1η Σειρά Ασκήσεων

Ονοματεπώνυμο: Φίλιππος Δουραχαλής
Α.Μ: 3170045

Άσκηση 1

Μέγεθος τομέα = 4096 Bytes
65536 ίχνη/επιφάνεια
8 πλακέτες διπλής όψης = 16 επιφάνειες
256 τομείς/ίχνος
SeekTime_{avg} = 10ms
Ταχύτητα περιστροφής 7200rpm

(α) Η χωρητικότητα κάθε ίχνους θα είναι:
$1 \text{ track} = 256 * 4096 = 1\,048\,576\,\text{B} = 1\,\text{MB}$

Η χωρητικότητα μιας επιφάνειας του δίσκου θα είναι:
$1 \text{ surface} = 65536 * 1048576 = 68\,719\,476\,736\,\text{B} = 64\,\text{GB}$

Η χωρητικότητα ολόκληρου του δίσκου θα είναι:
$TotalSize = 16 * 64 = 1024\,\text{GB} = 1\,\text{TB}$

(β) Ο αριθμός κυλίνδρων ισούται με τον αριθμό των ιχνών κάθε επιφάνειας του δίσκου.
Άρα ο δίσκος έχει συνολικά 65536 κυλίνδρους.

(γ) Η μέγιστη καθυστέρηση περιστροφής προκύπτει όταν χρειαζόμαστε μια πλήρη περιστροφή για να βρεθεί ο σωστός τομέας κάτω από την κεφαλή, άρα θα ισούται με:

$$RotationDelay_{max} = \frac{60}{7200} = 0,0083 = 8,3\text{msec}$$

Η μέση καθυστέρηση περιστροφής ισούται με το προηγούμενο αποτέλεσμα δια 2

$$RotationDelay_{avg} = \frac{60}{7200} * 0,5 = 0,00415 = 4,15\text{msec}$$

(δ) Σε μια περιστροφή ο δίσκος διαβάζει 1 ίχνος = 1 MB

Ο περιστρέφεται 7200 φορές το λεπτό, δηλαδή $\frac{7200}{60} = 120$ φορές ανα δευτερόλεπτο.
Άρα ο ρυθμός μεταφοράς θα είναι:

$$TransferRate = 120 * 1048576 = 125829120\frac{\text{B}}{\text{s}} = 120\frac{\text{MB}}{\text{s}}$$

Άσκηση 2

R(#a,b,c,d,e)
Μέγεθος γνωρίσματος a $= 10\,\text{B}$
Μέγεθος εγγραφής σχέσης $= 10 + 50 + 30 + 18 + 20 = 128\,\text{B}$
Μέγεθος σχέσης = N εγγραφές
Μέγεθος δείκτη $6\,\text{B}$
Μέγεθος σελίδας $= 1024\,\text{B}$

Κάθε σελίδα θα περιέχει $\frac{1024}{128} = 8$ εγγραφές της σχέσης.

Άρα χρειαζόμαστε συνολικά $\frac{N}{8}$ σελίδες για την αποθήκευση της σχέσης

Τέλος κάθε εγγραφή του ευρετηρίου θα έχει μέγεθος $10 + 6 = 16\,\text{B}$

(α) Ένα πυκνό ευρετήριο θα περιέχει όλες τις τιμές του γνωρίσματος πάνω στο οποίο ορίζεται.
Εφόσον το γνώρισμα **a** είναι πρωτεύον κλειδί, περιέχει μοναδικές τιμές, άρα το πυκνό ευρετήριο Θα περιέχει N εγγραφές.

Κάθε σελίδα μπορεί να περιέχει $\frac{1024}{16} = 64$ εγγραφές του ευρετηρίου.

Άρα τελικά για την αποθήκευσή του απαιτούνται $\frac{N}{64}$ σελίδες.

Για την αποθήκευση του ευρετηρίου και της σχέσης απαιτούνται συνολικά

$\frac{N}{8} + \frac{N}{64} = \frac{9N}{64}$ σελίδες.

