Μανιάτης Ανδρέας el18070 maniatis.andreas@gmail.com

1η σειρά γραπτών ασκήσεων

Άσκηση 1: Πλησιέστερο Ζεύγος Σημείων

(α) Στο πρόβλημα των 3 διαστάσεων γενικεύουμε τη μεθοδολογία χωρίζοντας τον 3D χώρο στα 2 με ένα επίπεδο. Έστω ότι γνωρίζουμε τις x, y, z συντεταγμένες χωρίζουμε με ένα επίπεδο το οποίο έχει σταθερή συντεταγμένη z, έστω το Z.
Βρίσκουμε αναδρομικά για τα δύο μισά τις μικρότερες αποστάσεις και καλούμε τη μικρότερη των 2 απόσταση δ.

Δημιουργούμε μια συμμετρική περιοχή γύρω από το Z, η οποία ορίζεται από εκατέρωθεν 2 (παράλληλα) επίπεδα τα οποία απέχουν το καθέτα απόσταση δ από το Z.
Αντίστοιχα, με τα τετράγωνα που δημιουργήσαμε στο 2D πρόβλημα τώρα δημιουργούμε κύβους κατάλληλης πλευράς, οι οποίοι περιέχουν το πολύ ένα σημείο.

Λόγω του δ χρειάζεται να υπολογίσουμε την απόσταση κάθε σημείου με συγκεκριμένο πάντα αριθμό σημείων. Θεωρούμε ότι οι αναδρομικές κλήσεις του αλγορίθμου επιστρέφουν ταξινομημένα τα σημεία, έστω στην κατεύθυνση y και άρα ξοδεύουμε γραμμικό χρόνο για τα πιθανά ζεύγη στο χώρο αυτό.
Θεωρώντας ότι οι αναδρομικές κλήσεις επιστρέφουν σωστές τιμές για τα 2 μισά και σύμφωνα με τη διαδικασία << ταιριάσματος >> ξέρουμε ότι παίρνουμε τη σωστή λύση και επομένως ο αλγόριθμος αποδεικνύεται επαγωγικά.
Ο γραμμικός χρόνος ταιριάσματος σε συνδυασμό με την αναδρομική διαίρεση σε 2 ίσα κομμάτια μας δίνει πολυπλοκότητα $O(n \log n)$.

(β) Κατασκευάζουμε ένα πλέγμα πλευράς ℓ και για κάθε κύβο (οποιωνδήποτε διαστάσεων) φτιάχνουμε μια λίστα όλων των σημείων που περιέχονται σε αυτόν.
Οι λίστες αυτές είναι ουσιαστικά buckets που προκύπτουν από μια κατάλληλη hashing συνάρτηση. Η κατασκευή αυτή γίνεται έτσι σε γραμμικό χρόνο.

Στη συνέχεια, βρίσκουμε για κάθε σημείο την απόσταση από τα σημεία της γειτονιάς του σε σταθερό χρόνο. Όπως αποδεικνύεται σε όλους τους σχετικούς αλγορίθμους «πλησιέστερου ζεύγους» η γειτονιά κάθε κύβου αποτελείται από σταθερό αριθμό κοντινότερων κύβων, λόγω κατασκευής τους.

Η προσέγγιση όμως είναι φραγμένη άρα για τη γειτονιά N που φτιάχνουμε ισχύει ότι:

- Όλα τα σημεία των οποίων η απόσταση από ένα σημείο είναι μικρότερη από $c\ell$ βρίσκονται στη γειτονιά του

Επομένως, μιλάμε όντως για περιορισμένο αριθμό αναζητήσεων. Και άρα και πιο σημαντικό:

- Καθένα στοιχείο πλησιέστερου ζεύγους θα βρίσκεται στη γειτονιά του άλλου

κάτι που εξασφαλίζει την ορθότητα του αλγορίθμου. Με βάση την παραπάνω ανάλυση η υπολογιστική πολυπλοκότητα είναι πράγματι $O(cn)$, αφού όσο μεγαλύτερο είναι το c τόσο αυξάνεται το μέγεθος της γειτονιάς και τα πεπερασμένα buckets που πρέπει να ψάξουμε.

