Η μέθοδος 'Διαίρει-και-Βασίλευε' (Divide-and-Conquer)

- Μία αναδρομική προσέγκιση
  - **Διαίρεσε** το πρόβλημα σε έναν αριθμό από υποπροβλήματα.
  - **Επίλυσε** τα υποπροβλήματα αναδρομικά.
  - **Συνδύασε** τις λύσεις των υποπροβλημάτων έτσι ώστε να δώσουν τη λύση στο αρχικό πρόβλημα.

Παράδειγμα: Ταξινόμηση μέσω συγχωνεύσεων (Merge Sort)

Ανάλυση: Βασισμένη σε *αναδρομικές σχέσεις*.

Η $n-$οστή δύναμη ενός $n \times n$ πίνακα

Είσοδος: Ένας $n \times n$ πίνακας A.

Έξοδος: $A^n = \underbrace{A \times A \times \cdots \times A}_{n \text{ φορές}}$

Ισχύει: Το $A^2 = A \times A$ μπορεί να υπολογιστεί σε $O(n^3)$ χρόνο.

- **Μία απλή επαναληπτική (iterative) μέθοδος**

```
(1)  RESULT = A
(2)  for i = 2 to n do
(3)          RESULT = RESULT × A
```

- **Ανάλυση**

Βήμα 1: $\Theta(n^2)$

Βήμα 2: $\Theta(n)$

Βήμα 3: $\Theta(n^4)$ [$n-1$ πράξεις κόστους $O(n^3)$ η κάθε μία.]

Σύνολο: $\Theta(n^4)$

Ερώτηση: Μπορούμε να λύσουμε το πρόβλημα γρηγορότερα;

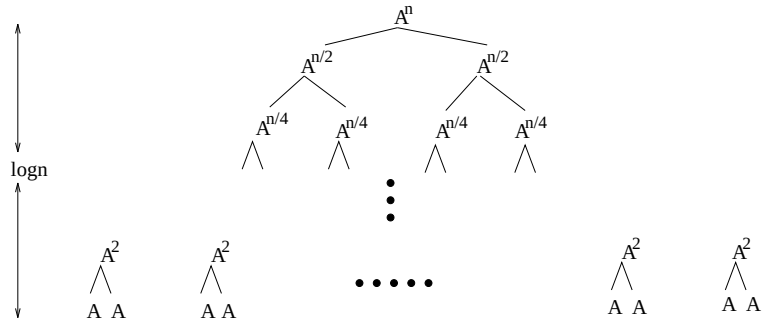
Μία λύση τύπου 'διαίρει-και-βασίλευε'

Υποθέτουμε ότι $n = 2^k$ (ή, $\log n = k$). Τότε,

$$A^n = \overbrace{A \times A \times \cdots \times A}^{n \text{ φορές}} = (\overbrace{A \times A \times \cdots \times A}^{\frac{n}{2} \text{ φορές}}) \times (\overbrace{A \times A \times \cdots \times A}^{\frac{n}{2} \text{ φορές}}) \quad \text{ή,}$$

$$A^n = (A^{\frac{n}{2}}) \times (A^{\frac{n}{2}}) = \left((A^{\frac{n}{4}}) \times (A^{\frac{n}{4}})\right) \times \left((A^{\frac{n}{4}}) \times (A^{\frac{n}{4}})\right) = \cdots \quad \text{ή,}$$

$$A^n = (A^{\frac{n}{2}})^2 = \left(A^{\frac{n}{4}})^2\right)^2 = \left(\left((A^{\frac{n}{8}})^2\right)^2\right)^2 = \left(\underbrace{\left(\left(A^2\right)^2\right)^2 \cdots}_{\log n}\right)^2$$



- **Ένας 'διαίρει-και-βασίλευε' αλγόριθμος**

```
RESULT = A
for i = 1 to log n do
        RESULT = RESULT × RESULT
```

Ανάλυση: $\Theta(n^3 \log n)$

Γενικά: Εάν ο πολλαπλασιασμός πινάκων μπορεί να πραγματοποιηθεί σε $O(f(n))$ χρόνο, τότε το A^n μπορεί να υπολογιστεί σε $O(f(n) \log n)$ χρόνο.

Σχόλιο: $f(n) = \Omega(n^2)$, δηλαδή, για τον πολλαπλασιασμό δύο $n \times n$ πινάκων χρειάζονται $\Omega(n^2)$ πράξεις.

