Αναδρομικοί Αλγόριθμοι

Αναδρομικός αλγόριθμος (recursive algorithm)

Επιλύει ένα πρόβλημα λύνοντας ένα ή περισσότερα στιγμιότυπα του **ίδιου προβλήματος**.

Αναδρομικοί Αλγόριθμοι

Αναδρομικός αλγόριθμος (recursive algorithm)

Επιλύει ένα πρόβλημα λύνοντας ένα ή περισσότερα στιγμιότυπα του **ίδιου προβλήματος**.

Παράδειγμα: Υπολογισμός παραγοντικού

```
int factorial (int N)
{
    if (N==0) return 1;
    return N*factorial(N-1);
}
```

$$N! = 1 \cdot 2 \cdot 3 \ldots (N-1) \cdot N = N \cdot (N-1)!$$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός παραγοντικού

$$N! = 1 \cdot 2 \cdot 3 \ldots (N-1) \cdot N = N \cdot (N-1)!$$

αναδρομικό πρόγραμμα

```
int factorial (int N)
{
    if (N==0) return 1;
    return N*factorial(N-1);
}
```

μη αναδρομικό πρόγραμμα

```
t = 1;
for (int i=1; i<=N; i++)
        t *= i;
```

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός παραγοντικού

$$N! = 1 \cdot 2 \cdot 3 \ldots (N-1) \cdot N = N \cdot (N-1)!$$

αναδρομικό πρόγραμμα

```
int factorial (int N)
{
    if (N==0) return 1;
    return N*factorial(N-1);
}
```

μη αναδρομικό πρόγραμμα

```
t = 1;
for (int i=1; i<=N; i++)
        t *= i;
```

• Κάθε αναδρομικό πρόγραμμα μπορεί να μετατραπεί σε ισοδύναμο μη αναδρομικό πρόγραμμα.

• Πολλές φορές όμως η χρήση αναδρομής δίνει πιο σύντομα ή/και πιο αποδοτικά προγράμματα.

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

⇨ Αν ο d είναι κοινός διαιρέτης των x, y τότε
$x = \kappa d$ και $y = \lambda d$, άρα $x - y = (\kappa - \lambda)d$
$\Rightarrow \gcd(x, y) \leq \gcd(x - y, y)$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

- Αν ο d είναι κοινός διαιρέτης των x, y τότε
  $x = \kappa d$ και $y = \lambda d$, άρα $x - y = (\kappa - \lambda)d$
  $\Rightarrow \gcd(x, y) \leq \gcd(x - y, y)$

- Αν ο d είναι κοινός διαιρέτης των $x - y, y$ τότε
  $x - y = \kappa d$ και $y = \lambda d$, άρα $x = (\kappa + \lambda)d$
  $\Rightarrow \gcd(x, y) \geq \gcd(x - y, y)$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

Επομένως $\gcd(x, y) = \gcd(x - y, y)$

$$\Rightarrow \gcd(x, y) = \gcd(y, x \bmod y)$$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$

όπου $x \geq y$ είναι θετικοί ακέραιοι

```
int gcd (int x, int y)
{
    if (y==0) return x;
    return gcd(y, x%y);
}
```

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$

όπου $x \geq y$ είναι θετικοί ακέραιοι

```
int gcd (int x, int y)
{
    if (y==0) return x;
    return gcd(y, x%y);
}
```

Παράδειγμα

gcd(128,40)=

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$

όπου $x \geq y$ είναι θετικοί ακέραιοι

```
int gcd (int x, int y)
{
    if (y==0) return x;
    return gcd(y, x%y);
}
```

Παράδειγμα

```
gcd(128,40)=
  gcd(40,8)=
```

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$

όπου $x \geq y$ είναι θετικοί ακέραιοι

```
int gcd (int x, int y)
{
    if (y==0) return x;
    return gcd(y, x%y);
}
```

Παράδειγμα

```
gcd(128,40)=
  gcd(40,8)=
    gcd(8,0)=
          8
```

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

Ιδιότητα: Αν $x \geq y$ τότε $x \bmod y < x/2$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

Ιδιότητα: Αν $x \geq y$ τότε $x \bmod y < x/2$

Απόδειξη:

· $y \leq x/2 \Rightarrow x \bmod y < y \leq x/2$

· $y > x/2 \Rightarrow x \bmod y = x - y < x/2$

Αναδρομικοί Αλγόριθμοι

Παράδειγμα: Υπολογισμός μέγιστου κοινού διαιρέτη

ακέραιοι x, y όπου $x > y$

$\gcd(x, y)$: ο μεγαλύτερος ακέραιος που τους διαιρεί ακριβώς

Αλγόριθμος του Ευκλείδη

Βασίζεται στον κανόνα $\gcd(x, y) = \gcd(y, x \bmod y)$
όπου $x \geq y$ είναι θετικοί ακέραιοι

Ιδιότητα: Αν $x \geq y$ τότε $x \bmod y < x/2$

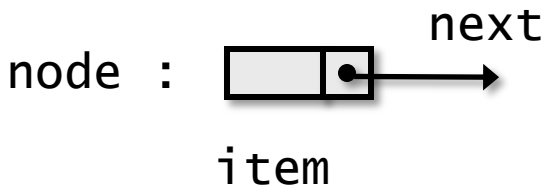
- ο αλγόριθμος του Ευκλείδη είναι **αποδοτικός** :
  απαιτεί $O(\lg x)$ επαναλήψεις

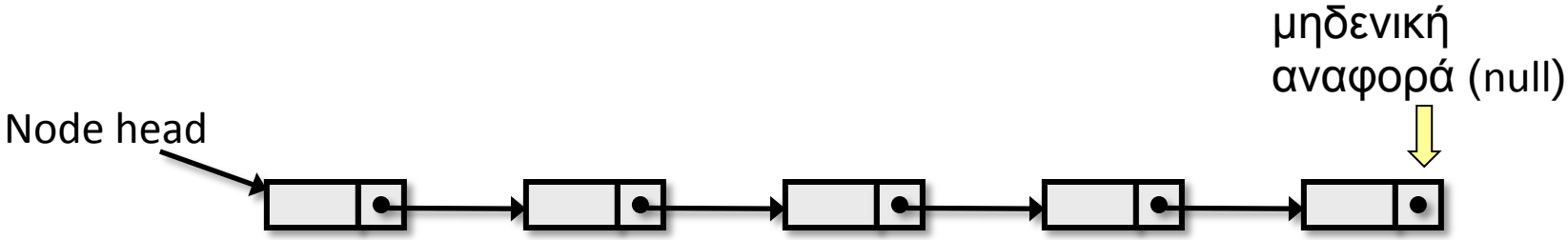
Αναδρομικοί Αλγόριθμοι

Αναδρομική συνάρτηση καταμέτρησης του πλήθους των κόμβων σε μία συνδεδεμένη λίστα

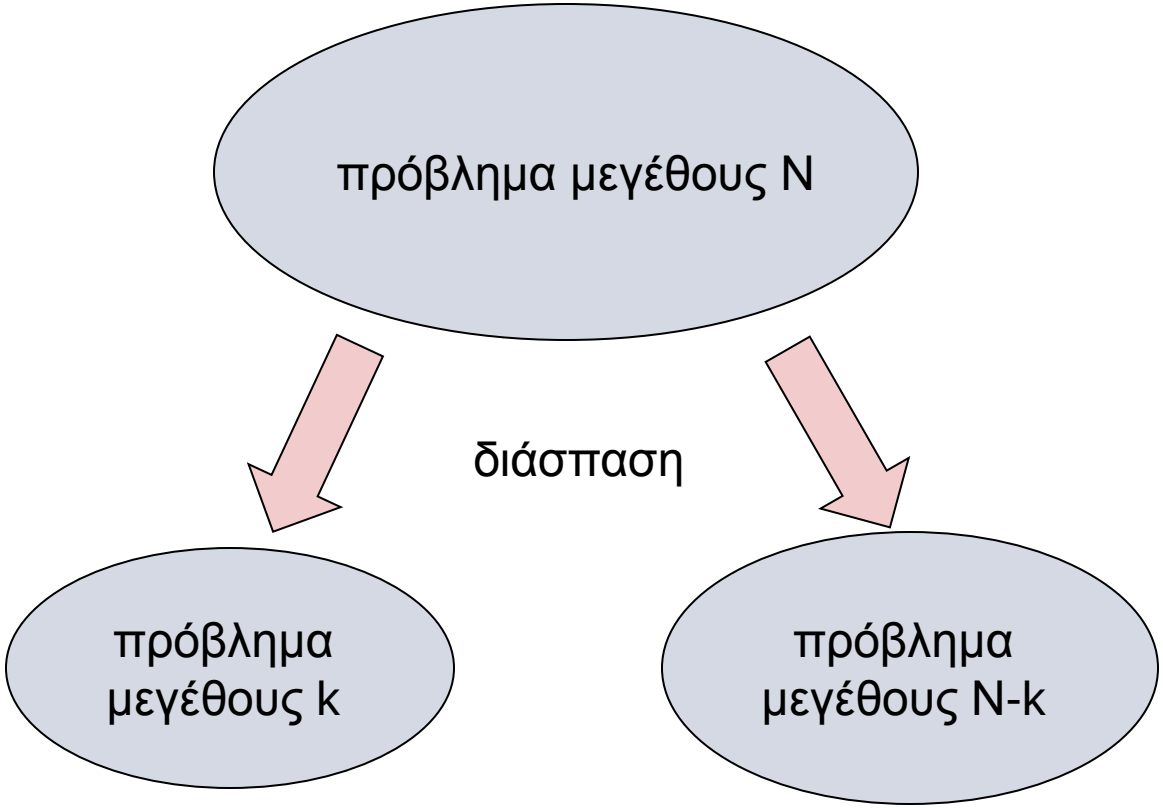
```
private class Node {
        Item item;
        Node next;
}

int count(Node x) {
        if (x == null) return 0;
        return 1 + count(x.next);
}
```