(β) Ένα αραιό ευρετήριο θα περιέχει μόνο την πρώτη τιμή του γνωρίσματος από κάθε σελίδα στην οποία αποθηκεύονται οι εγγραφές της σχέσης. Δηλαδή θα περιέχει συνολικά $\frac{N}{8}$ εγγραφές, εφόσον κάθε σελίδα της σχέσης περιέχει 8 τιμές του γνωρίσματος a.

Άρα για την αποθήκευση του αραιού ευρετηρίου χρειαζόμαστε $\frac{N}{8 * 64}$ σελίδες.

Τέλος για την αποθήκευση της σχέσης και του ευρετηρίου χρειαζόμαστε συνολικά $\frac{65N}{512}$ σελίδες του δίσκου.

Άσκηση 1.3

Οι αναδρομικές εξισώσεις του χρόνου εκτέλεσης για κάθε περίπτωση θα είναι οι εξής:

a. $T(n) = 4T(n/4) + 7n$

   Από Master Theorem:

$$f(n) = 7n$$
$$a = b = 4, \text{ άρα}$$
$$n^{\log_4 4} = n^1 = \Theta(n)$$

   εφόσον ισχύει ότι $7n = \Theta(n^{log_4 4}) = \Theta(n)$ τότε ισχύει η 2η περίπτωση του Θεωρήματος:

$$T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n \log n)$$

b. $T(n) = 4T(n/16) + n^{\frac{5}{4}}$

   Από Master Theorem:

$$f(n) = n^{5/4}$$
$$a = 4, b = 16 = a^2, \text{ άρα}$$
$$n^{\log_{16} 4} = n^{1/2}$$

Άρα η $f(n) = n^{5/4} = \Omega(n^{log_{16}4+\varepsilon}) = \Omega(n^{1/2+\varepsilon})$ αφού $f(n) > n^{log_{16}4}$

και επιπλέον

$$\begin{aligned} af\Big(\frac{n}{b}\Big) = 4f\Big(\frac{n}{16}\Big) = 4\Big(\Big(\frac{n}{16}\Big)^{5/4}\Big) = \frac{n^{5/4}}{256} \\ \leq cf(n), \text{με } c = \frac{1}{256} \end{aligned}$$

Επομένως ισχύει η 3η περίπτωση, δηλαδή $T(n) = \Theta(\sqrt[4]{n^5})$

c. $T(n) = 8T(n/4) + 6\sqrt{n}$

Από Master Theorem:

$$f(n) = 6\sqrt{n} = 6n^{1/2}$$
$$a = 8, b = 4, \text{ άρα}$$
$$n^{\log_4 8} = n^{3/2}$$

Αρα η $f(n) = 6\sqrt{n} = O(n^{log_4 8-\varepsilon}) = O(n^{3/2+\varepsilon})$ για $\varepsilon \simeq 0.5$
αφού $f(n) < n^{log_4 8}$
και η λύση είναι
$T(n) = \Theta(n^{log_4 8}) = \Theta(n^{3/2}) = \Theta(n\sqrt{n})$
βάσει της 1ης περίπτωσης του Θεωρήματος

d. $T(n) = 2T(n-2) + \Theta(1)$

Λύνουμε την εξίσωση με τη μέθοδο της αντικατάστασης:

Αρχικά παρατηρούμε πως η αναδρομική εξίσωση δημιουργεί ένα πλήρες δυαδικό δέντρο. Η αναδρομή σταματάει όταν $n - 2i = 0$, δηλαδή για $i = \frac{n}{2}$, που σημαίνει πως το δέντρο έχει n/2 + 1 επίπεδα. Όλοι οι κόμβοι ενός επιπέδου συνεισφέρουν σταθερό κόστος Θ(1). Εφόσον υπάρχουν $2^{n/2} - 1$ κόμβοι στο δέντρο, το συνολικό κόστος πρέπει να είναι $(2^{n/2} - 1)\Theta(1)$, άρα εικάζουμε ότι η T φράσεται άνω από την 2^n και άρα $T(n) = O(2^n)$

