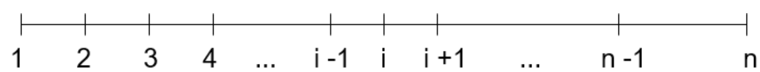
Λύσεις ασκήσεων φυλλαδίου 8 αλγορίθμων

Εαρινό 2020

Άσκηση 1

Κόστος εγκατάστασης σταθμού στην $i = C[i] \geq 0, 1 \leq i \leq n$.

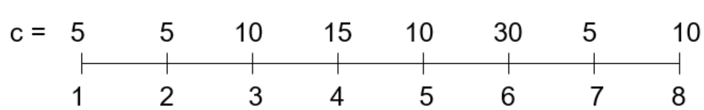


α′ Άπληστο κριτήριο: Επιλέξτε τις ακμές με άρτιο label για να τοποθετήσεις σταθμό (έστω c το κόστος ενός σταθμού). Αλγόριθμος:

1) Αν υπάρχει μόνο μία πόλη, τοποθέτησε τον σταθμό και επέστρεψε c.
2) Επίλεξε τη δεύτερη κατά σειρά πόλη από την αρχή του δρόμου και τοποθέτησε σταθμό.
3) Διέγραψε την προηγούμενη και την τρέχουσα πόλη.
4) Επανάλαβε τα βήματα 2 και 3 μέχρις ότου να μη μπορείς να επιλέξεις άλλη πόλη.
5) Επίστρεψε το συνολικό κόστος: $k \cdot c$, όπου k το πλήθος των σταθμών που τοποθετήθηκαν.

Ο αλγόριθμος είναι ορθός γιατί δεν αφήνει να υπάρξουν δύδο διαδοχικές πόλεις και καμία να μην έχει σταθμό. Ορθότητα: Ο αλγόριθμος είναι βέλτιστος. Αυτό είναι προφανές, καθώς, αν τοποθετούνταν λιγότεροι σταθμοί 2 διαδοχικές πόλεις θα έμεναν ακάλυπτες. Επίσης αν επιλέγονταν οι περιττές πόλεις, τότε θα τοποθετούνταν περισσότεροι σταθμοί αν το σύνολο των πόλεων ήταν περιττού πλήθους

β′



Ο άπληστος αλγόριθμος του ερωτήματος α θα επιστρέψει συνολικό κόστος $5+15+30+10 = 60$, αφού θα τοποθετήσει σταθμό στις πόλεις $2, 4, 6, 8$. Αυτή η λύση, όμως, δε βρίσκει βέλτιστη λύση, καθώς μπορούμε να επιλέξουμε τις $1, 3, 5, 7$ ή $2, 3, 5, 7$ με κόστος $30 < 60$.

γ′ Έχουμε ότι η λύση του αρχικού προβλήματος είναι η ελάχιστη τιμή των εναλλακτικών:
(+) αν έχει τοποθετηθεί σταθμό στην τελευταία πόλη
(-) αν δεν τοποθετείται σταθμός στην τελευταία.
Δηλαδή:

$$OPT(n) = min\{OPT^{+}(n), OPT^{-}(n)\}.$$

Ορίζω:

$$(1) OPT^{+}(i) = min\{C[i] + OPT^{+}(i-1), C[i] + OPT^{-}(i-1)\}$$

και

$$OPT^{-}(i) = OPT^{+}(i-1).$$

Άρα

$$OPT(i) = min\{OPT^{+}(i), OPT^{-}(i)\},$$

θέτω, επίσης, $OPT^{+}(0) = OPT^{-}(0) = OPT(0) = 0$. Η σχέση (1) προκύπτει από το γεγονός ότι επιλέγουμε το ελάχιστο κόστος μεταξύ των:

- κόστος σταθμού στην $i + Optimal$ κόστος της $i-1$ αν κι εκεί έχει εγκατασταθεί σταθμός.
- κόστος σταθμού στην $i + Optimal$ κόστος της $i-1$ αν εκεί δεν έχει εγκατασταθεί σταθμός.

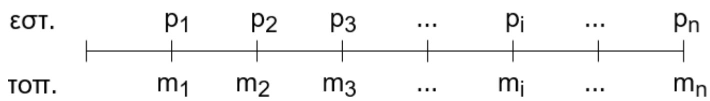
Η (2) είναι προφανής αφού δε βάλαμε σταθμό στον i θα πάρουμε το *optimal* κόστος που έχει η $i-1$ (προηγούμενη πόλη) η οποία επιβάλεται να έχει σταθμό.