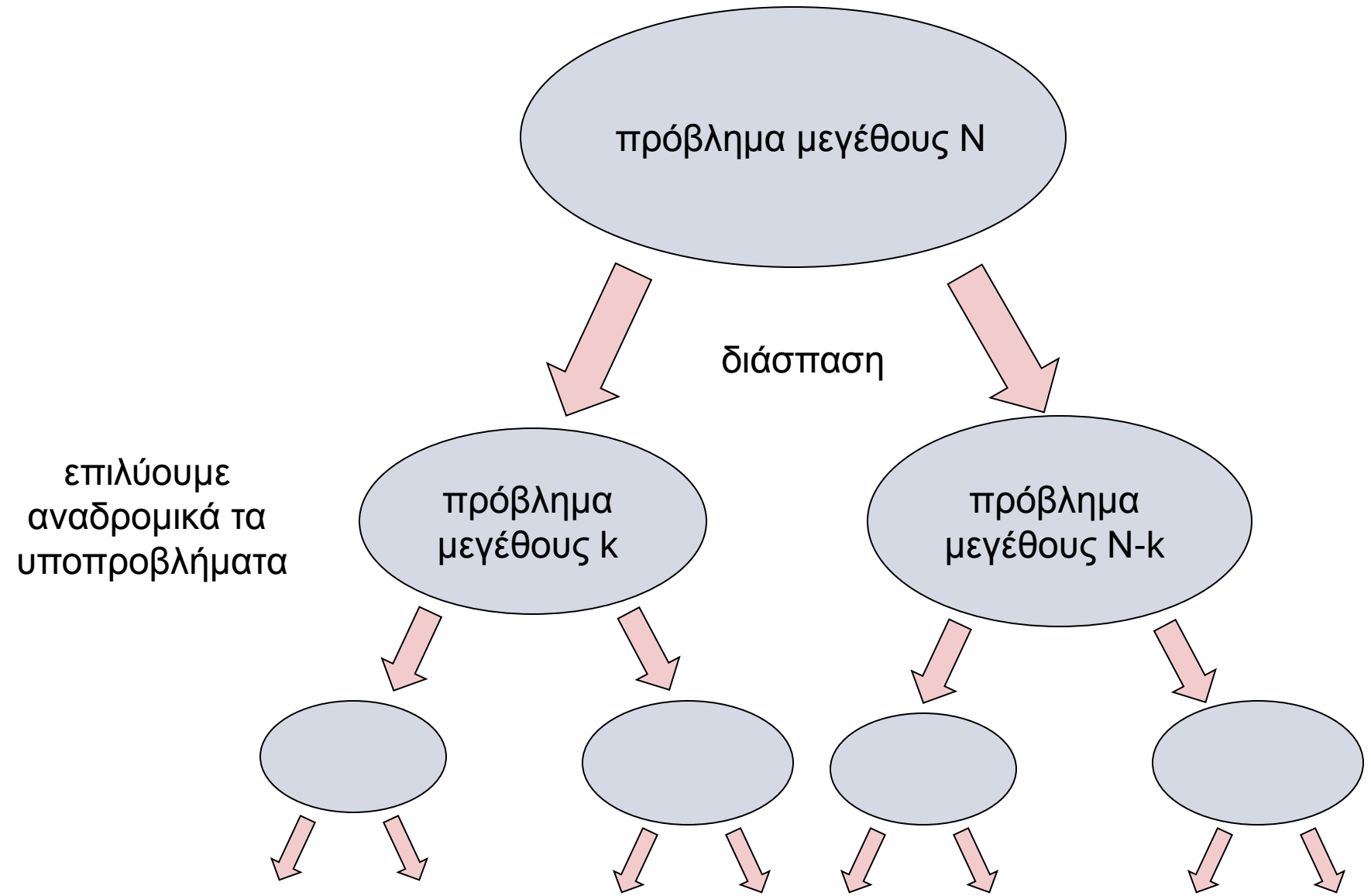




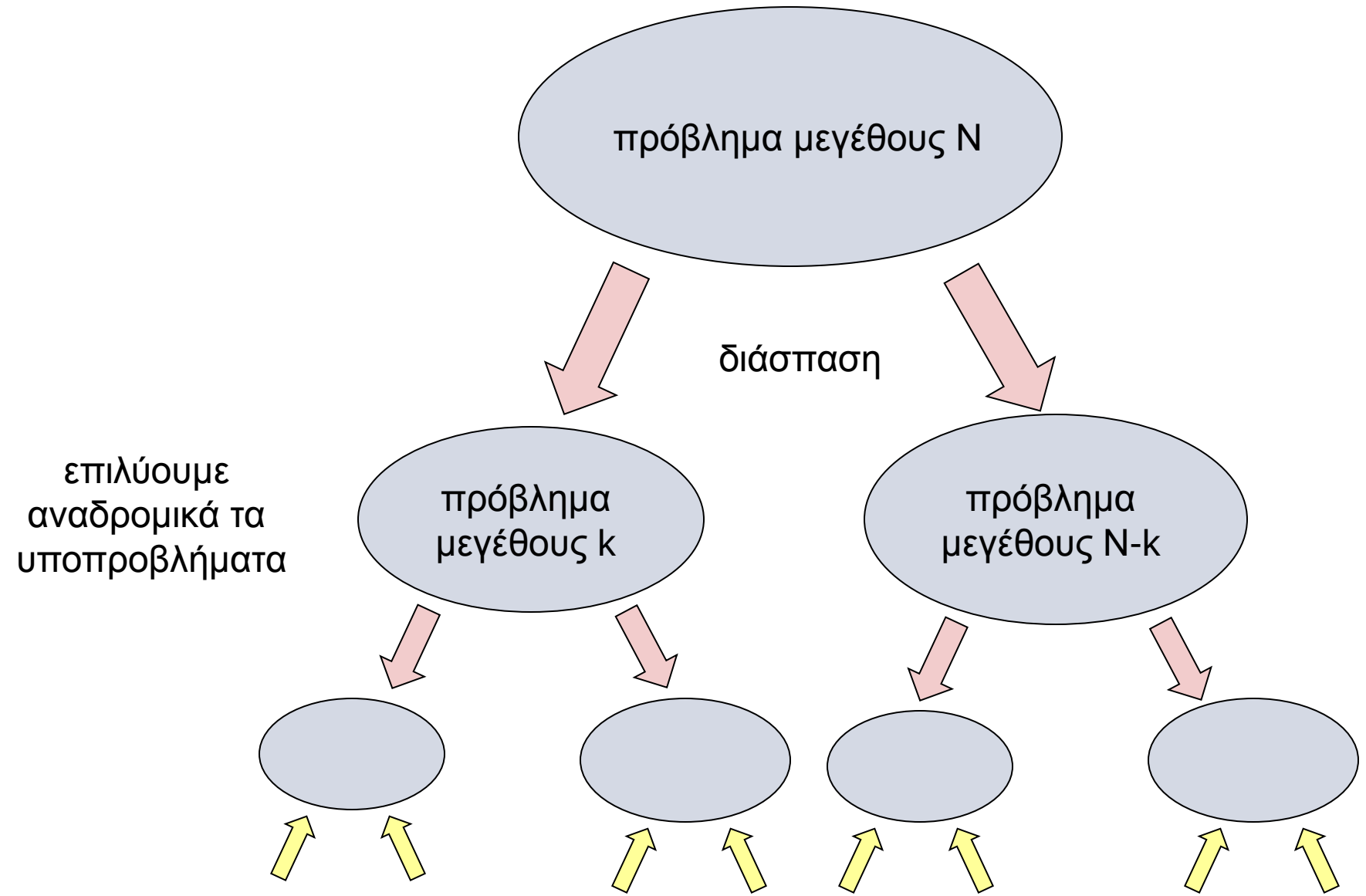
Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



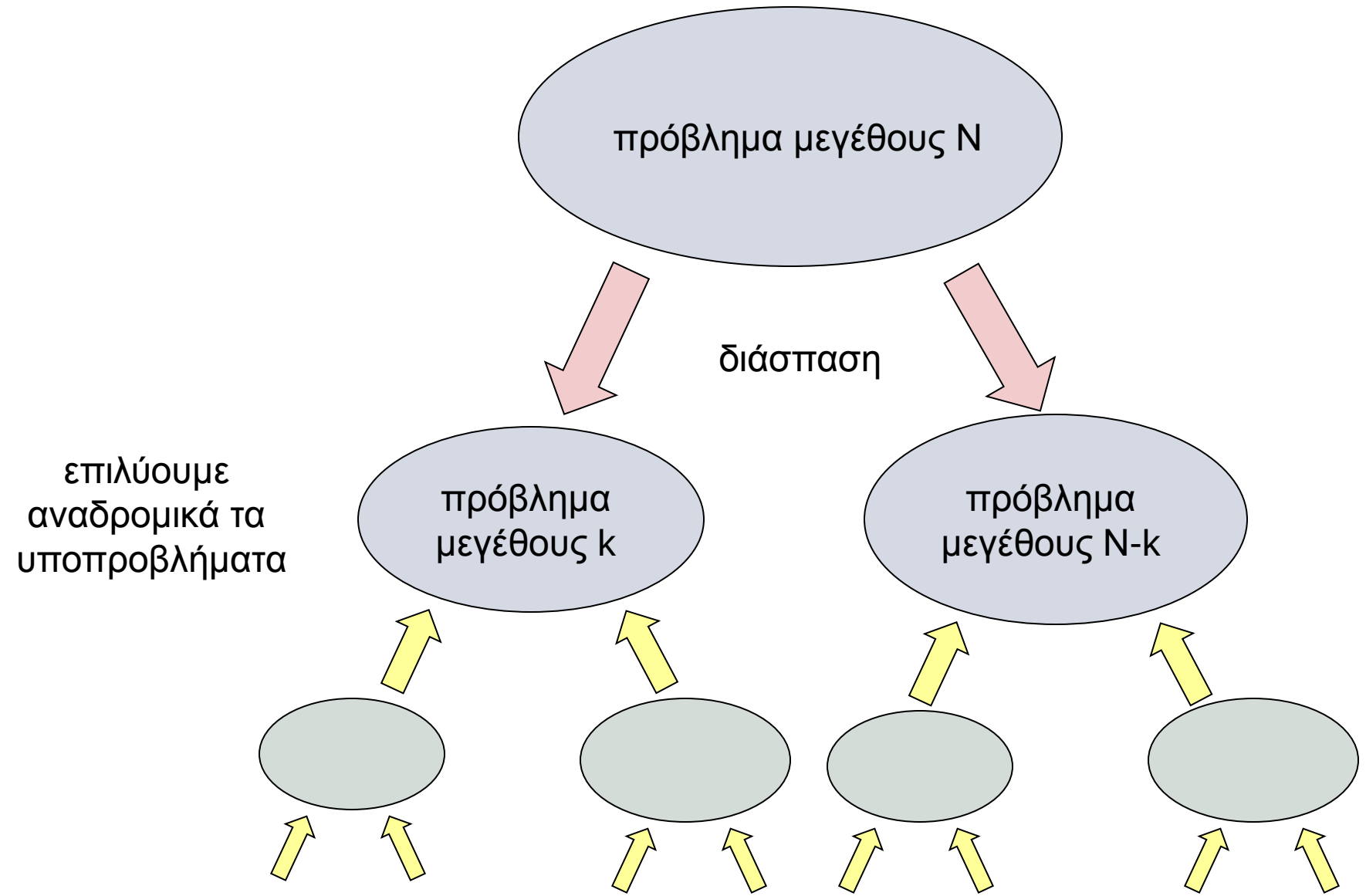
Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



