ΕΝΟΤΗΤΑ 1
ΕΙΣΑΓΩΓΗ

Δεδομένα

Σύνολο από πληροφορίες που πρέπει να αποθηκευτούν σε έναν υπολογιστή.

Τύπος Δεδομένων

Συναντάται ως:

- *Τύπος Δεδομένων του υλικού*

  Π.χ. αριθμοί σταθερής υποδιαστολής, προσημασμένοι αριθμοί σταθερής υποδιαστολής, αριθμοί κινητής υποδιαστολής, χαρακτήρες, κλπ.

- *Ιδεατός Τύπος Δεδομένων*

  Είδος δεδομένων που οι μεταβλητές μιας γλώσσας προγραμματισμού υψηλού επιπέδου μπορούν να αποθηκεύσουν και να επεξεργαστούν. Π.χ, real, int, float, double, array, κλπ.

- *Αφηρημένος Τύπος Δεδομένων*

  Τύποι δεδομένων που ``χτίζει'' ο προγραμματιστής για να διευκολύνει την επίλυση ενός προβλήματος, την σχεδίαση ενός αλγορίθμου ή την συγγραφή αποτελεσματικού και ευανάγνωστου κώδικα.

Αλγόριθμος

Αλγόριθμος είναι ένα πεπερασμένο σύνολο βημάτων/εντολών αυστηρά καθορισμένων (και εκτελέσιμων μέσα σε πεπερασμένο χρόνο), τα οποία αν ακολουθηθούν επιλύεται κάποιο πρόβλημα.

Είσοδος: Δεδομένα που δίνονται ως είσοδοι στον αλγόριθμο.

Έξοδος: ποσότητα που αποτελεί το αποτέλεσμα του αλγορίθμου.

Ευκρίνια/Αποτελεσματικότητα: Κάθε εντολή θα πρέπει να είναι απλή και ο τρόπος εκτέλεσής της να καθορίζεται χωρίς καμία αμφιβολία.

Περατότητα: Ο αλγόριθμος θα πρέπει να τερματίζει μετά από πεπερασμένα βήματα εκτέλεσης των εντολών του.

Πρόγραμμα

Υλοποίηση ενός αλγορίθμου σε κάποια γλώσσα προγραμματισμού.

Παράδειγμα

Εύρεση ενός στοιχείου Κ σε έναν ταξινομένο πίνακα.

Αλγόριθμοι (περιγραφή σε φυσική γλώσσα)

1. Ξεκινώντας από το πρώτο στοιχείο του πίνακα προσπέλασε κάθε στοιχείο του πίνακα και εξέτασε αν είναι το Κ, μέχρι είτε να βρεθεί το Κ, ή να εξεταστούν όλα τα στοιχεία του πίνακα.

2. Ξεκινώντας από το πρώτο στοιχείο του πίνακα προσπέλασε κάθε στοιχείο του πίνακα και εξέτασε αν είναι το Κ, μέχρι είτε να βρεθεί το Κ, ή να βρεθεί κάποιο στοιχείο μεγαλύτερο από το Κ ή να εξεταστούν όλα τα στοιχεία του πίνακα.

3. Δυαδική Αναζήτηση

Ξεκίνησε από το μεσαίο στοιχείο του πίνακα. Αν αυτό είναι το Κ επέστρεψε. Διαφορετικά, σύγκρινε το μεσαίο στοιχείο με το Κ και αν είναι μικρότερο το Κ βρίσκεται στο δεξί μισό του πίνακα, αν όχι τότε βρίσκεται στο αριστερό μισό του πίνακα. Σε κάθε περίπτωση μόνο το ένα μισό του πίνακα πρέπει να εξεταστεί. Επανέλαβε την διαδικασία αυτή μέχρι είτε να βρεθεί το Κ ή το προς εξέταση μέγεθος του πίνακα να μηδενιστεί

Παράδειγμα

Ύψωση ενός αριθμού x σε μια ακέραια δύναμη n.

