμοστούμε με την ιστορία της διάρρηξης με την οποία εισαγάγαμε το πρόβλημα του σακιδίου, όλα τα στοιχεία είναι χρυσά αντικείμενα, και ο διαρρήκτης θέλει να γεμίσει το σακίδιό του μέχρι επάνω.) Αυτή η ειδική περίπτωση είναι ταυτόσημη με την εύρεση ενός υποσυνόλου (ενός δεδομένου συνόλου) ακεραίων οι οποίοι έχουν άθροισμα ακριβώς $W$. Αφού αυτή είναι μια ειδική περίπτωση του προβλήματος ΣΑΚΙΔΙΟ, δεν μπορεί να είναι δυσκολότερο. Θα μπορούσε όμως να είναι πολυωνυμικό; Όπως προκύπτει, αυτό το πρόβλημα, που ονομάζεται ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ (Subset Sum), είναι επίσης πολύ δύσκολο.

Σε αυτό το σημείο θα μπορούσε κανείς να αναρωτηθεί: Αν το πρόβλημα ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ είναι μια ειδική περίπτωση η οποία τυχαίνει να είναι εξίσου δύσκολη με το γενικό πρόβλημα ΣΑΚΙΔΙΟ, γιατί μας ενδιαφέρει; Ο λόγος είναι η *απλότητα*. Στους περίπλοκους υπολογισμούς των αναγωγών μεταξύ προβλημάτων αναζήτησης που θα αναπτύξουμε σε αυτό το κεφάλαιο, τα εννοιολογικώς απλά προβλήματα όπως τα προβλήματα ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ και 3SAT είναι ανεκτίμητα.



## 8.2 NP-πλήρη προβλήματα

### Δύσκολα προβλήματα, εύκολα προβλήματα:

Με λίγα λόγια, ο κόσμος είναι γεμάτος από προβλήματα αναζήτησης, κάποια από τα οποία μπορούν να λυθούν αποδοτικά, ενώ άλλα φαίνεται να είναι πολύ δύσκολα. Αυτό απεικονίζεται στον πίνακα που ακολουθεί.

| Δύσκολα προβλήματα (**NP** πλήρη) | Εύκολα προβλήματα (στο **P**) |
|---|---|
| 3SAT | 2SAT, HORN SAT |
| TSP | MST |
| ΜΕΓΑΛΥΤΕΡΗ ΔΙΑΔΡΟΜΗ | ΣΥΝΤΟΜΟΤΕΡΗ ΔΙΑΔΡΟΜΗ |
| ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ | ΔΙΜΕΡΗΣ ΑΝΤΙΣΤΟΙΧΙΣΗ |
| ΣΑΚΙΔΙΟ | ΜΟΝΑΔΙΑΙΟ ΣΑΚΙΔΙΟ |
| ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ | ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ ΣΕ ΔΕΝΔΡΑ |
| ΑΚΕΡΑΙΟΣ ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ | ΓΡΑΜΜΙΚΟΣ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΣ |
| ΔΙΑΔΡΟΜΗ RUDRATA | ΔΙΑΔΡΟΜΗ EULER |
| ΙΣΟΡΡΟΠΗΜΕΝΗ ΤΟΜΗ | ΕΛΑΧΙΣΤΗ ΤΟΜΗ |

Αξίζει να μελετήσουμε αυτόν τον πίνακα. Στα δεξιά έχουμε προβλήματα τα οποία μπορούν να λυθούν αποδοτικά. Στα αριστερά έχουμε ένα σύνολο από «σκληρά καρύδια» τα οποία έχουν διαφύγει την αποδοτική λύση για πολλές δεκαετίες ή αιώνες.

Τα διάφορα προβλήματα στα δεξιά μπορούν να λυθούν με αλγορίθμους οι οποίοι είναι εξειδικευμένοι και ποικίλοι: δυναμικό προγραμματισμό, ροή, αναζήτηση σε γραφήματα, άπληστες τεχνικές. Αυτά τα προβλήματα είναι εύκολα για διάφορους λόγους.

Σε θλιβερή αντίθεση, τα προβλήματα στα δεξιά είναι *όλα δύσκολα για τον ίδιο λόγο!* Στον πυρήνα τους είναι όλα το ίδιο πρόβλημα, απλώς με διαφορετικές μεταμφιέσεις! Όλα είναι *ισοδύναμα:* όπως θα δούμε στην Ενότητα 8.3, καθένα από αυτά μπορεί να αναχθεί σε οποιοδήποτε από τα άλλα — και αντιστρόφως.

### P και NP

Ήρθε η ώρα να εισαγάγουμε μερικές σημαντικές έννοιες. Γνωρίζουμε τι είναι ένα πρόβλημα αναζήτησης: το καθοριστικό του χαρακτηριστικό είναι ότι οποιαδήποτε προτεινόμενη λύση μπορεί να ελεγχθεί γρήγορα για την ορθότητά της, με την έννοια ότι υπάρχει κάποιος αποδοτικός αλγόριθμος ελέγχου $C$ ο οποίος δέχεται ως είσοδο το δεδομένο στιγμιότυπο $I$ (τα δεδομένα τα οποία καθορίζουν το πρόβλημα που πρόκειται να λυθεί), καθώς και την προτεινόμενη λύση $S$, και εμφανίζει στην έξοδο τιμή `true`, αν και μόνο αν

η $S$ είναι πραγματικά μια λύση του στιγμιοτύπου $I$. Επιπλέον, ο χρόνος εκτέλεσης του αλγορίθμου $C(S, I)$ είναι φραγμένος από ένα πολυώνυμο ως προς $|I|$, το μήκος του στιγμιοτύπου. *Συμβολίζουμε την κλάση όλων αυτών των προβλημάτων αναζήτησης ως* **NP**.

## Γιατί P και NP;

Εντάξει, το γράμμα **P** αντιστοιχεί στη λέξη «polynomial» (πολυωνυμικός). Αλλά γιατί να χρησιμοποιούμε τα αρχικά **NP** (η συνήθης συντόμευση των λέξεων «no problem» (κανένα πρόβλημα) στις ηλεκτρονικές αίθουσες συνομιλίας — chatrooms) για να περιγράψουμε την κλάση των προβλημάτων αναζήτησης από τα οποία κάποια είναι τρομερά δύσκολα;

Τα γράμματα **NP** προέρχονται από τις λέξεις «nondeterministic polynomial time» (μη αιτιοκρατικός πολυωνυμικός χρόνος), ένας όρος που παραπέμπει στις απαρχές της θεωρίας πολυπλοκότητας. Διαισθητικά, σημαίνει ότι μια λύση σε οποιοδήποτε πρόβλημα αναζήτησης μπορεί να βρεθεί και να πιστοποιηθεί σε πολυωνυμικό χρόνο από ένα ειδικό (και αρκετά εξωπραγματικό) είδος αλγορίθμου, ο οποίος ονομάζεται *μη αιτιοκρατικός αλγόριθμος* (nondeterministic algorithm). Ένας τέτοιος αλγόριθμος έχει τη δύναμη να *μαντεύει* σωστά σε κάθε βήμα.

Παρεμπιπτόντως, ο αρχικός ορισμός του **NP** (και της πιο συνηθισμένης χρήσης του σήμερα) δεν αφορούσε μια κατηγορία προβλημάτων αναζήτησης, αλλά μια κατηγορία *προβλημάτων απόφασης*: αλγοριθμικές ερωτήσεις οι οποίες μπορούν να απαντηθούν με ναι ή όχι. Παράδειγμα: «Υπάρχει ανάθεση τιμών αληθείας η οποία να ικανοποιεί αυτόν το λογικό (Boolean) τύπο;» Αλλά και αυτό αντανακλά μια ιστορική πραγματικότητα: Την εποχή που αναπτυσσόταν η θεωρία της **NP**-πληρότητας, οι ερευνητές στη θεωρία των υπολογιστών ενδιαφερόντουσαν για τυπικές γλώσσες, έναν τομέα στον οποίο τέτοια προβλήματα απόφασης έχουν θεμελιώδη σημασία.

Έχουμε δει πολλά παραδείγματα προβλημάτων αναζήτησης **NP** τα οποία μπορούν να λυθούν σε πολυωνυμικό χρόνο. Σε τέτοιες περιπτώσεις, υπάρχει ένας αλγόριθμος ο οποίος δέχεται ως είσοδο ένα στιγμιότυπο $I$ και έχει χρόνο εκτέλεσης πολυωνυμικό ως προς $|I|$. Αν το στιγμιότυπο $I$ έχει λύση, ο αλγόριθμος επιστρέφει μια τέτοια λύση· και αν το $I$ δεν έχει λύση, ο αλγόριθμος ορθά το αναφέρει. *Η κατηγορία όλων των προβλημάτων αναζήτησης που μπορούν να λυθούν σε πολυωνυμικό χρόνο συμβολίζονται με το* **P**. Επομένως, όλα τα προβλήματα αναζήτησης στη δεξιά πλευρά του πίνακα ανήκουν στο **P**.

Υπάρχουν προβλήματα αναζήτησης τα οποία δεν μπορούν να λυθούν σε πολυωνυμικό χρόνο; Με άλλα λόγια, ισχύει **P** ≠ **NP**; Οι περισσότεροι ερευνητές αλγορίθμων πιστεύουν πως ναι. Είναι δύσκολο να πιστέψουμε ότι μπορεί πάντοτε να αποφευχθεί η εκθετική αναζήτηση, και ότι ένα απλό τέχνασμα θα σπάσει όλα αυτά τα δύσκολα προβλήματα, τα οποία παραμένουν άλυτα για δεκαετίες και αιώνες. Και υπάρχει ένας καλός λόγος για να πιστεύουν οι μαθηματικοί ότι **P** ≠ **NP** — η εργασία της εύρεσης μιας απόδειξης για ένα δεδομένο μαθηματικό ισχυρισμό είναι ένα πρόβλημα αναζήτησης, και ανήκει επομένως

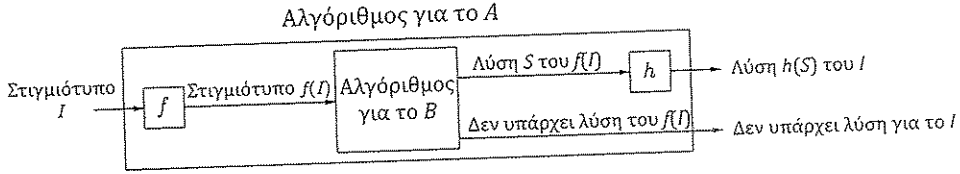


στο **NP** (έτσι κι αλλιώς, όταν η τυπική απόδειξη μιας μαθηματικής πρότασης γραφτεί με εξονυχιστικές λεπτομέρειες, μπορεί να ελεγχθεί μηχανικά, γραμμή προς γραμμή, από έναν αποδοτικό αλγόριθμο). Έτσι αν **P** = **NP** θα υπήρχε ένας αποδοτικός τρόπος απόδειξης των θεωρημάτων, εξαλείφοντας έτσι την ανάγκη για μαθηματικούς! Εν γένει, υπάρχουν διάφορες αιτίες για τις οποίες πιστεύεται ευρέως ότι **P** ≠ **NP**. Ωστόσο, η απόδειξη αυτού του γεγονότος είναι εξαιρετικά δύσκολη, ένας από τους βαθύτερους και πιο σημαντικούς άλυτους γρίφους των μαθηματικών.

## Πάλι αναγωγές

Ακόμη και αν δεχθούμε ότι **P** ≠ **NP**, τι ισχύει για τα συγκεκριμένα προβλήματα στην αριστερή πλευρά του πίνακα; Βασιζόμενοι στις ενδείξεις, πιστεύουμε ότι αυτά τα ιδιαίτερα προβλήματα δεν έχουν αποδοτικό αλγόριθμο (πέρα φυσικά από το ιστορικό γεγονός ότι πολλοί ευφυείς μαθηματικοί και επιστήμονες των υπολογιστών έχουν προσπαθήσει σκληρά και έχουν αποτύχει να βρουν κάποιον); Τέτοιες ενδείξεις παρέχουν οι *αναγωγές*, οι οποίες μεταφράζουν ένα πρόβλημα αναζήτησης σε κάποιο άλλο. Αυτό που επιδεικνύουν είναι ότι όλα τα προβλήματα στην αριστερή πλευρά του πίνακα, κατά μία έννοια, είναι *ακριβώς το ίδιο πρόβλημα*, με τη διαφορά ότι διατυπώνονται με διαφορετική γλώσσα. Θα χρησιμοποιήσουμε επίσης αναγωγές για να δείξουμε ότι αυτά τα προβλήματα είναι τα *δυσκολότερα* προβλήματα στο **NP** — αν έστω και ένα από αυτά έχει έναν αλγόριθμο πολυωνυμικού χρόνου, τότε *κάθε* πρόβλημα στο **NP** έχει αλγόριθμο πολυωνυμικού χρόνου. Έτσι, αν πιστεύουμε ότι **P** ≠ **NP**, τότε όλα αυτά τα προβλήματα αναζήτησης είναι δύσκολα.

Ορίσαμε τις αναγωγές στο Κεφάλαιο 7, και είδαμε αρκετά παραδείγματα από αυτές. Ας εξειδικεύσουμε τώρα αυτόν τον ορισμό για τα προβλήματα αναζήτησης. Μια *αναγωγή* από ένα πρόβλημα αναζήτησης $A$ σε ένα πρόβλημα αναζήτησης $B$ είναι ένας αλγόριθμος $f$ πολυωνυμικού χρόνου ο οποίος μετασχηματίζει κάθε στιγμιότυπο $I$ του $A$ σε ένα στιγμιότυπο $f(I)$ του $B$, μαζί με έναν άλλον αλγόριθμο πολυωνυμικού χρόνου $h$ ο οποίος απεικονίζει μια λύση $S$ του $f(I)$ σε μια λύση $h(S)$ του $I$· δείτε το διάγραμμα που ακολουθεί. Αν το $f(I)$ δεν έχει λύση, τότε δεν έχει ούτε το $I$. Από αυτές τις δύο διεργασίες μετάφρασης $f$ και $h$ έπεται ότι κάθε αλγόριθμος για το $B$ μπορεί να μετατραπεί σε αλγόριθμο για το $A$ αν τοποθετηθεί μεταξύ των $f$ και $h$.



## Οι δύο τρόποι χρήσης των αναγωγών

Μέχρι τώρα σε αυτό το βιβλίο ο σκοπός της αναγωγής ενός προβλήματος $A$ σε ένα πρόβλημα $B$ ήταν σαφής και έντιμος: Γνωρίζουμε πώς να λύσουμε αποδοτικά το πρόβλημα $B$, και θέλουμε να χρησιμοποιήσουμε αυτή τη γνώση για να λύσουμε το πρόβλημα $A$. Σε αυτό το κεφάλαιο, όμως, οι αναγωγές από το $A$ στο $B$ εξυπηρετούν έναν κάπως διεστραμμένο στόχο: γνωρίζουμε ότι το $A$ είναι δύσκολο, και χρησιμοποιούμε την αναγωγή για να αποδείξουμε ότι και το $B$ είναι δύσκολο!

Αν συμβολίσουμε μια αναγωγή από το $A$ στο $B$ με

$$A \rightarrow B$$

τότε μπορούμε να πούμε ότι η *δυσκολία* ρέει κατά τη διεύθυνση του βέλους, ενώ οι *αποδοτικοί αλγόριθμοι* κινούνται προς την αντίθετη κατεύθυνση. Αυτή η διάδοση της δυσκολίας είναι που μας κάνει να γνωρίζουμε ότι τα **NP**-πλήρη προβλήματα είναι δύσκολα: όλα τα άλλα προβλήματα αναζήτησης ανάγονται σε αυτά, και έτσι κάθε **NP**-πλήρες πρόβλημα περιέχει την πολυπλοκότητα όλων των προβλημάτων αναζήτησης. Αν έστω και ένα **NP**-πλήρες πρόβλημα ανήκει στο **P**, τότε **P** = **NP**.