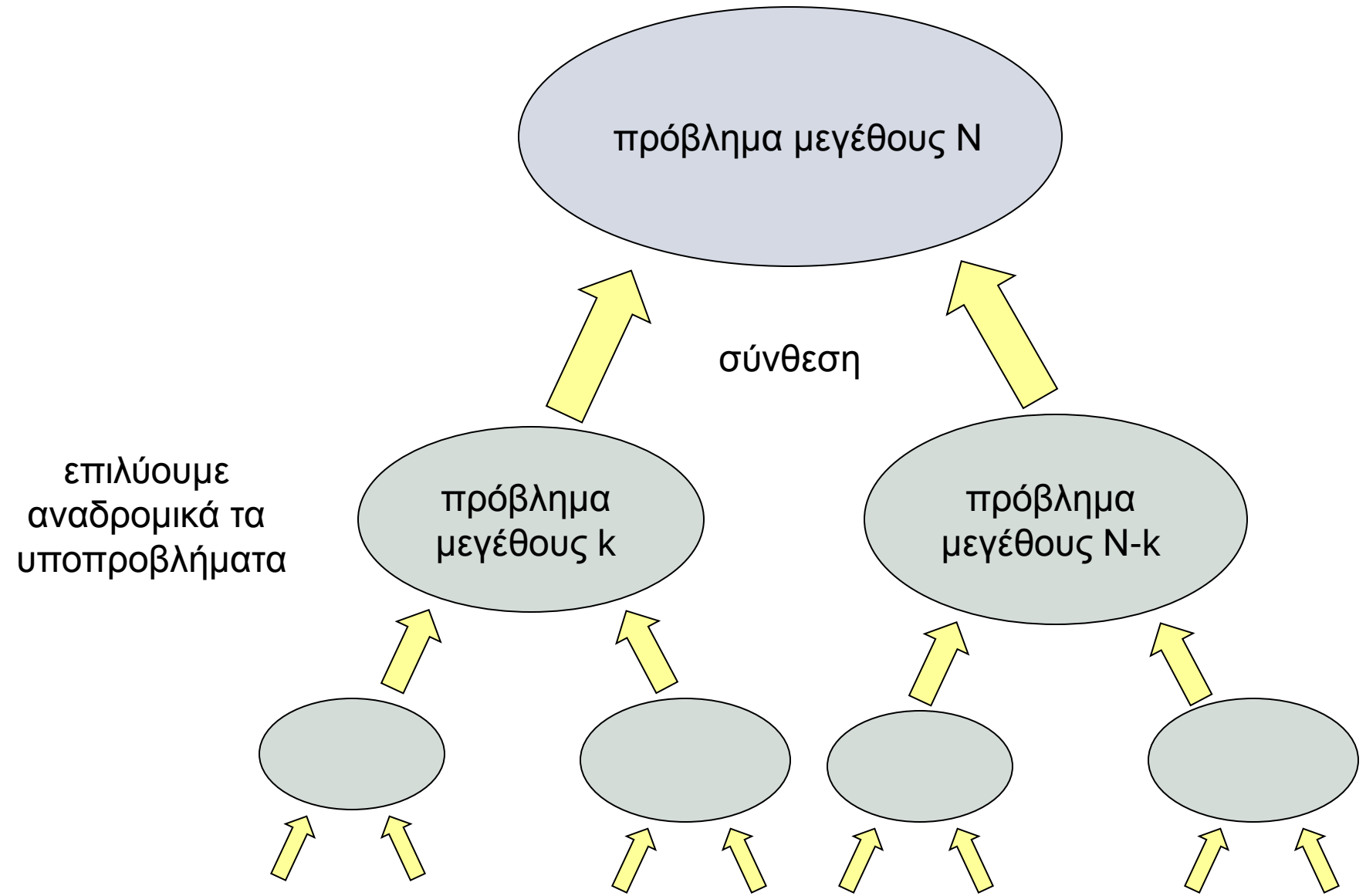
Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



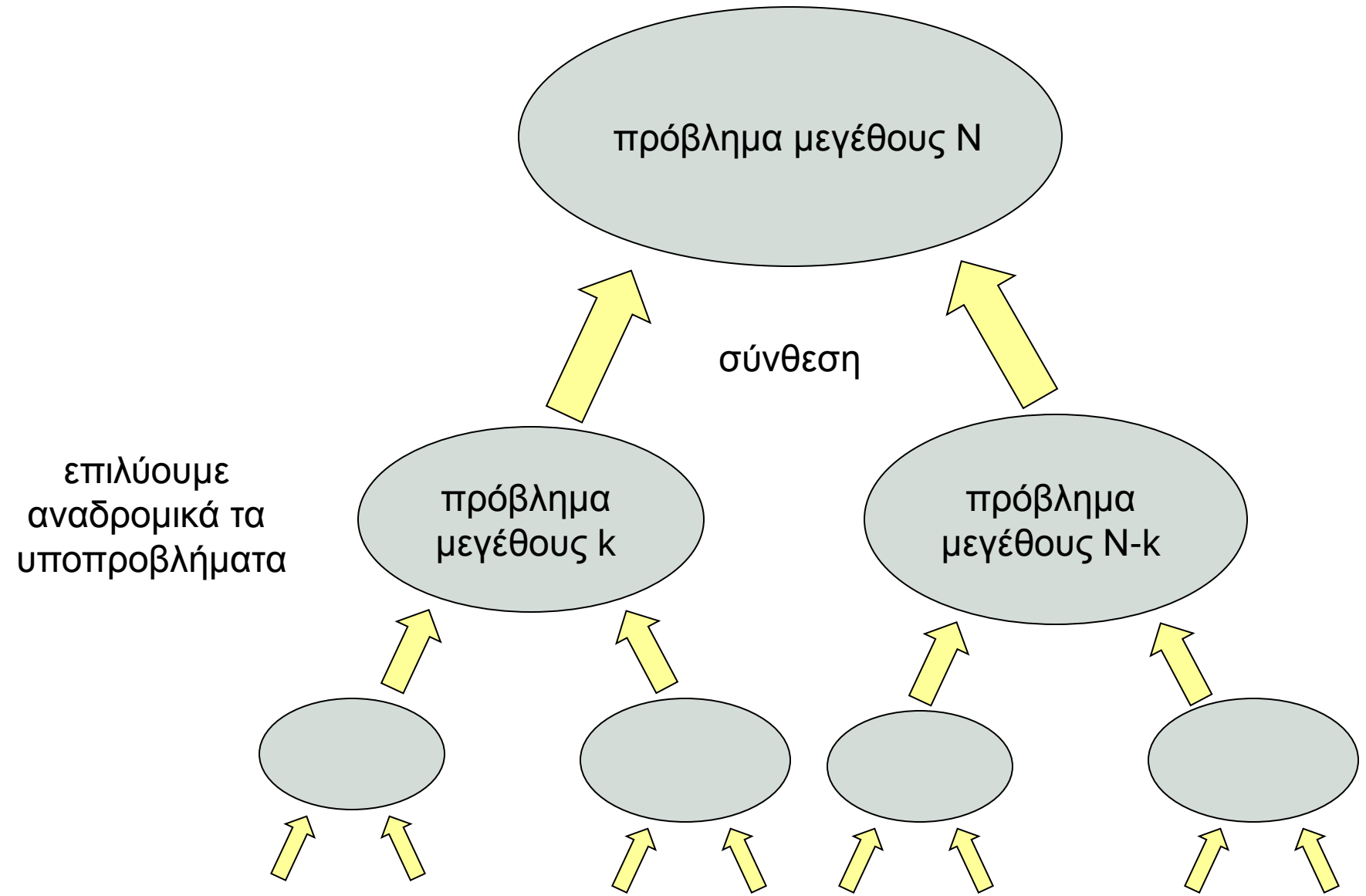
Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