Δυαδική αναζήτηση (Binary search)

Είσοδος: Ένα ταξινομημένο διάνυσμα A αποτελούμενο από n στοιχεία και ένα προς αναζήτηση στοιχείο key.

Έξοδος: Η θέση του στοιχείου key στον A. Εάν το key δεν υπάρχει στον A επιστρέφεται η τιμή 0.

- **Ένας τετριμμένος αλγόριθμος**

> Ερεύνησε το διάνυσμα από την αρχή προς το τέλος.
> Εάν συναντήσεις το στοιχείο key κατά τη διάρκεια της εξερεύνησης, τότε επέστρεψε τη θέση του.
> Αλλιώς, επέστρεψε 0.

Ανάλυση: $\Theta(n)$

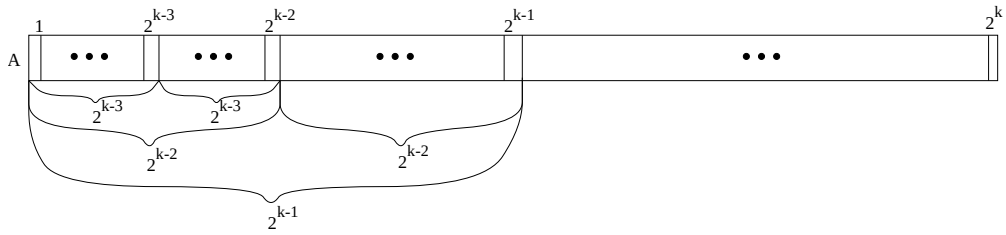
Πρόκειται για έναν άσχημα σχεδιασμένο αλγόριθμο ο οποίος δεν λαμβάνει υπ' όψιν του το γεγονός ότι το διάνυσμα είναι ταξινομημένο.

Ένας 'διαίρει-και-βασίλευε' αλγόριθμος

Υποθέστε ότι $n = 2^k$ (ή, $\log n = k$) και ότι το A είναι ταξινομημένο σε αύξουσα σειρά.

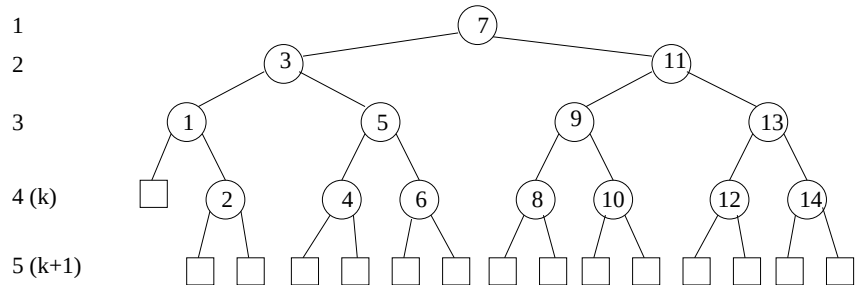
```
Binary_search(A, i, j, key)          /* Αναζητά το key μέσα στο A[i..j] */

if i > j return (0) and exit
if i ≤ j do
        middle = ⌊(i + j)/2⌋
        if key = A[middle] then return(middle) and exit
        else if key < A[middle] then Binary_search(A, i, middle − 1, key)
        else Binary_search(A, middle + 1, j, key)
```



Θεώρημα Εάν το n ανήκει στο διάστημα $[2^{k-1}, 2^k)$, τότε η $Binary\_search$ εκτελεί το πολύ k συγκρίσεις στοιχείων για μία αναζήτηση με θετικό αποτέλεσμα, και $k-1$ ή k συγκρίσεις για μία ανεπιτυχή αναζήτηση.

Απόδειξη Εξετάστε το δυαδικό δένδρο αποφάσεων (binary decision tree) που αντιστοιχεί στην $Binary\_search$.



- Μια επιτυχημένη αναζήτηση τελειώνει σε ένα ◯ κόμβο.
- Μια ανεπιτυχής αναζήτηση τελειώνει σε ένα □ κόμβο.
- ◯ κόμβοι υπάρχουν στα επίπεδα $1..k$.
- □ κόμβοι υπάρχουν στα επίπεδα k και $k+1$.

- Ο χρόνος που χρειάζεται για μια αναζήτηση με θετικό αποτέλεσμα είναι $O(\log n)$, ενώ ο χρόνος που χρειάζεται για μια ανεπιτυχή αναζήτηση είναι $\Theta(\log n)$.