Οι αναγωγές έχουν επίσης την εύχρηστη ιδιότητα ότι μπορούμε να τις *συνθέσουμε*.

$$\text{Αν } A \rightarrow B \text{ και } B \rightarrow C, \text{ τότε } A \rightarrow C.$$

Για να το δείτε αυτό, πρώτα απ' όλα παρατηρήστε ότι οποιαδήποτε αναγωγή καθορίζεται πλήρως από τις προεπεξεργαστικές και μετεπεξεργαστικές συναρτήσεις $f$ και $h$ (δείτε το διάγραμμα αναγωγής). Αν $(f_{AB}, h_{AB})$ και $(f_{BC}, h_{BC})$ ορίζουν τις αναγωγές από $A$ σε $B$ και από $B$ σε $C$, αντίστοιχα, τότε μια αναγωγή από το $A$ στο $C$ δίνεται από συνθέσεις αυτών των συναρτήσεων: η $f_{BC} \circ f_{AB}$ απεικονίζει ένα στιγμιότυπο του $A$ σε ένα στιγμιότυπο του $C$ και η $h_{AB} \circ h_{BC}$ στέλνει μια λύση του $C$ πίσω σε μια λύση του $A$.

Αυτό σημαίνει ότι, εφόσον γνωρίζουμε ότι ένα πρόβλημα $A$ είναι **NP**-πλήρες, μπορούμε να το χρησιμοποιήσουμε για να αποδείξουμε ότι ένα νέο πρόβλημα αναζήτησης $B$ είναι επίσης **NP**-πλήρες ανάγοντας απλώς το $A$ στο $B$. Μια τέτοια αναγωγή εδραιώνει το γεγονός ότι όλα τα προβλήματα στο **NP** ανάγονται στο $B$, μέσω του $A$.

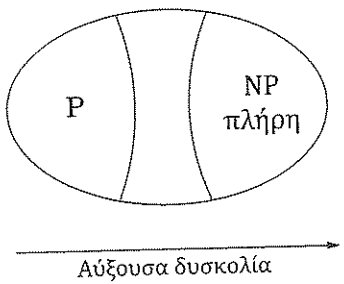
Και τώρα μπορούμε τελικά να ορίσουμε την κλάση των δυσκολότερων προβλημάτων αναζήτησης.

> *Ένα πρόβλημα αναζήτησης είναι* **NP**-*πλήρες* (NP-complete) *αν όλα τα άλλα προβλήματα αναζήτησης ανάγονται σε αυτό.*

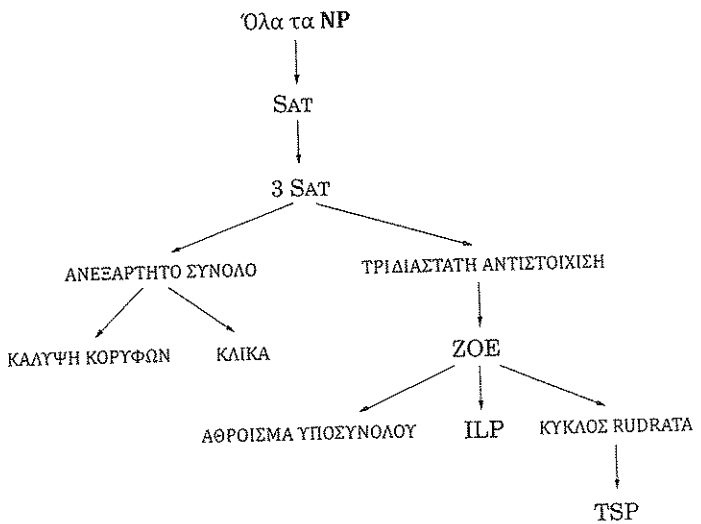
Πράγματι, πρόκειται για μια πολύ ισχυρή απαίτηση. Για να είναι ένα πρόβλημα **NP**-πλήρες, πρέπει να είναι χρήσιμο για την επίλυση κάθε προβλήματος αναζήτησης στον κόσμο! Είναι εντυπωσιακό ότι υπάρχουν τέτοια προβλήματα. Αλλά υπάρχουν, και η πρώτη στήλη του πίνακα που είδαμε νωρίτερα είναι γεμάτη με τα πλέον διάσημα παραδείγματα. Στην Ενότητα 8.3 θα δούμε πώς όλα αυτά τα προβλήματα ανάγονται το ένα στο άλλο, και επίσης γιατί όλα τα άλλα προβλήματα αναζήτησης ανάγονται σε αυτά.



**Εικόνα 8.6** Ο χώρος NP όλων των προβλημάτων αναζήτησης, αν υποθέσουμε ότι P ≠ NP.




**Εικόνα 8.7** Αναγωγές μεταξύ προβλημάτων αναζήτησης.



### Παραγοντοποίηση

Ένα τελευταίο σημείο: ξεκινήσαμε αυτό το βιβλίο με την παρουσίαση ενός άλλου διάσημου δύσκολου προβλήματος: το πρόβλημα ΠΑΡΑΓΟΝΤΟΠΟΙΗΣΗ (Factoring), δηλαδή η εργασία της εύρεσης όλων των πρώτων παραγόντων ενός δεδομένου ακεραίου. Αλλά η δυσκολία του προβλήματος ΠΑΡΑΓΟΝΤΟΠΟΙΗΣΗ είναι διαφορετικής φύσης από αυτή των άλλων δύσκολων προβλημάτων που μόλις είδαμε. Για παράδειγμα, κανείς δεν πιστεύει ότι το πρόβλημα ΠΑΡΑΓΟΝΤΟΠΟΙΗΣΗ είναι **NP**-πλήρες. Μια μεγάλη διαφορά έγκειται στο ότι στην περίπτωση του προβλήματος ΠΑΡΑΓΟΝΤΟΠΟΙΗΣΗ ο ορισμός δεν περιέχει το γνωστό όρο «ή αναφέρατε ότι δεν υπάρχει κανένας». Ένας ακέραιος μπορεί *πάντοτε* να αναλυθεί σε πρώτους.

Μια άλλη διαφορά (πιθανόν όχι τελείως άσχετη) είναι η εξής: όπως θα δούμε στο Κεφάλαιο 10, το πρόβλημα ΠΑΡΑΓΟΝΤΟΠΟΙΗΣΗ υποκύπτει στην ισχύ του *κβαντικού υπολογισμού* — ενώ το SAT, το TSP, και τα άλλα **NP**-πλήρη προβλήματα δε φαίνεται να υποκύπτουν.

## 8.3 Οι αναγωγές

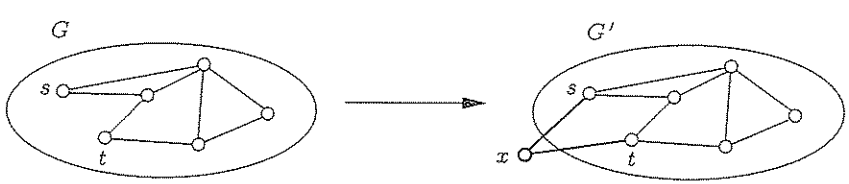
Τώρα θα δούμε ότι τα προβλήματα αναζήτησης της Ενότητας 8.1 μπορούν να αναχθούν το ένα στο άλλο όπως απεικονίζεται στην Εικόνα 8.7. Κατά συνέπεια, είναι όλα **NP**-πλήρη.

Πριν καταπιαστούμε με τις ειδικές αναγωγές στο δένδρο, ας ασχοληθούμε για προθέρμανση με τη συσχέτιση δύο εκδοχών του προβλήματος του Rudrata.

#### RUDRATA (S, T) → ΚΥΚΛΟΣ RUDRATA

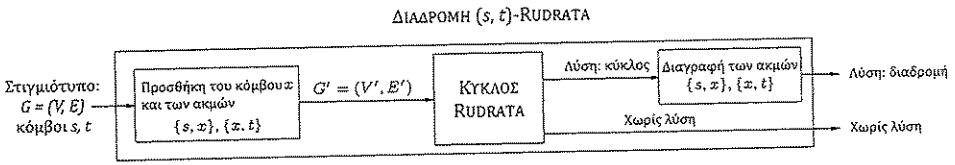
Θυμηθείτε το πρόβλημα ΚΥΚΛΟΣ RUDRATA: με δεδομένο ένα γράφημα, υπάρχει κύκλος ο οποίος διέρχεται από κάθε κορυφή μία μόνο φορά; Μπορούμε επίσης να διατυπώσουμε το πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA $(s, t)$, το οποίο σχετίζεται στενά, όπου καθορίζονται δύο κορυφές $s$ και $t$ και θέλουμε μια διαδρομή που να ξεκινά από την $s$ και να τελειώνει στην $t$, η οποία να διέρχεται από κάθε κορυφή μόνο μία φορά. Είναι δυνατόν το πρόβλημα ΚΥΚΛΟΣ RUDRATA να είναι ευκολότερο από το πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA $(s, t)$; Θα δείξουμε μέσω αναγωγής ότι η απάντηση είναι όχι.

Η αναγωγή απεικονίζει ένα στιγμιότυπο $(G = (V, E), s, t)$ του προβλήματος ΔΙΑΔΡΟΜΗ RUDRATA $(s, t)$ σε ένα στιγμιότυπο $G' = (V', E')$ του προβλήματος ΚΥΚΛΟΣ RUDRATA ως εξής: το γράφημα $G'$ είναι απλώς το $G$ με μια επιπλέον κορυφή $x$ και δύο νέες ακμές $\{s, x\}$ και $\{x, t\}$. Για παράδειγμα:






Οπότε, $V' = V \cup \{x\}$, και $E' = E \cup \{\{s, x\}, \{x, t\}\}$. Πώς θα ανακτήσουμε μια *διαδρομή* $(s, t)$ στο γράφημα $G$ με δεδομένο έναν *κύκλο* Rudrata στο $G'$; Εύκολα· απλώς διαγράφουμε τις ακμές $\{s, x\}$ και $\{x, t\}$ από τον κύκλο.




Για να επιβεβαιώσουμε την εγκυρότητα αυτής της αναγωγής, πρέπει να δείξουμε ότι λειτουργεί και στις δύο περιπτώσεις οι οποίες απεικονίζονται.

1. Όταν το στιγμιότυπο του προβλήματος ΚΥΚΛΟΣ RUDRATA έχει λύση.

Επειδή η νέα κορυφή $x$ έχει μόνο δύο γείτονες, τις κορυφές $s$ και $t$, ένας οποιοσδήποτε κύκλος στο γράφημα $G'$ πρέπει να διασχίζει διαδοχικά τις ακμές $\{t, x\}$ και $\{x, s\}$. Ο υπόλοιπος κύκλος τότε θα διασχίζει όλες τις άλλες κορυφές καθ' οδόν από την $s$ στην $t$. Επομένως, η διαγραφή των δύο ακμών $\{t, x\}$ και $\{x, s\}$ από τον κύκλο Rudrata δίνει μια διαδρομή Rudrata από την $s$ στην $t$ στο αρχικό γράφημα $G$.

2. Όταν το στιγμιότυπο του προβλήματος ΚΥΚΛΟΣ RUDRATA δεν έχει λύση.

Σε αυτή την περίπτωση πρέπει να δείξουμε ότι δεν μπορεί να έχει λύση ούτε το αρχικό στιγμιότυπο του ΔΙΑΔΡΟΜΗ RUDRATA $(s, t)$. Συνήθως είναι ευκολότερο να αποδείξουμε το αντιθετοαντίστροφο, δηλαδή να δείξουμε ότι αν υπάρχει μια $(s, t)$-διαδρομή Rudrata στο γράφημα $G$, τότε υπάρχει επίσης ένας κύκλος Rudrata στο γράφημα $G'$. Αλλά αυτό είναι εύκολο: απλώς προσθέτουμε τις δύο ακμές $\{t, x\}$ και $\{x, s\}$ στη διαδρομή Rudrata για να κλείσουμε τον κύκλο.

Μια τελευταία λεπτομέρεια, κρίσιμη αλλά συνήθως εύκολη στον έλεγχο, είναι ότι οι προεπεξεργαστικές και μετεπεξεργαστικές συναρτήσεις χρειάζονται πολυωνυμικό χρόνο ως προς το μέγεθος του στιγμιοτύπου $(G, s, t)$.

Είναι επίσης δυνατό να βαδίσουμε προς την αντίθετη κατεύθυνση, και να αναγάγουμε το πρόβλημα ΚΥΚΛΟΣ RUDRATA στο πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA $(s, t)$. Μαζί, αυτές οι αναγωγές δείχνουν ότι οι δύο παραλλαγές Rudrata είναι ουσιαστικά το ίδιο πρόβλημα — κάτι που δεν προκαλεί έκπληξη, δεδομένου ότι οι περιγραφές τους είναι σχεδόν ίδιες. Όμως οι περισσότερες από τις άλλες αναγωγές που θα δούμε αφορούν ζεύγη προβλημάτων τα οποία, εκ πρώτης όψεως, φαίνονται αρκετά διαφορετικά. Για να δείξουμε ότι ουσιαστικά είναι τα ίδια, οι αναγωγές μας θα πρέπει να πραγματοποιήσουν μια ευφυή μετάφραση μεταξύ τους.

### 3SAT → ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ

Δύσκολα θα μπορούσαμε να σκεφθούμε δύο περισσότερο διαφορετικά προβλήματα. Στο 3SAT η είσοδος είναι ένα σύνολο όρων, ο καθένας με τρία ή λιγότερα στοιχεία, για παράδειγμα

$$(\overline{x} \vee y \vee \overline{z})(x \vee \overline{y} \vee z)(x \vee y \vee z)(\overline{x} \vee \overline{y}),$$

και ο στόχος είναι να βρούμε μια ικανοποιούσα ανάθεση τιμών αληθείας. Στο ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ η είσοδος είναι ένα γράφημα και ένας αριθμός $g$, και το πρόβλημα είναι να βρεθεί ένα σύνολο από $g$ ζεύγη μη γειτονικών κορυφών. Πρέπει με κάποιον τρόπο να συσχετίσουμε τη λογική Boole με τα γραφήματα!

Ας σκεφθούμε. Για να σχηματιστεί μια ικανοποιούσα ανάθεση τιμών αληθείας, θα πρέπει να επιλέξουμε ένα στοιχείο από κάθε όρο και να του δώσουμε την τιμή true. Οι επιλογές μας όμως θα πρέπει να είναι συνεπείς: αν επιλέξουμε το στοιχείο $\overline{x}$ στον έναν όρο, δεν μπορούμε να επιλέξουμε $x$ σε κάποιον άλλον. Οποιαδήποτε συνεπής επιλογή στοιχείων, με ένα στοιχείο από κάθε όρο, καθορίζει μια ανάθεση τιμών αληθείας (μεταβλητές για τις οποίες κανένα επιλεγμένο στοιχείο δεν μπορεί να πάρει και τις δύο τιμές).

Έτσι, ας αναπαραστήσουμε έναν όρο, έστω τον $(x \vee \overline{y} \vee z)$, με ένα τρίγωνο του οποίου οι κορυφές έχουν τις ετικέτες $x, \overline{y}, z$. Γιατί τρίγωνο; Επειδή ένα τρίγωνο έχει τις τρεις κορυφές του συνδεδεμένες στο μέγιστο βαθμό, και έτσι μας αναγκάζει να επιλέξουμε μόνο μία από αυτές για το ανεξάρτητο σύνολο. Επαναλαμβάνουμε αυτή την κατασκευή για όλους τους όρους — ένας όρος με δύο στοιχεία θα αναπαρασταθεί απλώς από μια ακμή η οποία συνδέει τα στοιχεία. (Ένας όρος με ένα στοιχείο είναι ανόητος και μπορεί να αφαιρεθεί σε ένα στάδιο προεπεξεργασίας, αφού προσδιορίζει την τιμή της μεταβλητής.) Στο γράφημα που προκύπτει, ένα ανεξάρτητο σύνολο πρέπει να επιλέξει το πολύ ένα στοιχείο από κάθε ομάδα (όρο). Για να επιβάλουμε μία μόνο επιλογή από κάθε όρο, θέτουμε το στόχο $g$ ίσο με το πλήθος των όρων· στο παράδειγμά μας, $g = 4$.

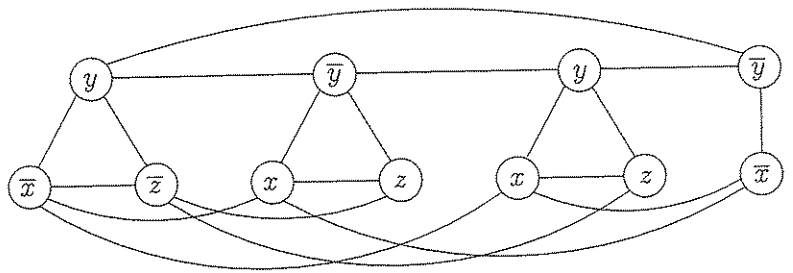
Το μόνο που λείπει τώρα είναι ένας τρόπος ο οποίος θα μας εμποδίζει να επιλέγουμε αντίθετα στοιχεία (δηλαδή, και το $x$ και το $\overline{x}$) σε διαφορετικούς όρους. Αλλά αυτό είναι εύκολο: τοποθετούμε μια ακμή ανάμεσα σε οποιεσδήποτε δύο κορυφές που αντιστοιχούν στα αντίθετα στοιχεία. Το γράφημα που προκύπτει για το παράδειγμά μας παρουσιάζεται στην Εικόνα 8.8.