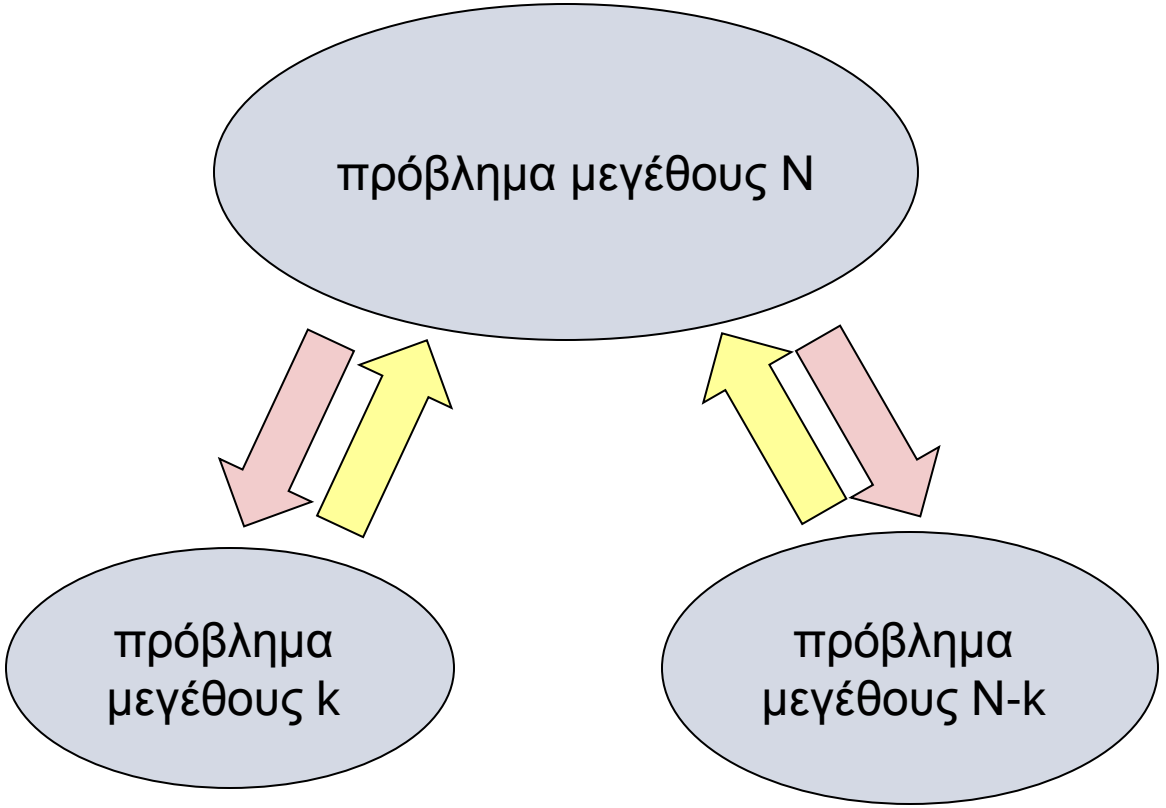
Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε



Αναδρομικοί Αλγόριθμοι: Διαίρει και βασίλευε

διάσπαση προβλήματος / σύνθεση λύσεων

επιλύουμε αναδρομικά τα υποπροβλήματα



Χρόνος εκτέλεσης : $T(N) = T(k) + T(N-k) + D(N) + C(N)$

χρόνος διάσπασης

χρόνος σύνθεσης

Αναδρομικοί Αλγόριθμοι

Απλό πρόβλημα : Εύρεση ελάχιστου στοιχείου ακολουθίας

128	233	88	9	12	45	33	28	56	8	90	102	3	245	356

μη αναδρομικό πρόγραμμα

```
t = a[0];
for (i = 1; i < N; i++)
    if (a[i] < t) t = a[i];
```

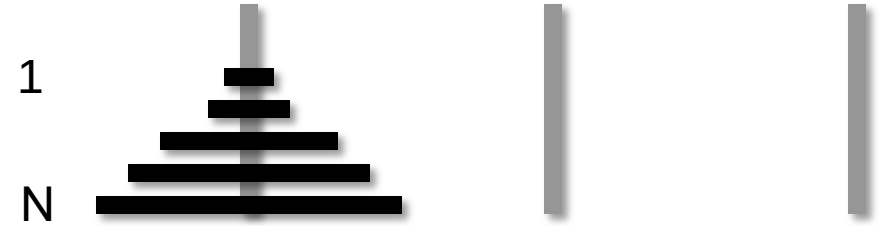
αναδρομικό πρόγραμμα
«διαίρει και βασίλευε»

```
int min(int a[], int l, int r)
{
        int u, v, m = (l+r)/2;
        if (l == r) return a[l];
        u = min(a, l, m);
        v = min(a, m+1, r);
        if (u<v) return u;
        else return v;
}
```

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Θέλουμε να μετακινήσουμε τους δίσκους κατά ένα πάσσαλο δεξιά, υπακούοντας στους παρακάτω κανόνες

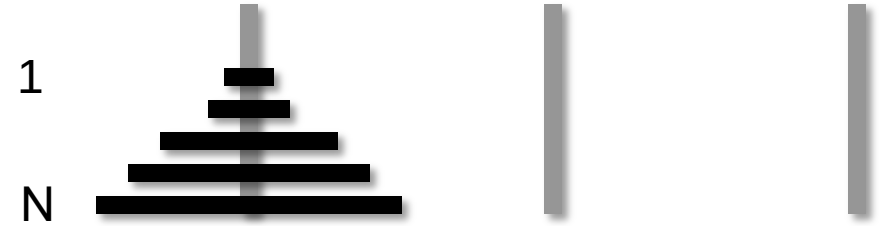
1) Μετακινούμε ένα δίσκο τη φορά

2) Ένας δίσκος δεν μπορεί να τοποθετηθεί πάνω από μικρότερο δίσκο

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Θέλουμε να μετακινήσουμε τους δίσκους κατά ένα πάσσαλο δεξιά, υπακούοντας στους παρακάτω κανόνες

1) Μετακινούμε ένα δίσκο τη φορά

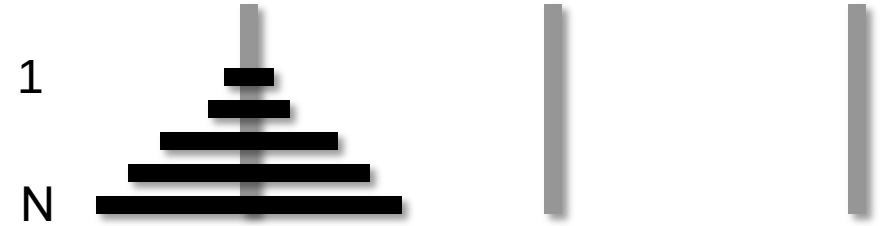
2) Ένας δίσκος δεν μπορεί να τοποθετηθεί πάνω από μικρότερο δίσκο

Η χρήση αναδρομής μας βοηθά να βρούμε μια κομψή λύση στο πρόβλημα.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Συμβολισμός

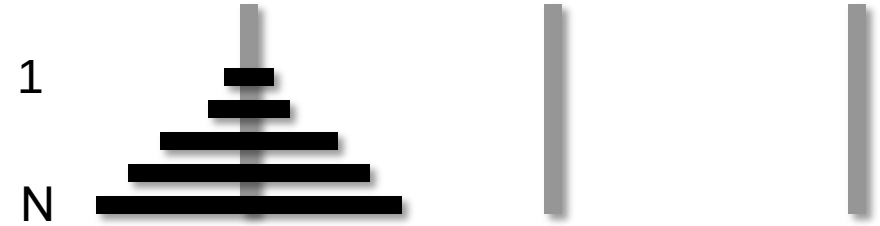
+1 : Μετακίνηση κατά 1 πάσσαλο δεξιά. Επιστρέφουμε στον αριστερότερο πάσσαλο αν ξεκινήσουμε από τον δεξιότερο.

-1 : Μετακίνηση κατά 1 πάσσαλο αριστερά. Επιστρέφουμε στον δεξιότερο πάσσαλο αν ξεκινήσουμε από τον αριστερότερο.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



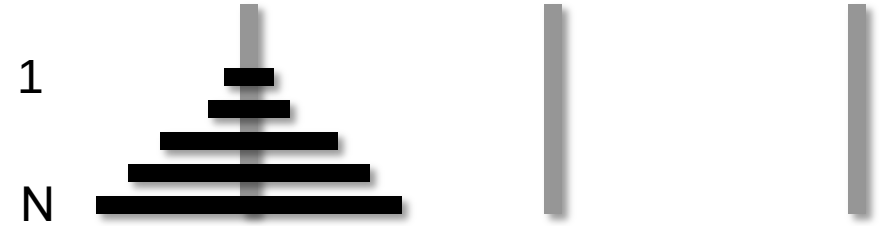
Ιδέα για αναδρομικό αλγόριθμο

Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

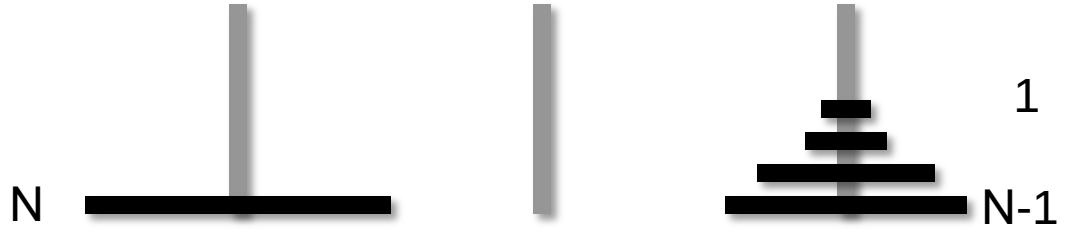
Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

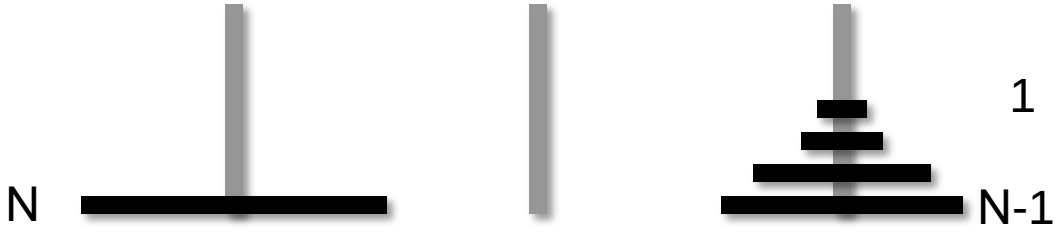
Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι

Ιδέα για αναδρομικό αλγόριθμο

Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι

Ιδέα για αναδρομικό αλγόριθμο

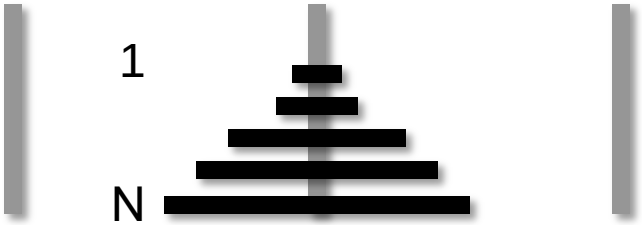
Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.
3. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

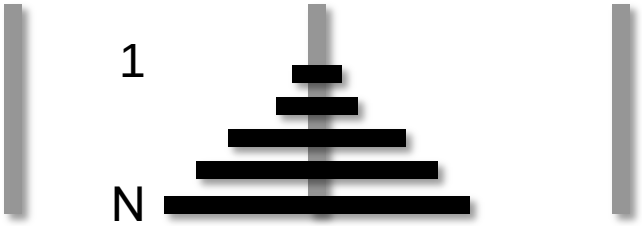
Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.
3. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

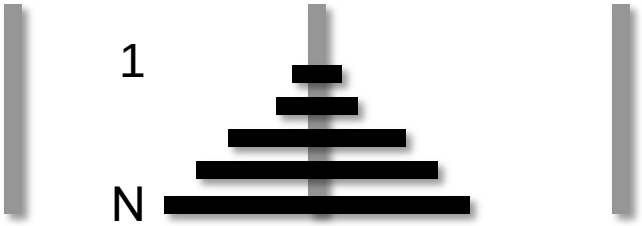
1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.
3. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

Η μετακίνηση των Ν-1 γίνεται με μαγικό τρόπο!