Αλγόριθμος 1 (Περιγραφή με ψευδο-κώδικα)

Συνάρτηση $power_1$(x: number, n: integer): επιστρέφει number

```
integer j = 0;
number y = 1;

while (j < n)
      y = y*x;
      j = j+1;
return y;
```

Υλοποίηση του Αλγ. 1 στη γλώσσα C

```
double power_1(double x, int n) {
    int j;
    double y = 1;

    j = n;                  /*  j = 0 */
    while (j > 0 ) {        /*  while (j < n) { */
        y = y*x;
        j = j -1;           /* j = j +1 */
    }
}
```

Κριτήρια Επιλογής Αλγορίθμων

- Ταχύτητα
- Απαιτούμενος χώρος μνήμης
- Ευκολία προγραμματισμού
- Γενικότητα

Μας ενδιαφέρει κυρίως η ταχύτητα και ο χώρος μνήμης που χρησιμοποιούν. Οι δύο αυτοί παράμετροι καθορίζουν την αποδοτικότητα ενός αλγόριθμου.

Είναι η $Power_1$ ο πιο αποδοτικός αλγόριθμος για το πρόβλημα μας?

Αλγόριθμος $Power_2$: είσοδος & έξοδος ίδια με πριν

```
Number y

if (n == 1) return x
y = power2(x, ⌊n/2⌋)

if n is odd
        return x*y*y
else
        return y*y
```

Μοντέλο Εργασίας

Ο υπολογιστής στον οποίο δουλεύουμε έχει έναν επεξεργαστή και ένα μεγάλο μπλοκ μνήμης.

Κάθε θέση μνήμης έχει μια αριθμητική διεύθυνση (διεύθυνση μνήμης).

Η μνήμη είναι τυχαίας προσπέλασης, το οποίο σημαίνει πως η πρόσβαση σε οποιαδήποτε θέση μνήμης (ανεξάρτητα από τη διεύθυνση στην οποία βρίσκεται) γίνεται με την ίδια ταχύτητα.

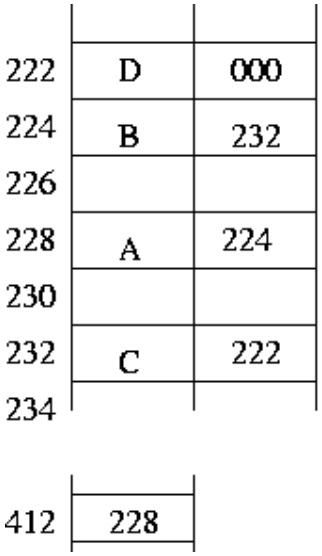
Συνεχόμενες θέσεις μνήμης μπορούν να αποθηκεύσουν τα πεδία μιας εγγραφής. Τα πεδία μιας εγγραφής μπορεί να μην είναι του ίδιου τύπου.

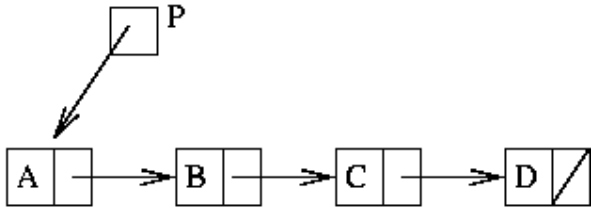
Συνεχόμενες θέσεις μνήμης μπορούν επίσης να αποθηκεύσουν ένα πίνακα (δηλαδή ένα σύνολο στοιχείων του ίδιου τύπου).

Μια μεταβλητή στην οποία είναι αποθηκευμένη μια διεύθυνση μνήμης ονομάζεται μεταβλητή δείκτη (ή δείκτης).

Γιατί οι δείκτες είναι πολύ χρήσιμοι;

Απλά Συνδεδεμένη Λίστα





Γιατί οι λίστες είναι πολύ χρήσιμες;

Σύγκριση με Πίνακες

Θετικά

- Εισαγωγή/διαγραφή νέων στοιχείων γίνεται πολύ εύκολα
- Ο συνολικός αριθμός στοιχείων δεν χρειάζεται να είναι γνωστός εξ αρχής

Αρνητικά

- Απαιτούν περισσότερη μνήμη (λόγω του πεδίου Next).
- Ανάκτηση στοιχείου για το οποίο είναι γνωστή η θέση του στη δομή (π.χ., ανάκτηση του 5ου στοιχείου) δεν μπορεί να γίνει σε σταθερό χρόνο.

