Φροντιστήριο 4

Άσκηση 1 (Το 0-1 knapsack πρόβλημα)

Έστω {1, …, *n*} αντικείμενα όπου το *i*-οστό αντικείμενο έχει βάρος *w*[*i*] κιλά και αξία *v*[*i*]. Έχουμε μια τσάντα η οποία μπορεί να σηκώσει βάρος μέχρι *W* κιλά. Ποια αντικείμενα θα πρέπει να τοποθετήσουμε στην τσάντα για να μεγιστοποιήσουμε την αξία του περιεχομένου της;

- Να λύσετε το πρόβλημα με αλγόριθμο δυναμικού προγραμματισμού.
- Να λύσετε το πρόβλημα με αλγόριθμο οπισθοδρόμησης.
- Να συγκρίνετε τους δύο αλγορίθμους ως προς την αποδοτικότητά τους.

Άσκηση 2 (Το κλασματικό knapsack πρόβλημα)

Θεωρήστε παραλλαγή του πιο πάνω προβλήματος όπου μπορούμε να επιλέξουμε ένα κλάσμα ενός αντικειμένου (π.χ. μισό κιλό χρυσόσκονης). Ποια είναι η πιο κερδοφόρα επιλογή αντικειμένων;

- Να λύσετε το πρόβλημα με άπληστο αλγόριθμο.
- Να αποδείξετε την ορθότητα του αλγορίθμου σας και να συζητήσετε τη δυνατότητα ύπαρξης άπληστου αλγορίθμου για το 0-1 knapsack πρόβλημα (Άσκηση 1).

Λύση Άσκησης 1

Λύση με δυναμικό προγραμματισμό

Χρησιμοποιούμε ένα πίνακα $V[1..n, 1..W]$, με μια σειρά για κάθε αντικείμενο και μια στήλη για κάθε βάρος μεταξύ 1 και W. Στη θέση $V[i,j]$ του πίνακα θα αποθηκεύουμε τη μέγιστη αξία συλλογής αντικειμένων από το σύνολο $1..i$ της οποία το βάρος δεν ξεπερνά το βάρος j.

Έχουμε τη σχέση

$$V[i,j] = \begin{cases} v_i, & \text{if } i = 1, w_i \le j \\ \max(V[i-1,j], V[i-1, j-w_i] + v_i), & \text{if } i > 1, w_i \le j \\ V[i-1,j], & \text{if } i > 1, w_i > j \\ 0, & \text{otherwise} \end{cases}$$

Υπολογίζουμε το $V[n,W]$, ξεκινώντας από το $V[1,1],\ldots,V[1,W]$ και προχωρώντας στο $V[2,\_]$, $V[3,\_],\ldots$.

Παράδειγμα

Έστω τσάντα μέγιστου βάρους 11 και πέντε αντικείμενα βάρους 1, 2, 5, 6, 7, και αξίας 1, 6, 18, 22, 28 αντίστοιχα. Εκτέλεση του αλγόριθμου δίνει τον πιο κάτω πίνακα

V	1	2	3	4	5	6	7	8	9	10	11
1	1	1	1	1	1	1	1	1	1	1	1
2	1	6	7	7	7	7	7	7	7	7	7
3	1	6	7	7	18	19	24	25	25	25	25
4	1	6	7	7	18	22	24	28	29	29	40
5	1	6	7	7	18	22	28	29	34	35	40

Εργασία 1

Να γράψετε αναλυτικά τον αλγόριθμο λύσης του προβλήματος και να υπολογίσετε τον χρόνο εκτέλεσής του.

Λύση με αλγόριθμο οπισθοδρόμησης

Ένα φορτίο είναι *i*-υποσχόμενο αν
1. περιέχει *i* αντικείμενα και
2. το βάρος του δεν ξεπερνά τη τιμή *W*.

Θεωρώντας ότι ξεκινούμε με το 0-υποσχόμενο διάνυσμα είναι δυνατό να κτίσουμε όλα τα 1-υποσχόμενα διανύσματα, 2-υποσχόμενα διανύσματα και ούτω καθεξής.

