Εθνικό Μετσόβιο Πολυτεχνείο
Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών
Τομέας Τεχνολογίας Πληροφορικής και Υπολογιστών
Αλγόριθμοι και Πολυπλοκότητα
1η Σειρά Γραπτών Ασκήσεων

Άσκηση 1: Πλησιέστερο Ζεύγος Σημείων

(α) Αρχικά, χρειάζεται να ταξινομήσουμε τα σημεία του χώρου κατά x, κατά y και κατά z σε χρόνο 3nlogn. Έπειτα, όμοια με την περίπτωση των 2 διαστάσεων, μπορούμε να χωρίσουμε τον χώρο σε 2 ημιχώρους (έστω κατά z) χρησιμοποιώντας το επίπεδο που βρίσκεται στο διάμεσο ύψος της ως προς z ταξινόμησης. Αναδρομικά υπολογίζεται η ελάχιστη απόσταση σε κάθε ημιχώρο και στο βήμα της συγχώνευσης απομένει να βρεθεί εάν υπάρχει κάποια απόσταση μικρότερη της ελάχιστης των δύο αναδρομικών κλήσεων μεταξύ σημείων εναλλάξ του διαχωριστικού επιπέδου. Οπότε, αν δ είναι η ελάχιστη απόσταση από τις αποστάσεις που βρέθηκαν στους 2 ημιχώρους, διαγράφουμε όλα τα σημεία σε απόσταση κατά z μεγαλύτερη από δ από το διαχωριστικό ημιεπίπεδο (σε γραμμικό χρόνο) και χωρίζουμε τον χώρο στη λωρίδα πλάτους 2δ σε κύβους πλευράς δ/2. Η μεγαλύτερη απόσταση μεταξύ σημείων σε έναν τέτοιο κύβο είναι $\frac{\sqrt{3}}{2}\delta < \delta$. Άρα, σε κάθε κύβο «χωράει» το πολύ ένα σημείο. Μπορεί, λοιπόν, να γίνει έλεγχος εάν 2 σημεία απέχουν λιγότερο από δ σε γραμμικό χρόνο, καθώς κάθε σημείο αρκεί να συγκριθεί με σταθερό αριθμό από γειτονικά του (ο ακριβής αριθμός δεν έχει σημασία για την παρούσα ανάλυση).

Ο συνολικός χρόνος εκτέλεσης του αναδρομικού αλγορίθμου περιγράφεται από τη σχέση $T(n) = 2T(n/2) + O(n)$, η οποία έχει λύση $T(n) = \Theta(n\log n)$ από το Master Theorem.

Άρα, ο συνολικός χρόνος εκτέλεσης του αλγορίθμου είναι $4n\log n = \Theta(n\log n)$.

Άσκηση 2: Πόρτες Ασφαλείας στο Κάστρο

Αρχικά θα βρεθεί ο διακόπτης που συνδέεται με την πρώτη πόρτα. Μπορεί να βρεθεί με logn συνδυασμούς διακοπτών, με δυαδική αναζήτηση, αλλάζοντας την κατάσταση των μισών διακοπτών αρχικά. Έπειτα, επαναλαμβάνεται η διαδικασία στους διακόπτες που επηρεάζουν την πόρτα (εάν αλλάξει η κατάστασή της στους διακόπτες των οποίων αλλάξαμε την κατάσταση, αλλιώς στους υπόλοιπους) συνεχώς μέχρι να βρεθεί ο ζητούμενος διακόπτης. Εάν εκτελεστεί η παραπάνω διαδικασία για όλες τις πόρτες, θα βρεθούν όλα τα ζεύγη διακόπτη – πόρτας σε $n\log n + (n-1)\log(n-1) + \ldots + 1 = O(n\log n)$ διαμορφώσεις διακοπτών. Η ορθότητα του αλγορίθμου οφείλεται στο γεγονός ότι δοκιμάζονται όλοι οι πιθανοί διακόπτες που είναι συνδεδεμένοι με κάθε πόρτα, έως ότου βρεθεί το σωστό ζεύγος.