Άσκηση 2: Πόρτες Ασφαλείας στο Κάστρο

Θεωρούμε ότι αρχίζουμε με όλους τους διακόπτες ανοιχτούς (θα μπορούσε οποιοσδήποτε άλλος τυχαίος συνδυασμός) δηλαδή έστω στο 0 (1 για κλειστό).
Ο αλγόριθμος που θα ακολουθήσουμε βασίζεται στη δυαδική αναζήτηση.

Αρχικά, πρέπει να βρούμε το διακόπτη και τη θέση για την πρώτη πόρτα.

Αντιστρέφουμε τους διακόπτες του πρώτου μισού, δηλαδή δοκιμάζουμε την ακολουθία:

$$\overbrace{1\;1\;\dots\;1}^{n/2}\;\overbrace{0\;0\;\dots\;0}^{n/2}$$

Αν η πρώτη πόρτα άνοιξε (αν ήταν κλειστή) ή αν έκλεισε (αν ήταν ανοιχτή) συνεχίζουμε την αναζήτηση στο αριστερό μισό αντιστρέφοντας ξανά τα πρώτα μισά, δηλαδή δοκιμάζουμε την ακολουθία

$$\overbrace{0\;0\;\dots\;0}^{n/4}\;\overbrace{1\;1\;\dots\;1}^{n/4}\;\overbrace{0\;\dots\;0}^{n/2}$$

Αν το κλείδωμά της δεν άλλαξε συνεχίζουμε στο δεξί μισό με την ακολουθία

$$\underbrace{1\;1\;\dots\;1}_{n/2}\;\underbrace{1\;\dots\;1}_{n/4}\;\underbrace{0\;\dots\;0}_{n/4}$$

Συνεχίζουμε αναδρομικά τη διαδικασία μέχρι να βρούμε τον διακόπτη και τη θέση του που ανοίγει την πρώτη πόρτα.
Επαναλαμβάνουμε τη διαδικασία για όλες τις πόρτες (n).
Λόγω χωρισμού των n διακοπτών στα 2 σε κάθε βήμα της αναζήτησης έχουμε χρόνο $\log(n)$ ο οποίος είναι και βέλτιστος όπως προκύπτει από το δένδρο αναζητήσεων.
Η διαδικασία γίνεται αναγκαστικά n φορές γιατί δεν μπορούμε να δούμε πίσω από πόρτες και άρα να χωρίσουμε το πρόβλημα σε υπο-προβλήματα. Άρα η πολυπλοκότητα είναι $O(n \log n)$.

Άσκηση 3 : Φόρτιση Ηλεκτρικών Αυτοκινήτων

Η στρατηγική που θα ακολουθήσουμε για την επίλυση του προβλήματος είναι δυαδική αναζήτηση στο χώρο λύσεων $\{1, \dots, n\}$, με σκοπό την εύρεση του μικρότερου δυνατού αριθμού υποδοχών φόρτισης.

Θα δείξουμε πώς μπορούμε να ελέγξουμε το αν ισχύει η απαίτηση κανένα αυτοκίνητο να μην περιμένει πάνω από d χρονικές μονάδες σε γραμμικό χρόνο.

- Το αυτοκίνητο με χρόνο άφιξης a_x συνδέεται στην υποδοχή στην οποία είχε συνδεθεί το αυτοκίνητο με χρόνο άφιξης a_{x-s}. Τα αυτοκίνητα με χρόνους άφιξης που παρεμβάλλονται μεταξύ αυτών των δύο θα συνδεθούν σε κάποια άλλη από τις $s-1$ υποδοχές αφού αυτές καταλήφθηκαν σε χρόνο προ του a_x και άρα θα ελευθερωθούν και πρώτες.