Ας ανακεφαλαιώσουμε την κατασκευή. Με δεδομένο ένα στιγμιότυπο $I$ του 3SAT, δημιουργούμε ένα στιγμιότυπο $(G, g)$ του προβλήματος ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ ως εξής:

- Το γράφημα $G$ έχει ένα τρίγωνο για κάθε όρο (ή απλώς μια ακμή, αν ο όρος έχει δύο στοιχεία), με κορυφές που φέρουν ως ετικέτες τα στοιχεία του όρου, και έχει επιπλέον ακμές ανάμεσα σε οποιεσδήποτε δύο κορυφές οι οποίες αναπαριστάνουν αντίθετα στοιχεία.
- Ο στόχος $g$ τίθεται ίσος με το πλήθος των όρων.



**Εικόνα 8.8** Το γράφημα που αντιστοιχεί στον τύπο $(\overline{x} \vee y \vee \overline{z})(x \vee \overline{y} \vee z)(x \vee y \vee z)(\overline{x} \vee \overline{y})$.




Προφανώς, αυτή η κατασκευή χρειάζεται πολυωνυμικό χρόνο. Ωστόσο, θυμηθείτε ότι για μια αναγωγή δεν χρειαζόμαστε μόνο έναν αποδοτικό τρόπο απεικόνισης στιγμιοτύπων του πρώτου προβλήματος σε στιγμιότυπα του δεύτερου (η συνάρτηση $f$ στο διάγραμμα της σελίδας 315), αλλά και έναν τρόπο ανακατασκευής μιας λύσης του πρώτου στιγμιοτύπου από οποιαδήποτε λύση του δεύτερου (η συνάρτηση $h$). Όπως πάντοτε, υπάρχουν δύο πράγματα για να αποδείξουμε.

1. Με δεδομένο ένα ανεξάρτητο σύνολο $S$ με $g$ κορυφές στο $G$, είναι δυνατό να ανακτήσουμε αποδοτικά μια ικανοποιούσα ανάθεση τιμών αληθείας στο $I$.

Για οποιαδήποτε μεταβλητή $x$, το σύνολο $S$ δεν μπορεί να περιέχει κορυφές οι οποίες έχουν ετικέτες και $x$ και $\overline{x}$, επειδή οποιοδήποτε τέτοιο ζεύγος κορυφών είναι συνδεδεμένο με μια ακμή. Έτσι, αναθέτουμε στη $x$ τιμή true αν το $S$ περιέχει μια κορυφή με ετικέτα $x$, και τιμή false αν το $S$ περιέχει μια κορυφή με ετικέτα $\overline{x}$ (αν το $S$ δεν περιέχει καμία, τότε αναθέτουμε στη $x$ μία από τις δύο τιμές). Επειδή το $S$ έχει $g$ κορυφές, πρέπει να έχει μία κορυφή ανά όρο· αυτή η ανάθεση τιμών αληθείας ικανοποιεί αυτά τα συγκεκριμένα στοιχεία, και έτσι ικανοποιεί όλους τους όρους.

2. Αν το γράφημα $G$ δεν έχει ανεξάρτητο σύνολο μεγέθους $g$, τότε ο λογικός (Boolean) τύπος $I$ δεν είναι ικανοποιήσιμος.

Συνήθως είναι πιο *απλό* να αποδείξουμε το αντιθετοαντίστροφο, ότι δηλαδή αν το στιγμιότυπο $I$ έχει μια ικανοποιούσα ανάθεση τιμών αληθείας τότε το γράφημα $G$ έχει ένα ανεξάρτητο σύνολο μεγέθους $g$. Αυτό είναι εύκολο: για κάθε όρο, επιλέγουμε ένα στοιχείο του οποίου η τιμή με την ικανοποιούσα ανάθεση είναι true (πρέπει να υπάρχει τουλάχιστον ένα τέτοιο στοιχείο), και προσθέτουμε την αντίστοιχη κορυφή στο $S$. Βλέπετε γιατί το $S$ πρέπει να είναι ανεξάρτητο;

## SAT → 3SAT

Πρόκειται για ένα ενδιαφέρον και συνηθισμένο είδος αναγωγής, από ένα πρόβλημα σε μια *ειδική περίπτωση* του εαυτού του. Θέλουμε να δείξουμε ότι το πρόβλημα παραμένει δύσκολο ακόμη και αν οι είσοδοί του περιοριστούν κατά κάποιον τρόπο — στην προκειμένη περίπτωση, ακόμη και αν όλοι οι όροι περιοριστούν ώστε να έχουν ≤ 3 στοιχεία. Τέτοιες αναγωγές τροποποιούν το δεδομένο στιγμιότυπο έτσι ώστε να απαλλαγεί από την απαγορευμένη δυνατότητα (όροι με ≥ 4 στοιχεία), ενώ ταυτόχρονα διατηρούν το στιγμιότυπο ουσιαστικά ίδιο, από την άποψη ότι μπορούμε να μεταφέρουμε μια λύση στο αρχικό στιγμιότυπο από οποιαδήποτε λύση του τροποποιημένου.

Ακολουθεί ένα τέχνασμα για την αναγωγή του SAT στο 3SAT: με δεδομένο ένα στιγμιότυπο $I$ του SAT, χρησιμοποιούμε ακριβώς το ίδιο στιγμιότυπο για το 3SAT, με τη διαφορά ότι ένας όρος με περισσότερα από τρία στοιχεία $(a_1 \vee a_2 \vee \cdots \vee a_k)$ (όπου τα $a_i$ είναι στοιχεία και $k > 3$), αντικαθίσταται από ένα σύνολο όρων,

$$(a_1 \vee a_2 \vee y_1)(\overline{y}_1 \vee a_3 \vee y_2)(\overline{y}_2 \vee a_4 \vee y_3)\cdots(\overline{y}_{k-3} \vee a_{k-1} \vee a_k),$$

όπου τα $y_i$ είναι νέες μεταβλητές. Ονομάζουμε $I'$ το στιγμιότυπο 3SAT το οποίο προκύπτει. Η μετατροπή από το $I$ στο $I'$ είναι προφανώς πολυωνυμικού χρόνου.

Γιατί λειτουργεί αυτή η αναγωγή; Το στιγμιότυπο $I'$ είναι ισοδύναμο με το $I$ ως προς την ικανοποιησιμότητα, επειδή για οποιαδήποτε ανάθεση τιμών στα $a_i$,

$$\left\{\begin{array}{c}(a_1 \vee a_2 \vee \cdots \vee a_k)\\ \text{ικανοποιείται}\end{array}\right\} \Leftrightarrow \left\{\begin{array}{c}\text{υπάρχει μια διευθέτηση των } y_i \text{ για την οποία οι}\\ (a_1 \vee a_2 \vee y_1)(\overline{y}_1 \vee a_3 \vee y_2)\cdots(\overline{y}_{k-3} \vee a_{k-1} \vee a_k)\\ \text{ικανοποιούνται όλοι}\end{array}\right\}$$

Για να το δείτε αυτό, υποθέστε καταρχήν ότι ικανοποιούνται όλοι οι όροι στα δεξιά. Τότε τουλάχιστον ένα από τα στοιχεία $a_1, \ldots, a_k$ πρέπει να είναι αληθές — διαφορετικά το $y_1$ θα έπρεπε να είναι αληθές, το οποίο με τη σειρά του θα ανάγκαζε το $y_2$ να είναι αληθές, και ούτω καθ' εξής, κάνοντας τελικά ψευδή τον τελευταίο όρο. Αλλά αυτό σημαίνει ότι ικανοποιείται επίσης και ο όρος $(a_1 \vee a_2 \vee \cdots \vee a_k)$.

Αντιστρόφως, αν ικανοποιείται ο όρος $(a_1 \vee a_2 \vee \cdots \vee a_k)$, τότε κάποια $a_i$ πρέπει να είναι αληθή. Θέτουμε την τιμή `true` στα $y_1, \ldots, y_{i-2}$ και στα υπόλοιπα την τιμή `false`. Αυτό διασφαλίζει ότι ικανοποιούνται όλοι οι όροι στα δεξιά.

Έτσι, οποιοδήποτε στιγμιότυπο του SAT μπορεί να μετασχηματιστεί σε ένα ισοδύναμο στιγμιότυπο του 3SAT. Στην πραγματικότητα, το 3SAT παραμένει δύσκολο ακόμη και με τον επιπλέον περιορισμό να μην εμφανίζεται καμία μεταβλητή σε περισσότερους από τρεις όρους. Για να το δείξουμε αυτό, πρέπει με κάποιον τρόπο να απαλλαγούμε από όποια μεταβλητή εμφανίζεται πάρα πολλές φορές.



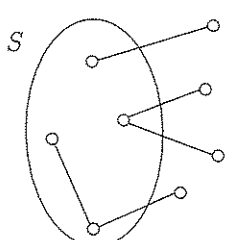
Ακολουθεί η αναγωγή από το 3SAT στην περιορισμένη έκδοσή του. Υποθέστε ότι στο στιγμιότυπο του 3SAT η μεταβλητή $x$ εμφανίζεται σε $k > 3$ όρους. Τότε αντικαθιστούμε την πρώτη της εμφάνιση με $x_1$, τη δεύτερη εμφάνιση με $x_2$, και ούτω καθεξής, αντικαθιστώντας κάθε μία από τις $k$ εμφανίσεις της με μια διαφορετική νέα μεταβλητή. Τέλος, προσθέτουμε τους όρους

$$(\overline{x}_1 \vee x_2)(\overline{x}_2 \vee x_3)\cdots(\overline{x}_k \vee x_1).$$

Και επαναλαμβάνουμε αυτό το βήμα για κάθε μεταβλητή η οποία εμφανίζεται περισσότερες από τρεις φορές.

Είναι εύκολο να δούμε ότι στο νέο τύπο καμία μεταβλητή δεν εμφανίζεται περισσότερες από τρεις φορές (και στην πραγματικότητα, κανένα στοιχείο δεν εμφανίζεται περισσότερες από δύο φορές). Επίσης, οι επιπλέον όροι που περιλαμβάνουν τις μεταβλητές $x_1$, $x_2$, ..., $x_k$ περιορίζουν αυτές τις μεταβλητές ώστε να έχουν την ίδια τιμή· βλέπετε γιατί; Επομένως το αρχικό στιγμιότυπο του 3SAT είναι ικανοποιήσιμο, αν και μόνο αν είναι ικανοποιήσιμο το περιορισμένο στιγμιότυπο.

**Εικόνα 8.9** Το $S$ είναι κάλυψη κορυφών, αν και μόνο αν το $V - S$ είναι ένα ανεξάρτητο σύνολο.




## ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ → ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ

Κάποιες αναγωγές βασίζονται στην επινοητικότητα του συσχετισμού δύο πολύ διαφορετικών προβλημάτων. Άλλες απλώς καταγράφουν το γεγονός ότι ένα πρόβλημα αποτελεί λεπτή μεταμφίεση ενός άλλου. Για να αναγάγουμε το ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ στην ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ πρέπει απλώς να παρατηρήσουμε ότι ένα σύνολο κόμβων $S$ αποτελεί κάλυψη κορυφών του γραφήματος $G = (V, E)$ (δηλαδή το $S$ αγγίζει κάθε κορυφή του $E$), αν και μόνο αν οι υπόλοιποι κόμβοι, $V - S$, είναι ένα ανεξάρτητο σύνολο του $G$ (Εικόνα 8.9).

Επομένως, για να λύσουμε ένα στιγμιότυπο $(G, g)$ του προβλήματος ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ, απλώς αναζητούμε μια κάλυψη κορυφών του $G$ με $|V| - g$ κόμβους. Αν υπάρχει μια τέτοια κάλυψη κορυφών, τότε παίρνουμε όλους τους κόμβους που *δεν* ανήκουν σε αυτή.

Αν δεν υπάρχει κάλυψη κορυφών, τότε το γράφημα $G$ δεν είναι δυνατό να έχει ένα ανεξάρτητο σύνολο μεγέθους $g$.

ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ ⟶ ΚΛΙΚΑ

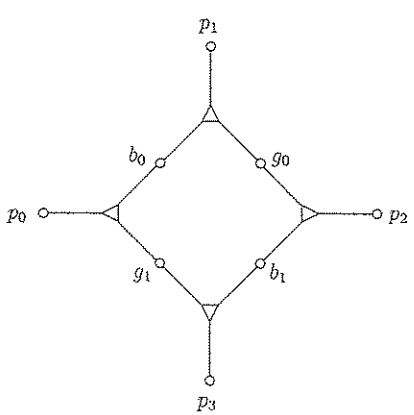
Τα προβλήματα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ και ΚΛΙΚΑ ανάγονται εύκολα το ένα στο άλλο. Ορίζουμε το *συμπλήρωμα* (compliment) ενός γραφήματος $G = (V, E)$ ως το $\overline{G} = (V, \overline{E})$, όπου το $\overline{E}$ περιέχει μόνο εκείνα τα μη διατεταγμένα ζεύγη κορυφών τα οποία δεν ανήκουν στο $E$. Τότε ένα σύνολο κόμβων $S$ είναι ένα ανεξάρτητο σύνολο του $G$, αν και μόνο αν το $S$ είναι μια κλίκα (clique) του $\overline{G}$. Για να το διατυπώσουμε με διαφορετικό τρόπο, αυτοί οι κόμβοι δεν έχουν ακμές μεταξύ τους στο γράφημα $G$, αν και μόνο αν έχουν όλες τις δυνατές ακμές μεταξύ τους στο γράφημα $\overline{G}$.

Επομένως, μπορούμε να αναγάγουμε το ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ στο πρόβλημα ΚΛΙΚΑ με απεικόνιση ενός στιγμιοτύπου $(G, g)$ του προβλήματος ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ στο αντίστοιχο στιγμιότυπο $(\overline{G}, g)$ του προβλήματος ΚΛΙΚΑ· η λύση και στα δύο είναι παρόμοια.

3SAT ⟶ ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ

Και πάλι, δύο πολύ διαφορετικά προβλήματα. Πρέπει να αναγάγουμε το 3SAT στο πρόβλημα της εύρεσης, μεταξύ των τριάδων αγόρι-κορίτσι-κατοικίδιο, ενός υποσυνόλου το οποίο να περιέχει το κάθε αγόρι, το κάθε κορίτσι, και το κάθε κατοικίδιο μόνο μία φορά. Εν ολίγοις, πρέπει να σχεδιάσουμε σύνολα τριάδων αγόρι-κορίτσι-κατοικίδιο τα οποία με κάποιον τρόπο να συμπεριφέρονται σαν λογικές (Boolean) μεταβλητές και πύλες!

Θεωρήστε το ακόλουθο σύνολο τεσσάρων τριάδων, όπου κάθε μία αναπαριστάνεται από έναν τριγωνικό κόμβο ο οποίος συνδέει ένα αγόρι, ένα κορίτσι και ένα κατοικίδιο:






Υποθέστε ότι τα δύο αγόρια $b_0$ και $b_1$ και τα δύο κορίτσια $g_0$ και $g_1$ δεν εμπλέκονται σε άλλες τριάδες. (Τα τέσσερα κατοικίδια $p_0$, ..., $p_3$ φυσικά θα ανήκουν επίσης και σε άλλες τριάδες· επειδή διαφορετικά το στιγμιότυπο δε θα είχε λύση.) Τότε, μια αντιστοίχιση πρέπει να περιέχει είτε τις δύο τριάδες $(b_0, g_1, p_0)$, $(b_1, g_0, p_2)$ είτε τις δύο τριάδες $(b_0, g_0, p_1)$, $(b_1, g_1, p_3)$, επειδή αυτοί είναι οι μόνοι τρόποι με τους οποίους αυτά τα δύο αγόρια και κορίτσια μπορούν να βρουν ένα ταίρι. Επομένως αυτό το «μικροεργαλείο» (gadget) έχει δύο δυνατές καταστάσεις: συμπεριφέρεται σαν μια λογική (Boolean) μεταβλητή!