Αφηρημένος Τύπος Δεδομένων

Αποτελείται από δύο μέρη:

- Ένα ή περισσότερα πεδία (σύνολα αντικειμένων, classes of mathematical objects)
- Ένα σύνολο λειτουργιών (mathematical operations) στα στοιχεία αυτών των πεδίων.

Παράδειγμα (εύρεση στοιχείου σε πίνακα)

- Τα δεδομένα μας είναι κάποιου τύπου, έστω key, και υπάρχει μια γραμμική διάταξη ανάμεσά τους:

  $\forall$ *u,v τύπου key, είτε* $u \prec v$, *ή* $v \prec u$, *ή* $v = u$.

- Έχουμε ένα σύνολο S από στοιχεία τύπου key και θέλουμε να απαντάμε το ερώτημα: $u \in S$?

Πεδία:

- στοιχεία τύπου key (π.χ., u,v)
- πεπερασμένα σύνολα στοιχείων τύπου key (π.χ., S)

Σύνολο λειτουργιών:

- Παροχή true ή false απάντησης στην ερώτηση «$u \in S$?”, όπου: u είναι στοιχείο τύπου key, S είναι πεπερασμένο σύνολο από στοιχεία τύπου key.

Δομή Δεδομένων

Μια δομή δεδομένων υλοποιεί έναν αφηρημένο τύπο δεδομένων.

Μια δομή δεδομένων επομένως συμπεριλαμβάνει:

- ένα σύνολο αποθηκευμένων δεδομένων που υφίσταται επεξεργασία από ένα σύνολο λειτουργιών, που καλούνται από το υπόλοιπο πρόγραμμα
- μια δομή αποθήκευσης
- ένα σύνολο από ορισμούς συναρτήσεων, όπου η κάθε συνάρτηση εκτελεί μια λειτουργία στο περιεχόμενο της δομής, και
- ένα σύνολο από αλγόριθμους, έναν αλγόριθμο για κάθε συνάρτηση.

Οι βασικές λειτουργίες επί των δομών δεδομένων είναι οι ακόλουθες:

- Προσπέλαση
- Εισαγωγή
- Διαγραφή
- Αναζήτηση
- Ταξινόμηση
- Αντιγραφή
- Συγχώνευση
- Διαχωρισμός

Μαθηματικό Υπόβαθρο

- Μονοτονία Συναρτήσεων

- Ακέραια Μέρη πραγματικών αριθμών
  $\lfloor x \rfloor, \lceil x \rceil, [x]$

- Πολυώνυμα – Πολυωνυμικές Συναρτήσεις
  $f(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$

- Λογάριθμοι & Εκθέτες – Ιδιότητες

- Χρήσιμοι Συμβολισμοί:
  $\lg n = \log_2 n$
  $\ln n = \log_e n$
  $\lg^k n = (\lg n)^k$
  $\lg \lg n = \lg(\lg n)$

- Παραγοντικά
  $n!$

Μαθηματική Επαγωγή

Παράδειγμα 1 – Βασική Μέθοδος

Ορθότητα Αλγόριθμου $Power_1$

Με επαγωγή στο j.

Συμβολίζουμε με y_j την τιμή της μεταβλητής y στην αρχή της j-οστής ανακύκλωσης, j = 1, …, n+1

Αρκεί να δείξουμε πως $y_j = x^{j-1}$.

Βάση επαγωγής

j = 1. Αρχικά, $y_1 = 1 = x^0 = x^{1-1} = x^{j-1}$.

Επαγωγική Υπόθεση

Έστω ότι για κάποιο k, $1 \leq k < n+1$, $y_k = x^{k-1}$.

Επαγωγικό Βήμα

Θα δείξουμε ότι ο ισχυρισμός ισχύει για (k+1): $y_{k+1} = x^k$.

Από αλγόριθμο: $y_{k+1} = y_k * x$.

Από επαγωγική υπόθεση: $y_k = x^{k-1}$.

Άρα, $y_{k+1} = x^k$, όπως απαιτείται.

Παράδειγμα 3: Ορθότητα Προγ/τος $\mathbf{Power_1}$

Με επαγωγή στο j, j = n, …, 0.