Για τον υπολογισμό του delay του a_x λοιπόν συγκρίνουμε το a_x με το $\text{delay} + a_{x-s} + 1$ και το αποθηκεύουμε. Τα πρώτα s αυτοκίνητα έχουν προφανώς μηδενικό delay. Έστω για την εξής αλληλουχία αφίξεων:

$\boxed{S=3}$

$1, 1, 1, 1, 1, 1, 1, 2, 4$

d $0, 0, 0, \boxed{1}, \boxed{1}, \boxed{1}$

$$\underset{a_1}{1} + \underset{\text{delay}}{0} + 1 - \underset{a_4}{1} = \underset{d_4}{1}$$

d $0, 0, 0, 1, 1, 1, \boxed{2}$

$$\underset{a_4}{1} + \underset{d_4}{1} + 1 - \underset{a_7}{1} = \underset{d_7}{2}$$

d $0, 0, 0, 1, 1, 1, 2, \boxed{1}$

$$\underset{a_5}{1} + \underset{d_5}{1} + 1 - \underset{a_8}{2} = \underset{d_8}{1}$$

d $0, 0, 0, 1, 1, 1, 2, 1, \boxed{0}$

$$\underset{a_6}{1} + \underset{d_6}{1} + 1 - 4 = -1 \rightarrow 0$$

αφού το delay ≥ 0.

Άρα λοιπόν ο έλεγχος της συνθήκης απαιτεί γραμμικό χρόνο και η δυαδική αναζήτηση χρειάζεται λογαριθμικό αριθμό βημάτων άρα η υπολογιστική πολυπλοκότητα του αλγορίθμου είναι $O(n \log n)$.

Άσκηση 4: Παραλαβή Πακέτων

1) Θα λύσουμε το πρόβλημα χρησιμοποιώντας άπληστο αλγόριθμο. Συγκεκριμένα, ταξινομούμε τα κλάσματα w_i/p_i και εξυπηρετούμε από το μεγαλύτερο στο μικρότερο, το οποίο προκύπτει και διαισθητικά αφού δίνουμε μεγαλύτερη προτεραιότητα στη μεγάλη σημασία w και το μικρό χρόνο p.

Θα αποδείξουμε ότι η άπληστη λύση είναι και η βέλτιστη με το επιχείρημα ανταλλαγής.

Έστω ότι η βέλτιστη λύση διαφέρει από την άπληστη σε ένα μόνο ζευγάρι δρομολόγησης για τα οποία ισχύει $\frac{w_i}{p_i} < \frac{w_j}{p_j}$, δηλαδή στην βέλτιστη λύση εξυπηρετείται πρώτα το i.

Ανταλλάσουμε την σειρά με την οποία εξυπηρετούνται οι δύο πελάτες και υπολογίζουμε την διαφορά στο συνολικό βεβαρυμένο χρόνο εξυπηρέτησης της βέλτιστης δρομολόγησης, θεωρώντας ότι η υπόλοιπη σειρά παραμένει ίδια.

$$w_1 p_1 + \dots + w_i (\overbrace{p_1 + \dots}^{t_0} + p_i) + w_j (\overbrace{p_1 + \dots}^{t_0} + p_i + p_j) = \Sigma$$

$$\Downarrow$$

$$w_1 p_1 + \dots + w_j (\overbrace{p_1 + \dots}^{t_0} + p_j) + w_i (\overbrace{p_1 + \dots}^{t_0} + p_i + p_j) = \Sigma'$$

- Θεωρούμε ότι οι ανταλλαγές που γίνονται είναι σε διαδοχικές θέσεις. Κάθε ακολουθία που δεν υπακούει την πλήρη διάταξη που αναφέραμε μπορεί να έρθει σε αυτήν με ανταλλαγές διαδοχικών θέσεων.

$$\Sigma' - \Sigma = (\cancel{w_j t_0} + \cancel{w_j p_j} + \cancel{w_i t_0} + \cancel{w_i p_i} + w_i p_j) - (\cancel{w_i t_0} + \cancel{w_i p_i} + \cancel{w_j t_0} + w_j p_i + \cancel{w_j p_j}) =$$

$$= w_i p_j - w_j p_i = p_i p_j \left(\frac{w_i}{p_i} - \frac{w_j}{p_j} \right) < 0$$

Έτσι, η βέλτιστη γίνεται ίδια με την άπληστη χωρίς αύξηση χρόνου και επομένως η άπληστη δρομολόγηση είναι και η βέλτιστη.