δ′ Αλγόριθμος:

1) Ξεκινάω από την αρχή του δρόμου με κατεύθυνση προς τα δεξιά. Κόστος αφετηρίας $= 0$.
2) for i = 1 to n{
   $OPT^{+}[i] = min\{C[i] + OPT^{+}[i-1], C[i] + OPT^{-}[i-1]\}$
   $OPT^{-}[i] = OPT^{+}[i-1]$
   $OPT[i] = min\{OPT^{+}[i], OPT^{-}[i]\}$
   }
3) return $OPT[n]$

Πολυπλοκότητα: $O(n)$
Για κάθε στοιχείο του πίνακα απαιτείται να το εξετάσω 1 μόνο φορά και να κάνω σταθερό πλήθος πράξεων. Γραμμική πολυπλοκότητα $O(n) \cdot c$, άρα $O(n)$.

Άσκηση 2



Ορίζω το υποπρόβλημα $D(i)$ ως το μέγιστο κέρδος που μπορεί να αποκομίσει η εταιρεία από τις τοποθεσίες 1 εώς i.
Είναι $D(i) = max\{D(i-1), p_i + D(i^*)\}$, όπου i^* είναι ο μεγαλύτερος δείκτης j τέτοιος ώστε $m_j \leq m_i - k$. Δηλαδή η πρώτη τοποθεσία που βρίσκεται πριν την i και απέχει τουλάχιστον k μίλια από αυτήν.
Πολυπλοκότητα: $O(nlogn)$.
Αφού λύνω n υποπροβλήματα και για το καθένα βρίσκω το i^* σε $O(logn)$ με δυαδική αναζήτηση.

Άσκηση 3

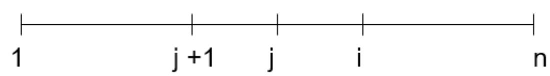
$s[1, ..., n]$

$$dict(w) = \begin{cases} true, & \text{αν } w \text{ έγκυρη λέξη.} \\ false, & \text{διαφορετικά.} \end{cases} \tag{1}$$

Ορίζω μία ακολουθία υποπροβλημάτων $s(i)$ για $0 \leq i \leq n$, όπου:

$$s(i) = \begin{cases} 1, & \text{αν } s[1, ..., i] \text{ έγκυρη ακολουθία λέξεων.} \\ 0, & \text{διαφορετικά.} \end{cases} \tag{2}$$

Είναι $S(i) = max_{0 \leq j \leq i}\{S(j) : dict(S[j+1, ..., i]) = true\}$ με $S(0) = 1$.

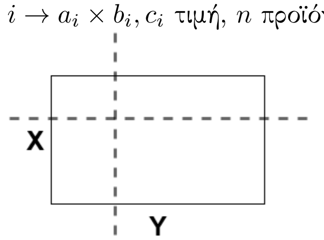


S	1			...		...	
	0	1	2		i		n

Συμπληρώνω τον πίνακα. Η ακολουθία S μπορεί να ανακατασκευαστεί ως ακολουθία έγκυρων λέξεων αν και μόνο αν $S(n) = 1$. Πολυπλοκότητα: $O(n^2$.

Άσκηση 4

$i \rightarrow a_i \times b_i, c_i$ τιμή, n προϊόντα



Ορίζω X, Y υποπροβλήματα. Για $1 \leq i \leq X$ και $1 \leq j \leq Y$ έστω $C(i, y)$ το μέγιστο κέρδος που μπορώ να πετύχω από κομμάτι υφάσματος διάστασης $i \times j$).
Ορίζω επίσης:

$$rect(i,j) = \begin{cases} max\{c_k\}, & \text{για όλα τα προϊόντα } k \text{ με } a_k = i \text{ και } b_k = j. \\ 0, & \text{αν δεν υπάρχει προϊόν με αυτές τις διαστάσεις.} \end{cases} \tag{3}$$

$C(i, j) = max\{max_{1 \leq k \leq i}\{C(k, j) + C(i-k, j)\}, max_{1 \leq h \leq j}\{C(i, h) + C(i, j-h)\}, rect(i, j)\}$

Αρχικοποίηση: $C(1, j) = max\{0, rect(1, j)\}$, $C(i, 1) = max\{0, rect(i, 1)\}$.
Επιστρέφουμε την τιμή $C(X, Y)$.