Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Αλγόριθμοι και Πολυπλοκότητα
Διδάσκοντες: Σ. Ζάχος, Δ. Φωτάκης

2η Σειρά Γραπτών Ασκήσεων - Σχέδιο Λύσεων

Άσκηση 1: Βίδες και Παξιμάδια

Ακολουθούμε την λογική του αλγόριθμου Quicksort. Δεν μπορούμε βέβαια να εφαρμόσουμε την Quicksort αυτούσια, γιατί δεν μπορούμε να συγκρίνουμε τις βίδες με τις βίδες και τα παξιμάδια με τα παξιμάδια. Δηλαδή το πρόβλημα εντοπίζεται στην χρήση του pivot για τα δύο σύνολα (πίνακες). Όμως μπορούμε να παρακάμψουμε αυτό το πρόβλημα.

Διαλέγουμε στην τύχη ένα *παξιμάδι*, το οποίο θα χρησιμοποιηθεί σαν *pivot για το partition των βιδών* (αφού δεν μπορούμε να συγκρίνουμε τις βίδες μεταξύ τους, θα συγκρίνουμε τις βίδες με το παξιμάδι), και βρίσκουμε (σε γραμμικό χρόνο) την βίδα που του αντιστοιχεί, και η οποία θα χρησιμοποιηθεί σαν *pivot για το partition των παξιμαδιών*. Από το partition των δύο συνόλων με αυτά τα pivots (σε γραμμικό χρόνο), προκύπτουν δύο υποσύνολα με βίδες και τα αντίστοιχα υποσύνολα με τα παξιμάδια, στα οποία εφαρμόζουμε την ίδια διαδικασία αναδρομικά. Ο χρόνος εκτέλεσης είναι αντίστοιχος της Quicksort, δηλαδή $\Theta(n \log n)$ στη μέση περίπτωση.

Άσκηση 2: Ανεφοδιασμός

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

  $$\textsc{Opt}(n+1) - \textsc{Opt}(n) \leq k \Rightarrow \textsc{Opt}(n+1) - \textsc{Greedy}(n) \leq k\,,$$

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

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 $\text{GREEDY}(m) \geq \text{OPT}(m)$. Αν λοιπόν d είναι η απόσταση που διανύουμε συνολικά, έχουμε ότι

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

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

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.
2. (β) Κάθε αντικείμενο x_i που περιλαμβάνεται στην άπληστη αλλά όχι στην βέλτιστη λύση προκαλεί, την στιγμή που επιλέγεται από τον άπληστο αλγόριθμο, (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}. Όμως κάθε στοιχείο που "αποκλείεται" εκείνη τη στιγμή λόγω του x_i έχει βάρος μικρότερο ή ίσο του βάρους του x_i. Επομένως, το "ισοζύγιο βάρους" ανάμεσα στα στοιχεία που περιλαμβάνονται στην άπληστη λύση αλλά όχι στην βέλτιστη 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.
   Αναλυτικά, 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. Ο άπληστος αλγόριθμος σε κάθε επανάληψη εισάγει στο σύνολο I ένα στοιχείο x_{g_i} που έχει βάρος μεγαλύτερο από το βάρος όλων των υπόλοιπων στοιχείων που περιέχει εκείνη τη στιγμή το σύνολο X. Έστω x_{g_i-1} (αντιστ. x_{g_i+1}) το βάρος του στοιχείου x_{g_i-1} (αντιστ. x_{g_i+1}) εφόσον αυτό ανήκει στο X τη στιγμή που το x_{g_i} επιλέγεται στην άπληστη λύση, και 0 διαφορετικά. Ισχύει λοιπόν ότι $x_{g_i} \geq x_{g_i-1}$ και $x_{g_i} \geq x_{g_i+1}$, και συνεπώς

   $$W(\{x_{g_i-1}, x_{g_i+1}\}) = x_{g_i-1} + x_{g_i+1} \leq 2x_{g_i} = 2W(\{x_{g_i}\})$$

   Χωρίζουμε το σύνολο X σε τριάδες της μορφής $\{x_{g_i-1}, x_{g_i}, x_{g_i+1}\}$, με $1 \leq i \leq k$ (μπορεί να ισχύει ότι $x_{g_1-1} = 0$ ή $x_{g_k+1} = 0$, ενώ η τριάδα γίνεται δυάδα αν το g_i είναι 1 ή n). Παρατηρούμε ότι η ένωση αυτών των τριάδων είναι το σύνολο X, λόγω του κριτηρίου τερματισμού του άπληστου αλγόριθμου. Για κάθε στοιχείο x_{g_i} στην άπληστη λύση, έστω I'_i τα στοιχεία της τριάδας $\{x_{g_i-1}, x_{g_i}, x_{g_i+1}\}$ στη βέλτιστη λύση. Αυτά μπορεί να είναι:
   - είτε το x_{g_i-1}
   - είτε το x_{g_i}
   - είτε το x_{g_i+1}
   - είτε τα $\{x_{g_i-1}, x_{g_i+1}\}$

   Σε κάθε περίπτωση ισχύει ότι $W(I'_i) \leq 2W(\{x_{g_i}\})$. Συνολικά:

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

3. 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 $X_n = X$. 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:

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

4. Παρατηρούμε ότι ο υπολογισμός του $S(X_n)$ απαιτεί τον υπολογισμό των $S(X_{n-2})$ και $S(X_{n-1})$. Επεκτείνοντας τον συλλογισμό μας, διαπιστώνουμε ότι ο υπολογισμός της βέλτιστης λύσης απαιτεί τον υπολογισμό των $S(X_i)$ για κάθε $i = 0, \ldots, n$. Για αυτό χρησιμοποιούμε δυναμικό προγραμματισμό. Ειδικότερα, θεωρούμε έναν πίνακα A με $n + 1$ θέσεις, του οποίου τα στοιχεία συμπληρώνονται σε χρόνο $\Theta(n)$ με βάση την παραπάνω αναδρομική σχέση. Σε ψευδοκώδικα:

```
A[0] = 0;
A[1] = x_1;
for i = 2 to n do
    A[i] = max{A[i-1], A[i-2] + x_i};
```

Άσκηση 4: Sponsored Search Auctions

(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

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

Αν αντιμεταθέσουμε τις διαφημίσεις i_ℓ και $i_{\ell+1}$, το συνολικό κέρδος μεταβάλλεται κατά:

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

όπου $C(\ell) = \prod_{j=1}^{\ell-1} c_{i_j}$. 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 αποδεικνύουμε 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 $\ell = 0, \ldots, k$. Έστω $G(A_i, \ell)$ 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.