Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

2η Σειρά Προτεινόμενων Ασκήσεων

Άσκηση 1: Το Πρόβλημα της Βιβλιοθήκης

Σε ένα φοιτητή δίνεται ένα υποχρεωτικό πρόγραμμα μελέτης για τις επόμενες T ημέρες. Με βάση το πρόγραμμα, ο φοιτητής διαβάζει ένα μόνο μάθημα κάθε ημέρα. Ο συνολικός αριθμός των μαθημάτων είναι n, $n < T$. Ο φοιτητής διαβάζει κάποια μαθήματα για περισσότερες από μία και όχι κατ' ανάγκη συνεχόμενες ημέρες (π.χ. 1η μέρα Αλγόριθμοι, 2η Γλώσσες, 3η Βάσεις, 4η Αλγόριθμοι, 5η Γλώσσες, 6η Αλγόριθμοι, 7η Αλγόριθμοι, 8η Βάσεις, 9η Βάσεις, κλπ.).

Η μελέτη κάθε μαθήματος απαιτεί το δανεισμό ενός συγκεκριμένου βιβλίου από τη βιβλιοθήκη. Ο κανονισμός δεν επιτρέπει στο φοιτητή να έχει "χρεωμένα" περισσότερα από k, $k < n$, βιβλία ταυτόχρονα. Επιπλέον, ο κανονισμός δεν επιτρέπει τον δανεισμό περισσότερων του ενός βιβλίου την ίδια ημέρα. Έτσι κάθε ημέρα που ο φοιτητής χρειάζεται ένα βιβλίο και δεν το έχει, πρέπει να πηγαίνει στη βιβλιοθήκη και να το δανείζεται (μόνο αυτό, δεν μπορεί να δανειστεί άλλα βιβλία την ίδια ημέρα). Αν μάλιστα ο φοιτητής έχει ήδη "χρεωμένα" k βιβλία, πρέπει να διαλέξει ποιο βιβλίο θα επιστρέψει. Κάθε επίσκεψη στη βιβλιοθήκη αποσπά τον φοιτητή από τη μελέτη του. Το ζητούμενο λοιπόν είναι η πολιτική επιστροφής βιβλίων που ελαχιστοποιεί τον αριθμό των επισκέψεων στη βιβλιοθήκη.

Να διατυπώσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο / πολιτική επιστροφής βιβλίων που εξασφαλίζει τον ελάχιστο αριθμό επισκέψεων στη βιβλιοθήκη, και να αποδείξετε ότι ο αλγόριθμός σας πράγματι υπολογίζει τη βέλτιστη λύση. Ποιος είναι ο χρόνος εκτέλεσης του αλγορίθμου σας; Μπορείτε να συνδέσετε το πρόβλημα της βιβλιοθήκης με κάποια (σημαντική) πρακτική εφαρμογή;

Λύση. Πρόκειται για το πρόβλημα επιλογής των σελίδων που διατηρούνται στην "γρήγορη" μνήμη σε υπολογιστές με ιεραρχία μνήμης (το πρόβλημα είναι γνωστό ως *paging* ή *cache maintenance*). Η βέλτιστη πολιτική είναι είναι να επιστρέφουμε το βιβλίο που θα χρειαστούμε τελευταίο (από αυτά που έχουμε). Δείτε την Ενότητα 4.3, σελ. 164-171, στο βιβλίο των Kleinberg και Tardos. □

Άσκηση 2: Ανεφοδιασμός

Ένας οδηγός αποφασίζει να κάνει ένα ταξίδι από την πόλη x στην πόλη y με το αυτοκίνητό του, το οποίο έχει αυτονομία κίνησης ως προς τη βενζίνη που μπορεί να αποθηκεύσει, k χιλιόμετρα. Ο οδηγός διαθέτει χάρτη στον οποίο αναφέρονται όλα τα βενζινάδικα της διαδρομής και οι αποστάσεις μεταξύ τους, και επιθυμεί να υπολογίσει ένα πλάνο ανεφοδιασμού που ελαχιστοποιεί τον συνολικό αριθμό στάσεων για ανεφοδιαμό (δεδομένου βέβαια ότι μεταξύ δύο διαδοχικών ανεφοδιασμών διανύεται απόσταση που δεν ξεπερνά τα k χιλιόμετρα, ώστε το αυτοκίνητο να μην μείνει χωρίς βενζίνη). Να διατυπώσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο που υπολογίζει ένα βέλτιστο πλάνο ανεφοδιασμού. Να προσδιορίσετε την υπολογιστική πολυπλοκότητα και να αποδείξετε την ορθότητα του αλγορίθμου σας. *Σημείωση:* Μπορείτε να υποθέσετε ότι δύο διαδοχικά βενζινάδικα απέχουν μεταξύ τους το πολύ k χιλιόμετρα.

Λύση. Για να λύσουμε το πρόβλημα θα χρησιμοποιήσουμε την άπληστη μέθοδο. Ο άπληστος αλγόριθμος είναι ο εξής: ''Βάζω βενζίνη στο τελευταίο βενζινάδικο που μπορώ να φτάσω πριν μου τελειώσουν τα αποθέματα στο ντεπόζιτο''. Θα αποδείξουμε ότι αυτός ο αλγόριθμος δίνει τη βέλτιστη λύση (ουσιαστικά θα αποδείξουμε ότι το πρόβλημα έχει την ιδιότητα της *άπληστης επιλογής* ως προς το συγκεκριμένο κριτήριο).

Ας ονομάσουμε τον άπληστο αλγόριθμο GREEDY, και ας ονομάσουμε OPT κάποιον αλγόριθμο που υπολογίζει την βέλτιστη λύση. Συμβολίζουμε με GREEDY(n) και OPT(n) την απόσταση που διανύουμε από την αρχή της διαδρομής μέχρι τη n-οστή στάση όταν ακολουθούμε τον αλγόριθμο GREEDY και OPT αντίστοιχα. Με επαγωγή, θα δείξουμε ότι $\forall n \geq 0$, GREEDY$(n) \geq$ OPT(n). Δείτε ότι αυτό ουσιαστικά αποτελεί διατύπωση της ιδιότητας της άπληστης επιλογής, αφού εξασφαλίζει ότι υπάρχει μια βέλτιστη λύση που συμφωνεί με τις επιλογές του άπληστου αλγόριθμου.