Αρκεί να δείξω ότι $y_{n-j+1}=x^{n-j}$

Βάση Επαγωγής

j = n:

$y_{n-n+1} = y_1 = 1 = x^0 = x^{n-n}$, όπως απαιτείται.

Επαγωγική Υπόθεση

Έστω ότι για κάποιο k, $n \geq k > 0$, ισχύει ότι $y_{n-k+1} = x^{n-k}$.

Επαγωγικό Βήμα

Θα δείξουμε ότι ο ισχυρισμός ισχύει για (k-1): $y_{n-k+2} = x^{n-k+1}$.

Από αλγόριθμο: $y_{n-k+2} = y_{n-k+1} * x$.

Από επαγωγική υπόθεση: $y_{n-k+1} = x^{n-k}$.

Άρα, $y_{n-k+2} = x^{n-k+1}$, όπως απαιτείται.

Παράδειγμα 3 – Αιγυπτιακός Πολλαπλασιασμός - Ισχυρή Επαγωγή

Δίδεται η συνάρτηση:

$$m(x,y) = \begin{cases} 0, \text{ αν } & y = 0 \\ m(x+x, y/2), \text{ αν y ζυγός } \& \neq 0 \\ x + m(x, y-1), \text{ διαφορετικά} \end{cases}$$

? $m(x,y) = x*y$, $\forall$ ακέραιο x,y

Απόδειξη

Με επαγωγή στο y.

Βάση της επαγωγής

Αν $y = 0$, $m(x,y) = 0$, αλλά και $x*y = 0$, οπότε ισχύει.

Επαγωγική Υπόθεση

Φιξάρουμε μια τιμή του $y > 0$. Υποθέτουμε ότι $m(x,z) = x*z$, για κάθε z, $0 \leq z < y$.

Επαγωγικό Βήμα

Αποδεικνύουμε τον ισχυρισμό για την τιμή y: $m(x*y) = x*y$.

- y περιττός: $m(x,y) = x + m(x,y-1)$. Επαγωγική υπόθεση ($z = y-1 < y$): $m(x,y-1) = x*(y-1)$. Άρα: $m(x,y) = x + x*(y-1) = x + x*y - x = x*y$
- y άρτιος: $m(x,y) = m(x+x, y/2)$. Επαγωγική υπόθεση ($z = y/2 < y$): $m(x,y) = (x+x)*y/2 = x*y$. Άρα: $m(x,y) = x*y$.

Ανάλυση Αλγορίθμων

Εμπειρικός τρόπος μέτρησης της επίδοσης

Θετικά: απλότητα

Αρνητικά:

- Δύσκολο να προβλεφθεί η συμπεριφορά για άλλα σύνολα δεδομένων.
- Ο χρόνος επεξεργασίας εξαρτάται από το υλικό, τη γλώσσα προγραμματισμού και το μεταφραστή, αλλά και από το πόσο δεινός είναι ο προγραμματιστής.

Θεωρητικός τρόπος μέτρησης της επίδοσης

Εισάγεται μια μεταβλητή n που εκφράζει το μέγεθος του προβλήματος.

Παραδείγματα

- Στο πρόβλημα ανυψώσεως σε δύναμη το μέγεθος του προβλήματος είναι το n: η δύναμη στην οποία πρέπει να υψωθεί ο δεδομένος αριθμός.
- Σε ένα πρόβλημα ταξινόμησης ενός πίνακα, το μέγεθος του προβλήματος είναι ο αριθμός στοιχείων του πίνακα.

Ανάλυση Αλγορίθμων

Ο χρόνος επεξεργασίας και ο απαιτούμενος χώρος μνήμης εκτιμώνται με τη βοήθεια μιας συνάρτησης f(n) που εκφράζει την χρονική πολυπλοκότητα ή την πολυπλοκότητα χώρου.

Μας ενδιαφέρει κυρίως η γενική συμπεριφορά αυτής της συνάρτησης, δηλαδή η τάξη της.

Συμβολισμός Ο

Έστω f(n) και g(n) δύο συναρτήσεις. Η f(n) είναι της τάξεως O(g(n)), f(n) = O(g(n)), αν υπάρχουν θετικοί ακέραιοι c και n_0, έτσι ώστε για κάθε $n \geq n_0$ να ισχύει:

$$0 \leq f(n) \leq cg(n)$$

Παράδειγμα

Έστω $f(n) = an^2 + bn$, όπου a,b θετικές σταθερές.