Επαγωγικη απόδειξη

Για $n = 0$ η βασική περίπτωση ισχύει.
Έστω ότι $T(n) \leq c2^n - d$
Δείχνουμε ότι ισχύει και για $n-2$

$$
\begin{aligned}
T(n) &\leq 2(c2^{n-2} - d) + k \\
&= c2^{n-1} - 2d + k \\
&\leq c2^n - 2d + k \\
&\leq c2^n - d
\end{aligned}
$$

Άρα εφόσον $c2^n - d < c2^n$, $T(n) = O(2^n)$

Από τους παραπάνω χρόνους εκτέλεσης χειρότερης περίπτωσης των αλγορίθμων, παρατηρούμε πως ο d έχει τον χειρότερο.

Γνωρίζουμε επίσης ότι $n^{3/2} > n^{5/4}$ και άρα αρκεί να συγκρίνουμε την $n^{5/4}$ με την $n \log n$ για να αποφασίσουμε ποιά είναι καλύτερη.

$$
\begin{aligned}
\lim_{n \to \infty} \frac{n^{5/4}}{nlogn} &= \lim_{n \to \infty} (\frac{n^1/4}{logn}) \\
&\overset{\frac{\infty}{\infty}}{=} \lim_{n \to \infty} \frac{1}{4} (\frac{n^{-3/4}}{1/n}) \\
&= \lim_{n \to \infty} \frac{1}{4} n^{1/4} \\
&= \infty
\end{aligned}
$$

Άρα το $n \log n$ αποτελεί καλύτερο άνω φράγμα, οπότε επιλέγουμε τον αλγόριθμο b για να λύσουμε το Π.

Άσκηση 1.4

Έστω ο πίνακας A με $|A| = n$. Χρησιμοποιούμε τον αλγόριθμο της MergeSort για να ταξινομήσουμε τον A.

Σε κάθε βήμα της αναδρομής ο πίνακας διαιρείται σε δύο υποπίνακες L και R οι οποίοι στη συνέχεια συγχωνεύονται. Κατά τη διαδικασία της συγχώνευσης οι L και R είναι ταξινομημένοι (αφού είτε αποτελούνται από 1 στοιχείο στη τερματική περίπτωση ή έχουν προκύψει από προηγούμενη συγχώνευση και άρα έχουν ταξινομηθεί ήδη). Άρα σε αυτό το επίπεδο θέλουμε να μετρήσουμε πόσες αντιστροφές υπάρχουν συνολικά μεταξύ των πινάκων και να επιστρέψουμε το πλήθος τους

Στο στάδιο της συγχώνευσης συγκρίνουμε διαδοχικά τα $mid-l+1$ στοιχεία του L με τα $r-mid$ στοιχεία του R για να καθορίσουμε ποιό είναι μικρότερο. Σε αυτό το βήμα είναι εύκολο να βρούμε πόσες αντιστροφές υπάρχουν στους δύο αυτούς πίνακες, αφού όταν βρούμε ένα στοιχείο i του L που είναι μεγαλύτερο από κάποιο στοιχείο j του R, το ίδιο θα ισχύει για τα $i, i+1, ..., mid$ στοιχεία του L (λόγω ταξινόμησης).

Επομένως είμαστε σίγουροι πως σε αυτό το επιπέδο υπάρχουν τόσες αντιστροφές όσες και τα $mid-l+1-i$ στοιχεία του L που απομένουν.