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και Ν δίσκοι



Ιδέα για αναδρομικό αλγόριθμο

Ας υποθέσουμε ότι ξέρουμε πώς να μετακινήσουμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο δεξιά ή αριστερά.

1. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.
2. Μετακινούμε το δίσκο Ν κατά ένα πάσσαλο δεξιά.
3. Μετακινούμε του πρώτους Ν-1 δίσκους κατά 1 πάσσαλο αριστερά.

αναδρομικά!

Η μετακίνηση των Ν-1 γίνεται με μαγικό τρόπο!

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

hanoi(3,+1)

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
```

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

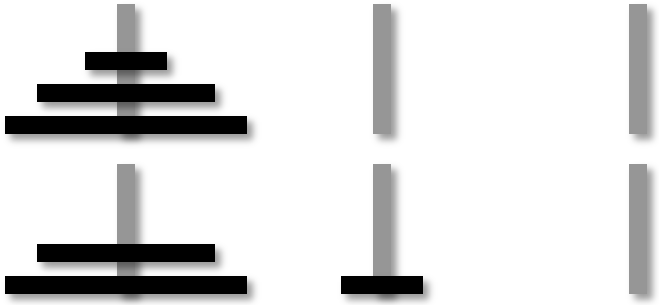
```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
```

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
```

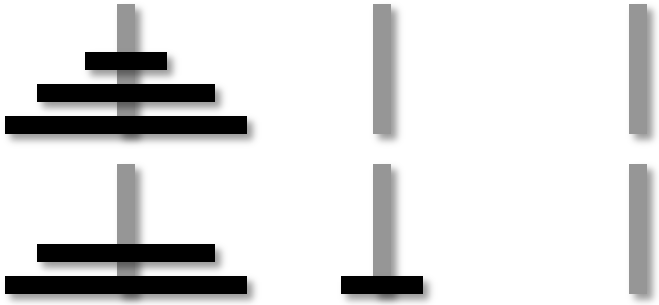
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
```



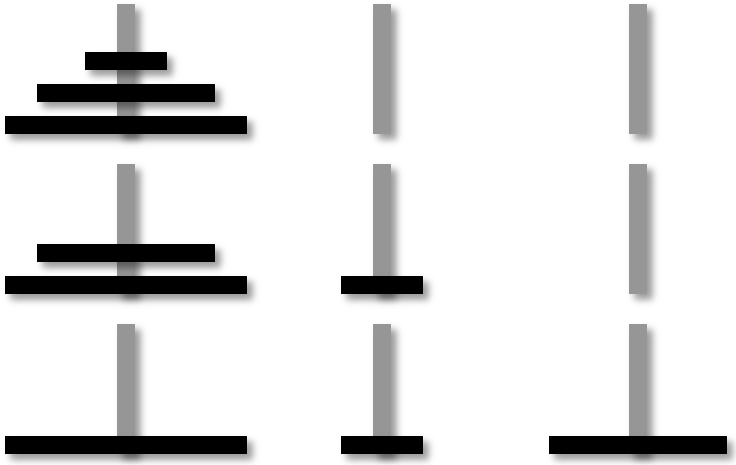
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
```



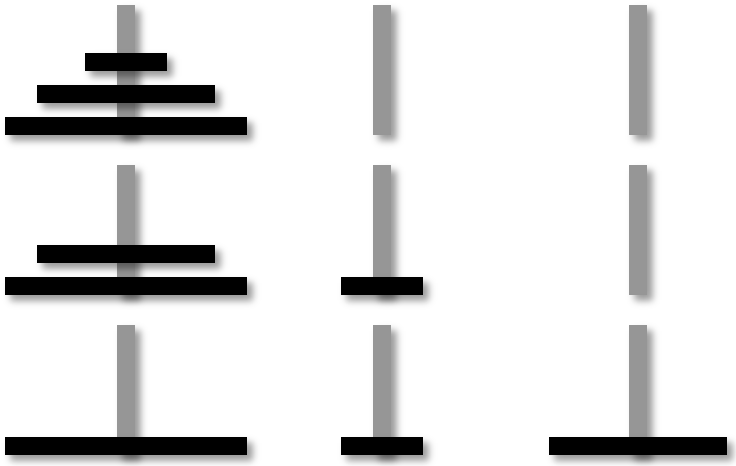
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
```



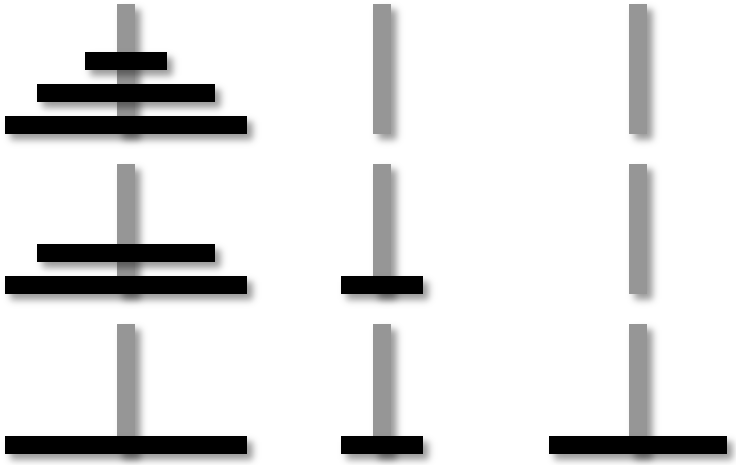
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
```



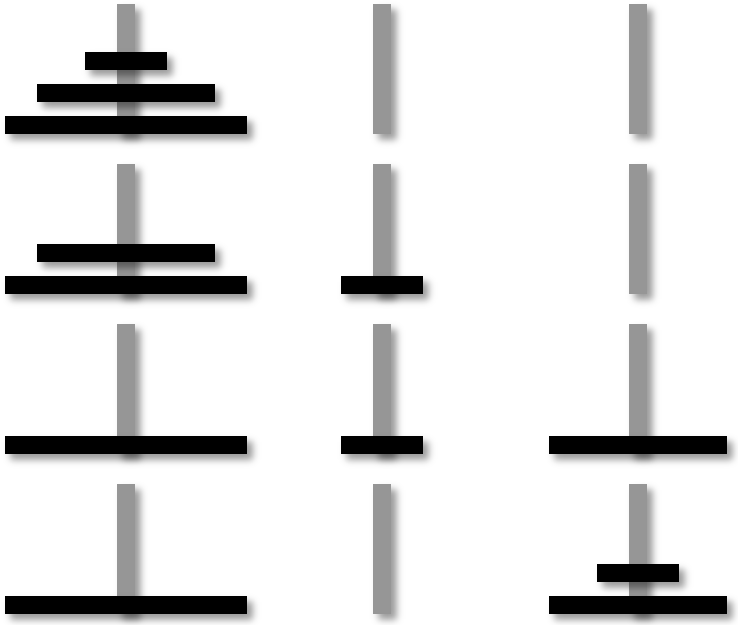
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
```



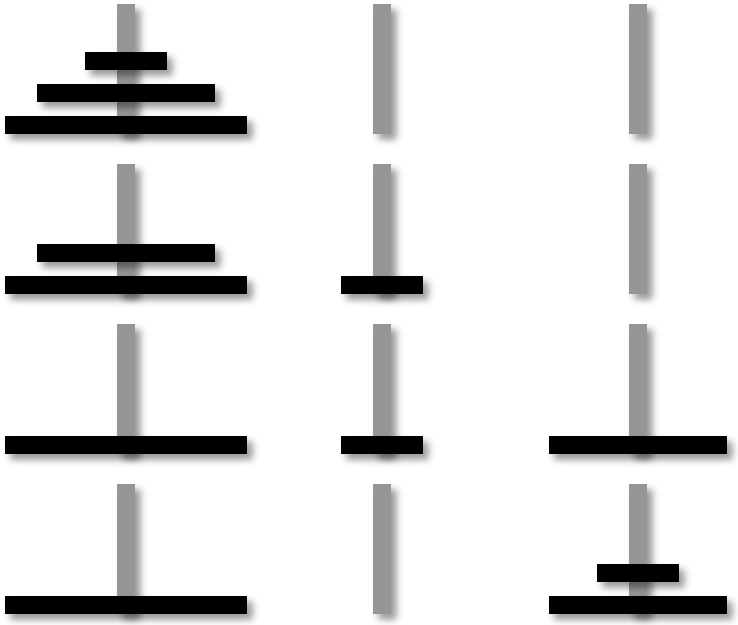
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
```

