ΑΛΓΟΡΙΘΜΟΙ

• ΦΡΟΝΤΙΣΤΗΡΙΟ 8 •

Άσκηση 1 / 7 (Δυναμικός Προγραμματισμός)

Δίνεται ένα σύνολο n δραστηριοτήτων. Κάθε δραστηριότητα έχει έναρξη και λήξη που δίνονται από το ζεύγος (s_i, f_i) και αξία v_i. Να επιλέξεται κάποιες από αυτές έτσι ώστε να μην επικαλύπτονται και να μεγιστοποιείται η συνολική αξία.

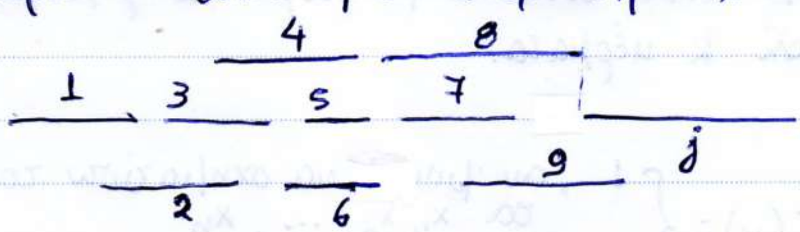
Λύση

Δραστηριότητα $i \to (s_i, f_i), v_i$

Ορίζω ως $p(j)$ (εκφράζει μια προηγούμενη δραστηριότητα, όσο πιο κοντά στο j) το μεγαλύτερο δείκτη $i<j$ τέτοιο ώστε η δραστηριότητα i να ΜΗΝ επικαλύπτεται με τη j. Τυπικά, δηλαδή, βρίσκουμε το μεγαλύτερο f_i που είναι μικρότερο από το s_j.

Θέλουμε να ξέρουμε εάν φιξάρουμε μία δραστηριότητα είναι, από τις υπόλοιπες, εκείνη που τελειώνει πιο αργά.

Ορίζω ως $OPT(j)$ τη μέγιστη δυνατή αξία που μπορώ να πετύχω αν έχω διαθέσιμες τις j δραστηριότητες, τότε αν η λύση περιέχει την j-οστή δραστηριότητα παίρνω αξία v_j συν ό,τι καλύτερο μπορώ να πάρω με τις δραστηριότητες $1, 2, \dots, p(j)$. Διαφορετικά η βέλτιστη αξία που μπορώ να πάρω είναι αυτή που μπορώ να πετύχω με τις δραστηριότητες $1, 2, \dots, j-1$.



$p(j) = 8$

$$OPT(j) = \max\{ v_j + OPT(p(j)), OPT(j-1) \}$$

Αλγόριθμος:

$O(n\log n)$ Ταξινομώ τις δραστηριότητες κατά αύξουσα σειρά του f_i

$O(n)$ Υπολογίζω τον πίνακα p

Για κάθε j από 1 μέχρι n κάνω

$O(n)$ $OPT(j) = \max\{v_j + OPT(p(j)), OPT(j-1)\}$

Επιστρέφω το μέγιστο $OPT(j)$

Πολυπλοκότητα: $O(n\log n)$ (λόγω της ταξινόμησης).

Άσκηση 2/7 (Δυναμικός Προγραμματισμός)

Δίνονται οι υποδιαιρέσεις (αξίες κερμάτων) $x_1, x_2, \dots, x_n$ κάποιου νομίσματος και ένα ποσό V. Δώστε αλγόριθμο δυναμικού προγραμματισμού που αποφασίζει κατά πόσο είναι δυνατό να σχηματίσουμε το ποσό V με αυτά τα κέρματα στις ακόλουθες περιπτώσεις και υπολογίστε την πολυπλοκότητά του:

α. Από κάθε x_i μπορούμε να χρησιμοποιήσουμε όσα κέρματα θέλουμε.

β. Κάθε x_i μπορούμε να το χρησιμοποιήσουμε το πολύ μία φορά.

γ. Από κάθε x_i μπορούμε να χρησιμοποιήσουμε όσα κέρματα θέλουμε αλλά συνολικά μπορούμε να χρησιμοποιήσουμε το πολύ k κέρματα.