Άρα αρκεί να τροποποιήσουμε την MergeSort ώστε να κρατάει το πλήθος αυτό μέχρι όλα τα στοιχεία να είναι στις σωστές τους θέσεις και αρα να έχουν μετρηθεί όλες οι αντιστροφές στοιχείων των συστοιχειών L και R σε κάθε αναδρομικό βήμα ως εξής:

(Σημείωση: Ως βάση χρησιμοποιείται ο αλγόριθμος του βιβλίου "Εισαγωγή στους Αλγορίθμους)

Algorithm 1 Άσκηση 1.4

```
procedure SORT(A, left, right)
    inv = 0
    if left < right then
        q = (right − left)/2
        inv = inv + SORT(A, left, q)
        inv = inv + SORT(A, q + 1, right)
        inv = inv + MERGE(A, left, q, right)
    return inv

procedure MERGE(A, l, mid, r)
    inv = 0
    lengthL = mid − l + 1
    lengthR = r − mid
    L[1...lengthL + 1]
    R[1...lengthR + 1]
    for i = 1 to lengthL do
        L[i] = A[left + i − 1]
    for j = 1 to lengthR do
        R[j] = A[mid + j]
    L[lengthL + 1] = ∞                ▷ Φρουρος τελους συστοιχίας L
    R[lengthR + 1] = ∞                ▷ Φρουρος τελους συστοιχίας R
    i = 1
    j = 1
    for k = left to right do
        if L[i] ≤ R[j] then
            A[k] = L[i]
            i = i + 1
        else
            ▷Κάθε στοιχείο του L μετά το i θα είναι επίσης μεγαλύτερο του R[j]
            A[k] = R[j]
            j = j + 1
            inv = inv + lengthL - i
    return inv
```

Πολυπλοκότητα Αλγορίθμου

Ο αλγόριθμος παρουσιάζει ίδια πολυπλοκότητα με τον αλγόριθμο της Merge-

Sort, δηλαδή
$T(n) = 2T(n/2) + cn$
Χρησιμοποιώντας το Θεώρημα Κυρίαρχου Όρου παίρνουμε:

$f(n) = cn$
$a = 2, b = 2$, άρα
$n^{\log_2 2} = n$

Αφού $f(n) = \Theta(n)$, ισχύει ότι $T(n) = \Theta(nlogn)$

Άσκηση 1.6

Έστω σύνολο $W = w_1, w_2, ..., w_n$ που περιέχει τα βάρη κάθε αντικειμένου w_i για i = 1, ..., n και $C \in \mathbb{N}$ η χωρητικότητα μίας κούτας.

Έστω ακόμη το σύνολο $A \subseteq N$ που περιέχει τα αντικείμενα μιας βέλτιστης λύσης.

Το πρόβλημα μπορεί να λυθεί αντίστοιχα με το διακριτό πρόβλημα του σακιδίου, με την διαφορά ότι μας αρκεί ο αλγόριθμος να επιστρέφει true αν βρεί κάποιο υποσύνολο με άθροισμα στοιχείων C.

Δεδομένου ότι το A αποτελεί βέλτιστη λύση, αφαιρώντας το στοιχείο w_n από το W, η λύση του προβλήματος για $W - w_n$ είναι το σύνολο A$\backslash w_n$ που είναι επίσης βέλτιστο. Άρα αρκεί να πάρουμε περιπτώσεις, αναλόγως αν η λύση (το A) περιέχει το w_n ή όχι

Αν συμβολίσουμε f(n, C) τη λύση που επιστρέφει ο αλγόριθμος για είσοδο το σύνολο W με $|W| = n$ και τη χωρητικότητα C, παίρνουμε τις περιπτώσεις για το f όπως περιγράφηκαν πιο πάνω:

