ΑΛΓΟΡΙΘΜΟΙ

• ΦΡΟΝΤΙΣΤΗΡΙΟ 7 -

Άσκηση 1/6 (Δυναμικός Προγραμματισμός).
Δίνονται αντικείμενα $1, 2, \dots, n$ με θετικά βάρη $w_1, w_2, \dots, w_n$ αντίστοιχα. Πρέπει να επιλέξετε κάποια από αυτά ώστε να μεγιστοποιείται το συνολικό βάρος, υπό τον περιορισμό ότι αν επιλέξετε το αντικείμενο i, δεν επιτρέπεται να επιλέξετε το αντικείμενο $i+1$.
Δώστε αλγόριθμο δυναμικού προγραμματισμού που να υπολογίζει το μέγιστο άθροισμα βαρών και εκτιμήστε την πολυπλοκότητά του.

Λύση

Κάθε αντικείμενο $i \rightarrow$ έχει βάρος w_i

$$\underline{1} \quad \underline{100} \quad \underline{10^4} \;\Big|\; \underline{1} \quad \underline{1}$$
i

→ Δε μπορούμε να ταξινομήσουμε τον πίνακα που περιέχει τα βάρη των αντικειμένων, γιατί τότε θα έπρεπε να κρατήσουμε και να βάλουμε περιορισμούς για το ποια στοιχεία δεν μπορούμε να πάρουμε κάθε φορά (αν πάρω το i δεν μπορώ να πάρω το $i+1$).

Έστω $OPT(i)$ το μέγιστο βάρος που μπορώ να εξασφαλίσω επιλέγοντας τα αντικείμενα $1, 2, \dots, i$.
Θέλω να υπολογίσω το $OPT(n)$ που είναι:
το μέγιστο βάρος που μπορώ να πετύχω αν έχω στη διάθεσή μου όλα τα n αντικείμενα.
Τώρα θέλω να σπάσω το πρόβλημα σε υποπροβλήματα, σκέφτομαι ως εξής:
"Το nοστό αντικείμενο θα περιέχεται στην λύση μου;" Αυτό εξαρτάται από το τι έχουμε ήδη πάρει, δηλαδή αν δεν

έχω πάρει το $n-1$, μπορώ να πάρω το n κτλ.

Για το $OPT(n)$ έχουμε:

$$OPT(n) = \max\{ OPT(n-1),\ w_n + OPT(n-2) \}$$

δεν περιλαμβάνουμε το nοστό στοιχείο και λύνουμε το $OPT(n-1)$

βάρος nοστού αντικειμένου

λύνουμε το $OPT(n-2)$ οπότε αυτό μας επιτρέπει να πάρουμε και το nοστό του οποίου προστείθεται το βάρος

→ Στον δυναμικό προγραμματισμό βρίσκουμε πρώτα την αξία κάθε λύσης και μετά επιστρέφουμε τη βέλτιστη.

Γενικά έχουμε: $OPT(i) = \max\{ OPT(i-1),\ w_i + OPT(i-2) \}$

με αρχικές συνθήκες $OPT(0) = 0$,

$OPT(1) = w_1$.

Αλγόριθμος:

```
max_weight (w1, w2, ... wn)
    OPT(0) = 0
    OPT(1) = w1
    for i=2 to n do
        OPT(i) = max{OPT(i-1), wi + OPT(i-2)}
    return OPT(n)
```

OPT

0	w_1		• • • •		
0	1	2			n

$$OPT(2) = \max\{ OPT(1),\ w_2 + OPT(0) \}$$

Ο αλγόριθμος έχει πολυπλοκότητα $O(n)$, απλός επαναληπτικός αλγόριθμος n επαναλήψεων όπου κάθε επανάληψη είναι σταθερού χρόνου

Άσκηση 2/6 (Δυναμικός προγραμματισμός)

Δίνεται ακολουθία $x = (x_1, x_2, \dots, x_n)$, η οποία δεν περιέχει στοιχεία που επαναλαμβάνονται. Να βρείτε μία υποακολουθία της στην οποία τα στοιχεία είναι σε φθίνουσα σειρά κι έχει το μέγιστο δυνατό άθροισμα.