Λύση

α. Έστω

$$C(v) = \begin{cases} 1, & \text{αν μπορώ να σχηματίσω το } V \text{ από τα } x_1, x_2, \dots, x_n \\ 0, & \text{διαφορετικά} \end{cases}$$

Αν μπορώ να σχηματίσω το V, θα μπορώ να σχηματίσω τουλάχιστον 1 από τα ποσά $V-x_1$ ή $V-x_2$ ή... ή $V-x_n$. Άρα η $C(V)$ θα είναι το λογικό OR (ή max) των $C(V-x_1)$, $C(V-x_2)$, ..., $C(V-x_n)$, οπότε η αναδρομική εξίσωση που χρειαζόμαστε είναι:

$$C(V) = \max_{1\le i\le n}\{C(V-x_i)\} \text{ με } C(0)=1 \text{ και } C(V)=0 \ \forall V<0$$

Αλγόριθμος:

```
C(0) = 1
for i=1 to V do
    C(i) = max_{1≤j≤n} { C(i - x_j) }
return C(V)
```

Πολυπλοκότητα $O(nV)$.

B. Έστω $C(V,i) = \begin{cases} 1, & \text{αν μπορώ να σχηματίσω το } V \text{ από τα } x_1, x_2, \dots, x_i \\ 0, & \text{διαφορετικά.} \end{cases}$

πρέπει να εκφράσω τον περιορισμό ότι από κάθε αξία μπορώ να πάρω μόνο ένα κέρμα /συγκεκριμένο σύνολο νομισμάτων.

Είναι: $C(V,i) = \max_{1\le i\le n}\{C(V-x_i, i-1), C(V, i-1)\}$,

με $C(0,i)=1 \ \forall i$ και $C(V,0)=0, \ \forall V\neq 0$
και $C(V,i)=0 \ \forall V<0$.

Σε αυτόν τον αλγόριθμο θα γεμίσω δισδιάστατο πίνακα

Αλγόριθμος:

Αρχικοποιώ τη πρώτη γραμμή και τη πρώτη στήλη πίνακα C διαστάσεων $(n+1)\times(V+1)$.

for $j=1$ to V:
  for $i=1$ to n
    $C(j,i) = \max\{C(j-x_i, i-1), C(j, i-1)\}$
return $C(V,n)$.

Πολυπλοκότητα: $O(nV)$

γ. Έστω $C(V,k) = \begin{cases} 1, & \text{αν μπορώ να σχηματίσω το } V \text{ από τα } x_1, x_2, \dots, x_n \text{ με το πολύ } k \text{ κέρματα} \\ 0, & \text{διαφορετικά} \end{cases}$

Αν μπορώ να σχηματίσω το V από τα $x_1, x_2, \dots, x_n$ με k το πολύ κέρματα τότε τουλάχιστον ένα από τα ποσά $V-x_1, V-x_2, \dots, V-x_n$ θα σχηματίζεται με $k-1$ το πολύ κέρματα. Άρα:

$$C(V,k) = \max_{1\le j\le n}\{C(V-x_j, k-1)\} \quad \text{με} \quad \begin{aligned} &C(0,k)=1 \ \forall k\ge 0 \\ &C(V,0)=0 \ \forall V\ne 0 \\ &C(V,k)=0 \ \forall V<0 \end{aligned}$$

Αλγόριθμος:
  $c(0,j) = 1 \quad \forall j\in\{0,1,\dots,k\}$
  $c(i,0) = 0 \quad \forall i\in\{1,2,\dots,V\}$
  for $i=1$ to V do
    for $j=1$ to k do
      $C(i,j) = \max_{1\le t\le n}\{C(i-x_t, k-1)\}$

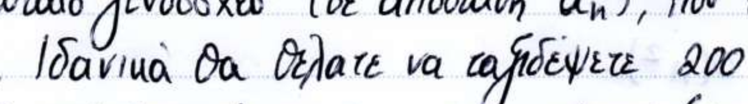
  return $C(V,k)$

Πολυπλοκότητα: $O(nVk)$

Άσκηση 3/7 (Δυναμικός Προγραμματισμός)