1. $f(n-1, C)$ υποθέτωντας ότι η λύση δεν περιέχει το n και

2. $f(n-1, C-w_n)$ αν η λύση το περιέχει

H f(i, y) θα πρέπει να γίνεται 1 αν υπάρχει υποσύνολο i στοιχείων με άθροισμα y και 0 αν δεν υπάρχει. Το υποσύνολο αυτό μπορεί να έχει ή όχι το αντικείνο n και άρα:

$$f(n, C) = f(n-1, C) \vee f(n-1, C-w_n)$$

a) Αναδρομική εξίσωση

Ορίζουμε την f ως εξής:

$$f(i,y) = \begin{cases} 1 & y = 0 \\ 0 & i = 0 \text{ και } y \neq 0 \\ f(i-1,y) \vee f(i-1,y-w_i) & i > 1 \\ f(i-1,y) & w_i > y \end{cases}$$

Με αυτόν τον τρόπο κατασκευάζουμε αναδρομικά τη λύση της f, σταματώντας όταν το βάρος y=0, δηλαδή όταν έχουμε προσθέσει στην λύση μας k αντικείμενα του W με άθροισμα y ή όταν δεν έχουμε άλλα αντικείμενα να προσθέσουμε στη λύση.

Εφόσον έχουμε την αναδρομική λύση εκθετικού χρόνου, μπορούμε να κατασκευάσουμε έναν επαναληπτικό αλγόριθμο ως εξής:

Algorithm 2 Άσκηση 1.6

1: **for** $i = 1$ **to** n **do**
2:     $f(i,0) \leftarrow true$
3: **for** $i = 1$ **to** C **do**
4:     $f(1,i) \leftarrow w_i == i$
5: **for** $i = 2$ **to** n **do**
6:     **for** $y = 1$ **to** C **do**
7:         **if** $w_i > y$ **then**
8:             $f(i,y) \leftarrow f(i-1,y)$
9:         **else**
10:             $f(i,y) \leftarrow f(i-1,y) \vee f(i-1,y-w_i)$
    return $f(n,C)$

Το τελευταίο κελί του πίνακα f που δημιουργείται αναπαριστά ολόκληρο το προβλημα. Αν είναι true, σημαίνει πως τουλάχιστον ένα απο τα 2 υποπροβλήματα και κατά συνέπεια το αρχικό πρόβλημα έχει βέλτιστη λύση

b) Πολυπλοκότητα Αλγορίθμου

Στις γραμμές 1-4 αρχικοποιείται ένας πίνακας μεγέθους n x (C+1). Η πρώτη στήλη του πίνακα (για C = 0) περιέχει true αφού σε αυτή τη περίπτωση αρκεί να μην επιλέξουμε κανένα στοιχείο. Η πρώτη γραμμή του πίνακα γίνεται false, με εξαίρεση το κελί όπου το βάρος του πρώτου στοιχείου του W ισούται με το y, καθώς τότε αυτό ανήκει στη λύση του υποπροβλήματος $(1, w_1)$ και άρα είναι true. Στις γραμμές 5-10 υπολογίζουμε τις υπόλοιπες τιμές της f βάσει των περιπτώσεων που αναφέρονται παραπάνω.

Τα δύο πρώτα for τρέχουν n και C φορές, σε χρόνο O(n) και O(C) αντίστοιχα, ενώ οι πράξεις μέσα σε κάθε επανάληψη εκτελούνται σε σταθερό χρόνο O(1). Το for της γραμμής 8 εκτελείται C φορές για κάθε στοιχείο του συνόλου W δηλαδή συνολικά (n-1)*C φορές (έχουμε ήδη υπολογίσει τις τιμές του πρώτου στοιχείου του πίνακα W), ενώ οι πράξεις μέσα του εκτελούνται επίσης σε σταθερό χρόνο.

Άρα συνολικά ο χρόνος εκτέλεσης του αλγορίθμου είναι

$$n * O(1) + C * O(1) + (n - 1) * C * O(1) = O(nC)$$

Σχετικά με την πολυπλοκότητα χώρου, κατα την εκτέλεση του αλγορίθμου δημιουργείται μόνο ο πίνακας f ο οποίος αποτελείται απο (n * (C+1)) κελιά και άρα απαιτεί χώρο O(n*(C+1)) = O(n*C)