Με δεδομένο ένα *i*-υποσχόμενο διάνυσμα υπάρχουν δύο περιπτώσεις
1. είτε αυτό μπορεί να γίνει (*i*+1)-υποσχόμενο διάνυσμα με την προσθήκη ενός αντικειμένου,
2. είτε έχουμε φτάσει σε αδιέξοδο, δηλαδή σε μια τοποθέτηση στην οποία οποιαδήποτε προσθήκη αντικειμένου συνεπάγεται συγκέντρωση βάρους > *W*.

Στη δεύτερη περίπτωση
1. σημειώνουμε την αξία του συγκεντρωμένου φορτίου (για μελλοντικές συγκρίσεις) και
2. εφαρμόζουμε οπισθοδρόμηση.

Εργασία 2

Να γράψετε αναλυτικά τον αλγόριθμο λύσης του προβλήματος και να υπολογίσετε τον χρόνο εκτέλεσής του.

Λύση Άσκησης 2

Θέτουμε $a[i]=v[i]/w[i]$ για κάθε αντικείμενο. Υποθέτουμε ότι τα αντικείμενα είναι ταξινομημένα σε φθίνουσα σειρά ως προς την τιμή a. Αν όχι τα ταξινομούμε με ένα από τους γνωστούς αλγόριθμους ταξινόμησης.

Ακολουθεί άπληστος αλγόριθμος για το πρόβλημα:

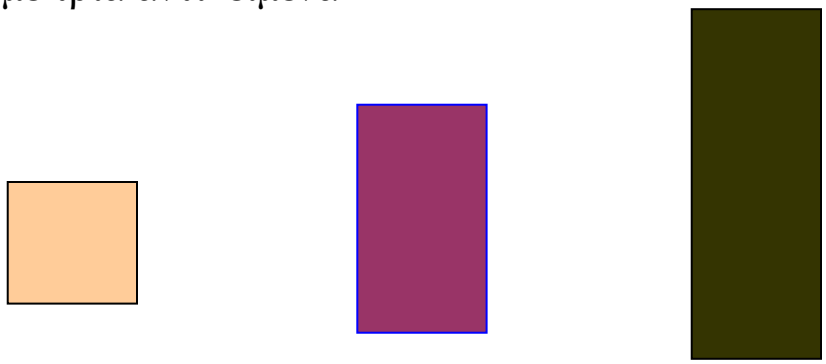
```
i =1;
w =0;
A = ∅;
while (i<=n && w!= W){
     if (w+w[i] <= W )
          A = A∪{(i,w[i])};
          w = w + w[i];
     else
          A = A∪{(i, W-w)};
          w = W;
          i=i+1;
}
```

Ο αλγόριθμος επιλέγει αντικείμενα βάσει του κριτηρίου ποιο έχει τον μεγαλύτερο λόγο αξίας ανά βάρος. Όταν φθάσει σε κάποιο αντικείμενο του οποίου το βάρος είναι μεγαλύτερο από την υπολειπόμενη χωρητικότητα του σάκου, επιλέγει μόνο μέρος αυτού.

Εργασία 2
Να αποδείξετε την ορθότητα του αλγορίθμου.

Ακολουθεί αντιπαράδειγμα για την καταλληλότητα του αλγορίθμου στην περίπτωση του προβλήματος 0-1 knapsack:

Θεωρούμε τρία αντικείμενα



w[1] = 10	w[2] = 20	w[3] = 30
v[1] = 60	v[2] = 80	v[3] = 90
a[1] = 6	a[2] = 4	a[3] = 3

και τσάντα με W = 50.

Ο άπληστος αλγόριθμος επιλέγει τα δύο πρώτα αντικείμενα.
Συνολικό κέρδος: 140

Η βέλτιστη λύση όμως περιέχει τα δύο τελευταία αντικείμενα.