Έτσι, για να μετασχηματίσουμε ένα στιγμιότυπο του 3SAT σε στιγμιότυπο από την ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ, ξεκινάμε με τη δημιουργία ενός αντιγράφου του προηγούμενου μικροεργαλείου για *κάθε* μεταβλητή $x$. Ονομάζουμε τους κόμβους που προκύπτουν $p_{x1}$, $b_{x0}$, $g_{x1}$, και ούτω καθεξής. Η επιδιωκόμενη ερμηνεία είναι ότι το αγόρι $b_{x0}$ αντιστοιχίζεται με το κορίτσι $g_{x1}$ αν $x$ = `true`, και με το κορίτσι $g_{x0}$ αν $x$ = `false`.

Κατόπιν πρέπει να δημιουργήσουμε τριάδες οι οποίες κατά κάποιον τρόπο θα μιμούνται τους όρους. Για κάθε όρο, έστω τον $c = (x \vee \overline{y} \vee z)$, εισάγουμε ένα νέο αγόρι $b_c$ και ένα νέο κορίτσι $g_c$. Αυτά θα συμμετάσχουν σε τρεις τριάδες, μία για κάθε στοιχείο του όρου. Και τα κατοικίδια σε αυτές τις τριάδες πρέπει να αντανακλούν τους τρεις τρόπους με τους οποίους μπορεί να ικανοποιηθεί ο όρος: (1) $x$ = `true`, (2) $y$ = `false`, (3) $z$ = `true`. Για τον (1) έχουμε την τριάδα $(b_c, g_c, p_{x1})$, όπου $p_{x1}$ είναι το κατοικίδιο $p_1$ στο μικροεργαλείο για το $x$. Ορίστε γιατί επιλέξαμε το $p_1$: αν $x$ = `true`, τότε το αγόρι $b_{x0}$ αντιστοιχίζεται με το κορίτσι $g_{x1}$ και το αγόρι $b_{x1}$ με το κορίτσι $g_{x0}$, και έτσι λαμβάνονται τα κατοικίδια $p_{x0}$ και $p_{x2}$. Οπότε σε αυτή την περίπτωση το αγόρι $b_c$ και το κορίτσι $g_c$ μπορούν να αντιστοιχιστούν με το κατοικίδιο $p_{x1}$. Αν όμως $x$ = `false`, τότε λαμβάνονται τα κατοικίδια $p_{x1}$ και $p_{x3}$, οπότε δεν μπορούν το κορίτσι $g_c$ και το αγόρι $b_c$ να προσαρμοστούν με αυτόν τον τρόπο. Κάνουμε το ίδιο πράγμα για τα άλλα δύο στοιχεία του όρου, και οδηγούμαστε στις τριάδες που περιλαμβάνουν το αγόρι $b_c$ και το κορίτσι $g_c$ είτε με τα κατοικίδια $p_{y0}$ ή $p_{y2}$ (για τη μορφή άρνησης της μεταβλητής $y$), είτε με το $p_{z1}$ ή με το $p_{z3}$ (για τη μεταβλητή $z$).

Πρέπει να διασφαλίσουμε ότι για κάθε παρουσία ενός στοιχείου σε έναν όρο $c$ υπάρχει ένα διαφορετικό κατοικίδιο για να το αντιστοιχίσουμε με το αγόρι $b_c$ και το κορίτσι $g_c$. Αλλά αυτό είναι εύκολο: από μια προγενέστερη αναγωγή μπορούμε να υποθέσουμε ότι κανένα στοιχείο δεν εμφανίζεται περισσότερες από δύο φορές, οπότε κάθε μικροεργαλείο μεταβλητών έχει αρκετά κατοικίδια, δύο για τις παρουσίες με άρνηση και δύο για τις παρουσίες χωρίς άρνηση.

Η αναγωγή τώρα φαίνεται πλήρης: από οποιαδήποτε αντιστοίχιση μπορούμε να ανακτήσουμε μια ικανοποιούσα ανάθεση τιμών αληθείας — εξετάζουμε απλώς κάθε μικροεργαλείο μεταβλητών και ελέγχουμε με ποιο κορίτσι αντιστοιχίστηκε το αγόρι $b_{x0}$. Και από μια ικανοποιούσα ανάθεση τιμών αληθείας μπορούμε να αντιστοιχίσουμε το μικροεργαλείο που ισοδυναμεί σε κάθε μεταβλητή $x$ έτσι ώστε να επιλεγούν οι τριάδες

($b_{x0}$, $g_{x1}$, $p_{x0}$) και ($b_{x1}$, $g_{x0}$, $p_{x2}$) αν $x$ = `true`, ή να επιλεγούν οι τριάδες ($b_{x0}$, $g_{x0}$, $p_{x1}$) και ($b_{x1}$, $g_{x1}$, $p_{x3}$) αν $x$ = `false`· και για κάθε όρο $c$ αντιστοιχίζουμε το αγόρι $b_c$ και το κορίτσι $b_c$ με το κατοικίδιο το οποίο αντιστοιχεί σε ένα από τα στοιχεία που τον ικανοποιούν.

Αλλά παραμένει ένα τελευταίο πρόβλημα: στην αντιστοίχιση που ορίστηκε στο τέλος της προηγούμενης παραγράφου, *κάποια κατοικίδια μπορεί να μείνουν χωρίς να αντιστοιχιστούν*. Στην πραγματικότητα, αν υπάρχουν $n$ μεταβλητές και $m$ όροι, τότε ακριβώς $2n - m$ κατοικίδια θα απομείνουν χωρίς να αντιστοιχιστούν (μπορείτε να επιβεβαιώσετε ότι αυτός ο αριθμός θα είναι σίγουρα θετικός, επειδή έχουμε το πολύ τρεις παρουσίες για κάθε μεταβλητή, και τουλάχιστον δύο στοιχεία σε κάθε όρο). Αλλά αυτό είναι εύκολο να διορθωθεί: Προσθέτουμε $2n - m$ νέα ζευγάρια αγοριών-κοριτσιών που είναι «λάτρεις όλων των ζώων», και τα αντιστοιχίζουμε μέσω τριάδων με όλα τα κατοικίδια!

### ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ → ZOE

Θυμηθείτε ότι στο ZOE μάς δίνεται ένας πίνακας **A**, διαστάσεων $m \times n$, με καταχωρίσεις 0 – 1, και πρέπει να βρούμε ένα 0 – 1 διάνυσμα $\mathbf{x} = (x_1, \ldots, x_n)$ τέτοιο ώστε οι $m$ εξισώσεις

$$\mathbf{Ax} = \mathbf{1}$$

να ικανοποιούνται, όπου με **1** υποδηλώνουμε το διάνυσμα στήλης που αποτελείται από άσους. Πώς μπορούμε να εκφράσουμε το πρόβλημα ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ σε αυτό το πλαίσιο;

Τα προβλήματα ZOE και ILP είναι πολύ χρήσιμα προβλήματα ακριβώς επειδή παρέχουν μια μορφή στην οποία μπορούν να εκφραστούν πολλά συνδυαστικά προβλήματα. Σε μια τέτοια διατύπωση, θεωρούμε ότι οι μεταβλητές 0 – 1 περιγράφουν μια λύση, και γράφουμε εξισώσεις οι οποίες εκφράζουν τους περιορισμούς του προβλήματος.

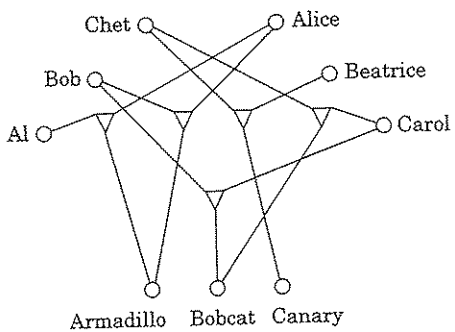
Για παράδειγμα, ορίστε πώς θα εκφράσουμε ένα στιγμιότυπο από την ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ ($m$ αγόρια, $m$ κορίτσια, $m$ κατοικίδια, και $n$ τριάδες αγόρι-κορίτσι-κατοικίδιο) στη γλώσσα του προβλήματος ZOE. Έχουμε μεταβλητές 0 – 1 $x_1, \ldots, x_n$, μία για κάθε τριάδα, όπου $x_i = 1$ σημαίνει ότι έχει επιλεγεί η $i$-οστή τριάδα για την αντιστοίχιση και $x_i = 0$ σημαίνει ότι δεν έχει επιλεγεί.

Τώρα το μόνο που πρέπει να κάνουμε είναι να γράψουμε εξισώσεις οι οποίες να δηλώνουν ότι η λύση που περιγράφεται από τις $x_i$ είναι μια έγκυρη αντιστοίχιση. Για κάθε αγόρι (ή κορίτσι, ή κατοικίδιο), υποθέστε ότι οι τριάδες οι οποίες περιέχουν το αγόρι (ή το κορίτσι ή το κατοικίδιο) είναι εκείνες που έχουν αρίθμηση $j_1, j_2, \ldots, j_k$· τότε η κατάλληλη εξίσωση είναι

$$x_{j_1} + x_{j_2} + \cdots + x_{j_k} = 1,$$



η οποία δηλώνει ότι στην αντιστοίχιση πρέπει να συμπεριληφθεί μόνο μία από αυτές τις τριάδες. Για παράδειγμα, ακολουθεί ο πίνακας **A** για ένα στιγμιότυπο από την ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ που είδαμε νωρίτερα




$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

Οι πέντε στήλες του **A** αντιστοιχούν στις πέντε τριάδες, ενώ οι εννέα γραμμές είναι για τους Al, Bob, Chet, Alice, Beatrice, Carol, Armadillo, Bobcat, και Canary, αντίστοιχα.

Είναι απλό να ισχυριστούμε ότι οι λύσεις των δύο στιγμιοτύπων μεταφράζονται αμφίδρομα.

### ZOE → ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ

Αυτή είναι μια αναγωγή ανάμεσα σε δύο ειδικές περιπτώσεις του ILP: το ένα με πολλές εξισώσεις αλλά μόνο με συντελεστές 0 – 1, και το άλλο με μία μόνο εξίσωση αλλά αυθαίρετους ακέραιους συντελεστές. Η αναγωγή βασίζεται σε μια απλή και δοκιμασμένη στο χρόνο ιδέα: τα διανύσματα με 0 – 1 μπορούν να κωδικοποιούν αριθμούς!

Για παράδειγμα, με δεδομένο αυτό το στιγμιότυπο του ZOE:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

αναζητούμε ένα σύνολο στηλών του **A** που, αν προστεθούν μεταξύ τους, κάνουν το διάνυσμα με όλες τις συνιστώσες να είναι 1. Αλλά αν θεωρήσουμε ότι οι στήλες είναι δυαδικοί ακέραιοι (αν τους διαβάσουμε από επάνω προς τα κάτω), αναζητούμε ένα υποσύνολο των ακεραίων 18, 5, 4, 8 το οποίο να αποτελείται από ακεραίους με άθροισμα το δυαδικό ακέραιο $11111_2 = 31$. Και αυτό είναι ένα στιγμιότυπο του προβλήματος ΑΘΡΟΙΣΜΑ ΥΠΟΣΥΝΟΛΟΥ. Η αναγωγή είναι πλήρης!

Εκτός από μια μικρή λεπτομέρεια, η οποία συνήθως αμαυρώνει τη στενή σχέση ανάμεσα σε διανύσματα με 0 – 1 και δυαδικούς ακεραίους: το *κρατούμενο*. Λόγω του κρατουμένου, το άθροισμα δυαδικών ακεραίων με 5 bit μπορεί να είναι 31 (για παράδειγμα, 5 + 6 + 20 = 31 ή, στο δυαδικό, $00101_2 + 00110_2 + 10100_2 = 11111_2$) αν και το άθροισμα των αντιστοίχων διανυσμάτων δεν είναι (1, 1, 1, 1, 1). Αλλά αυτό είναι εύκολο να διορθωθεί: Θεωρούμε τα διανύσματα στήλης όχι ως ακεραίους στη βάση 2, αλλά ως ακεραίους στη βάση $n + 1$ — κατά ένα παραπάνω από το πλήθος των στηλών. Με αυτόν τον τρόπο, επειδή προστίθενται το πολύ $n$ ακέραιοι και όλα τα ψηφία τους είναι 0 και 1, δεν μπορεί να υπάρχει κρατούμενο και η αναγωγή μας λειτουργεί.

### ZOE → ILP

Το 3SAT είναι μια ειδική περίπτωση του SAT — ή, το SAT είναι μια γενίκευση του 3SAT. Με τη φράση *ειδική περίπτωση* εννοούμε ότι τα στιγμιότυπα του 3SAT αποτελούν ένα υποσύνολο των στιγμιοτύπων του SAT (ειδικότερα, είναι αυτά χωρίς μακροσκελείς όρους), και ο ορισμός της λύσης είναι ίδιος και στα δύο προβλήματα (μια ανάθεση τιμών η οποία ικανοποιεί όλους τους όρους). Συνεπώς, υπάρχει μια αναγωγή από το 3SAT στο SAT, στην οποία η είσοδος δεν υφίσταται μετασχηματισμό και η λύση του στιγμιοτύπου προορισμού παραμένει επίσης αμετάβλητη. Με άλλα λόγια, οι συναρτήσεις $f$ και $h$ του διαγράμματος αναγωγής (στη σελίδα 315) είναι και οι δύο η ταυτοτική συνάρτηση.

Αυτό ακούγεται αρκετά τετριμμένο, αλλά είναι ένας πολύ χρήσιμος και συνηθισμένος τρόπος για να δείξουμε ότι ένα πρόβλημα είναι **NP**-πλήρες: Απλώς παρατηρούμε ότι πρόκειται για μια γενίκευση ενός γνωστού **NP**-πλήρους προβλήματος. Για παράδειγμα, το πρόβλημα ΚΑΛΥΨΗ ΣΥΝΟΛΟΥ είναι **NP**-πλήρες επειδή είναι μια γενίκευση του προβλήματος ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ (και επίσης, παρεμπιπτόντως, του προβλήματος ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ). Στην Άσκηση 8.10 θα δούμε περισσότερα παραδείγματα.

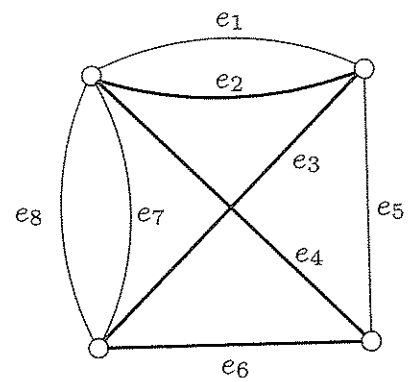
Συνήθως χρειάζεται λίγη δουλειά για να δείξουμε ότι ένα πρόβλημα είναι μια ειδική περίπτωση ενός άλλου. Η αναγωγή από το ZOE στο ILP είναι ένα σχετικό παράδειγμα. Στο LP αναζητούμε ένα ακέραιο διάνυσμα $\mathbf{x}$ το οποίο ικανοποιεί την ανίσωση $\mathbf{Ax} \leq \mathbf{b}$, για δεδομένο πίνακα $\mathbf{A}$ και διάνυσμα $\mathbf{b}$. Για να γράψουμε ένα στιγμιότυπο του ZOE σε αυτή ακριβώς τη μορφή, πρέπει να ξαναγράψουμε κάθε εξίσωση του στιγμιοτύπου ZOE ως δύο ανισώσεις (θυμηθείτε τους μετασχηματισμούς της Ενότητας 7.1.4), και να προσθέτουμε για κάθε μεταβλητή $x_i$ τις ανισώσεις $x_i \leq 1$ και $-x_i \leq 0$.