- Ισχύει για $n = 0$, αφού GREEDY$(0) =$ OPT$(0) = 0$.
- Θα δείξουμε ότι $\forall n \geq 0$, αν GREEDY$(n) \geq$ OPT(n), τότε GREEDY$(n+1) \geq$ OPT$(n+1)$.
  Η βέλτιστη λύση που αντιστοιχεί στο OPT(n) είναι βέβαια εφικτή, άρα πρέπει να εξασφαλίζει βενζίνη για τη μετάβαση από τη στάση n στη στάση $n+1$. Συνεπώς,

$$\text{OPT}(n+1) - \text{OPT}(n) \leq k \Rightarrow \text{OPT}(n+1) - \text{GREEDY}(n) \leq k\,,$$

  λόγω της επαγωγικής υπόθεσης GREEDY$(n) \geq$ OPT(n). Δηλαδή ο GREEDY εξασφαλίζει εξασφαλίζει ότι καλύπτουμε την απόσταση OPT$(n+1)$, και ανεφοδιαζόμαστε (για $(n+1)$-οστή φορά) είτε στο ίδιο βενζινάδικο με τον OPT είτε σε κάποιο επόμενό του. Σε κάθε περίπτωση, GREEDY$(n+1) \geq$ OPT$(n+1)$.

Έστω τώρα ότι ο αλγόριθμος OPT λύνει το πρόβλημα κάνοντας m ενδιάμεσες στάσεις για ανεφοδιασμό. Αφού ο αλγόριθμος OPT είναι βέλτιστος, ο GREEDY κάνει τουλάχιστον m στάσεις. Παραπάνω αποδείξαμε ότι GREEDY$(m) \geq$ OPT(m). Αν λοιπόν d είναι η απόσταση μεταξύ των πόλεων x και y, έχουμε ότι

$$d - \text{OPT}(m) \leq k \Rightarrow d - \text{GREEDY}(m) \leq k$$

Άρα ο GREEDY εξασφαλίζει ότι έπειτα από τον βέλτιστο αριθμό m στάσεων, έχουμε αρκετή βενζίνη για να φτάσουμε στο τέλος της διαδρομής. □

Άσκηση 3: Ανεξάρτητα Σύνολα με Βάρη

Έστω $X = (x_1, \ldots, x_n)$ μία ακολουθία n θετικών ακεραίων. Το σύνολο δεικτών $I \subseteq \{1, \ldots, n\}$ ονομάζεται *ανεξάρτητο* αν για κάθε ζεύγος δεικτών $i, j \in I$, $|i - j| > 1$, δηλαδή το I δεν περιέχει διαδοχικούς δείκτες. Το *βάρος* $W(J)$ ενός συνόλου δεικτών $J \subseteq \{1, \ldots, n\}$ είναι το άθροισμα των αντίστοιχων x_i, δηλ. $W(J) = \sum_{i \in J} x_i$. Το ζητούμενο είναι ο αποδοτικός υπολογισμός ενός ανεξάρτητου συνόλου μέγιστου βάρους για δεδομένη ακολουθία X.

1. Έστω ότι χρησιμοποιούμε τον ακόλουθο άπληστο αλγόριθμο: αρχικά $I = \emptyset$ και $J = \{1, \ldots, n\}$. Ενόσω $J \neq \emptyset$, προσθέτουμε τον δείκτη $j \in J$ με τη μεγαλύτερη τιμή x_j στο I, και αφαιρούμε από το J τους δείκτες $j-1$, j, και $j+1$. Το αποτέλεσμα είναι το σύνολο I.
   (α) Βρείτε ένα απλό παράδειγμα για το οποίο ο παραπάνω άπληστος αλγόριθμος αποτυγχάνει να υπολογίσει τη βέλτιστη λύση.
   (β) Να δείξετε ότι το ανεξάρτητο σύνολο I που υπολογίζεται από τον παραπάνω άπληστο αλγόριθμο έχει βάρος τουλάχιστον το μισό του βάρους του βέλτιστου ανεξάρτητου συνόλου.

2. Να δείξετε ότι για τον υπολογισμό ενός ανεξάρτητου συνόλου μέγιστου βάρους ισχύει η αρχή της βελτιστότητας, και να διατυπώσετε αναδρομική σχέση που συνδέει τη βέλτιστη λύση ενός στιγμιότυπου με τη βέλτιστη λύση κατάλληλα επιλεγμένων επιμέρους στιγμιοτύπων του.
3. Να διατυπώσετε αλγόριθμο δυναμικού προγραμματισμού με χρονική πολυπλοκότητα $\mathrm{O}(n)$ που υπολογίζει ένα ανεξάρτητο σύνολο μέγιστου βάρους.

Λύση. (1.α) Για οσοδήποτε μικρό $\varepsilon > 0$, θεωρούμε το σύνολο $X = \{1, 1+\varepsilon, 1\}$, για το οποίο η άπληστη λύση έχει βάρος $1+\varepsilon$ και η βέλτιστη λύση έχει βάρος 2.

(1.β) Διαισθητικά, έστω αντικείμενο x_i που περιλαμβάνεται στην άπληστη, αλλά όχι στην βέλτιστη, λύση. Η επιλογή του x_i από τον άπληστο αλγόριθμο προκαλεί (την διαγραφή από το J και έτσι) τον "αποκλεισμό" από την άπληστη λύση *κανενός ή περισσοτέρων* από τα "γειτονικά" του x_{i-1} και x_{i+1} (βλ. ότι κάποιο(α) από τα x_{i-1} και x_{i+1} μπορούν να έχουν ήδη διαγραφεί από το J πριν την επιλογή του x_i). Όμως κάθε στοιχείο που "αποκλείεται" εκείνη τη στιγμή λόγω του x_i έχει βάρος μικρότερο ή ίσο του βάρους του x_i [1]. Επομένως, το "ισοζύγιο βάρους" ανάμεσα στα στοιχεία που περιλαμβάνονται στην άπληστη λύση, αλλά όχι στην βέλτιστη λύση, και στα στοιχεία που "αποκλείονται" λόγω αυτών είναι 1 προς 2 ή καλύτερο. Με αυτόν τον τρόπο, ο άπληστος αλγόριθμος εξασφαλίζει ότι το συνολικό του βάρος είναι τουλάχιστον το μισό του συνολικού βάρους του βέλτιστου ανεξάρτητου συνόλου.