Πρόκειται να κάνετε ένα μεγάλο ταξίδι, ξεκινώντας από τη χιλιομετρική θέση 0. Στη διαδρομή του ταξιδιού υπάρχουν n ξενοδοχεία, στις χιλιομετρικές θέσεις $a_1 < a_2 < \dots < a_n$, όπου κάθε a_i μετριέται από το σημείο εκκίνησης. Οι μόνες θέσεις όπου επιτρέπεται να σταματήσετε είναι αυτά τα ξενοδοχεία, αλλά μπορείτε να επιλέξετε ποια θα είναι αυτά. Πρέπει οπωσδήποτε να σταματήσετε στο τελευταίο ξενοδοχείο (σε απόσταση a_n), που είναι ο προορισμός σας. Ιδανικά θα θέλατε να ταξιδεύετε 200 χιλιόμετρα ημερησίως αλλά αυτό μπορεί να μην είναι εφικτό (μιας και εξαρτάται από τις αποστάσεις των ξενοδοχείων μεταξύ τους). Αν ταξιδεύετε x χιλιόμετρα στη διάρκεια μιας ημέρας, η ποινή για την συγκεκριμένη ημέρα θα είναι $(200-x)^2$. Θέλετε να προγραμματίσετε το ταξίδι σας έτσι ώστε να ελαχιστοποιήσετε τη συνολική ποινή, δηλαδή το άθροισμα των ημερησίων ποινών για όλες τις ημέρες του ταξιδιού. Δώστε έναν αποδοτικό αλγόριθμο ο οποίος να προσδιορίζει τη βέλτιστη ακολουθία των ξενοδοχείων στα οποία πρέπει να σταματήσετε.

Λύση



$x\ km \rightarrow$ κόστος $(200-x)^2$

Θεωρώ μία στάση i και το υποπρόβλημα της εύρεσης της βέλτιστης λύσης όταν βρίσκομαι στην i είναι:

$$D(i) = \min_{0 \leq j \leq i} \{ D(j) + (200 - (a_i - a_j))^2 \}, \text{ με } D(0)=0$$

Αλγόριθμος:

```
D(0) = 0
For i=1 to n do{
```

```
    min ← ∞
    for j=0 to i-1 do{
        t ← D(j) + (200-(a_i - a_j))^2
        if t<min then
            min ← t }
    D(i) ← min}
return D(n)
```

Πολυπλοκότητα : $O(n^2)$.

Άσκηση 4/7 (Δυναμικός Προγραμματισμός)

Διαθέτουμε budget K και υπάρχουν διαθέσιμα αντικείμενα με αξίες $a_1, a_2, \ldots, a_n$. Ποια είναι η μέγιστη αξία ενός συνόλου αντικειμένων που μπορούμε να επιλέξουμε χωρίς να ξεπερνά το budget μας;

α. Προτείνετε έναν άπληστο αλγόριθμο για το πρόβλημα και δώστε ένα αντιπαράδειγμα που να δείχνει ότι ο αλγόριθμός σας δεν είναι σωστός.

β. Δώστε αλγόριθμο δυναμικού προγραμματισμού για το πρόβλημα και υπολογίστε την πολυπλοκότητά του.

γ. Πώς θα άλλαζε ο αλγόριθμος ή/και η πολυπλοκότητά του αν υπήρχαν διαθέσιμα b αντίγραφα από κάθε αντικείμενο;

δ. Τι θα κάναμε αν έπρεπε να επιστρέψουμε και το βέλτιστο σύνολο εκτός από τη βέλτιστη τιμή;

Λύση

Το πρόβλημα μοιάζει με το πρόβλημα του σακιδίου με χωρητικότητα ίση με K και βάρη αντικειμένων ίσα με τις αξίες τους

α. Άπληστη επιλογή: Πάρε σε κάθε βήμα το ακριβότερο αντικείμενο που μπορείς να αγοράσεις. Ο αλγόριθμος ΔΕΝ είναι σωστός.

Αναπαράδειγμα:

→ Έστω αντικείμενα με αξίες 30, 20, 20 και budget $K=40$.

Greedy → 1ο αντικείμενο → αξία = 30

OPT → 2ο και 3ο αντικείμενο → αξία = 20 + 20 = 40.

→ Έστω αντικείμενα με αξίες 4, 5, 8 και $K = 10$.

Greedy → αξία = 8

OPT → αξία = 5 + 4 = 9.

β. Ορίζω ως υποπρόβλημα $OPT(i,j)$ τη μέγιστη αξία που μπορώ να πετύχω αν είναι διαθέσιμα τα αντικείμενα $a_1, a_2, \dots, a_i$ και διαθέτω budget j. Το ζητούμενο είναι $OPT(n,k)$.

γ. Θεωρώ ℓ αντικείμενα με αξία a_1, ℓ αντικείμενα με αξία a_2 κ.ο.κ και εφαρμόζω τον παραπάνω αλγόριθμο. Ο πίνακας που θα υπολογίσω έχει γραμμές για κάθε αντικείμενο a_i, άρα η πολυπλοκότητα είναι $\Theta(n\ell k)$.

δ. Κάνουμε ό,τι ακριβώς κάναμε για το πρόβλημα Knapsack που είδαμε στο μάθημα (Διάλεξη 13).