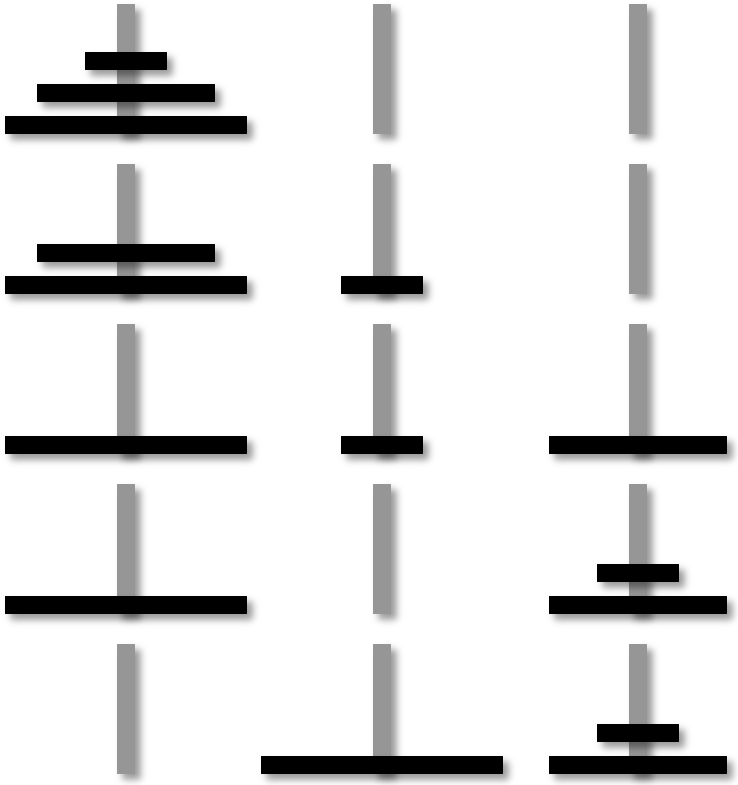


```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
```

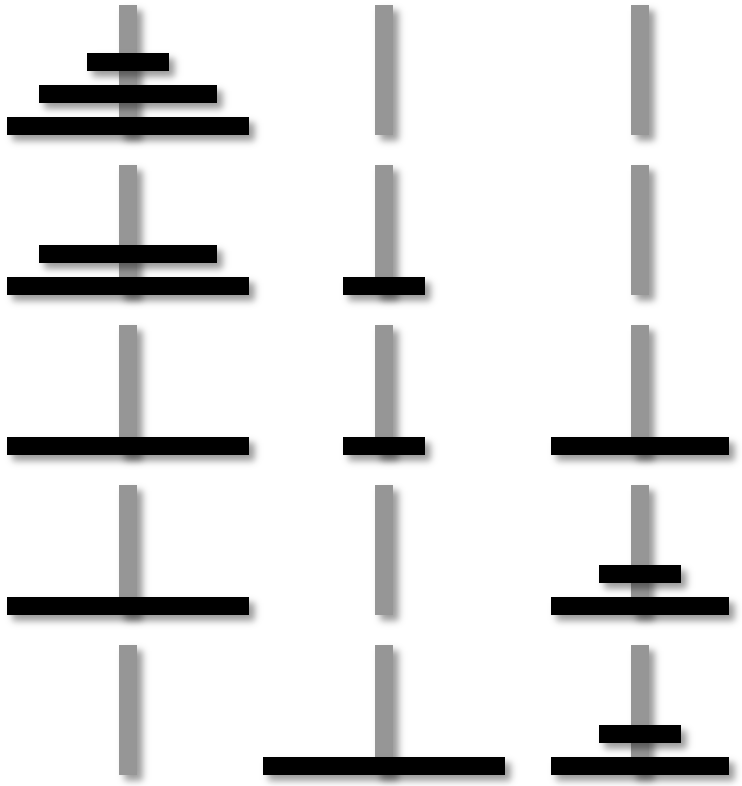


```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```



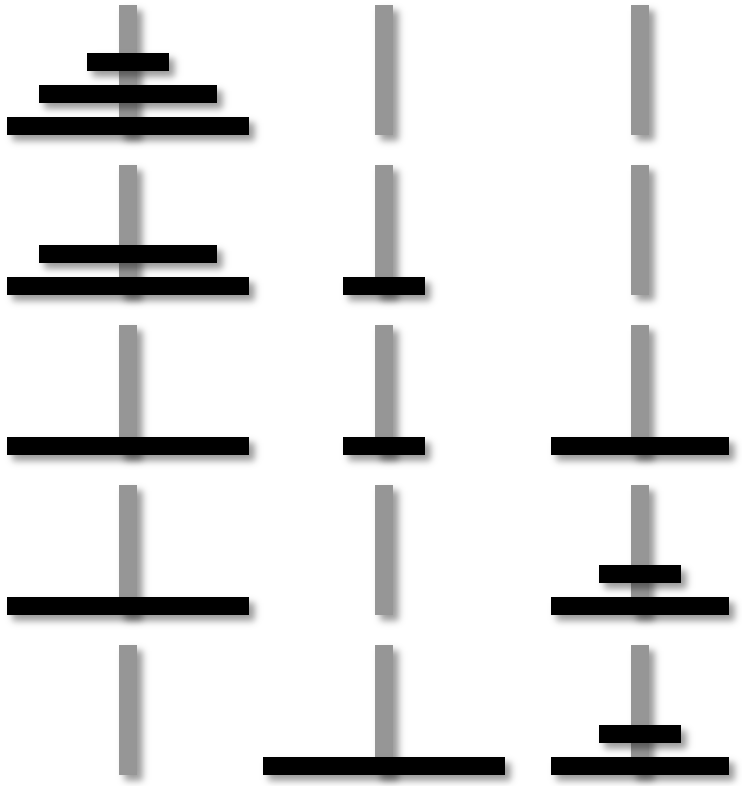
```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
```

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```



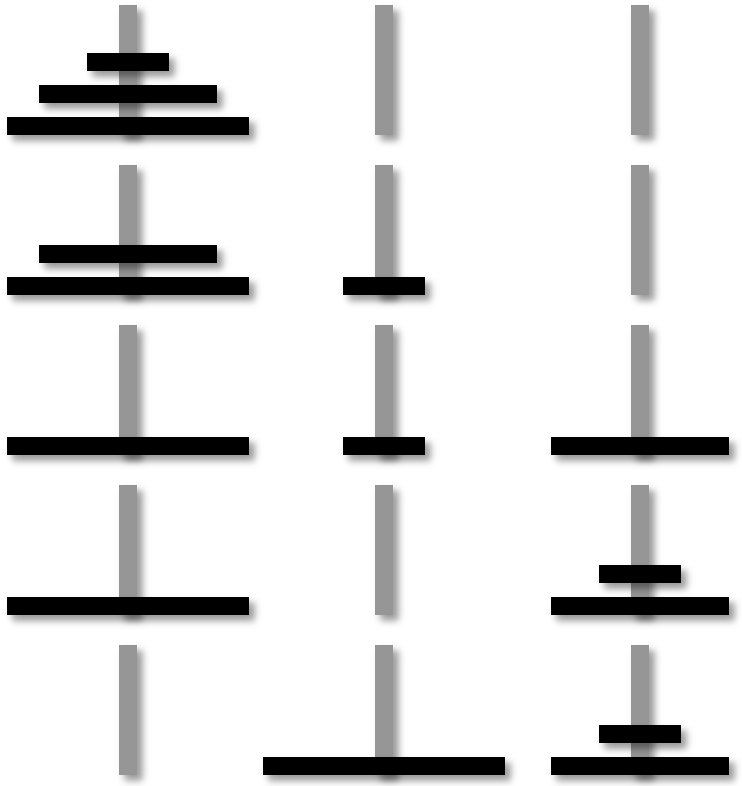
```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
  hanoi(2,-1)
```

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```



```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
```

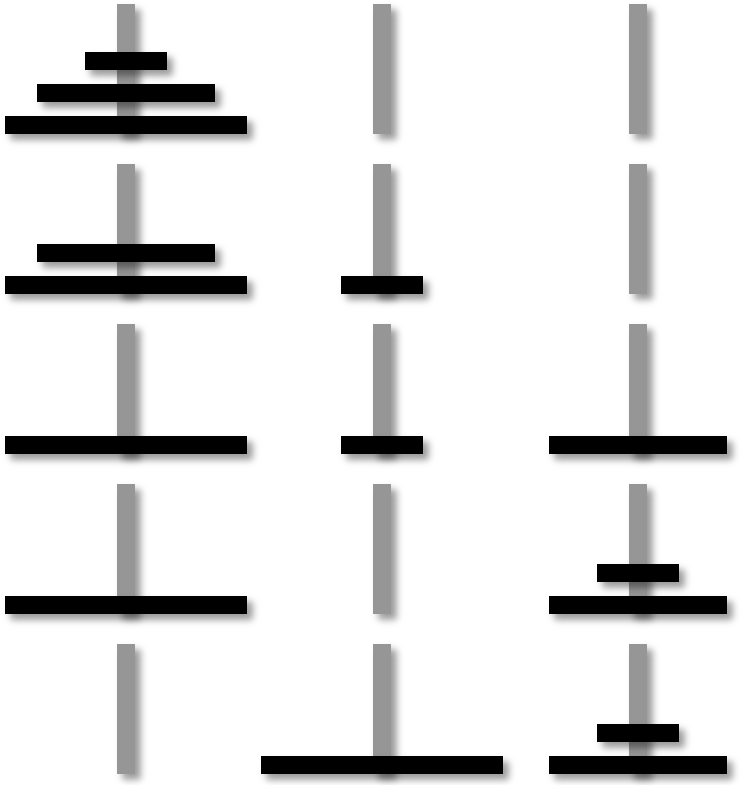
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```