Τυπικά, έστω ότι ο άπληστος αλγόριθμος επιστρέφει το σύνολο $I = \{x_{g_1}, \ldots, x_{g_k}\}$, ενώ η βέλτιστη λύση είναι το σύνολο I', που μπορεί να διαφέρει από το I. Για κάθε στοιχείο x_{g_i} που επιλέγεται από τον άπληστο αλγόριθμο, δημιουργούμε ένα σύνολο G_i που περιλαμβάνει τα στοιχεία που διαγράφονται από το J κατά την επιλογή του x_{g_i}. Δηλαδή είναι $G_i = \{x_{g_i-1}, x_{g_i}, x_{g_i+1}\}$, αν τα x_{g_i-1} και x_{g_i+1} ανήκουν στο J κατά την επιλογή του x_{g_i}, $G_i = \{x_{g_i-1}, x_{g_i}\}$ (αντιστ. $G_i = \{x_{g_i}, x_{g_i+1}\}$), αν μόνο το x_{g_i-1} (αντιστ. το x_{g_i+1}) ανήκει στο J κατά την επιλογή του x_{g_i}, και $G_i = \{x_{g_i}\}$, αν κανένα από τα x_{g_i-1}, x_{g_i+1} δεν ανήκει στο J κατά την επιλογή του x_{g_i}. Λόγω του κριτηρίου τερματισμού του άπληστου αλγόριθμου, τα σύνολα G_i αποτελούν μια διαμέριση του X. Επιπλέον, το συνολικό βάρος των στοιχείων του συνόλου $G_i \setminus \{x_{g_i}\}$ δεν ξεπερνά το διπλάσιο του x_{g_i} (δηλ. αν $G_i = \{x_{g_i-1}, x_{g_i}, x_{g_i+1}\}$, έχουμε ότι $x_{g_i-1} + x_{g_i+1} \leq 2x_{g_i}$, αντίστοιχα και στις άλλες περιπτώσεις).

Για κάθε στοιχείο x_{g_i} που επιλέγεται από τον άπληστο αλγόριθμο, θεωρούμε το σύνολο $G'_i = I' \cap G_i$ που περιλαμβάνει τα στοιχεία του G_i που ανήκουν στην βέλτιστη λύση. Παρατηρούμε ότι το συνολικό βάρος $W(G'_i)$ των στοιχείων κάθε συνόλου G'_i δεν υπερβαίνει το $2x_{g_i}$. Επιπλέον, επειδή τα σύνολα G_i αποτελούν μια διαμέριση του X, τα σύνολα G'_i αποτελούν μια διαμέριση της βέλτιστης λύσης. Έτσι έχουμε ότι:

$$W(I') = \sum_{i=1}^{k} W(G'_i) \leq \sum_{i=1}^{k} 2x_{g_i} = 2W(I)$$

(2) Η αρχή της βελτιστότητας ισχύει γιατί αν η ακολουθία $x_{o_1}, \ldots, x_{o_k}$, $o_1 < \cdots < o_k$, αποτελεί βέλτιστη λύση για το σύνολο $X = \{x_1, \ldots, x_n\}$, κάθε υπακολουθία $x_{o_1}, \ldots, x_{o_j}$ με $o_j = q$ αποτελεί βέλτιστη λύση για το υποσύνολο $X' = \{x_1, \ldots, x_q\}$.

Με βάση αυτή την παρατήρηση, θεωρούμε τα υποσύνολα $X_i = \{x_1, \ldots, x_i\}$ που αποτελούνται από τα πρώτα i στοιχεία του X, για κάθε $i = 0, \ldots, n$. Βέβαια είναι $X_0 = \emptyset$ και $X_n = X$.

[1] Αν ισχύει ότι $x_{i-1} > x_i$ (αντίστ. ότι $x_{i+1} > x_i$), το x_{i-1} (αντίστ. το x_{i+1}) πρέπει να έχει διαγραφεί από το J στο παρελθόν, λόγω της επιλογής του x_{i-2} (αντίστ. του x_{i+2}). Διαφορετικά ο άπληστος αλγόριθμος δεν θα επέλεγε το x_i. Στο x_i "χρεώνουμε" λοιπόν τον "αποκλεισμό" μόνο των στοιχείων που διαγράφονται από το J την στιγμή που το x_i επιλέγεται από τον άπληστο αλγόριθμο. Καθένα από αυτά τα στοιχεία (το πολύ 2) έχει βάρος μικρότερο ή ίσο του x_i.

Έστω $S(X_i)$ το συνολικό βάρος της βέλτιστης λύσης για το υποσύνολο X_i. Υποθέτουμε ότι γνωρίζουμε τα $S(X_{n-1})$ και $S(X_{n-2})$, και εστιάζουμε στην επιλογή του στοιχείου x_n στην βέλτιστη λύση. Μπορούμε είτε να επιλέξουμε το x_n, αποκλείοντας αναγκαστικά το x_{n-1}, οπότε "κερδίζουμε" συνολικό βάρος $S(X_{n-2}) + x_n$, είτε να αγνοήσουμε το x_n, οπότε "κερδίζουμε" συνολικό βάρος $S(X_{n-1})$. Προφανώς επιλέγουμε το καλύτερο από τα δύο! Με άλλα λόγια, δείξαμε ότι $S(X_n) = \max\{S(X_{n-1}), S(X_{n-2})+x_n\}$. Γενικεύοντας, καταλήγουμε ότι το βέλτιστο συνολικό βάρος $S(X_n)$ υπολογίζεται από την αναδρομική σχέση:

$$S(X_i) = \begin{cases} \max\{S(X_{i-1}), S(X_{i-2}) + x_i\} & \text{όταν } i \geq 2 \\ x_1 & \text{όταν } i = 1 \\ 0 & \text{όταν } i = 0 \end{cases}$$

(3) Μπορούμε να υπολογίσουμε το μέγιστο βάρος ενός ανεξάρτητου συνόλου $X(S_n)$ καθώς και ένα ανεξάρτητο σύνολο μέγιστου βάρους από την παραπάνω αναδρομική σχέση σε χρόνο $\Theta(n)$, με δυναμικό προγραμματισμό. □