### ZOE → ΚΥΚΛΟΣ RUDRATA

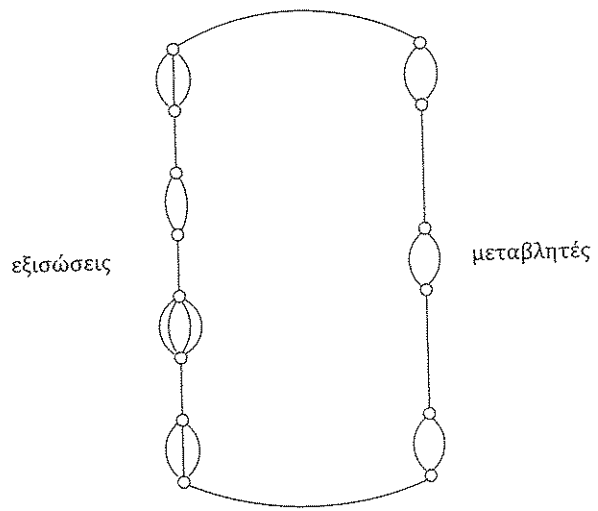
Στο πρόβλημα ΚΥΚΛΟΣ RUDRATA αναζητούμε σε ένα γράφημα έναν κύκλο ο οποίος επισκέπτεται κάθε κορυφή μόνο μία φορά. Θα αποδείξουμε ότι είναι **NP**-πλήρες σε δύο στάδια: πρώτα θα αναγάγουμε το ZOE σε μια γενίκευση του προβλήματος ΚΥΚΛΟΣ RUDRATA, με την ονομασία ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ (Rudrata cycle with paired edges), και στη συνέχεια θα δείξουμε πώς θα απαλλαγούμε από τα επιπλέον χαρακτηριστικά αυτού του προβλήματος και θα το αναγάγουμε στο απλό πρόβλημα ΚΥΚΛΟΣ RUDRATA.



**Εικόνα 8.10** Κύκλος Rudrata με συζευγμένες ακμές:
$C = \{(e_1, e_3), (e_5, e_6), (e_4, e_5), (e_3, e_7), (e_3, e_8)\}$.




**Εικόνα 8.11** Αναγωγή του ZOE στο πρόβλημα ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ.



Σε ένα στιγμιότυπο του προβλήματος ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ μας δίνεται ένα γράφημα $G = (V, E)$ και ένα σύνολο $C \subseteq E \times E$ με ζεύγη ακμών. Αναζητούμε έναν κύκλο ο οποίος (1) επισκέπτεται όλες τις κορυφές μία φορά, όπως πρέπει να κάνει ένας κύκλος Rudrata, και (2) για κάθε ζεύγος ακμών $(e, e')$ του $C$, διασχίζει είτε την ακμή $e$ είτε την $e'$ — *μόνο μία* από αυτές. Στο απλό παράδειγμα της Εικόνας 8.10 παρουσιάζεται μια λύση με έντονη γραμμή. Παρατηρήστε ότι επιτρέπουμε δύο ή περισσότερες παράλληλες ακμές ανάμεσα σε δύο κόμβους — μια δυνατότητα η οποία δεν έχει νόημα στα περισσότερα προβλήματα γραφημάτων — καθώς τώρα τα διαφορετικά αντίγραφα μιας ακμής μπορούν να συζευχθούν με άλλα αντίγραφα ακμών με τρόπους που πράγματι έχουν διαφορά.

Τώρα ας δούμε την αναγωγή του ΖΟΕ στο πρόβλημα ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ. Με δεδομένο ένα στιγμιότυπο του ΖΟΕ, $\mathbf{Ax} = \mathbf{1}$ (όπου $\mathbf{A}$ είναι ένας πίνακας $m \times n$ με καταχωρίσεις 0 – 1, και επομένως περιγράφει $m$ εξισώσεις με $n$ μεταβλητές), το γράφημα που κατασκευάζουμε έχει την πολύ απλή δομή που παρουσιάζεται στην Εικόνα 8.11: ένας κύκλος που συνδέει $m + n$ συλλογές παράλληλων ακμών. Για κάθε μεταβλητή $x_i$ έχουμε δύο παράλληλες ακμές (αντιστοιχούν σε $x_i = 1$ και $x_i = 0$). Και για κάθε εξίσωση $x_{j_1} + \cdots + x_{j_k} = 1$ η οποία περιλαμβάνει $k$ μεταβλητές έχουμε $k$ παράλληλες ακμές, μία για κάθε μεταβλητή που εμφανίζεται στην εξίσωση. Αυτό είναι όλο το γράφημα. Προφανώς, οποιοσδήποτε κύκλος Rudrata σε αυτό το γράφημα πρέπει να διασχίζει $m + n$ συλλογές παράλληλων ακμών μία προς μία, επιλέγοντας μία ακμή από κάθε συλλογή. Με αυτόν τον τρόπο, ο κύκλος «επιλέγει» για κάθε μεταβλητή μια τιμή — 0 ή 1 — και, για κάθε εξίσωση, μια μεταβλητή η οποία εμφανίζεται σ' αυτή.

Η όλη αναγωγή δεν μπορεί να είναι τόσο απλή, φυσικά. Πρέπει επίσης κάπου να αντανακλαστεί η δομή του πίνακα $\mathbf{A}$ (και όχι μόνο οι διαστάσεις του), και έχει απομείνει μόνο μία θέση: το σύνολο $C$ των ζευγών των ακμών που είναι τέτοιο ώστε να διασχίζεται μόνο μία ακμή σε κάθε ζεύγος. Για κάθε εξίσωση (θυμηθείτε ότι συνολικά υπάρχουν $m$), και για κάθε μεταβλητή $x_i$ που εμφανίζεται σε αυτή, προσθέτουμε στο σύνολο $C$ το ζεύγος $(e, e')$ όπου $e$ είναι η ακμή η οποία αντιστοιχεί στην εμφάνιση της $x_i$ στη συγκεκριμένη εξίσωση (στην αριστερή πλευρά της Εικόνας 8.11), και $e'$ είναι η ακμή η οποία αντιστοιχεί στην ανάθεση της μεταβλητής $x_i = 0$ (στη δεξιά πλευρά της Εικόνας). Με αυτό ολοκληρώνεται η κατασκευή.

Πάρτε οποιαδήποτε λύση αυτού του στιγμιοτύπου του προβλήματος ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ. Όπως αναφέρθηκε πριν, η λύση αυτή επιλέγει μια τιμή για κάθε μεταβλητή και μια μεταβλητή για κάθε εξίσωση. Ισχυριζόμαστε ότι αυτές οι τιμές που έχουν επιλεγεί είναι μια λύση του αρχικού στιγμιοτύπου του ΖΟΕ. Αν μια μεταβλητή $x_i$ έχει τιμή 1, τότε η ακμή $x_i = 0$ δε διασχίζεται, και έτσι όλες οι ακμές που σχετίζονται με την $x_i$ στην πλευρά της εξίσωσης πρέπει να διασχιστούν (καθώς είναι συζευγμένες στο $C$ με την ακμή $x_i = 0$). Έτσι, σε κάθε εξίσωση μόνο μία από τις μεταβλητές εμφανίζεται να έχει τιμή 1 — που είναι το ίδιο πράγμα με το να πούμε ότι όλες οι εξισώσεις ικανοποιούνται.



Η άλλη κατεύθυνση είναι επίσης απλή: από μια λύση του στιγμιοτύπου του ΖΟΕ μπορούμε εύκολα να πάρουμε έναν κατάλληλο κύκλο Rudrata.

## Απαλλαγή από τα ζεύγη ακμών

Μέχρι τώρα έχουμε μια αναγωγή από το ΖΟΕ στο ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ· αλλά στην πραγματικότητα ενδιαφερόμαστε για το πρόβλημα ΚΥΚΛΟΣ RUDRATA το οποίο είναι μια ειδική περίπτωση του προβλήματος με συζευγμένες ακμές: η περίπτωση στην οποία το σύνολο των ζευγών $C$ είναι κενό. Για να επιτύχουμε το στόχο μας, χρειαζόμαστε, ως συνήθως, να βρούμε έναν τρόπο για να απαλλαγούμε από το ανεπιθύμητο χαρακτηριστικό — στην προκειμένη περίπτωση τα ζεύγη ακμών.

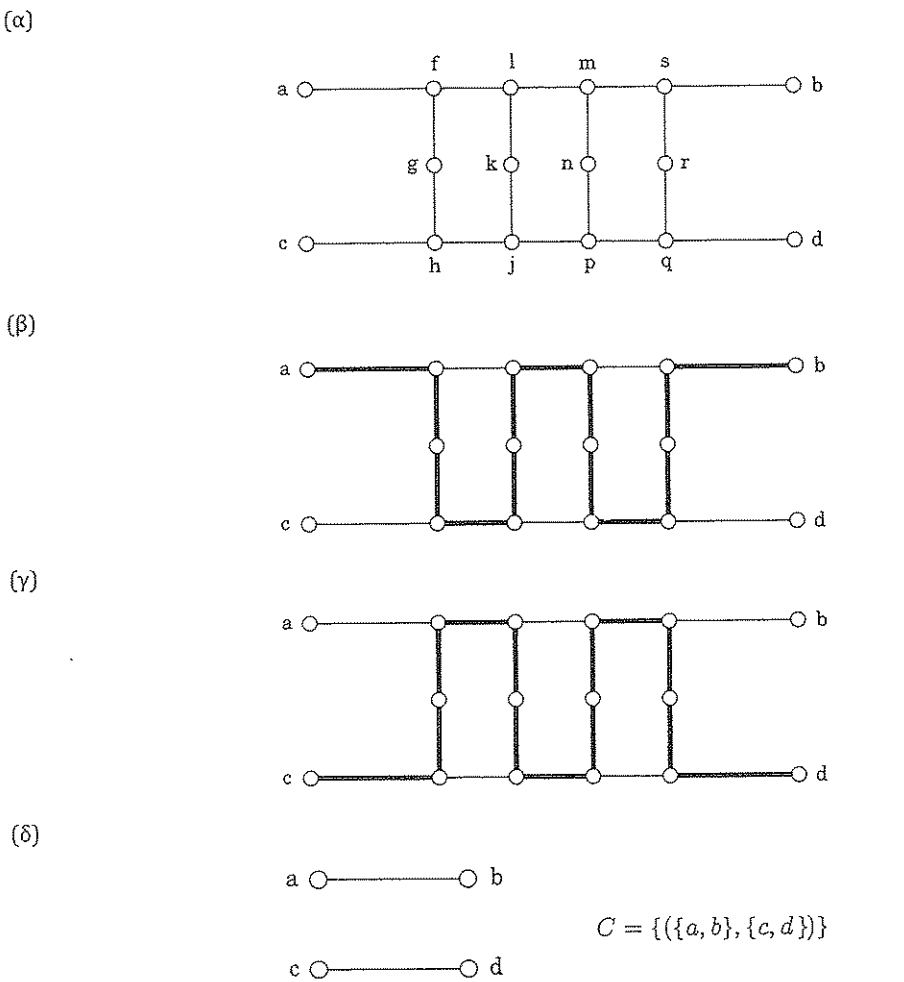
Θεωρήστε το γράφημα που παρουσιάζεται στην Εικόνα 8.12, και υποθέστε ότι είναι τμήμα ενός μεγαλύτερου γραφήματος $G$ με τέτοιον τρόπο ώστε μόνο τα τέσσερα άκρα $a, b, c, d$ να αγγίζουν το υπόλοιπο γράφημα. Ισχυριζόμαστε ότι αυτό το γράφημα έχει την ακόλουθη σημαντική ιδιότητα: *σε οποιονδήποτε κύκλο Rudrata του $G$ το υπογράφημα που φαίνεται πρέπει να διασχιστεί με έναν από τους δύο τρόπους που παρουσιάζονται με έντονη γραμμή στην Εικόνα 8.12 (β) και (γ).* Ορίστε γιατί. Υποθέστε ότι ο κύκλος αρχικά εισέρχεται στο υπογράφημα από την κορυφή $a$ και συνεχίζει μέχρι την $f$. Τότε πρέπει να συνεχίσει με την κορυφή $g$, επειδή η $g$ έχει βαθμό 2 και έτσι πρέπει να δεχθεί επίσκεψη αμέσως μετά από την επίσκεψη που δέχεται κάποιος από τους γειτονικούς της κόμβους — διαφορετικά δεν υπάρχει τρόπος να συμπεριληφθεί στον κύκλο. Επομένως πρέπει να προχωρήσουμε στον κόμβο $h$, και εδώ φαίνεται ότι έχουμε μια επιλογή. Θα μπορούσαμε να συνεχίσουμε στον $j$ ή να επιστρέψουμε στον $c$. Αλλά αν ακολουθήσουμε τη δεύτερη επιλογή, πώς θα επισκεφθούμε το υπόλοιπο υπογράφημα; (Ένας κύκλος Rudrata δεν πρέπει να αφήσει κορυφή χωρίς επίσκεψη.) Είναι εύκολο να δούμε ότι αυτό θα ήταν αδύνατο, και έτσι από την κορυφή $h$ δεν έχουμε άλλη επιλογή παρά να συνεχίσουμε στην $j$ και από εκεί να επισκεφθούμε το υπόλοιπο γράφημα όπως φαίνεται στην Εικόνα 8.12 (β). Για λόγους συμμετρίας, αν ο κύκλος Rudrata εισέλθει σε αυτό το υπογράφημα από την κορυφή $c$, πρέπει να το διασχίσει όπως φαίνεται στην Εικόνα 8.12 (γ). Και αυτοί είναι οι μοναδικοί τρόποι.

Αλλά αυτή η ιδιότητα μάς λέει κάτι σημαντικό: αυτό το «μικροεργαλείο» συμπεριφέρεται όπως ακριβώς δύο ακμές $\{a, b\}$ και $\{c, d\}$ οι οποίες είναι συζευγμένες στο πρόβλημα ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ (δείτε την Εικόνα 8.12 (γ)).

Η υπόλοιπη αναγωγή είναι πλέον ξεκάθαρη: για να αναγάγουμε το πρόβλημα ΚΥΚΛΟΣ RUDRATA ΜΕ ΣΥΖΕΥΓΜΕΝΕΣ ΑΚΜΕΣ στο πρόβλημα ΚΥΚΛΟΣ RUDRATA διασχίζουμε τα ζεύγη του $C$ ένα προς ένα. Για να απαλλαγούμε από κάθε ζεύγος $(\{a, b\}, \{c, d\})$ αντικαθιστούμε τις δύο ακμές με το μικροεργαλείο της Εικόνας 8.12 (α). Για οποιοδήποτε άλλο ζεύγος του $C$ το οποίο περιλαμβάνει την ακμή $\{a, b\}$ αντικαθιστούμε την ακμή $\{a, b\}$ με τη νέα ακμή $\{a, f\}$, όπου η $f$ προέρχεται από το μικροεργαλείο: η διάσχιση της $\{a, f\}$ είναι από τώρα

και στο εξής μια ένδειξη ότι θα διασχιστεί η ακμή $\{a, b\}$ στο παλιό γράφημα. Παρομοίως, η ακμή $\{c, h\}$ αντικαθιστά την ακμή $\{c, d\}$. Μετά από $|C|$ τέτοιες αντικαταστάσεις (οι οποίες πραγματοποιούνται σε πολυωνυμικό χρόνο, καθώς κάθε αντικατάσταση προσθέτει μόνο 12 κορυφές στο γράφημα) έχουμε τελειώσει, και οι κύκλοι Rudrata στο γράφημα που προκύπτει θα είναι σε αμφιμονοσήμαντη αντιστοιχία με τους κύκλους Rudrata στο αρχικό γράφημα που συμμορφώνεται με τους περιορισμούς του $C$.

**Εικόνα 8.12** Ένα μικροεργαλείο το οποίο επιβάλλει συζευγμένη συμπεριφορά.






## Κύκλος Rudrata → TSP

Με δεδομένο ένα γράφημα $G = (V, E)$, κατασκευάστε το ακόλουθο στιγμιότυπο του TSP: το σύνολο των πόλεων είναι το ίδιο με το $V$, και η απόσταση ανάμεσα στις πόλεις $u$ και $v$ είναι 1 αν η $\{u, v\}$ είναι μια ακμή του $G$ και $1 + \alpha$ διαφορετικά, για κάποιο $\alpha > 1$ το οποίο θα προσδιοριστεί. Ο προϋπολογισμός του στιγμιοτύπου του TSP είναι ίσος με το πλήθος των κόμβων $|V|$.

Είναι εύκολο να δείτε ότι αν το γράφημα $G$ έχει έναν κύκλο Rudrata, τότε ο ίδιος κύκλος θα είναι επίσης μια διαδρομή στα όρια του προϋπολογισμού του στιγμιοτύπου του TSP· και ότι αντιστρόφως, αν το γράφημα $G$ δεν έχει κύκλο Rudrata, τότε δεν υπάρχει λύση: η φθηνότερη δυνατή περιήγηση TSP έχει κόστος τουλάχιστον $n + \alpha$ (πρέπει να χρησιμοποιήσει τουλάχιστον μία ακμή μήκους $1 + \alpha$, και το συνολικό μήκος όλων των άλλων $n - 1$ ακμών είναι τουλάχιστον $n - 1$). Έτσι το πρόβλημα ΚΥΚΛΟΣ RUDRATA ανάγεται στο TSP.