```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
```

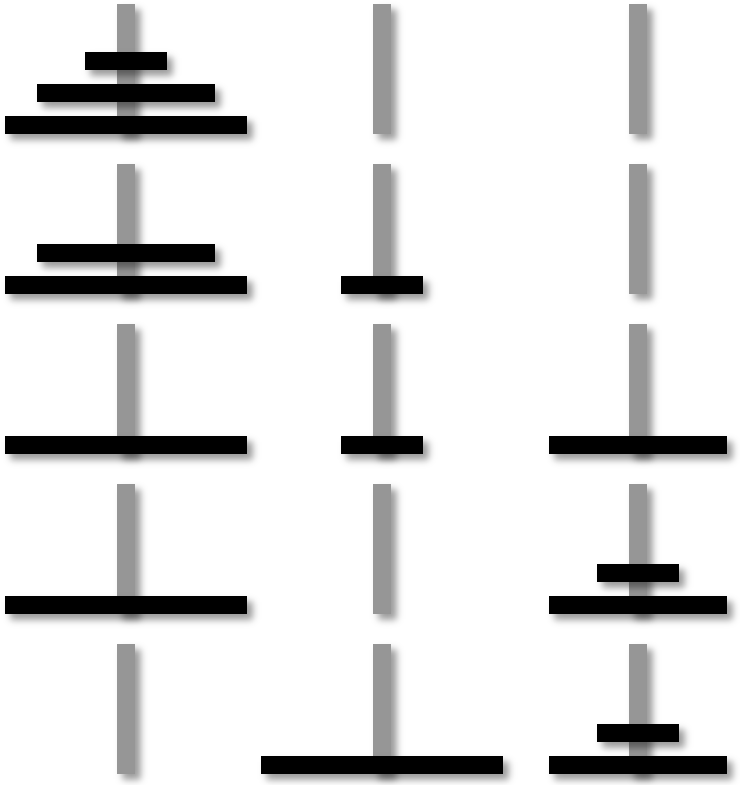
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
```



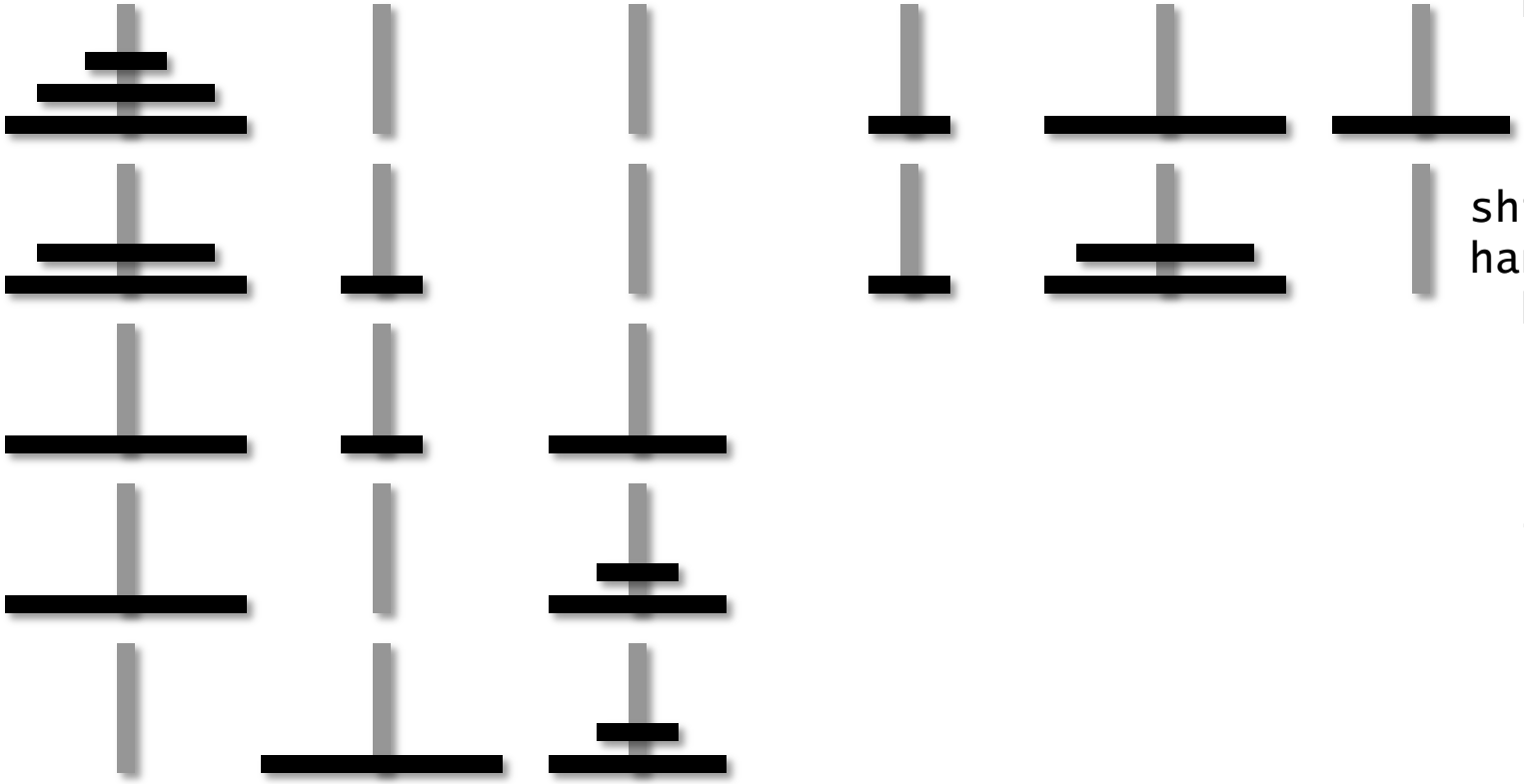
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
  shift(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
```



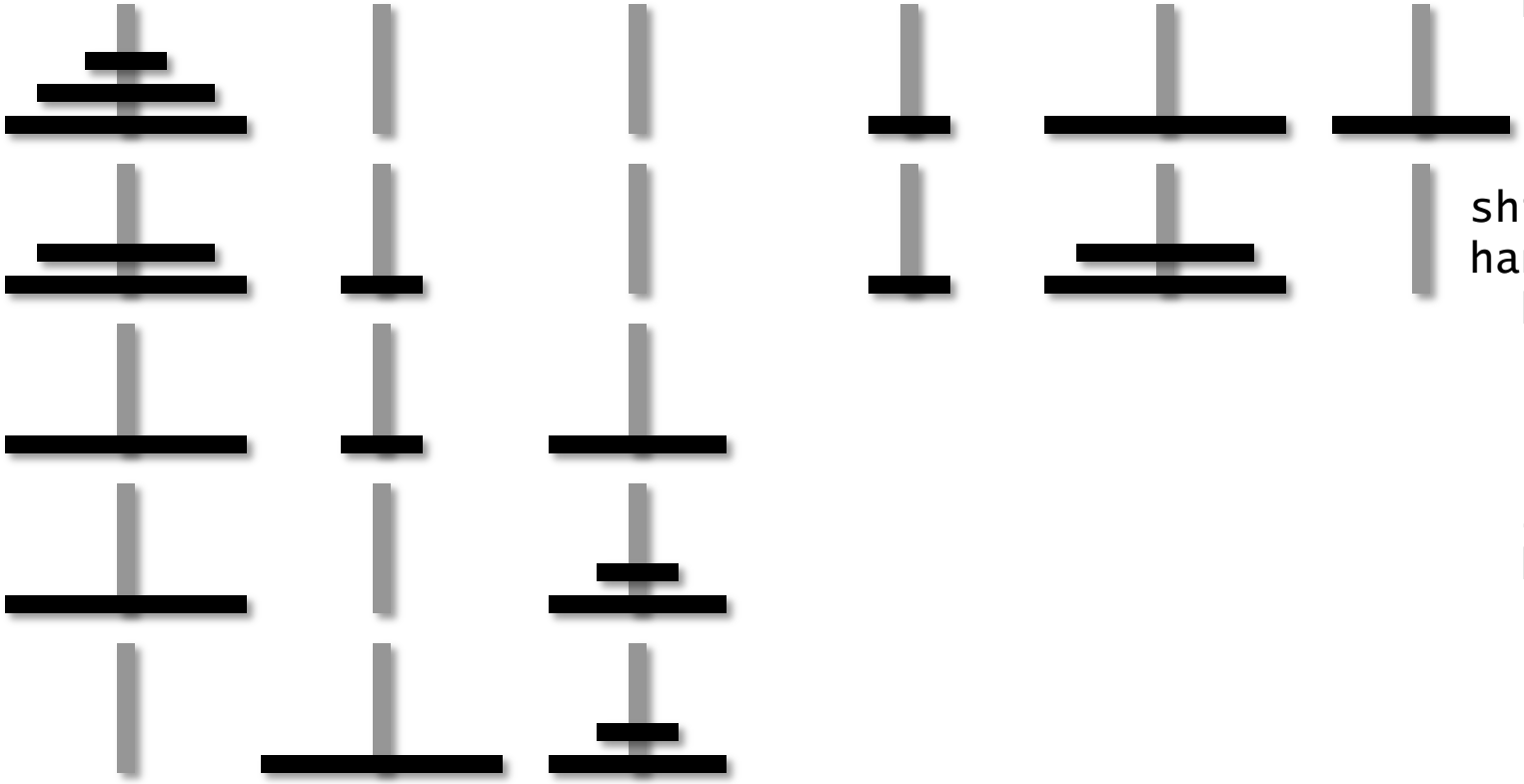
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
shift(3,+1)
hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
```

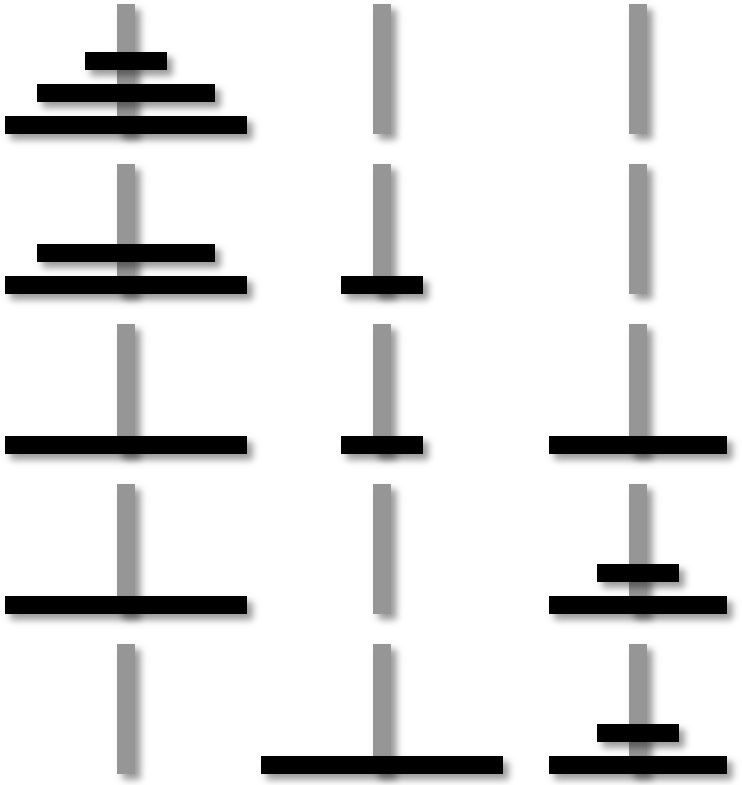


```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
shift(3,+1)
hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
```