Άσκηση 4: Αλυσίδα Εστιατορίων

Ως υπεύθυνος δικτύου μιας αλυσίδας εστιατορίων, πρέπει να επιλέξετε που θα ανοίξουν καταστήματα κατά μήκος της νέας εθνικής οδού. Έχουν προεπιλεγεί n υποψήφιες θέσεις, και για κάθε υποψήφια θέση i, $i = 1, \ldots, n$, έχει υπολογισθεί το αναμενόμενο κέρδος από το κατάστημα στη θέση i με βάση το αν ανοίξουν καταστήματα στις γειτονικές θέσεις. Αν δεν ανοίξει κατάστημα σε καμία από τις θέσεις $i-1$ και $i+1$, το αναμενόμενο κέρδος από το κατάστημα στην θέση i είναι a_i, αν ανοίξει κατάστημα σε μία από τις θέσεις $i-1$ και $i+1$, το αναμενόμενο κέρδος είναι b_i, και αν ανοίξουν καταστήματα σε αμφότερες τις θέσεις $i-1$ και $i+1$, το αναμενόμενο κέρδος είναι c_i (τα c_1 και c_n δεν ορίζονται, και για κάθε θέση i, ισχύει ότι $a_i \geq b_i \geq c_i \geq 0$). Να διατυπώσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο, που με είσοδο τις τριάδες (a_i, b_i, c_i), $i = 1, \ldots, n$, επιλέγει τις θέσεις όπου θα ανοίξουν καταστήματα ώστε να μεγιστοποιηθεί το κέρδος.

Λύση. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό (αφήνεται ως άσκηση να διατυπώσετε και να αιτιολογήσετε την αρχή της βελτιστότητας για αυτό το πρόβλημα). Για κάθε $i = 1, \ldots, n$, θεωρούμε το υποσύνολο $A_i = \{1, \ldots, i\}$ που αποτελείται από τις πρώτες i θέσεις. Θα διατυπώσουμε μια αναδρομική σχέση που συνδέει το βέλτιστο κέρδος από τις θέσεις στο σύνολο A_i με το βέλτιστο κέρδος από τις θέσεις στο σύνολο A_{i-1}. Η διαφορά σε σχέση με τα προηγούμενα προβλήματα είναι ότι το επιπλέον κέρδος από το κατάστημα στη θέση i εξαρτάται από το αν ανοίγουν καταστήματα στις θέσεις $i-1$ και $i+1$. Για αυτό διακρίνουμε τρεις διαφορετικές περιπτώσεις σε σχέση με το άνοιγμα καταστημάτων στις θέσεις i και $i+1$, και υπολογίζουμε διαφορετικά το βέλτιστο κέρδος για καθεμία από αυτές. Συγκεκριμένα, έστω ότι:

- $P_0(i)$ είναι το βέλτιστο κέρδος από τις θέσεις στο σύνολο A_i δεδομένου ότι δεν ανοίγει κατάστημα στη θέση i.
- $P_{1,0}(i)$ είναι το βέλτιστο κέρδος από τις θέσεις στο σύνολο A_i δεδομένου ότι ανοίγει κατάστημα στη θέση i και δεν ανοίγει κατάστημα στη θέση $i+1$.
- $P_{1,1}(i)$ είναι το βέλτιστο κέρδος από τις θέσεις στο σύνολο A_i δεδομένου ότι ανοίγει κατάστημα τόσο στη θέση i όσο και στη θέση $i+1$.

Για $i = 1$, προφανώς έχουμε ότι:

$$P_0(1) = 0, \quad P_{1,0}(1) = a_1, \quad P_{1,1}(1) = b_1$$

Για $i \geq 2$, τα $P_0(i)$, $P_{1,0}(i)$, και $P_{1,1}(i)$ υπολογίζονται αναδρομικά από τα $P_0(i-1)$, $P_{1,0}(i-1)$, και $P_{1,1}(i-1)$ ως εξής:

$$\begin{aligned} P_0(i) &= \max\{P_0(i-1), P_{1,0}(i-1)\} \\ P_{1,0}(i) &= \max\{a_i + P_0(i-1), b_i + P_{1,1}(i-1)\} \\ P_{1,1}(i) &= \max\{b_i + P_0(i-1), c_i + P_{1,1}(i-1)\} \end{aligned}$$

Το μέγιστο κέρδος είναι $\max\{P_0(n), P_{1,0}(n)\}$. Το μέγιστο κέρδος και η βέλτιστη λύση μπορούν να υπολογιστούν με δυναμικό προγραμματισμό σε χρόνο $\Theta(n)$. □

Άσκηση 5: Αντιπροσωπεία Φορητών Υπολογιστών

Επιθυμούμε να βελτιστοποιήσουμε την λειτουργία μιας νέας αντιπροσωπείας φορητών υπολογιστών για τις επόμενες n ημέρες. Για κάθε ημέρα i, $1 \leq i \leq n$, υπάρχει (ασφαλής) πρόβλεψη d_i του αριθμού των πωλήσεων. Όλες οι πωλήσεις λαμβάνουν χώρα το μεσημέρι, και όσοι φορητοί υπολογιστές δεν πωληθούν, αποθηκεύονται. Υπάρχει δυνατότητα αποθήκευσης μέχρι S υπολογιστών, και το κόστος είναι C για κάθε υπολογιστή που αποθηκεύεται και για κάθε ημέρα αποθήκευσης. Το κόστος μεταφοράς για την προμήθεια νέων υπολογιστών είναι K ευρώ, ανεξάρτητα από το πλήθος των υπολογιστών που προμηθευόμαστε, και οι νέοι υπολογιστές φθάνουν λίγο πριν το μεσημέρι (άρα αν πωληθούν αυθημερόν, δεν χρειάζονται αποθήκευση). Αρχικά δεν υπάρχουν καθόλου υπολογιστές στην αντιπροσωπεία.

Το ζητούμενο είναι να προσδιορισθούν οι παραγγελίες (δηλ. πόσους υπολογιστές θα παραγγείλουμε και πότε) ώστε να ικανοποιηθούν οι προβλεπόμενες πωλήσεις με το ελάχιστο δυνατό συνολικό κόστος (αποθήκευσης και μεταφοράς). Να διατυπώσετε αλγόριθμο με χρόνο εκτέλεσης πολυωνυμικό στο nS για αυτό το πρόβλημα. Να αιτιολογήσετε την ορθότητα και την πολυπλοκότητα του αλγορίθμου σας.