Σε αυτή την αναγωγή εισαγάγαμε την παράμετρο $\alpha$ επειδή μεταβάλλοντάς την μπορούμε να πάρουμε δύο ενδιαφέροντα αποτελέσματα. Αν $\alpha = 1$ τότε όλες οι αποστάσεις είναι 1 ή 2, οπότε αυτό το στιγμιότυπο του TSP ικανοποιεί την τριγωνική ανισότητα: αν $i, j, k$ είναι πόλεις, τότε $d_{ij} + d_{jk} \geq d_{ik}$ (απόδειξη: η σχέση $a + b \geq c$ ισχύει για οποιουσδήποτε αριθμούς $1 \leq a, b, c \leq 2$). Πρόκειται για μια ειδική περίπτωση του TSP που έχει πρακτική σημασία και η οποία, όπως θα δούμε στο Κεφάλαιο 9, κατά μία έννοια είναι ευκολότερη, επειδή μπορεί να *προσεγγιστεί* αποδοτικά.

Αν, από την άλλη πλευρά, η παράμετρος $\alpha$ είναι μεγάλη, τότε το στιγμιότυπο του TSP που προκύπτει μπορεί να μην ικανοποιεί την τριγωνική ανισότητα, αλλά έχει μια άλλη σημαντική ιδιότητα: είτε έχει μια λύση με κόστος $n$ ή μικρότερο, είτε όλες οι λύσεις του έχουν κόστος τουλάχιστον $n + \alpha$ (το οποίο τώρα μπορεί να είναι αυθαίρετα μεγαλύτερο από το $n$). Δεν μπορεί να υπάρχει τίποτε ενδιάμεσα! Όπως θα δούμε στο Κεφάλαιο 9, αυτή η σημαντική ιδιότητα *χάσματος* (gap) συνεπάγεται ότι δεν είναι δυνατό να υπάρξει προσεγγιστικός αλγόριθμος, εκτός και αν **P = NP**.

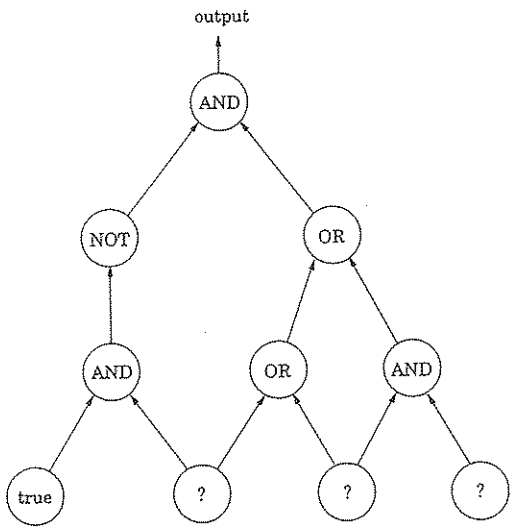
## Οποιοδήποτε Πρόβλημα στο NP → SAT

Στην Εικόνα 8.7 αναγάγαμε το SAT σε διάφορα προβλήματα αναζήτησης. Τώρα θα ολοκληρώσουμε τον κύκλο και θα ισχυριστούμε ότι όλα αυτά τα προβλήματα — και στην πραγματικότητα όλα τα προβλήματα στο **NP** — ανάγονται στο SAT.

Ειδικότερα, θα δείξουμε ότι όλα τα προβλήματα στο **NP** μπορούν να αναχθούν σε μια γενίκευση του SAT την οποία θα ονομάσουμε SAT ΚΥΚΛΩΜΑΤΟΣ (Circuit SAT). Στο SAT ΚΥΚΛΩΜΑΤΟΣ μας δίνεται ένα *λογικό* (Boolean) *κύκλωμα* (δείτε την Εικόνα 8.13 και θυμηθείτε την Ενότητα 7.7), ένα DAG (προσανατολισμένο άκυκλο γράφημα) του οποίου οι κορυφές είναι *πύλες* (gates) πέντε διαφορετικών τύπων:

- Οι πύλες AND και οι πύλες OR έχουν βαθμό εισόδου 2.

**Εικόνα 8.13** Ένα στιγμιότυπο του προβλήματος SAT ΚΥΚΛΩΜΑΤΟΣ.




- Οι πύλες NOT έχουν βαθμό εισόδου 1.
- Οι πύλες *γνωστής εισόδου* δεν έχουν εισερχόμενες ακμές και φέρουν ετικέτες `false` ή `true`.
- Οι πύλες *άγνωστης εισόδου* δεν έχουν εισερχόμενες ακμές και φέρουν ετικέτα «?».

Μία από τις τερματικές κορυφές του DAG καθορίζεται ως πύλη *εξόδου* (output gate).

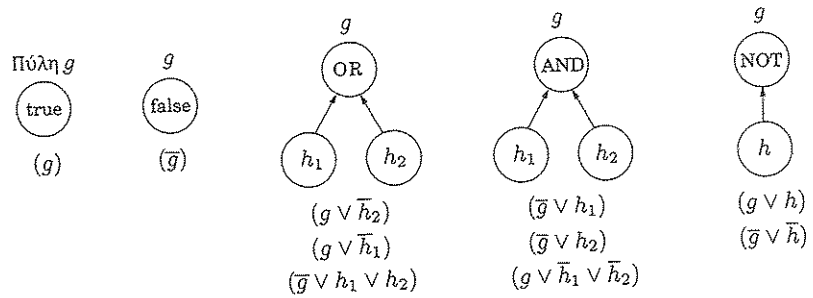
Με δεδομένη μια ανάθεση τιμών στις άγνωστες εισόδους, μπορούμε να αποτιμήσουμε τις πύλες του κυκλώματος με τοπολογική σειρά, χρησιμοποιώντας τους κανόνες της λογικής Boole (όπως `false` $\vee$ `true` = `true`), μέχρι να πάρουμε την τιμή στην πύλη εξόδου. Αυτή είναι η τιμή του κυκλώματος για τη συγκεκριμένη ανάθεση τιμών στις εισόδους. Για παράδειγμα, το κύκλωμα της Εικόνας 8.13 αποτιμάται σε τιμή `false` για την ανάθεση τιμών `true`, `false`, `true` (από αριστερά προς τα δεξιά).

Έτσι, το SAT ΚΥΚΛΩΜΑΤΟΣ είναι το ακόλουθο πρόβλημα αναζήτησης: Με δεδομένο ένα κύκλωμα, βρείτε μια ανάθεση τιμών αληθείας για τις άγνωστες εισόδους τέτοια ώστε η πύλη εξόδου να αποτιμάται σε τιμή `true`, ή αναφέρετε ότι δεν υπάρχει τέτοια ανάθεση. Για παράδειγμα, αν μας είχε δοθεί το κύκλωμα της Εικόνας 8.13, θα μπορούσαμε να έχουμε επιστρέψει την ανάθεση (`false`, `true`, `true`) επειδή, αν αντικαταστήσουμε αυτές τις τιμές στις άγνωστες εισόδους (από αριστερά προς τα δεξιά), η έξοδος γίνεται `true`.



Το SAT ΚΥΚΛΩΜΑΤΟΣ είναι μια γενίκευση του SAT. Για να δείτε γιατί, παρατηρήστε ότι το SAT αναζητά μια ικανοποιούσα ανάθεση τιμών αληθείας για ένα κύκλωμα το οποίο έχει την εξής απλή δομή: μια δέσμη από πύλες AND στην κορυφή συνδέουν τους όρους, και το αποτέλεσμα αυτού του μεγάλου AND είναι η έξοδος. Κάθε όρος είναι η διάζευξη (OR) των στοιχείων του. Και κάθε στοιχείο είναι είτε μια άγνωστη πύλη εισόδου είτε η άρνησή της (NOT). Δεν υπάρχουν γνωστές πύλες εισόδου.

Πηγαίνοντας προς την άλλη κατεύθυνση, το SAT ΚΥΚΛΩΜΑΤΟΣ μπορεί επίσης να αναχθεί στο SAT. Ορίστε πώς μπορούμε να ξαναγράψουμε οποιοδήποτε κύκλωμα σε συζευκτική κανονική μορφή —(conjunctive normal form — η σύζευξη AND των όρων): για κάθε πύλη $g$ του κυκλώματος δημιουργούμε μια μεταβλητή $g$ και μοντελοποιούμε την επίδραση της πύλης χρησιμοποιώντας μερικούς όρους:




(Βλέπετε γιατί αυτοί οι όροι πραγματικά επιβάλλουν ακριβώς το επιθυμητό αποτέλεσμα;) Και για να ολοκληρώσουμε, αν $g$ είναι η πύλη εξόδου, την αναγκάζουμε να είναι `true` με την προσθήκη του όρου ($g$). Το στιγμιότυπο του SAT το οποίο προκύπτει είναι ισοδύναμο με το δεδομένο στιγμιότυπο του προβλήματος SAT ΚΥΚΛΩΜΑΤΟΣ: οι ικανοποιούσες αναθέσεις τιμών αληθείας αυτής της συζευκτικής κανονικής μορφής βρίσκονται σε αμφιμονοσήμαντη αντιστοιχία με εκείνες του κυκλώματος.

Τώρα που γνωρίζουμε ότι το SAT ΚΥΚΛΩΜΑΤΟΣ ανάγεται στο SAT ας στραφούμε στην κύρια δουλειά μας, να δείξουμε ότι *όλα* τα προβλήματα αναζήτησης ανάγονται στο SAT ΚΥΚΛΩΜΑΤΟΣ. Έτσι, υποθέστε ότι το $A$ είναι ένα πρόβλημα στο **NP**. Πρέπει να ανακαλύψουμε μια αναγωγή από το $A$ στο SAT ΚΥΚΛΩΜΑΤΟΣ. Αυτό ακούγεται πολύ δύσκολο, *επειδή δεν γνωρίζουμε σχεδόν τίποτε για το A*!

Το μόνο που γνωρίζουμε για το $A$ είναι ότι πρόκειται για πρόβλημα αναζήτησης, οπότε πρέπει να χρησιμοποιήσουμε αυτή τη γνώση. Το κύριο χαρακτηριστικό ενός προβλήματος αναζήτησης είναι ότι οποιαδήποτε λύση μπορεί να ελεγχθεί γρήγορα: υπάρχει ένας

αλγόριθμος $C$ ο οποίος ελέγχει, με δεδομένο ένα στιγμιότυπο $I$ και μια προτεινόμενη λύση $S$, αν η $S$ είναι λύση του $I$ ή όχι. Επιπλέον, ο $C$ παίρνει αυτή την απόφαση σε πολυωνυμικό χρόνο που είναι συνάρτηση του μήκους του $I$ (μπορούμε να υποθέσουμε ότι η ίδια η λύση $S$ είναι κωδικοποιημένη ως μια δυαδική συμβολοσειρά, και γνωρίζουμε ότι το μήκος αυτής της συμβολοσειράς είναι πολυωνυμικό ως προς το μήκος του $I$).

Θυμηθείτε τώρα το επιχείρημά μας στην Ενότητα 7.7, ότι οποιοσδήποτε πολυωνυμικός αλγόριθμος μπορεί να μετατραπεί σε κύκλωμα· ένα κύκλωμα του οποίου οι πύλες εισόδου κωδικοποιούν την είσοδο στον αλγόριθμο. Φυσικά, για οποιοδήποτε μήκος εισόδου (πλήθος των bit εισόδου) το κύκλωμα θα προσαρμοστεί στο κατάλληλο πλήθος εισόδων, αλλά το συνολικό πλήθος των πυλών του κυκλώματος θα είναι πολυωνυμικό ως προς το πλήθος των εισόδων. Αν ο εν λόγω πολυωνυμικός αλγόριθμος επιλύει ένα πρόβλημα το οποίο απαιτεί μια απάντηση ναι ή όχι (όπως στην περίπτωση του $C$: «Κωδικοποιεί η $S$ μια λύση του στιγμιοτύπου το οποίο είναι κωδικοποιημένο με το $I$;»), τότε αυτή η απάντηση δίνεται στην πύλη εξόδου.

Συμπεραίνουμε ότι, με δεδομένο οποιοδήποτε στιγμιότυπο $I$ του προβλήματος $A$, μπορούμε να κατασκευάσουμε σε πολυωνυμικό χρόνο ένα κύκλωμα του οποίου οι γνωστές είσοδοι είναι τα bit του $I$ και οι άγνωστες είσοδοι είναι τα bit του $S$, έτσι ώστε η έξοδος να έχει τιμή `true`, αν και μόνο αν οι άγνωστες είσοδοι προοιωνίζονται μια λύση $S$ του $I$. Με άλλα λόγια, *οι ικανοποιούσες αναθέσεις τιμών αληθείας στις άγνωστες εισόδους του κυκλώματος βρίσκονται σε αμφιμονοσήμαντη αντιστοιχία με τις λύσεις του στιγμιοτύπου $I$ του $A$*. Η αναγωγή είναι πλήρης.

## Μη επιλύσιμα προβλήματα

Τουλάχιστον ένα **NP**-πλήρες πρόβλημα μπορεί να λυθεί από *κάποιον* αλγόριθμο — το δυσάρεστο είναι ότι αυτός ο αλγόριθμος θα είναι εκθετικός. Όπως όμως προκύπτει, υπάρχουν ωραιότατα υπολογιστικά προβλήματα για τα οποία *δεν υπάρχει κανένας αλγόριθμος*!

Ένα διάσημο πρόβλημα αυτού του είδους είναι μια αριθμητική εκδοχή του SAT. Με δεδομένη μια πολυωνυμική εξίσωση πολλών μεταβλητών, όπως για παράδειγμα η εξίσωση

$$x^3yz + 2y^4z^2 - 7xy^5z = 6,$$

υπάρχουν ακέραιες τιμές των $x$, $y$, $z$ οι οποίες να την ικανοποιούν; Δεν υπάρχει αλγόριθμος ο οποίος να λύνει αυτό το πρόβλημα. Κανένας αλγόριθμος, πολυωνυμικός, εκθετικός, διπλά εκθετικός, ή χειρότερος! Τέτοια προβλήματα ονομάζονται *μη επιλύσιμα* (unsolvable).

Το πρώτο μη επιλύσιμο πρόβλημα ανακαλύφθηκε το 1936 από τον Alan M.Turing, τότε φοιτητή των μαθηματικών στο Cambridge της Αγγλίας. Όταν το διατύπωσε ο Turing, δεν υπήρχαν υπολογιστές ή γλώσσες προγραμματισμού (στην πραγματικότητα, μπορούμε να ισχυριστούμε ότι αυτά τα πράγματα εμφανίστηκαν αργότερα *ακριβώς επειδή* είχε αυτή τη λαμπρή ιδέα ο Turing). Όμως σήμερα μπορούμε να το διατυπώσουμε με γνωστούς όρους.



## Μη επιλύσιμα προβλήματα (*Συνέχεια*)

Υποθέστε ότι σας δίνεται ένα πρόγραμμα στην αγαπημένη σας γλώσσα προγραμματισμού, μαζί με μια συγκεκριμένη είσοδο. Θα μπορέσει ποτέ να τερματιστεί το πρόγραμμα, εφόσον ξεκινήσει να λειτουργεί με αυτή την είσοδο; Πρόκειται για μια πολύ εύλογη ερώτηση. Πολλοί από εμάς θα ήταν κατενθουσιασμένοι αν είχαμε έναν αλγόριθμο, ας τον ονομάσουμε `terminates(p, x)`, ο οποίος θα δεχόταν ένα αρχείο με ένα πρόγραμμα p ως είσοδο, και ένα αρχείο δεδομένων x, και μετά από υπολογισμούς τελικά θα μας έλεγε αν θα μπορούσε να σταματήσει το p από τη στιγμή που θα ξεκινούσε με το x.

Πώς όμως θα γράφατε το πρόγραμμα `terminates`; (Αν δεν έχετε συναντήσει αυτό το πρόβλημα μέχρι τώρα, αξίζει να το σκεφθείτε για λίγο, για να εκτιμήσετε τη δυσκολία της συγγραφής ενός τέτοιου «καθολικού ανιχνευτή ατέρμονων βρόχων»).

Λοιπόν, δεν μπορείτε. *Τέτοιος αλγόριθμος δεν υπάρχει*!