Χρησιμοποιώντας ταξινόμηση και έπειτα το άπληστο κριτήριο πετυχαίνουμε πολυπλοκότητα $O(n\log n + n) = O(n \log n)$

Σ) Αρχικά, ταξινομούμε και πάλι τους πελάτες σε φθίνουσα σειρά $\frac{w_i}{p_i}$. Λόγω όμως των πολλών μεταβλητών (υπαλλήλων) δεν μπορούμε να λύσουμε το πρόβλημα με άπληστο κριτήριο, αλλά με δυναμικό προγραμματισμό.

Οι μεταβλητές που χρειαζόμαστε για να λύσουμε αναδρομικά είναι ο αριθμός των πελατών που ανατίθενται στους υπαλλήλους με τη σειρά που αναφέραμε και ο συνολικός χρόνος αναμονής (current) που θα επιβαρυνθεί κάθε πελάτης που εισέρχεται στην ουρά εξυπηρέτησης κάθε υπαλλήλου.

Αν $F(i)$ ο ελάχιστος συνολικός βεβαρυμένος χρόνος για τους πρώτους i πελάτες και $T_1(i)$ και $T_2(i)$ οι χρόνοι αναμονής που θα περιμένει ο $(i+1)$-οστός πελάτης αν εξυπηρετηθεί από τον πρώτο και τον δεύτερο υπάλληλο αντίστοιχα έχουμε

$$F(i) = F(i-1) + w(i) \cdot \left(p(i) + \min\{ T_1(i-1), T_2(i-1) \} \right)$$

και $$T_1(i) = \begin{cases} T_1(i-1), & \text{αν } T_1(i-1) > T_2(i-1) \\ T_1(i-1) + p(i), & \text{διαφορετικά} \end{cases}$$

, ομοίως για $T_2(i)$.

Έστω, το εξής παράδειγμα με ήδη ταξινομημένα (w_i, p_i):

i	wi	pi
1	4	1
2	5	2
3	3	2
4	3	3
5	2	3

i	F	T_1	T_2
0	0	0	0
1	4	1	0
2	14	1	2
3	23	3	2
4	38	3	5
5	50	6	5

Έτσι, η λύση βρίσκεται αποδοτικά bottom-up.

Αν είχαμε m υπαλλήλους θα είχαμε m $T_{i \in \{1,\dots,m\}}$ στήλες και η λύση θα επεκτεινόταν εντελώς φυσικά.

Λόγω της εύρεσης του min η πολυπλοκότητα θα ήταν $O(m \cdot i)$.

Άσκηση 5 : Ελάχιστη Διαταραχή Ακολουθίας

1) • Όσο αυξάνεται το K το $D(a_K)$ αυξάνεται ή παραμένει σταθερό μέχρι το set των K αριθμών να περιέχει το μέγιστο και ελάχιστο στοιχείο οπότε έπειτα δεν μπορεί να αυξηθεί άλλο.

• Το συνολικό $D(a)$ αποτελεί άθροισμα n στοιχείων όπου όλοι προσθεταίοι είναι φραγμένοι από την ποσότητα max − min.

• Όσο πιο "αργά" η K-αδα περιέχει και το min και το max τόσες λιγότερες φορές θα εμφανιστεί το (max − min) στο άθροισμα και άρα περισσότεροι προσθεταίοι μικρότεροί του, αφού όταν το φτάνουμε όλοι οι προσθεταίοι έπειτα είναι ίσοι με αυτό. Άρα ή το min ή το max είναι το τελευταίο στοιχείο.