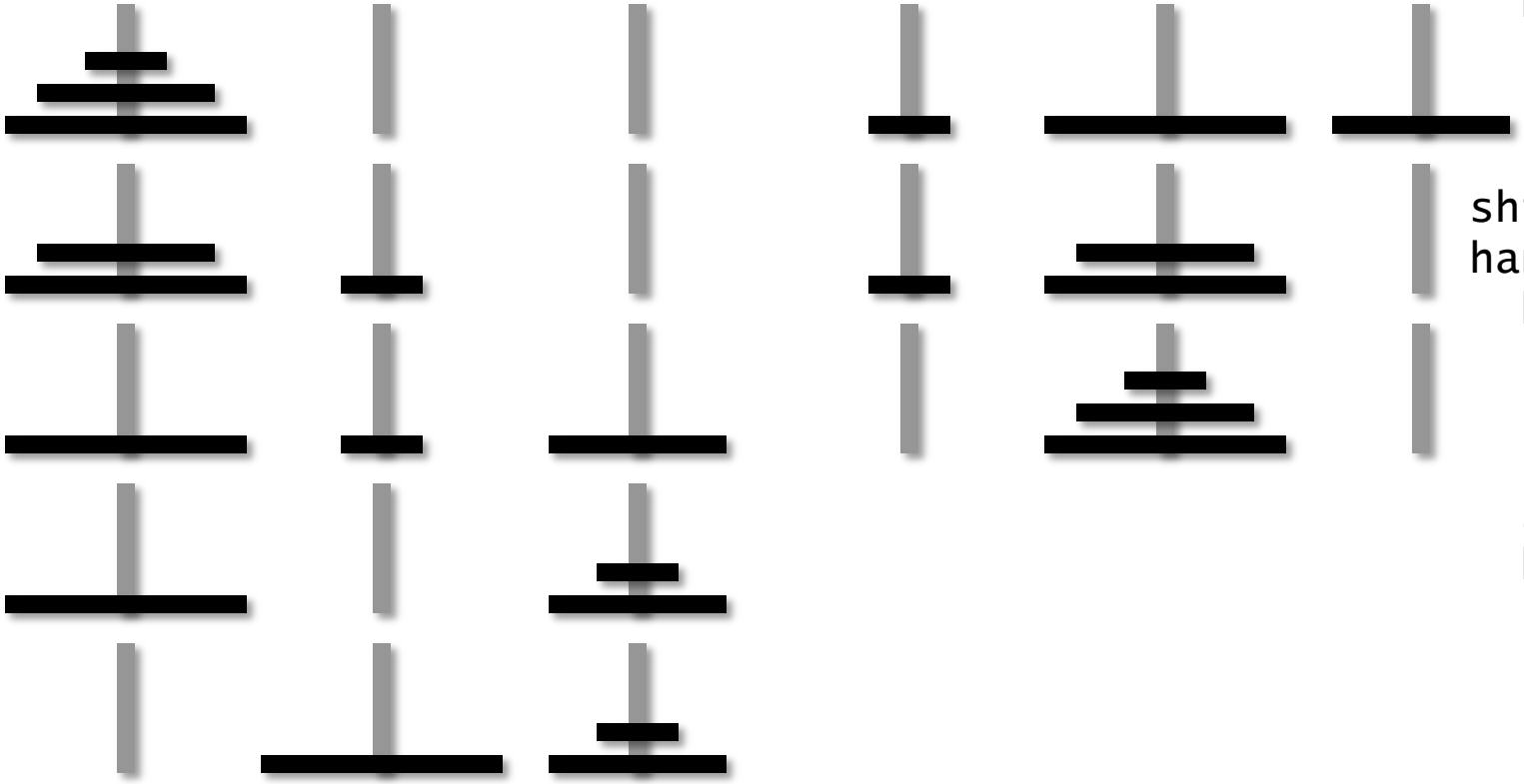
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```



```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
shift(3,+1)
hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
```

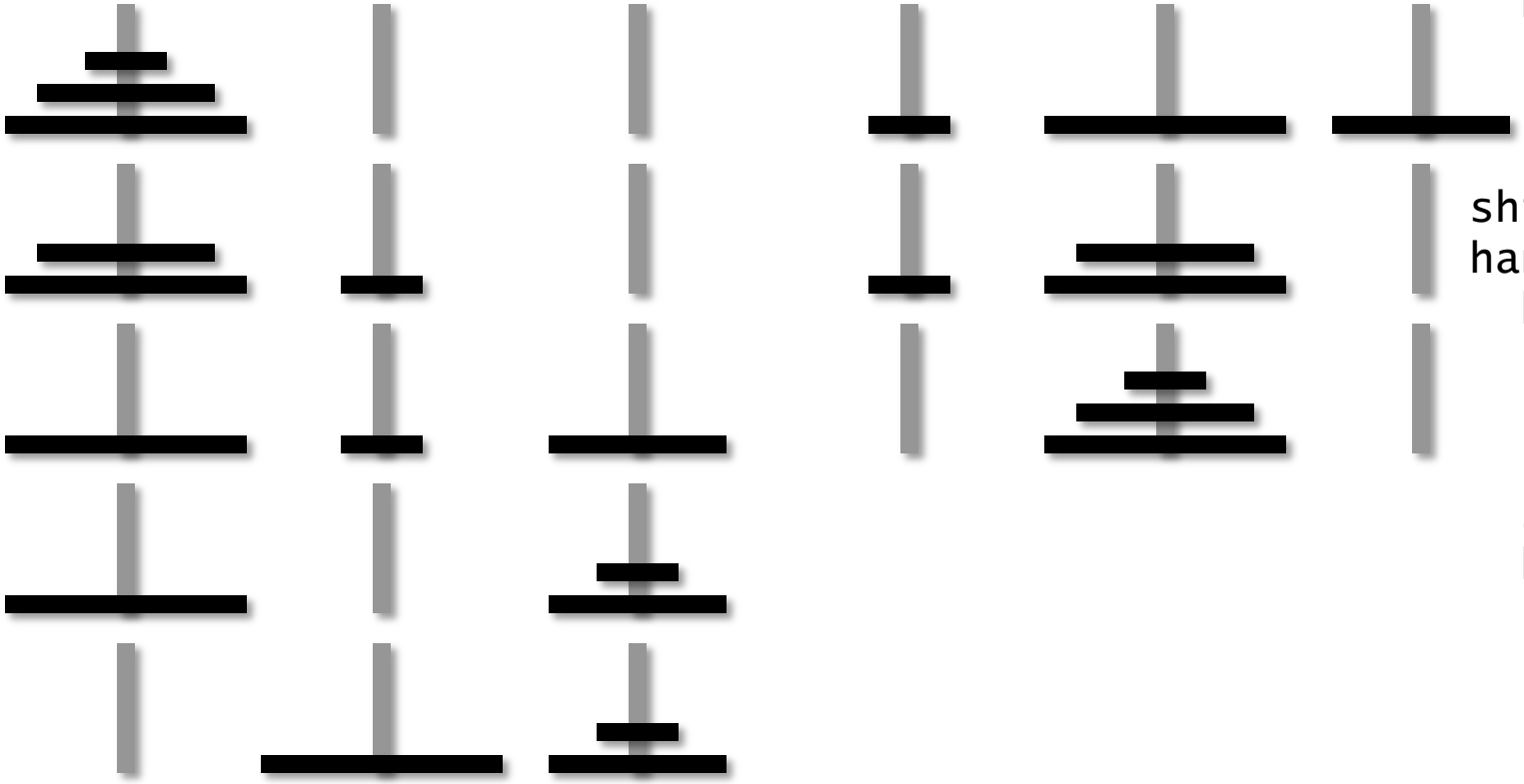
```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
shift(3,+1)
hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
```



```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

```
hanoi(3,+1)
  hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
shift(3,+1)
hanoi(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
    shift(2,-1)
    hanoi(1,+1)
      hanoi(0,-1)
      shift(1,+1)
      hanoi(0,-1)
```



Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και N δίσκοι

Θέλουμε να μετακινήσουμε τους δίσκους κατά ένα πάσσαλο δεξιά, υπακούοντας στους παρακάτω κανόνες

1) Μετακινούμε ένα δίσκο τη φορά

2) Ένας δίσκος δεν μπορεί να τοποθετηθεί πάνω από μικρότερο δίσκο

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

Αριθμός κινήσεων :

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και N δίσκοι

Θέλουμε να μετακινήσουμε τους δίσκους κατά ένα πάσσαλο δεξιά, υπακούοντας στους παρακάτω κανόνες

1) Μετακινούμε ένα δίσκο τη φορά

2) Ένας δίσκος δεν μπορεί να τοποθετηθεί πάνω από μικρότερο δίσκο

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

Αριθμός κινήσεων :

$$T(N) = 2T(N-1) + 1,\ N \geq 2$$
$$T(1) = 1$$

Αναδρομικοί Αλγόριθμοι

Οι Πύργοι του Ανόι

3 πάσσαλοι και N δίσκοι

Θέλουμε να μετακινήσουμε τους δίσκους κατά ένα πάσσαλο δεξιά, υπακούοντας στους παρακάτω κανόνες

1) Μετακινούμε ένα δίσκο τη φορά

2) Ένας δίσκος δεν μπορεί να τοποθετηθεί πάνω από μικρότερο δίσκο

```
void hanoi(int N, int d)
{
    if (N==0) return;
    hanoi(N-1,-d);
    shift(N,d);
    hanoi(N-1,-d);
}
```

Αριθμός κινήσεων :

$$T(N) = 2T(N-1) + 1,\ N \geq 2$$
$$T(1) = 1$$

$$\Rightarrow T(N) = 2^N - 1$$