Λύση

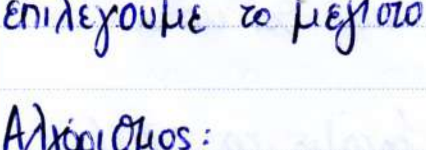
Ορίζω το υποπρόβλημα με βάση το οποίο θα χτίσω την αναδρομική εξίσωση που υπολογίζει και επιστρέφει τη βέλτιστη λύση.

Έστω $c(j)$ το κόστος της βέλτιστης υποακολουθίας μέχρι τη θέση j, η οποία περιέχει το στοιχείο $x(j)$.

Έχουμε:

$$c(j) = \max_{1 \le i \le j} \{ c(i) \} + x_j$$

Η βέλτιστη υποακολουθία είναι μία υποακολουθία μέχρι κάποια θέση i συν το x_j. Η θέση i θα είναι εκείνη που μεγιστοποιεί το άθροισμα μέχρι εκείνη τη θέση.



Επίσης πρέπει τα στοιχεία που επιλέγω να είναι σε φθίνουσα σειρά, οπότε πρέπει υποχρεωτικά να έχουμε $x_i > x_j$

Θέλω να βρω τα $c(j)$ για κάθε j από 1 μέχρι n.
Άρα για κάθε θέση j χρειάζεται να γίνουν $j-1$ έλεγχοι (1 για κάθε $i \in [1, j]$). Όταν υπολογίσουμε ΟΛΑ τα $c(j)$ επιλέγουμε το μέγιστο.

Αλγόριθμος:

```
max_decreasing_subsequence (x1, x2, ..., xn)
    C(1) = x1.
    For j=2 to n do
```

```
c(j) = x_j
For i=1 to j-1
    if x_i > x_j then
        if x_j + c(i) > c(j)
            c(j) = x_j + c(i)
return max{c(j)}
```

← ΠΡΟΣΟΧΗ ΟΧΙ C(N) ΑΠΑΡΑΙΤΗΤΑ, γιατί μπορεί να μην είναι αυτό με την μέγιστη υποακολουθία

Πολυπλοκότητα:
$O(n^2)$ - 2 loop
το ένα εμφωλευμένο στο άλλο.

Άσκηση 3/6 (Δυναμικός Προγραμματισμός)

Ξεκινάμε από έναν ακέραιο n στον οποίο μπορούμε να εφαρμόσουμε τις πράξεις:

- Διαίρεση με το 2, 3, 5 (εφόσον το υπόλοιπο της είναι 0).
- Αφαίρεση μιας μονάδας.

Υπολογίστε το ελάχιστο πλήθος πράξεων που απαιτούνται για να φτάσουμε στο 1, δίνοντας αλγόριθμο δυναμικού προγραμματισμού.

Λύση

Ορίζω ως $OPT(i)$ το ελάχιστο πλήθος πράξεων που απαιτούνται ξεκινώντας από το i για να φτάσουμε στο 1.

Είναι: $OPT(i) = 1 + \min\{OPT(i-1), OPT(i/t)\}$,
$t \in \{2,3,5\}$ και $i/t \in \mathbb{Z}$

↑ ελάχιστο πλήθος βημάτων που μένουν από τη στιγμή που θα κάνω το πρώτο βήμα.

Έχουμε $OPT(1) = 0$ και ζητάμε το $OPT(n)$.

Αλγόριθμος:
$OPT(1) = 0$

for $i = 2$ to n

$$OPT(i) = 1 + \min\{OPT(i-1), OPT(i/t)\}$$

$t \in \{2, 3, 5\}$ και $OPT(x) = +\infty$, αν $x = i/t \notin \mathbb{Z}$

return $OPT(n)$

<u>Πολυπλοκότητα</u>: $O(n)$.