• Μετακινώντας, λοιπόν στο τέλος το (max − min) προκύπτει εξαιτίας της αρχής της βελτιστότητας για τους πρώτους προσθεταίους ότι με αυτόν τον τρόπο έχουμε τη βέλτιστη λύση.

2) Θα λύσουμε το πρόβλημα με τη χρήση bottom-up δυναμικού προγραμματισμού.

Αρχικά, φτιάχνουμε μια ταξινομημένη λίστα των αριθμών της ακολουθίας. Έστω, οι δείκτες i και j οι οποίοι δείχνουν στην πρώτη και την τελευταία θέση της ταξινομημένης λίστας αντίστοιχα.

Με βάση το άπληστο κριτήριο του (α) ερωτήματος καταστρώνουμε την αναδρομική εξίσωση

$$\boxed{F(i,j) = (j-i) + \min\{F(i+1,j),\ F(i,j-1)\}}$$

όπου $F(i,j)$ είναι η ελάχιστη συνολική διαταραχή που χρησιμοποιεί τα στοιχεία i μέχρι j της ταξινομημένης λίστας.

Οι δύο επιλογές που φαίνονται μέσα στο min αποτελούν την απόφαση του αν θα τοποθετήσουμε στο τέλος της βέλτιστης ακολουθίας το παρόν ελάχιστο στοιχείο με index i ή το μέγιστο με j.

Το $j-i$ αποτελεί τη διαταραχή κάθε στοιχείου που αποκλείεται.

Κατασκευάζουμε έναν πίνακα, ο οποίος γεμίζει σταδιακά.

Έστω για $a = (6, 2, 3, 1, 3, 3)$ με την ταξινομημένη λίστα και πίνακα :

	i					j
S	1	2	3	3	3	6

i\j	1	2	3	4	5	6
1		1	3	3	3	8
2			1	1	1	5
3				0	0	3
4					0	3
5						3
6						

- Σκοπός μας είναι να βρούμε το στοιχείο $F(1,6)$
- Σε κάθε min της αναδρομής κρατάμε την επιλογή ώστε να ξέρουμε αν βάλαμε στο τέλος το μικρότερο ή το μεγαλύτερο.
- Γεμίζουμε διαδοχικές διαγωνίους.

1) Αρχικά, γεμίζουμε τα στοιχεία $(1,2)$, $(2,3)$, ... με τις διαφορές $s(2)-s(1)$, $s(3)-s(2)$, ...

2) $F(1,3) = 2 + \min\{F(2,3), F(1,2)\} = 3$

- Όταν έχουμε ισοπαλίες όπως εδώ μπορούμε να φτιάξουμε εναλλακτικές βέλτιστες ακολουθίες.

...

Άρα βρήκαμε σωστά το ελάχιστο συνολικό κόστος που είναι 8. Για τη βέλτιστη ακολουθία κάνουμε trace-back τα σημειωμένα βελάκια.

- Στον πίνακα φαίνονται μόνο όσο βελάκια οδηγούν στο $F(1,6)$ αλλά κανονικά τα κρατάμε όλα.

Αρχικά πάμε από το $(1,6)$ στο $(1,5)$ άρα πάει στο τέλος το 6. Έπειτα, πάμε από το $(1,5)$ στο $(2,5)$ άρα το 1 πάει στο τέλος. Καταλήγουμε εντέλει στην ακολουθία $3, 3, 2, 1, 6$.

- Η σειρά των πρώτων 2 δεν παίζει ρόλο άρα κοιτάμε πίσω $n-2$ φορές.

Η πολυπλοκότητα του αλγορίθμου είναι $O(n\log n + \frac{n^2}{2}) = O(n^2)$.

Η ορθότητα εξασφαλίζεται από την αρχή της βελτιστότητας και από το γεγονός ότι αφού οι δείκτες i και j δείχνουν πάντα στο μικρότερο και μεγαλύτερο διαθέσιμο στοιχείο τηρείται το άπληστο κριτήριο (βήμα) που θεμελιώσαμε. ■