Να και η απόδειξη: Υποθέστε ότι πραγματικά είχατε ένα τέτοιο πρόγραμμα `terminates (p, x)`. Τότε θα μπορούσατε να το χρησιμοποιήσετε ως υπορουτίνα του ακόλουθου κακού προγράμματος:

```
Συνάρτηση paradox(z:αρχείο)

1: if terminates(z, z) goto 1
```

Παρατηρήστε τι κάνει η συνάρτηση paradox: τερματίζει, αν και μόνο αν το πρόγραμμα $z$ *δεν* τερματίζει όταν του δοθεί ως είσοδος ο δικός του κώδικας.

Θα πρέπει να μυρίζεστε τα προβλήματα. Και αν τοποθετούσαμε αυτό το πρόγραμμα σε ένα αρχείο με το όνομα paradox και εκτελούσαμε το `paradox(paradox)`; Θα μπορούσε ποτέ να τερματίσει; Ή όχι; Καμία απάντηση δεν είναι δυνατή. Αφού καταλήξαμε σε αυτή την αντίφαση υποθέτοντας ότι υπάρχει ένας αλγόριθμος ο οποίος μπορεί να μας πει αν ένα πρόγραμμα τερματίζεται, πρέπει να συμπεράνουμε ότι αυτό το πρόβλημα δεν μπορεί να λυθεί από κανέναν αλγόριθμο.

Παρεμπιπτόντως, όλα αυτά μας λένε κάτι σημαντικό για τον προγραμματισμό: Ποτέ δε θα αυτοματοποιηθεί, και πάντοτε θα βασίζεται στην πειθαρχία, την επινοητικότητα, και τη σκληρή δουλειά. Γνωρίζουμε τώρα ότι δεν είστε σε θέση να πείτε αν ένα πρόγραμμα έχει έναν ατέρμονα βρόχο. Μπορείτε όμως να πείτε αν έχει υπερχείλιση χώρου προσωρινής αποθήκευσης (buffer overrun); Βλέπετε πώς μπορείτε να χρησιμοποιήσετε την αδυναμία επίλυσης του «προβλήματος τερματισμού» για να δείξετε ότι και αυτό το πρόβλημα είναι μη επιλύσιμο;

## Ασκήσεις

8.1. *Σύγκριση βελτιστοποίησης και αναζήτησης.* Θυμηθείτε το πρόβλημα του περιοδεύοντος πωλητή:

> TSP
> *Είσοδος:* Ένας πίνακας αποστάσεων· ένας προϋπολογισμός $b$
> *Έξοδος:* Μια περιήγηση η οποία διέρχεται από όλες τις πόλεις και έχει μήκος $\leq b$, αν υπάρχει τέτοια περιήγηση.

Η εκδοχή βελτιστοποίησης αυτού του προβλήματος ζητά άμεσα τη συντομότερη διαδρομή.

> TSP-OPT
> *Είσοδος:* Ένας πίνακας αποστάσεων
> *Έξοδος:* Η συντομότερη διαδρομή η οποία διέρχεται από όλες τις πόλεις.

Δείξτε ότι, αν μπορεί να λυθεί σε πολυωνυμικό χρόνο το πρόβλημα TSP, τότε μπορεί να λυθεί και το TSP-OPT.

8.2. *Σύγκριση αναζήτησης και λήψης απόφασης.* Υποθέστε ότι έχετε μια διαδικασία η οποία εκτελείται σε πολυωνυμικό χρόνο και σας λέει αν ένα γράφημα έχει κύκλο Rudrata ή όχι. Δείξτε ότι μπορείτε να τη χρησιμοποιήσετε για να αναπτύξετε έναν αλγόριθμο πολυωνυμικού χρόνου για το πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA (ο οποίος επιστρέφει την πραγματική διαδρομή, αν υπάρχει).

8.3. ΤΣΙΓΓΟΥΝΙΚΟ SAT (Stingy SAT) είναι το ακόλουθο πρόβλημα: με δεδομένο ένα σύνολο όρων (καθένας από τους οποίους είναι μια διάζευξη στοιχείων) και έναν ακέραιο $k$, βρείτε μια ικανοποιούσα ανάθεση τιμών αληθείας στην οποία το πολύ $k$ μεταβλητές έχουν τιμή `true`, αν υπάρχει τέτοια ανάθεση. Αποδείξτε ότι το ΤΣΙΓΓΟΥΝΙΚΟ SAT είναι **NP**-πλήρες.

8.4. Θεωρήστε το πρόβλημα ΚΛΙΚΑ περιορισμένο σε γραφήματα στα οποία κάθε κορυφή έχει βαθμό το πολύ 3. Ονομάστε αυτό το πρόβλημα ΚΛΙΚΑ-3.

(α) Αποδείξτε ότι το ΚΛΙΚΑ-3 ανήκει στο **NP**.

(β) Ποιο είναι το λάθος στην ακόλουθη απόδειξη της **NP**-πληρότητας του προβλήματος ΚΛΙΚΑ-3; Γνωρίζουμε ότι το πρόβλημα ΚΛΙΚΑ για γενικά γραφήματα είναι **NP**-πλήρες, οπότε αρκεί να παρουσιάσουμε μια αναγωγή από το ΚΛΙΚΑ-3 στο ΚΛΙΚΑ. Με δεδομένο ένα γράφημα $G$ με κορυφές βαθμού $\leq 3$ και μια παράμετρο $g$, η αναγωγή αφήνει το γράφημα και την παράμετρο αμετάβλητα: προφανώς η έξοδος της αναγωγής είναι μια δυνατή είσοδος για το πρόβλημα ΚΛΙΚΑ. Επιπλέον, η απάντηση και στα δύο προβλήματα είναι παρόμοια. Αυτό αποδεικνύει την ορθότητα της αναγωγής και, επομένως, την **NP**-πληρότητα του προβλήματος ΚΛΙΚΑ-3.



(γ) Είναι αλήθεια ότι το πρόβλημα ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ παραμένει **NP**-πλήρες ακόμη και όταν περιορίζεται σε γραφήματα στα οποία κάθε κορυφή έχει βαθμό το πολύ 3. Ονομάστε αυτό το πρόβλημα VC-3. Ποιο είναι το λάθος στην ακόλουθη απόδειξη της **NP**-πληρότητας του προβλήματος ΚΛΙΚΑ-3;

Παρουσιάζουμε μια αναγωγή από το VC-3 στο ΚΛΙΚΑ-3. Με δεδομένο ένα γράφημα $G = (V, E)$ με βαθμούς κόμβων φραγμένους από το 3 και μια παράμετρο $b$, δημιουργούμε ένα στιγμιότυπο του ΚΛΙΚΑ-3 αφήνοντας το γράφημα αμετάβλητο και αλλάζοντας την παράμετρο σε $|V| - b$. Τώρα, ένα υποσύνολο $C \subseteq V$ είναι κάλυψη κορυφών στο γράφημα $G$, αν και μόνο αν το συμπληρωματικό σύνολο $V - C$ είναι μια κλίκα στο $G$. Επομένως το γράφημα $G$ έχει κάλυψη κορυφών μεγέθους $\leq b$, αν και μόνο αν έχει μια κλίκα με μέγεθος $\geq |V| - b$. Αυτό αποδεικνύει την ορθότητα της αναγωγής και, κατά συνέπεια, την **NP**-πληρότητα του προβλήματος ΚΛΙΚΑ-3.

(δ) Περιγράψτε έναν αλγόριθμο $O(|V|)$ για το πρόβλημα ΚΛΙΚΑ-3.

8.5. Δώστε μια απλή αναγωγή από την ΤΡΙΔΙΑΣΤΑΤΗ ΑΝΤΙΣΤΟΙΧΙΣΗ στο SAT, και μια άλλη από τον ΚΥΚΛΟ RUDRATA στο SAT. (*Υπόδειξη:* Στην τελευταία περίπτωση μπορείτε να χρησιμοποιήσετε μεταβλητές $x_{ij}$ των οποίων η διαισθητική σημασία είναι «η κορυφή $i$ είναι η $j$-οστή κορυφή του κύκλου Rudrata»· κατόπιν θα πρέπει να γράψετε όρους που να εκφράζουν τους περιορισμούς του προβλήματος.)

8.6. Στη σελίδα 323 είδαμε ότι το πρόβλημα 3SAT παραμένει **NP**-πλήρες όταν περιορίζεται σε τύπους στους οποίους κάθε στοιχείο εμφανίζεται το πολύ δύο φορές.

(α) Δείξτε ότι αν κάθε στοιχείο εμφανίζεται το πολύ *μία φορά*, τότε το πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο.

(β) Δείξτε ότι το πρόβλημα ΑΝΕΞΑΡΤΗΤΟ ΣΥΝΟΛΟ παραμένει **NP**-πλήρες ακόμη και στην ειδική περίπτωση όπου όλοι οι κόμβοι του γραφήματος έχουν βαθμό μέχρι 4.

8.7. Θεωρήστε μια ειδική περίπτωση του προβλήματος 3SAT στην οποία όλοι οι όροι έχουν ακριβώς τρία στοιχεία και κάθε μεταβλητή εμφανίζεται ακριβώς τρεις φορές. Δείξτε ότι αυτό το πρόβλημα μπορεί να λυθεί σε πολυωνυμικό χρόνο. (*Υπόδειξη:* Δημιουργήστε ένα διμερές γράφημα με όρους στα αριστερά, μεταβλητές στα δεξιά, και ακμές όποτε μια μεταβλητή εμφανίζεται σε έναν όρο. Χρησιμοποιήστε την Άσκηση 7.30 για να δείξετε ότι αυτό το γράφημα έχει μια αντιστοίχιση.)

8.8. Στο πρόβλημα ΑΚΡΙΒΕΣ 4SAT (Exact 4SAT) η είσοδος είναι ένα σύνολο όρων όπου ο καθένας είναι μια διάζευξη ακριβώς τεσσάρων στοιχείων, και κάθε μεταβλητή εμφανίζεται το πολύ μία φορά σε κάθε όρο. Ο στόχος είναι να βρείτε μια ικανοποιούσα ανάθεση τιμών, αν υπάρχει. Αποδείξτε ότι το πρόβλημα ΑΚΡΙΒΕΣ 4SAT είναι **NP**-πλήρες.

8.9. Στο πρόβλημα ΕΠΙΤΥΧΕΣ ΣΥΝΟΛΟ (Hitting set) μας δίνεται μια οικογένεια συνόλων $\{S_1, S_2, ..., S_n\}$ και ένας προϋπολογισμός $b$, και επιθυμούμε να βρούμε ένα σύνολο $H$ μεγέθους $\leq b$ το οποίο τέμνει κάθε σύνολο $S_i$, αν υπάρχει τέτοιο σύνολο $H$. Με άλλα λόγια, θέλουμε $H \cap S_i \neq 0$ για κάθε $i$.

Δείξτε ότι το πρόβλημα ΕΠΙΤΥΧΕΣ ΣΥΝΟΛΟ είναι **NP**-πλήρες.

8.10. *Απόδειξη της* **NP***-πληρότητας με γενίκευση.* Αποδείξτε για τα προβλήματα που ακολουθούν ότι είναι **NP**-πλήρη δείχνοντας ότι αποτελούν *γενίκευση* κάποιου **NP**-πλήρους προβλήματος που έχουμε δει σε αυτό το κεφάλαιο.

(α) ΙΣΟΜΟΡΦΙΣΜΟΣ ΥΠΟΓΡΑΦΗΜΑΤΩΝ (Subgraph isomorphism): Με δεδομένα ως είσοδο δύο γράφήματα $G$ και $H$, προσδιορίστε αν το γράφημα $G$ είναι υπογράφημα του $H$ (δηλαδή, αν με τη διαγραφή ορισμένων κορυφών και ακμών από τον $H$ παίρνουμε ένα γράφημα το οποίο, εκτός από τη διαφορετική ονομασία των κορυφών, είναι παρόμοιο με το γράφημα $G$), και αν ναι, επιστρέψτε την αντίστοιχη απεικόνιση του $V(G)$ στο $V(H)$.

(β) ΜΕΓΑΛΥΤΕΡΗ ΔΙΑΔΡΟΜΗ (Longest path): Με δεδομένο ένα γράφημα $G$ και έναν ακέραιο $g$, βρείτε στο $G$ μια απλή διαδρομή μήκους $g$.

(γ) ΜΕΓΙΣΤΟ SAT (Max SAT): Με δεδομένο έναν τύπο CNF και έναν ακέραιο $g$, βρείτε μια ανάθεση τιμών η οποία να ικανοποιεί τουλάχιστον $g$ όρους.

(δ) ΠΥΚΝΟ ΥΠΟΓΡΑΦΗΜΑ (Dense subgraph): Με δεδομένο ένα γράφημα και δύο ακεραίους $a$ και $b$, βρείτε ένα σύνολο $a$ κορυφών του $G$ τέτοιο ώστε να υπάρχουν τουλάχιστον $b$ ακμές ανάμεσά τους.

(ε) ΑΡΑΙΟ ΥΠΟΓΡΑΦΗΜΑ (Sparse subgraph): Με δεδομένο ένα γράφημα και δύο ακεραίους $a$ και $b$, βρείτε ένα σύνολο $a$ κορυφών στο $G$ τέτοιο ώστε να υπάρχουν το πολύ $b$ ακμές ανάμεσά τους.

(στ) ΚΑΛΥΨΗ ΣΥΝΟΛΟΥ (Set cover): (Αυτό το πρόβλημα γενικεύει *δύο* γνωστά **NP**-πλήρη προβλήματα.)

(ζ) ΑΞΙΟΠΙΣΤΟ ΔΙΚΤΥΟ (Reliable network): Μας δίνονται δύο πίνακες $m \times n$, ένας πίνακας αποστάσεων $d_{ij}$, και ένας πίνακας *απαιτήσεων συνδετικότητας* (connectivity requirement) $r_{ij}$, καθώς και ένας προϋπολογισμός $b$· πρέπει να βρούμε ένα γράφημα $G = (\{1, 2, ..., n\}, E)$ τέτοιο, ώστε (1) το συνολικό κόστος όλων των ακμών να είναι $b$ ή μικρότερο, και (2) ανάμεσα σε οποιεσδήποτε δύο διαφορετικές κορυφές $i$ και $j$ να υπάρχουν $r_{ij}$ διαδρομές ξένες ως προς τις κορυφές. (*Υπόδειξη:* Υποθέστε ότι όλες οι αποστάσεις $d_{ij}$ είναι 1 ή 2, $b = n$, και όλες οι τιμές $r_{ij}$ είναι 2. Ποιο πολύ γνωστό **NP**-πλήρες πρόβλημα είναι αυτό;)

8.11. Υπάρχουν πολλές παραλλαγές του προβλήματος του Rudrata, ανάλογα με το αν το γράφημα είναι προσανατολισμένο ή όχι, και αν αναζητείται κύκλος ή διαδρομή. Να αναγάγετε το πρόβλημα ΠΡΟΣΑΝΑΤΟΛΙΣΜΕΝΗ ΔΙΑΔΡΟΜΗ RUDRATA (Directed rudrata path) σε καθένα από τα ακόλουθα.



(α) Το πρόβλημα (μη προσανατολισμένη) ΔΙΑΔΡΟΜΗ RUDRATA.

(β) Το πρόβλημα (μη προσανατολισμένη) ΔΙΑΔΡΟΜΗ RUDRATA ($s$, $t$), το οποίο είναι σαν το πρόβλημα ΔΙΑΔΡΟΜΗ RUDRATA με τη διαφορά ότι τα άκρα της διαδρομής καθορίζονται στην είσοδο.

8.12. Το πρόβλημα $k$-ΕΠΙΚΑΛΥΠΤΙΚΟ ΔΕΝΔΡΟ ($k$-Spanning tree) είναι το ακόλουθο.

*Είσοδος:* Ένα μη προσανατολισμένο γράφημα $G = (V, E)$
*Έξοδος:* Ένα επικαλυπτικό δένδρο του $G$ στο οποίο κάθε κόμβος έχει βαθμό $\leq k$, αν υπάρχει τέτοιο δένδρο.

Δείξτε ότι για οποιοδήποτε $k \geq 2$:

(α) Το $k$-ΕΠΙΚΑΛΥΠΤΙΚΟ ΔΕΝΔΡΟ είναι ένα πρόβλημα αναζήτησης.