Λύση. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Για την αρχή της βελτιστότητας, θεωρούμε μια βέλτιστη ακολουθία παραγγελιών $(p_1, \ldots, p_{n-1}, p_n)$ για n ημέρες, όπου μετά το τέλος της n-οστής ημέρας δεν υπάρχουν αδιάθετοι υπολογιστές. Αν $d_n > S$, τότε η μόνη τιμή που μπορεί να πάρει το p_n σε μια βέλτιστη λύση είναι d_n (όλοι οι υπολογιστές που πωλούνται την n-οστή ημέρα παραγγέλνονται την ίδια ημέρα με κόστος K). Σε αυτή την περίπτωση, η υπακολουθία $(p_1, \ldots, p_{n-1})$ είναι μια βέλτιστη πολιτική παραγγελιών για τις $n-1$ πρώτες ημέρες, όπου στο τέλος της $(n-1)$-οστής ημέρας δεν υπάρχουν αδιάθετοι υπολογιστές. Αν $d_n \leq S$, το p_n μπορεί να είναι είτε d_n (όταν $K \leq d_n C$), οπότε ισχύει ότι και παραπάνω, είτε 0 (όταν $K > d_n C$), οπότε η υπακολουθία $(p_1, \ldots, p_{n-1})$ είναι μια βέλτιστη πολιτική παραγγελιών για τις $n-1$ πρώτες ημέρες, υπό την προϋπόθεση ότι στο τέλος της ημέρας $n-1$ υπάρχουν d_n αδιάθετοι υπολογιστές (που θα αποθηκευθούν με κόστος $d_n C$).

Έστω λοιπόν $\text{cost}[i, s]$, $1 \leq i \leq n$, $0 \leq s \leq S$, το ελάχιστο κόστος παραγγελιών για τις πρώτες i ημέρες, υπό την προϋπόθεση ότι στο τέλος της ημέρας i υπάρχουν s διαθέσιμοι υπολογιστές (το κόστος αποθήκευσής τους δεν συμπεριλαμβάνεται στο $\text{cost}[i, s]$). Σύμφωνα με τα παραπάνω, το $\text{cost}[i, s]$ δίνεται από την αναδρομική σχέση:

$$\text{cost}[i, s] = \begin{cases} \min\{\text{cost}[i-1, s+d_i] + (s+d_i)C, \text{cost}[i-1, 0] + K\} & \text{αν } i \geq 2 \text{ και } s + d_i \leq S \\ \text{cost}[i-1, 0] + K & \text{αν } i \geq 2 \text{ και } s + d_i > S \\ K & \text{αν } i = 1 \end{cases}$$

Το βέλτιστο κόστος δίνεται από το $\text{cost}[n, 0]$, το οποίο υπολογίζεται από την παραπάνω εξίσωση σε χρόνο $O(nS)$. Η ακολουθία των παραγγελιών μπορεί να υπολογισθεί στον ίδιο χρόνο ανατρέχοντας στις επιλογές που οδήγησαν στον υπολογισμό του βέλτιστου κόστους $\text{cost}[n, 0]$. □

Άσκηση 6: Κοπή Υφασμάτων (DPV 6.14)

Δίνεται ένα ορθογώνιο κομμάτι υφάσματος με διαστάσεις $X \times Y$, όπου X και Y θετικοί ακέραιοι, και μια λίστα με n προϊόντα τα οποία μπορούν να κατασκευασθούν με την χρήση του υφάσματος. Για κάθε προϊόν i γνωρίζουμε ότι χρειάζεται ένα ορθογώνιο κομμάτι υφάσματος διαστάσεων $a_i \times b_i$, και ότι η τελική τιμή πώλησης είναι c_i (τα a_i, b_i, c_i είναι θετικοί ακέραιοι, αν σας διευκολύνει, μπορείτε να υποθέσετε ότι δεν υπάρχουν προϊόντα i, j με $a_i \geq a_j, b_i \geq b_j$, και $c_i \leq c_j$). Κάθε προϊόν μπορεί να δημιουργηθεί σε οσαδήποτε αντίγραφα (μπορεί και κανένα). Έχουμε στη διάθεσή μας μια μηχανή που μπορεί να κόψει οποιοδήποτε ορθογώνιο κομμάτι υφάσματος σε δύο κομμάτια, είτε οριζόντια είτε κατακόρυφα (κάθε κόψιμο ακολουθεί μια ευθεία κάθετη προς τις πλευρές του υφάσματος και δεν μπορεί να διακοπεί). Να διατυπώσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο που προσδιορίζει την βέλτιστη εκμετάλλευση του υφάσματος, δηλ. υπολογίζει μια στρατηγική κοπής ώστε να μεγιστοποιηθεί το συνολικό κέρδος από την πώληση των προϊόντων που προκύπτουν από τα κομμάτια του υφάσματος. Να αιτιολογήσετε την ορθότητα και να προσδιορίσετε την χρονική πολυπλοκότητα του αλγορίθμου σας.

Λύση. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Για την αρχή της βελτιστότητας, θεωρούμε μια βέλτιστη λύση που κόβει το ύφασμα π.χ. στην θέση k, $1 \leq k < X$, στην οριζόντια διάσταση. Τότε η βέλτιστη εκμετάλλευση του υφάσματος προκύπτει αν θεωρήσουμε την βέλτιστη εκμετάλλευση των δύο τμημάτων υφάσματος που προκύπτουν (ένα με διαστάσεις $k \times Y$ και ένα με διαστάσεις $(X - k) \times Y$). Μάλιστα η βέλτιστη εκμετάλλευση για τα δύο τμήματα υπολογίζεται ανεξάρτητα.

Έτσι το βέλτιστο κέρδος $G(x, y)$, $1 \leq x \leq X$, $1 \leq y \leq Y$, που μπορούμε να έχουμε από ένα κομμάτι υφάσματος με διαστάσεις $x \times y$ δίνεται από την αναδρομική σχέση:

$$G(x,y) = \begin{cases} 0 & \text{αν για κάθε } i \in [n],\ a_i > x \vee b_i > y \\ \max\{\ \max_{1 \leq k < x}\{G(k,y) + G(x-k,y)\}, & \\ \qquad \max_{1 \leq k < y}\{G(x,k) + G(x,k-y)\}, & \text{διαφορετικά} \\ \qquad \max_{i: a_i \leq x \wedge b_i \leq y}\{c_i\}\ \} & \end{cases}$$

Το βέλτιστο κέρδος $G(X, Y)$ υπολογίζεται από την παραπάνω αναδρομική εξίσωση σε χρόνο $O(X \cdot Y \cdot (n + X + Y))$ με εφαρμογή δυναμικού προγραμματισμού. □

Άσκηση 7: Δρομολόγηση Εργασιών

Θεωρούμε σύνολο εργασιών $A = \{1, \ldots, n\}$ που είναι υποψήφιες να εκτελεστούν σε έναν υπολογιστή. Κάθε εργασία j έχει χρόνο εκτέλεσης t_j λεπτά και προθεσμία εκτέλεσης d_j (θεωρούμε ότι οι χρόνοι εκτέλεσης και οι προθεσμίες είναι θετικοί φυσικοί αριθμοί και ότι $t_j \leq d_j$ για κάθε j). Η εκτέλεση μιας εργασίας j αποφέρει κέρδος p_j, εφόσον ολοκληρωθεί εμπρόθεσμα, και κέρδος 0, διαφορετικά.