Συμπέρασμα

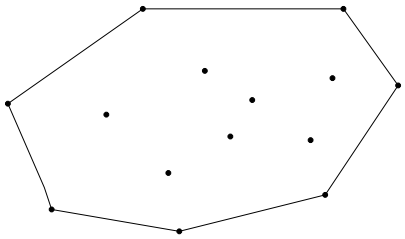
	Επιτυχής αναζήτηση	**Ανεπιτυχής αναζήτηση**
Καλύτερη περίπτωση	$\Theta(1)$	$\Theta(\log n)$
Χειρότερη περίπτωση	$\Theta(\log n)$	$\Theta(\log n)$
Μέση περίπτωση	$\Theta(\log n)$	$\Theta(\log n)$

Ερώτηση Θα βοηθούσε εάν χωρίζαμε το διάνυσμα σε 3 μέρη (*τριαδική αναζήτηση ternary search*);

Προσδιορισμός του ?convex hull?

- Το *convex hull* ενός συνόλου από n σημεία του επιπέδου είναι μία ακολουθία από σημεία του συνόλου η οποία ορίζει μια κυρτή convex 'γραμμή' που περικλείει όλα τα σημεία.

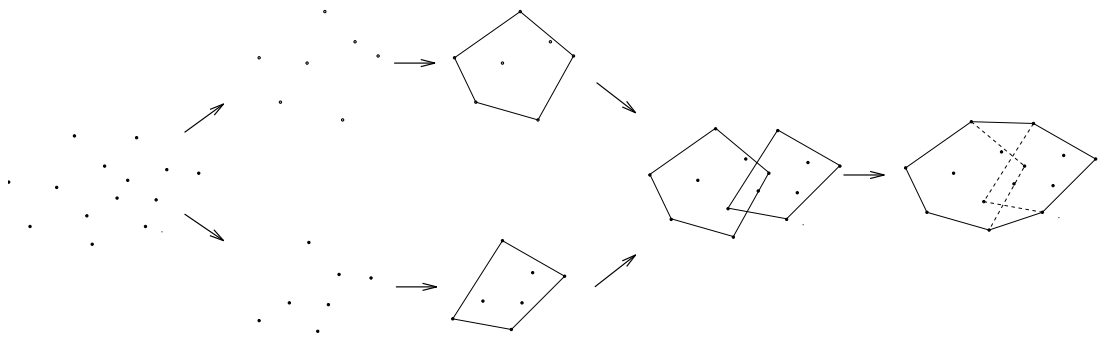
Παράδειγμα



- **Ένας 'διαίρει-και-βασίλευε' αλγόριθμος**

  - Χώρισε τα σημεία σε 2 σύνολα, το καθένα από $n/2$ σημεία.
  - Αναδρομικά υπολόγισε τα δύο convex hulls.
  - Συνδύασε τα 2 convex hulls.

Παράδειγμα



Πρόβλημα Δεν είναι εύκολο να συνδυαστούν οι λύσεις.

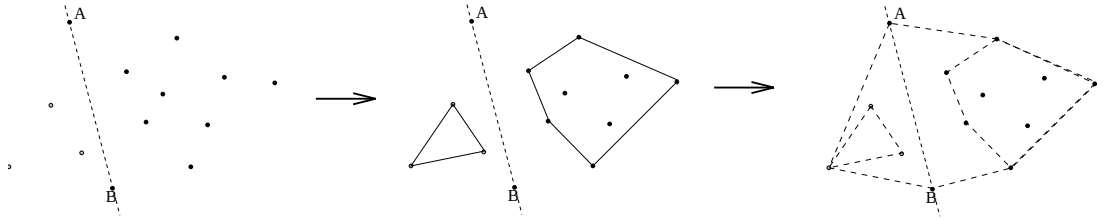
Εναλλακτική λύση Χώρισε τα σημεία διαφορετικά.

Ιδιότητα Τα σημεία με τις μέγιστες και ελάχιστες y-συντεταγμένες ανήκουν στο convex hull. Έστω ότι τα σημεία αυτά είναι τα A και B, αντίστοιχα.

- Χώρισε τα σημεία σε 2 σύνολα με βάση τη θέση του σε σχέση με την γραμμή AB.

  Λύσε τα 2 μικρότερα προβλήματα.

  Συνδύασε τις λύσεις τους.



Ανάλυση

Μέση περίπτωση: $O(n \log n)$

Χειρότερη περίπτωση: $O(n^2)$