(β) Το $k$-ΕΠΙΚΑΛΥΠΤΙΚΟ ΔΕΝΔΡΟ είναι **NP**-πλήρες (*Υπόδειξη:* Ξεκινήστε με $k = 2$ και εξετάστε τη σχέση ανάμεσα σε αυτό το πρόβλημα και τη ΔΙΑΔΡΟΜΗ RUDRATA.)

8.13. Προσδιορίστε ποια από τα ακόλουθα προβλήματα είναι **NP**-πλήρη και ποια μπορούν να λυθούν σε πολυωνυμικό χρόνο. Σε κάθε πρόβλημα σας δίνεται ένα μη προσανατολισμένο γράφημα $G = (V, E)$, μαζί με:

(α) Ένα σύνολο $L \subseteq V$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο τέτοιο ώστε το σύνολο των φύλλων του να περιλαμβάνει το σύνολο $L$.

(β) Ένα σύνολο $L \subseteq V$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο τέτοιο ώστε το σύνολο των φύλλων του να είναι ακριβώς το σύνολο $L$.

(γ) Ένα σύνολο $L \subseteq V$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο τέτοιο ώστε το σύνολο των φύλλων του να περιλαμβάνεται στο σύνολο $L$.

(δ) Έναν ακέραιο $k$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο με $k$ ή λιγότερα φύλλα.

(ε) Έναν ακέραιο $k$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο με $k$ ή περισσότερα φύλλα.

(στ) Έναν ακέραιο $k$, και πρέπει να βρείτε ένα επικαλυπτικό δένδρο με ακριβώς $k$ φύλλα.

(*Υπόδειξη:* Όλες οι αποδείξεις **NP**-πληρότητας είναι με γενίκευση, εκτός από μία.)

8.14. Αποδείξτε ότι το ακόλουθο πρόβλημα είναι **NP**-πλήρες: με δεδομένο ένα μη προσανατολισμένο γράφημα $G = (V, E)$ και έναν ακέραιο $k$, επιστρέψτε μια κλίκα μεγέθους $k$ *καθώς και* ένα ανεξάρτητο σύνολο μεγέθους $k$, με την προϋπόθεση ότι υπάρχουν.

8.15. Δείξτε ότι το ακόλουθο πρόβλημα είναι **NP**-πλήρες.

ΜΕΓΙΣΤΟ ΚΟΙΝΟ ΥΠΟΓΡΑΦΗΜΑ
*Είσοδος:* Δύο γραφήματα $G_1 = (V_1, E_1)$ και $G_2 = (V_2, E_2)$· ένας προϋπολογισμός $b$.
*Έξοδος:* Δύο σύνολα κόμβων $V_1' \subseteq V_1$ και $V_2' \subseteq V_2$ των οποίων η διαγραφή αφήνει τουλάχιστον $b$ κόμβους σε κάθε γράφημα, και κάνει τα δύο γραφήματα όμοια.

8.16. Νοιώθουμε την ανάγκη να πειραματιστούμε και θέλουμε να δημιουργήσουμε ένα νέο πιάτο. Υπάρχουν διάφορα συστατικά από τα οποία μπορούμε να επιλέξουμε, και θα θέλαμε να χρησιμοποιήσουμε όσο το δυνατόν περισσότερα από αυτά, αλλά κάποια συστατικά δεν ταιριάζουν με κάποια άλλα. Αν υπάρχουν $n$ πιθανά συστατικά (αριθμημένα από το 1 μέχρι το $n$), γράφουμε έναν πίνακα $n \times n$ ο οποίος δίνει τη *δυσαρμονία* ανάμεσα σε οποιοδήποτε ζεύγος συστατικών. Αυτή η *δυσαρμονία* είναι ένας πραγματικός αριθμός μεταξύ 0.0 και 1.0, όπου 0.0 σημαίνει «τα πάνε τέλεια μεταξύ τους» και 1.0 σημαίνει «δεν τα πάνε καθόλου καλά μεταξύ τους». Ακολουθεί ένα παράδειγμα πίνακα όπου υπάρχουν πέντε πιθανά συστατικά.

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.0 | 0.4 | 0.2 | 0.9 | 1.0 |
| 2 | 0.4 | 0.0 | 0.1 | 1.0 | 0.2 |
| 3 | 0.2 | 0.1 | 0.0 | 0.8 | 0.5 |
| 4 | 0.9 | 1.0 | 0.8 | 0.0 | 0.2 |
| 5 | 1.0 | 0.2 | 0.5 | 0.2 | 0.0 |

Σε αυτή την περίπτωση, τα συστατικά 2 και 3 τα πάνε αρκετά καλά μεταξύ τους ενώ τα συστατικά 1 και 5 έχουν άσχημη αντίθεση. Παρατηρήστε ότι αυτός ο πίνακας είναι αναγκαστικά συμμετρικός, και ότι οι διαγώνιες καταχωρίσεις είναι πάντοτε 0.0. Οποιοδήποτε σύνολο συστατικών επιφέρει μια *ποινή* η οποία είναι το *άθροισμα όλων των τιμών δυσαρμονίας ανάμεσα στα ζεύγη των συστατικών.* Για παράδειγμα, το σύνολο των συστατικών {1, 3, 5} επιφέρει μια ποινή 0.2 + 1.0 + 0.5 = 1.7. Θέλουμε αυτή η ποινή να είναι μικρή.

ΠΕΙΡΑΜΑΤΙΚΗ ΚΟΥΖΙΝΑ
*Είσοδος:* $n$, το πλήθος των συστατικών από τα οποία γίνεται η επιλογή· $D$, ο πίνακας «δυσαρμονίας» $n \times n$· κάποιος αριθμός $p \geq 0$
*Έξοδος:* Το μέγιστο πλήθος συστατικών που μπορούμε να επιλέξουμε με ποινή $\leq p$.



Δείξτε ότι αν το πρόβλημα ΠΕΙΡΑΜΑΤΙΚΗ ΚΟΥΖΙΝΑ είναι επιλύσιμο σε πολυωνυμικό χρόνο, τότε θα είναι και το 3SAT.

8.17. Δείξτε ότι για οποιοδήποτε πρόβλημα Π στο **NP**, υπάρχει ένας αλγόριθμος ο οποίος επιλύει το Π σε χρόνο $O(2^{p(n)})$, όπου $n$ είναι το μέγεθος του στιγμιοτύπου εισόδου και $p(n)$ είναι ένα πολυώνυμο (το οποίο μπορεί να εξαρτάται από το Π).

8.18. Δείξτε ότι αν **P** = **NP** τότε το κρυπτοσύστημα RSA (Ενότητα 1.4.2) μπορεί να «σπάσει» σε πολυωνυμικό χρόνο.

8.19. Ο *χαρταετός* (kite) είναι ένα γράφημα με άρτιο πλήθος κορυφών, έστω $2n$, όπου $n$ από τις κορυφές σχηματίζουν μια κλίκα και οι υπόλοιπες $n$ κορυφές είναι συνδεδεμένες σε μια «ουρά» που αποτελείται από μια διαδρομή η οποία συνδέεται με μία από τις κορυφές της κλίκας. Με δεδομένο ένα γράφημα και ένα στόχο $g$, το πρόβλημα ΧΑΡΤΑΕΤΟΣ αναζητά ένα υπογράφημα που είναι χαρταετός και περιέχει $2g$ κόμβους. Αποδείξτε ότι το πρόβλημα ΧΑΡΤΑΕΤΟΣ είναι **NP**-πλήρες.

8.20. Σε ένα μη προσανατολισμένο γράφημα $G = (V, E)$, λέμε ότι το $D \subseteq V$ είναι ένα *κυρίαρχο σύνολο* (dominating set) αν κάθε $v \in V$ είτε ανήκει στο $D$, είτε γειτνιάζει με ένα τουλάχιστο μέλος του $D$. Στο πρόβλημα ΚΥΡΙΑΡΧΟ ΣΥΝΟΛΟ, η είσοδος είναι ένα γράφημα και ένας προϋπολογισμός $b$, και ο σκοπός είναι να βρεθεί στο γράφημα ένα κυρίαρχο σύνολο με μέγεθος το πολύ $b$, αν υπάρχει τέτοιο. Αποδείξτε ότι αυτό το πρόβλημα είναι **NP**-πλήρες.

8.21. *Δημιουργία ακολουθίας με υβριδισμό.* Μια πειραματική διαδικασία για την αναγνώριση μιας νέας ακολουθίας DNA την εξερευνά επανειλημμένα για να προσδιορίσει ποια $k$-μερή (υποσυμβολοσειρές μήκους $k$) περιέχει. Βασιζόμενοι σε αυτές τις υποσυμβολοσειρές, μπορούμε να ανακατασκευάσουμε την πλήρη ακολουθία.

Ας διατυπώσουμε αυτή τη διαδικασία ως συνδυαστικό πρόβλημα. Για οποιαδήποτε συμβολοσειρά $x$ (η ακολουθία DNA), έστω ότι το $\Gamma(x)$ υποδηλώνει το πολυσύνολο όλων των $k$-μερών της. Ειδικότερα, το $\Gamma(x)$ περιέχει ακριβώς $|x| - k + 1$ στοιχεία.

Τώρα το πρόβλημα της ανακατασκευής μπορεί εύκολα να διατυπωθεί: με δεδομένο ένα πολυσύνολο συμβολοσειρών, μήκους $k$ η κάθε μία, βρείτε μια συμβολοσειρά $x$ τέτοια ώστε το $\Gamma(x)$ να είναι ακριβώς αυτό το πολυσύνολο.

(α) Δείξτε ότι το πρόβλημα ανακατασκευής ανάγεται στη ΔΙΑΔΡΟΜΗ RUDRATA. (*Υπόδειξη:* Κατασκευάστε ένα προσανατολισμένο γράφημα με έναν κόμβο για κάθε $k$-μερές, και με μια ακμή από τον κόμβο $a$ στον $b$ αν οι τελευταίοι $k - 1$ χαρακτήρες του $a$ ταιριάζουν με τους πρώτους $k - 1$ χαρακτήρες του $b$.)

(β) Υπάρχουν όμως καλύτερα νέα. Δείξτε ότι το ίδιο πρόβλημα ανάγεται επίσης στη ΔΙΑΔΡΟΜΗ EULER (Euler path). (*Υπόδειξη:* Αυτή τη φορά, χρησιμοποιήστε ένα προσανατολισμένο γράφημα για κάθε $k$-μερές.)

8.22. Στο χρονοπρογραμματισμό εργασιών, είναι σύνηθες να χρησιμοποιείται μια αναπαράσταση γραφήματος με έναν κόμβο για κάθε εργασία και μια προσανατολισμένη ακμή από την εργασία $i$ στην εργασία $j$ αν η $i$ αποτελεί προϋπόθεση για την $j$. Αυτό το προσανατολισμένο γράφημα απεικονίζει τους περιορισμούς που αφορούν τις προτεραιότητες των εργασιών στο πρόβλημα του χρονοδιαγράμματος. Προφανώς, ένα χρονοδιάγραμμα είναι εφικτό, αν και μόνο αν το γράφημα είναι άκυκλο· αν δεν είναι, θα θέλαμε να προσδιορίσουμε το μικρότερο πλήθος περιορισμών που πρέπει να απορριφθούν έτσι ώστε να γίνει άκυκλο.

Με δεδομένο ένα προσανατολισμένο γράφημα $G = (V, E)$, ένα υποσύνολο $E' \subseteq E$ ονομάζεται *σύνολο τόξου ανάδρασης* (feedback arc set) αν η αφαίρεση των κορυφών $E'$ καθιστά το γράφημα $G$ άκυκλο.

ΣΥΝΟΛΟ ΤΟΞΟΥ ΑΝΑΔΡΑΣΗΣ (FAS): Με δεδομένο ένα προσανατολισμένο γράφημα $G = (V, E)$ και έναν προϋπολογισμό $b$, βρείτε ένα σύνολο τόξου ανάδρασης με ακμές $\leq b$, αν υπάρχει.

(α) Δείξτε ότι το FAS ανήκει στο **NP**.

Μπορούμε να δείξουμε ότι το FAS είναι **NP**-πλήρες με αναγωγή από την ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ. Με δεδομένο ένα στιγμιότυπο $(G, b)$ του προβλήματος ΚΑΛΥΨΗ ΚΟΡΥΦΩΝ, όπου $G$ είναι ένα μη προσανατολισμένο γράφημα και θέλουμε μια κάλυψη κορυφών μεγέθους $\leq b$, κατασκευάζουμε ένα στιγμιότυπο $(G', b)$ του FAS ως εξής. Αν το γράφημα $G = (V, E)$ έχει $n$ κορυφές $v_1, \ldots, v_n$, τότε κάνουμε το γράφημα $G' = (V', E')$ ένα προσανατολισμένο γράφημα με $2n$ κορυφές $w_1, w_1', \ldots, w_n, w_n'$, και $n + 2|E|$ (προσανατολισμένες) ακμές:

- $(w_i, w_i')$ για κάθε $i = 1, 2, \ldots, n$.
- $(w_i', w_j)$ και $(w_j', w_i)$ για κάθε $(v_i, v_j) \in E$.

(β) Δείξτε ότι αν το γράφημα $G$ περιέχει μια κάλυψη κορυφών μεγέθους $b$, τότε το γράφημα $G'$ περιέχει ένα σύνολο τόξου ανάδρασης μεγέθους $b$.

(γ) Δείξτε ότι αν το γράφημα $G'$ περιέχει ένα σύνολο τόξου ανάδρασης μεγέθους $b$, τότε το γράφημα $G$ περιέχει μια κάλυψη κορυφών μεγέθους (το πολύ) $b$. (*Υπόδειξη:* Με δεδομένο ένα σύνολο τόξου ανάδρασης μεγέθους $b$ στο γράφημα $G'$, μπορεί να χρειαστεί να το τροποποιήσετε πρώτα ελαφρώς για να πάρετε ένα άλλο το οποίο θα έχει πιο εύχρηστη μορφή, αλλά είναι του ίδιου μεγέθους ή μικρότερο. Κατόπιν, αποδείξτε ότι το γράφημα $G$ πρέπει να περιέχει μια κάλυψη κορυφών με ίδιο μέγεθος όπως το τροποποιημένο σύνολο τόξου ανάδρασης.)



8.23. Στο πρόβλημα ΔΙΑΔΡΟΜΕΣ ΞΕΝΩΝ ΚΟΜΒΩΝ (Node-disjoint paths) η είσοδος είναι ένα μη προσανατολισμένο γράφημα στο οποίο κάποιες κορυφές έχουν επισημανθεί ειδικά: ένα ορισμένο πλήθος «αφετηριών» $s_1, s_2, \ldots, s_k$ και ένα ίσο πλήθος «προορισμών» $t_1, t_2, \ldots, t_k$. Ο στόχος είναι να βρεθούν $k$ διαδρομές ξένες ως προς τους κόμβους (δηλαδή διαδρομές οι οποίες δεν έχουν κοινούς κόμβους) όπου η $i$-οστή διαδρομή πηγαίνει από τον κόμβο $s_i$ στον κόμβο $t_i$. Δείξτε ότι αυτό το πρόβλημα είναι **NP**-πλήρες.

Παρουσιάζουμε μια ακολουθία από προοδευτικά ισχυρότερες υποδείξεις.

i) Αναγωγή από το 3SAT.

ii) Για έναν τύπο 3SAT με $m$ όρους και $n$ μεταβλητές, χρησιμοποιήστε $k = m + n$ αφετηρίες και προορισμούς. Εισαγάγετε ένα ζεύγος αφετηρίας/προορισμού $(s_x, t_x)$ για κάθε μεταβλητή $x$, και ένα ζεύγος αφετηρίας/προορισμού για κάθε όρο $c$.

iii) Για κάθε όρο 3SAT, εισαγάγετε 6 νέες ενδιάμεσες κορυφές, μία για κάθε στοιχείο το οποίο υπάρχει στο συγκεκριμένο όρο και μία για το συμπλήρωμά του.

iv) Παρατηρήστε ότι αν η διαδρομή από τον κόμβο $s_c$ στον κόμβο $t_c$ διέρχεται μέσω κάποιας ενδιάμεσης κορυφής η οποία αναπαριστάνει, για παράδειγμα, μια παρουσία της μεταβλητής $x$, τότε καμία άλλη διαδρομή δεν μπορεί να διέρχεται μέσω αυτής της κορυφής. Από ποια κορυφή θα θέλατε να αναγκαστεί να διέρχεται η άλλη διαδρομή;