Ο υπολογιστής μπορεί να εκτελεί μόνο μία εργασία κάθε χρονική στιγμή. Η εκτέλεση μιας εργασίας j απαιτεί την δέσμευση του υπολογιστή για χρονικό διάστημα t_j λεπτών χωρίς διακοπή. Ένα σύνολο εργασιών $F \subseteq A$ καλείται *εφικτό* αν οι εργασίες του F μπορούν να δρομολογηθούν ώστε να ολοκληρωθούν όλες εμπρόθεσμα.

(α) Να δείξετε ότι για κάθε εφικτό σύνολο εργασιών F, μια μέθοδος δρομολόγησης που εξασφαλίζει ότι όλες οι εργασίες του F ολοκληρώνονται εμπρόθεσμα είναι η δρομολόγηση τους σε αύξουσα σειρά προθεσμιών (αγνοώντας τις εργασίες εκτός F).

(b) Να διατυπώσετε έναν όσο το δυνατόν πιο αποδοτικό αλγόριθμο που υπολογίζει ένα εφικτό σύνολο εργασιών μέγιστου κέρδος. Να αιτιολογήσετε την ορθότητα και να προσδιορίσετε την χρονική πολυπλοκότητα του αλγορίθμου σας.

Λύση. (α) Το ζητούμενο αποδεικνύεται με ένα επιχείρημα ανταλλαγής. Έστω F ένα εφικτό σύνολο που περιλαμβάνει k εργασίες. Αγνοούμε τις άλλες εργασίες του A, και αριθμούμε τις εργασίες του F σε αύξουσα σειρά προθεσμιών $d_1 \leq d_2 \leq \cdots \leq d_k$.

Έστω $\sigma' = (i_1, \ldots, i_k)$ μια δρομολόγηση των εργασιών του F όπου όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα. Ισχύει δηλαδή ότι για κάθε $i_j \in F$, $\sum_{\ell=1}^{j} t_{i_\ell} \leq d_{i_j}$. Μια τέτοια δρομολόγηση υπάρχει, γιατί το F είναι εφικτό, αλλά δεν συμφωνεί κατ' ανάγκη με την δρομολόγηση των εργασιών του F σε αύξουσα σειρά προθεσμιών. Τροποποιώντας την $(i_1, \ldots, i_k)$ όπου είναι απαραίτητο, θα δείξουμε ότι και στην δρομολόγηση $\sigma = (1, \ldots, k)$ σε αύξουσα σειρά προθεσμιών, όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα.

Έστω p, $1 \leq p < k$, η πρώτη θέση στην οποία η δρομολόγηση $\sigma = (1, \ldots, k)$ διαφέρει από την $\sigma' = (i_1, \ldots, i_k)$. Έχουμε λοιπόν ότι για $\ell = 1, \ldots, p-1$, $\ell = i_\ell$. Στην θέση p όμως, η σ δρομολογεί την εργασία p, ενώ η σ' δρομολογεί την εργασία $i_p > p$, και δρομολογεί την εργασία p στην θέση $q > p$ (δηλ. $i_q = p$). Επειδή οι εργασίες δρομολογούνται σε αύξουσα σειρά προθεσμιών στην σ, και επειδή οι δρομολογήσεις σ και σ' ταυτίζονται στις πρώτες $p-1$ θέσεις τους, οι εργασίες στις θέσεις $p, \ldots, q$ της σ' έχουν προθεσμία μεγαλύτερη ή ίση του d_p. Δηλαδή για $\ell = p, \ldots, q$, $d_p \leq d_{i_\ell}$.

Θα τροποποιήσουμε την σ', και θα φτιάξουμε μια εφικτή δρομολόγηση που ταυτίζεται με την σ στις πρώτες p θέσεις. Αφού η δρομολόγηση σ' είναι εφικτή, η εργασία p ολοκληρώνεται εμπρόθεσμα, δηλ. ισχύει ότι $\sum_{\ell=1}^{q} t_{i_\ell} \leq d_p$. Ανταλλάσσουμε τις p και i_p στην δρομολόγηση σ'. Δρομολογούμε δηλαδή την εργασία p στην θέση p (όπως στην σ) και την εργασία i_p στην θέση q. Και στη νέα δρομολόγηση, η εργασία p και οι εργασίες $i_p, \ldots, i_{q-1}$ ολοκληρώνονται όχι αργότερα από τη χρονική στιγμή $\sum_{\ell=1}^{q} t_{i_\ell} \leq d_p$. Αφού για κάθε $\ell = p, \ldots, q$, $d_p \leq d_{i_\ell}$, η εργασία p και οι εργασίες $i_p, \ldots, i_{q-1}$ ολοκληρώνονται εμπρόθεσμα. Παρατηρούμε ακόμη ότι ο χρόνος ολοκλήρωσης των υπόλοιπων εργασιών (δηλ. των εργασιών $1, \ldots, p-1$ και $i_{q+1}, \ldots, i_k$) δεν επηρεάζεται από αυτή την ανταλλαγή. Άρα και στη νέα δρομολόγηση, όλες οι εργασίες ολοκληρώνονται εμπρόθεσμα.

(β) Για τη διατύπωση της λύσης, (ταξινομούμε και) αριθμούμε τις εργασίες σε αύξουσα σειρά με βάση τις προθεσμίες τους, δηλ. $d_1 \leq d_2 \leq \cdots \leq d_n$. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Αρχικά διατυπώνουμε την αρχή της βελτιστότητας: αν η ακολουθία $(i_1, \ldots, i_k)$, με $i_1 < \cdots < i_k$ λόγω του (α), αποτελεί βέλτιστη λύση για το σύνολο εργασιών $A = \{1, \ldots, n\}$ και χρόνο ολοκλήρωσης της τελευταίας εργασίας D, η υπακολουθία $(i_1, \ldots, i_{k-1})$ αποτελεί βέλτιστη λύση για το υποσύνολο εργασιών $A' = \{1, \ldots, i_{k-1}\}$ και χρόνο ολοκλήρωσης της τελευταίας εργασίας $\min\{d_{i_{k-1}}, D - t_k\}$.