? $f(n) = O(n^2)$.

Ψάχνουμε για c & n_0 τ.ω.

$$an^2 + bn \leq cn^2, \text{ για κάθε } n >= n_0$$

$0 \leq (c-a)n^2 - bn \Rightarrow$

$0 \leq n\,[(c-a)n - b] \Rightarrow$

$(c-a)n - b \geq 0$

Αν επιλέξουμε $c = a+1$ και οποιοδήποτε $n_0 \geq b$, η ανισότητα $(c-a)n - b \geq 0$ ισχύει.

Συμβολισμός Ω

Έστω f(n) και g(n) δύο συναρτήσεις. Η f(n) είναι της τάξεως Ω(g(n)), f(n) = Ω(g(n)), αν υπάρχουν θετικοί ακέραιοι c και n_0, έτσι ώστε για κάθε $n \geq n_0$ να ισχύει:

$$0 \leq cg(n) \leq f(n)$$

Παράδειγμα

Θα αποδείξω ότι f(n) = Ω(n).

$an^2 + bn \geq cn \Rightarrow$

$an^2 + (b-c)n \geq 0 \Rightarrow$

$an + b-c \geq 0.$

Αν επιλέξουμε c = b ισχύει ότι $an \geq 0$, για κάθε $n \geq 0$, άρα για c = b & $n_0 = 0$, ο ισχυρισμός αποδεικνύεται.

Συμβολισμός Θ

Έστω f(n) και g(n) δύο συναρτήσεις. Η f(n) είναι της τάξεως Θ(g(n)), f(n) = Θ(g(n)), αν f(n) = O(g(N)) και f(n) = Ω(g(n)).

Παράδειγμα

Αποδεικνύουμε πως f(n) = $\Omega(n^2)$, οπότε f(n) = $\Theta(n^2)$.

Ψάχνουμε για c & n_0 τ.ω.

$$an^2 + bn \geq cn^2, \text{ για κάθε } n >= n_0$$

$0 \geq (c-a)n^2 - bn \Rightarrow$

$0 \geq n\,[(c-a)n - b] \Rightarrow$

$(c-a)n - b \leq 0.$

Αν επιλέξουμε c = 0 ισχύει ότι -an-b ≤ 0, για κάθε n, άρα για c = 0 & n_0 = 0, ο ισχυρισμός αποδεικνύεται.

Συνήθεις Κατηγορίες Χρονικής Πολυπλοκότητας

O(1): σταθερή πολυπλοκότητα

O(logn): λογαριθμική πολυπλοκότητα

O(n): γραμμική πολυπλοκότητα

O(n logn): πολυπλοκότητα O(n log n) (συνήθης πολυπλοκότητα βέλτιστων αλγόριθμων ταξινόμησης)

O(n^2): τετραγωνική πολυπλοκότητα

O(n^3): κυβική πολυπλοκότητα

O(n^k), για κάποιο προκαθορισμένο ακέραιο k: πολυωνυμική πολυπλοκότητα

O(2^n): εκθετική πολυπλοκότητα

Ιδιότητες Ο, Ω, Θ

Ανακλαστική Ιδιότητα

$f(n) = O(f(n))$

$f(n) = \Omega(f(n))$

$f(n) = \Theta(f(n))$

Μεταβατική Ιδιότητα

$f(n) = O(g(n)) \ \& \ g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$

$f(n) = \Omega(g(n)) \ \& \ g(n) = \Omega(h(n)) \Rightarrow f(n) = \Omega(h(n))$

$f(n) = \Theta(g(n)) \ \& \ g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n))$

Συμμετρική Ιδιότητα

$f(n) = \Theta(g(n))$ αν και μόνο αν $g(n) = \Theta(f(n))$

Αντίστροφη Συμμετρική Ιδιότητα

$f(n) = O(g(n))$ αν και μόνο αν $g(n) = \Omega(f(n))$

Καλή άσκηση για το σπίτι

Αποδείξτε τις παραπάνω ιδιότητες.

```
ERROR: syntaxerror
OFFENDING COMMAND: --%ztokenexec_continue--

STACK:

false
```