Άσκηση 1.1

Οι παρακάτω συναρτήσεις είναι χωρισμένες (ομαδοποιημένες) ανά κλάση και ανά αύξουσα σειρά πολυπλοκότητας.

Θ(1)	10^{100}, 8^{1000}
Θ(loglogn)	$log\sqrt{logn}$, loglogn
Θ(logn)	$\sum_{k=1}^{n}\frac{1}{k}$, logn
Θ($\sqrt{n}$)	$n^{\frac{1}{2}}$, $\sqrt{2}^{logn}$, $5\sqrt{n}$
Θ($n^{\frac{2}{3}}$)	$n^{\frac{2}{3}}$,
Θ(n)	n-200, 100n+logn
Θ(nlogn)	nlogn, logn!, $logn^{n}$
Θ(n^2)	$\binom{n}{2}$, $8n^2$,
Θ(5^{logn})	5^{logn}
Θ(3^{log^2n})	3^{log^2n}
Θ(n^{k+1})	n^{k+1}
Θ(2^n)	2^n , $\sum_{k=0}^{n}\binom{n}{k}$
Θ(n!)	n!, (n-1)!
Θ(n^{logn})	n^{logn}

Άσκηση 1.2

Ερώτηση 1:

Αρχικά θα πρέπει να βρούμε, με τη βοήθεια ενός εξαντλητικού αλγορίθμου, όλα τα πιθανά υποσύνολα του S που αποτελούνται από k στοιχεία έκαστο και έπειτα ο αλγόριθμος για το κάθε ένα θα ελέγχει αν το άθροισμά του είναι ίσο με το M. Αν έστω και ένα από αυτά τα υποσύνολα έχει άθροισμα το M, τότε ο αλγόριθμος θα δώσει θετική απάντηση.

Ο αριθμός των υποσυνόλων που πρέπει να ελεγχθούν είναι οι συνδυασμοί των n στοιχείων ανά k.

Δηλαδή: $\binom{n}{k} = \frac{n!}{k!(n-k)!} \leq \frac{n^k}{k!} = O(n^k)$

Ερώτηση 2:

Σε αυτή την περίπτωση, πάλι με την βοήθεια ενός εξαντλητικού αλγορίθμου, βρίσκουμε όλα τα πιθανά υποσύνολα του S και έπειτα στο καθένα προσθέτουμε τα στοιχεία του και ελέγχουμε αν το άθροισμα ισούται με M.

Το σύνολο όλων των πιθανών υποσυνόλων του S είναι 2^n και σε κάθε υποσύνολο πρέπει να προσθέσουμε τα ίδια του τα στοιχεία, το οποίο έχει πολυπλοκότητα O(n), επομένως η συνολική πολυπλοκότητα είναι $O(n2^n)$.

Άσκηση 1.3.

α. T(n) = $5T\left(\frac{n}{5}\right)$ + 7n

Εφαρμόζουμε το Master Theorem. Έχουμε:

f(n)=7n.

Είναι f(n) = $\Theta(n^{\log_5 5}) = \Theta(n^1)$ = Θ(n).

Επομένως T(n) = $\Theta(n^{\log_5 5} \log n) = \Theta(n \log n)$ (από $2^η$ περίπτωση του Master Theorem)

β. T(n) = $4T\left(\frac{n}{16}\right)$ + $n^{\frac{5}{4}}$

Εφαρμόζουμε το Master Theorem. Έχουμε:

- f(n)= $n^{\frac{5}{4}}$
- $n^{\log_{16} 4} = n^{\frac{1}{2}}$

Είναι f(n) = $\Omega(n^{\frac{1}{2}+\varepsilon})$ για ε=1 > 0.

Επίσης ισχύει ότι $4(\frac{n}{16})^{\frac{5}{4}}$ = $4\frac{n^{\frac{5}{4}}}{32}$ = $\frac{n^{\frac{5}{4}}}{8}$ (για σταθερά c = $\frac{1}{8}$ < 1)

Επομένως T(n) = $\Theta(n^{\frac{5}{4}})$ (από 3η περίπτωση του Master Theorem).

γ. T(n) = $8T\left(\frac{n}{4}\right)$ + $6\sqrt{n}$

Εφαρμόζουμε το Master Theorem. Έχουμε:

- f(n)= $6\sqrt{n}$ = $6n^{\frac{1}{2}}$
- $n^{\log_4 8}$ = $n^{\frac{3}{2}}$

Είναι f(n) = $O(n^{\frac{3}{2}-\varepsilon})$ για ε = 1 > 0

Επομένως T(n) = $\Theta(n^{\frac{3}{2}})$ (από 1η περίπτωση του Master Theorem)

δ. T(n) = $2T(n-2)$ + $O(1)$ = $2T(n-2)$ + 1

Γνωρίζουμε από την εκφώνηση ότι T(0)=1.

Δεν μπορούμε σε αυτή τη περίπτωση να εφαρμόσουμε το Master Theorem αφού η αναδρομική σχέση δεν έχει την ζητούμενη από το Master Theorem μορφή.

Λύνω την αναδρομική:

T(n)=2T(n-2)+1 = 2(2(T(n-4)+1)+1) = 2(2(T(n-6)+1)+1)+1 =

$$2^k T(n-2k) + 2^2 * 1 + 2^1 * 1 + 2^0 * 1 = 2^k T(n-2k) + \sum_{i=0}^{k-1} 2^i$$

Επειδή ξέρουμε ότι T(0)=1 θέτω $(n-2k) = 0$

Άρα για $k = \frac{n-1}{2}$: T(n) =$2^{\frac{n-1}{2}} T(0) + \sum_{i=0}^{\frac{n-3}{2}} 2^i = 2^{\frac{n-1}{2}} + (2^{\frac{n-1}{2}} - 1) =>$ (από το διωνυμικό θεώρημα)

$$Άρα\ T(n) = 2^{\frac{n-1}{2}} + 2^{\frac{n-1}{2}} - 1.$$

Πολυπλοκότητα: $O(2^n)$

Άσκηση 1.4.

Αλγόριθμος:

- Αρχικά ταξινομώ τα n στοιχεία του πίνακα. Απαιτούμενος χρόνος: O(nlogn)
- Έπειτα χωρίζω τον πίνακα σε 2 ίσους υποπίνακες.
- Τέλος λύνω αναδρομικά το πρόβλημα για τον κάθε υποπίνακα. Κρατάω έναν counter ο οποίος αυξάνεται για το κάθε ζεύγος θέσεων (i, j), το οποίο αν i < j ικανοποιεί τη ζητούμενη συνθήκη A[i] > A[j]. Μετά συγχωνεύω τους υποπίνακες συγκρίνοντας τα γειτονικά στοιχεία, και αυξάνω τον counter αναλόγως. Απαιτούμενος χρόνος: O(n) για την διαπέραση του πίνακα, O(1) για την σύγκριση των αριθμών, Θ(nlogn) για την συγχώνευση.

Άρα συνολικά η πολυπλοκότητα του αλγορίθμου είναι:

O(nlogn) + O(n) + O(1) = O(nlogn)

Counter=0

sort(A[1 . . . n]);

ΑΛΓΟΡ(A[1 . . . n]){

k=$\frac{n}{2}$

counter=ΑΛΓΟΡ(A[1 . . . k)

counter=ΑΛΓΟΡ(A[k . . . n)

if(A[$\frac{k}{2}$] > A[$\frac{k}{2}+1$])

return counter+1

```
else
    return counter

}
```