Θεωρούμε λοιπόν τα υποσύνολα $A_i = \{1, \ldots, i\}$ που αποτελούνται από τις πρώτες i εργασίες στο σύνολο A, για κάθε $i = 0, \ldots, n$ (προφανώς είναι $A_0 = \emptyset$ και $A_n = A$). Έστω $P(A_i, D)$, $0 \leq D \leq d_i$, το βέλτιστο κέρδος από το υποσύνολο εργασιών A_i με χρόνο ολοκλήρωσης D για την τελευταία εργασία. Υποθέτουμε ότι γνωρίζουμε τα $P(A_{n-1}, d_{n-1})$ και $P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\})$, και εστιάζουμε στην εργασία n. Λόγω του (α), αν η εργασία n επιλεγεί, θα εκτελεστεί τελευταία. Μπορούμε είτε να επιλέξουμε την εργασία n, και να έχουμε κέρδος $p_n + P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\})$, είτε να μην επιλέξουμε την εργασία n, και να έχουμε κέρδος $P(A_{n-1}, d_{n-1})$. Προφανώς επιλέγουμε

το καλύτερο από τα δύο. Με άλλα λόγια:

$$P(A_n, d_n) = \max\{p_n + P(A_{n-1}, \min\{d_n - t_n, d_{n-1}\}), P(A_{n-1}, d_{n-1})\}$$

Γενικεύοντας, καταλήγουμε ότι το βέλτιστο κέρδος $P(A_n, d_n)$ υπολογίζεται από την αναδρομική σχέση:

$$P(A_i, D) = \begin{cases} \max\{p_i + P(A_{i-1}, D - t_i), P(A_{i-1}, D)\} & \text{όταν } i \geq 1 \text{ και } 1 \leq D \leq d_i \\ P(A_i, d_i) & \text{όταν } i \geq 1 \text{ και } d_i < D \\ 0 & \text{όταν } i = 0 \text{ ή } D \leq 0 \end{cases}$$

Ο χρόνος για τον υπολογισμό του βέλτιστου κέρδους $P(A_n, d_n)$ είναι $\Theta(n \log n)$ για την ταξινόμηση των εργασιών σε αύξουσα σειρά με βάση τις προθεσμίες τους, και $\Theta(nd_n)$ για τον υπολογισμό (με δυναμικό προγραμματισμό) των τιμών της παραπάνω αναδρομικής σχέσης και των εργασιών που εκτελούνται, δηλαδή $\Theta(n \log n + nd_n)$ συνολικά. □

Άσκηση 8: Sponsored Search Auctions

Σε κάθε ερώτημα, μια μηχανή αναζήτησης (βλ. Google, Yahoo, κλπ.) επιλέγει k από ένα σύνολο n διαφημίσεων, $k < n$, τις ταξινομεί σε μία λίστα k θέσεων, τη μία κάτω από την άλλη, και τις εμφανίζει στα δεξιά των αποτελεσμάτων της αναζήτησης. Οι διαφημιζόμενοι πληρώνουν για να εμφανιστούν σε αυτές τις θέσεις, και σε αυτό οφείλεται το μεγαλύτερο μέρος του τζίρου των μηχανών αναζήτησης.

Η μηχανή αναζήτησης επιλέγει ποιες διαφημίσεις θα εμφανιστούν σε ποιες θέσεις γνωρίζοντας την "αξία" $v_i \in \mathbb{N}$ κάθε διαφήμισης i, την "ποιότητά" της $q_i \in [0, 1]$, και την "τάση" $c_i \in [0, 1]$ που δημιουργεί στον χρήστη να συνεχίσει με την επόμενη διαφήμιση στη λίστα.

Ειδικότερα, υποθέτουμε ότι ο χρήστης εξετάζει τη λίστα $(i_1, \ldots, i_k)$ των διαφημίσεων σύμφωνα με το παρακάτω μοντέλο, όπου οι θέσεις της λίστας αριθμούνται από πάνω προς τα κάτω:

1. Ο χρήστης ξεκινάει εξετάζοντας τη διαφήμιση i_1 στην πρώτη θέση.
2. Όταν ο χρήστης εξετάζει τη διαφήμιση i_j στην j θέση της λίστας, συνεχίζει στο site του διαφημιζόμενου (κάνοντας click στη διαφήμιση i_j) με πιθανότητα q_{i_j}. Σε αυτή την περίπτωση, η μηχανή αναζήτησης έχει κέρδος v_{i_j}.
3. Ανεξάρτητα του αν έκανε click στη διαφήμιση i_j ή όχι, ο χρήστης συνεχίζει εξετάζοντας τη διαφήμιση i_{j+1} στη θέση $j + 1$ της λίστας με πιθανότητα c_{i_j}. Διαφορετικά (δηλ. με πιθανότητα $1 - c_{i_j}$), ο χρήστης εγκαταλείπει την λίστα.
4. Σε κάθε περίπτωση, ο χρήστης εγκαταλείπει την λίστα όταν έχει εξετάσει τη διαφήμιση i_k στη θέση k.

Με βάση το παραπάνω μοντέλο, η μηχανή αναζήτησης επιλέγει τις διαφημίσεις και τη σειρά που θα εμφανιστούν ώστε να μεγιστοποιήσει το αναμενόμενο κέρδος. Πιο συγκεκριμένα, με βάση το παραπάνω μοντέλο, η πιθανότητα ότι ο χρήστης εξετάζει τη διαφήμιση i_j στη θέση j της λίστας, δεδομένου ότι οι διαφημίσεις $i_1, \ldots, i_{j-1}$ βρίσκονται πριν από την i_j στη λίστα, είναι $q_{i_j} \prod_{\ell=1}^{j-1} c_{i_\ell}$. Έτσι η μηχανή αναζήτησης επιλέγει τη λίστα διαφημίσεων $(i_1, \ldots, i_k)$ που μεγιστοποιεί την ποσότητα:

$$\sum_{j=1}^{k} v_{i_j} q_{i_j} \prod_{\ell=1}^{j-1} c_{i_\ell} ,$$

(α) Να δείξετε ότι η βέλτιστη λύση ταξινομεί τις διαφημίσεις σε φθίνουσα σειρά του λόγου $\frac{vq}{1-c}$. Δηλαδή στη βέλτιστη λύση έχουμε:

$$\frac{v_{i_1} q_{i_1}}{1-c_{i_1}} \geq \frac{v_{i_2} q_{i_2}}{1-c_{i_2}} \geq \cdots \geq \frac{v_{i_k} q_{i_k}}{1-c_{i_k}}$$

(β) Να διατυπώσετε αλγόριθμο που υπολογίζει τη βέλτιστη λύση σε χρόνο $\mathrm{O}(n \log n + nk)$.

Λύση. (α) Θα δείξουμε ότι αν οι διαφημίσεις δεν είναι σε φθίνουσα σειρά ως προς $\frac{vq}{1-c}$, μπορούμε να βελτιώσουμε το συνολικό κέρδος εναλλάσοντας τις θέσεις τους. Για την απόδειξη, θα χρησιμοποιήσουμε απαγωγή σε άτοπο.

Αναλυτικά, έστω μια βέλτιστη λύση, στην οποία οι διαφημίσεις δεν εμφανίζονται σε φθίνουσα σειρά ως προς το $\frac{vq}{1-c}$, και έστω ℓ ο μικρότερος δείκτης για τον οποίο ισχύει ότι

$$\frac{v_{i_\ell} q_{i_\ell}}{1-c_{i_\ell}} < \frac{v_{i_{\ell+1}} q_{i_{\ell+1}}}{1-c_{i_{\ell+1}}}$$

΄Εστω $C(\ell) = \prod_{j=1}^{\ell-1} c_{i_j}$. Αν αντιμεταθέσουμε τις διαφημίσεις i_ℓ και $i_{\ell+1}$, το συνολικό κέρδος μεταβάλλεται κατά:

$$\begin{aligned} C(\ell)[v_{i_\ell} q_{i_\ell} + v_{i_{\ell+1}} q_{i_{\ell+1}} c_{i_\ell} - v_{i_{\ell+1}} q_{i_{\ell+1}} - v_{i_\ell} q_{i_\ell} c_{i_{\ell+1}}] = \\ = C(\ell)[v_{i_\ell} q_{i_\ell}(1-c_{i_{\ell+1}}) + v_{i_{\ell+1}} q_{i_{\ell+1}}(c_{i_\ell} - 1)]\,, \end{aligned}$$

Αφού η αρχική λύση είναι βέλτιστη, το συνολικό κέρδος δεν βελτιώνεται από την αντιμετάθεση των διαφημίσεων i_ℓ και $i_{\ell+1}$, δηλαδή ισχύει ότι:

$$v_{i_\ell} q_{i_\ell}(1-c_{i_{\ell+1}}) + v_{i_{\ell+1}} q_{i_{\ell+1}}(c_{i_\ell} - 1) \geq 0$$

Κατά συνέπεια, έχουμε ότι:

$$\frac{v_{i_\ell} q_{i_l}}{1-c_{i_\ell}} \geq \frac{v_{i_{\ell+1}} q_{i_{\ell+1}}}{1-c_{i_{\ell+1}}}\,,$$

που έρχεται σε αντίφαση με την αρχική μας υπόθεση.

(β) Για τη διατύπωση της λύσης, υποθέτουμε ότι έχουμε ταξινομήσει τις διαφημίσεις σε φθίνουσα σειρά ως προς $\frac{vq}{1-c}$. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό. Αρχικά παρατηρούμε ότι ισχύει η αρχή της βελτιστότητας. Πράγματι, αν η ακολουθία $(i_1, \ldots, i_k)$, με $i_1 < \cdots < i_k$ λόγω του (α), αποτελεί βέλτιστη λύση για το σύνολο διαφημίσεων $A = \{1, \ldots, n\}$ και λίστα μήκους k, κάθε υπακολουθία $(i_j, \ldots, i_k)$ αποτελεί βέλτιστη λύση για το υποσύνολο διαφημίσεων $A' = \{i_j, \ldots, n\}$ και λίστα μήκους $k - j + 1$.

Θεωρούμε λοιπόν τα υποσύνολα $A_i = \{i, \ldots, n\}$ που αποτελούνται από τις τελευταίες $n - i + 1$ διαφημίσεις στο σύνολο A, για κάθε $i = n+1, \ldots, 1$ (προφανώς είναι $A_{n+1} = \emptyset$ και $A_1 = A$), και όλα τα δυνατά μήκη λίστας $\ell = 0, \ldots, k$. ΄Εστω $G(A_i, \ell)$ το βέλτιστο κέρδος από το υποσύνολο διαφημίσεων A_i με λίστα μήκους ℓ. Υποθέτουμε ότι γνωρίζουμε τα $G(A_2, k-1)$ και $G(A_2, k)$, και εστιάζουμε στην επιλογή της διαφήμισης 1. Λόγω του (α), αν η διαφήμιση 1 επιλεγεί στη λίστα, θα εμφανιστεί στην πρώτη θέση. Μπορούμε είτε να επιλέξουμε την διαφήμιση 1, και να έχουμε κέρδος $v_1 q_1 + c_1 G(A_2, k-1)$, είτε να μην επιλέξουμε την διαφήμιση 1 και να έχουμε κέρδος $G(A_2, k)$. Προφανώς επιλέγουμε το καλύτερο από τα δύο! Με άλλα λόγια, δείξαμε ότι

$G(A_1, k) = \max\{G(A_2, k), v_1 q_1 + c_1 G(A_2, k-1)\}$. Γενικεύοντας, καταλήγουμε ότι το βέλτιστο κέρδος $G(A_1, k)$ υπολογίζεται από την αναδρομική σχέση:

$$G(A_i, \ell) = \begin{cases} \max\{G(A_{i+1}, \ell), v_i q_i + c_i G(A_{i+1}, \ell - 1)\} & \text{όταν } i \leq n \text{ και } \ell > 0 \\ 0 & \text{όταν } i = n+1 \text{ ή } \ell = 0 \end{cases}$$

Ο χρόνος για τον υπολογισμό του βέλτιστου κέρδους $G(A_1, k)$ είναι $\Theta(n \log n)$ για την ταξινόμηση των διαφημίσεων και $\Theta(nk)$ για τον υπολογισμό των τιμών της παραπάνω αναδρομικής σχέσης και της λίστας διαφημίσεων με δυναμικό προγραμματισμό, δηλαδή $\Theta(n \log n + nk)$ συνολικά. □