Άσκηση 3: Φόρτιση Ηλεκτρικών Αυτοκινήτων

Για την επίλυση του προβλήματος ορίζουμε αρχικά το πρόβλημα απόφασης «Αρκούν k σταθμοί φόρτισης για να είναι το πολύ d η μέγιστη καθυστέρηση;». Η επίλυση του προβλήματος απόφασης μπορεί να γίνει σε γραμμικό χρόνο, χρησιμοποιώντας μια ουρά προτεραιότητας, ως εξής:
Για κάθε χρόνο άφιξης α_i, εάν οι k σταθμοί φόρτισης είναι κατειλημμένοι, τοποθετούμε τον α_i στην ουρά προτεραιότητας. Για να γνωρίζουμε εάν οι σταθμοί φόρτισης είναι κατειλημμένοι χρειάζεται ένας μετρητής, ο οποίος θα λαμβάνει τιμές στο διάστημα [0, k]. Επίσης, χρειάζεται να αποθηκεύεται σε μια μεταβλητή η τρέχουσα χρονική στιγμή, ώστε όταν το επόμενο στοιχείο της εισόδου έχει τιμή μεγαλύτερη από την τρέχουσα χρονική στιγμή, να γίνεται ανανέωσή της, μηδενισμός του μετρητή των κατειλημμένων σταθμών φόρτισης και εξαγωγή των k πρώτων στοιχείων από την ουρά προτεραιότητας. Τέλος, κάθε φορά που παίρνουμε ένα στοιχείο από την ουρά προτεραιότητας, γνωρίζουμε ότι η καθυστέρησή του είναι τρέχουσα χρονική στιγμή – τιμή στοιχείου + 1. Εάν η ποσότητα αυτή είναι μεγαλύτερη από d, η απάντηση στο πρόβλημα απόφασης είναι ΟΧΙ. Εναλλακτικά, εφόσον έχει γίνει επεξεργασία όλων των στοιχείων και η ουρά προτεραιότητας είναι άδεια, η απάντηση είναι ΝΑΙ. Η παραπάνω διαδικασία επεξεργάζεται κάθε στοιχείο 2 φορές στη χειρότερη περίπτωση, αποδίδοντας χρόνο εκτέλεσης $O(2n) = O(n)$.
Από τη φύση του προβλήματος είναι εύκολο να συμπεράνουμε ότι χρειάζεται τουλάχιστον 1 και το πολύ n σταθμοί φόρτισης. Χρησιμοποιώντας το παραπάνω πρόβλημα απόφασης και δυαδική αναζήτηση για τους σταθμούς s* στο διάστημα [1, n], μπορούμε να βρούμε το ελάχιστο s* σε χρόνο O(nlogn).

Η ορθότητα του αλγορίθμου οφείλεται στο γεγονός ότι αν δεν εξασφαλίζεται η ζητούμενη καθυστέρηση από k σταθμούς φόρτισης, δε μπορεί να εξασφαλιστεί ούτε με λιγότερους από k σταθμούς. Η διάταξη αυτή επιτρέπει τη λύση του προβλήματος με τη χρήση δυαδικής αναζήτησης.

Η χρονική πολυπλοκότητα είναι nlogn γιατί η δυαδική αναζήτηση χρειάζεται χρόνο logn και κάθε βήμα χρειάζεται γραμμικό χρόνο, πράγμα το οποίο φαίνεται στην περιγραφή του προβλήματος απόφασης.

Άσκηση 4: Παραλαβή Πακέτων

1. Για την περίπτωση ενός υπαλλήλου, μπορεί να χρησιμοποιηθεί το άπληστο κριτήριο δρομολόγησης των δεμάτων κατά φθίνουσα σειρά των λόγων $\frac{w_i}{p_i}$. Ο χρόνος εκτέλεσης του αλγορίθμου είναι n (υπολογισμός των λόγων) + nlogn (ταξινόμηση) = O(nlogn). Για την απόδειξη ορθότητας χρησιμοποιούμε το παρακάτω επιχείρημα ανταλλαγής:

   Έστω ότι τα στοιχεία έχουν ταξινομηθεί βάσει των λόγων $\frac{w}{p}$, ώστε $\frac{w_i}{p_i} \leq \frac{w_j}{p_j} \leftrightarrow i > j$ στη λύση του άπληστου αλγορίθμου και έστω μια άλλη λύση στην οποία έχει ανταλλαγεί το πρώτο στοιχείο με κάποιο άλλο διατηρώντας τη διάταξη των υπολοίπων. Τα κόστη των 2 λύσεων θα είναι:

   (1) $w_1p_1 + w_2(p_1 + p_2) + \ldots + w_n\sum_{i=1}^{n} p_i$ (άπληστος αλγόριθμος)
   (2) $w_kp_k + w_2(p_k + p_2) + w_3(\sum_{i=1}^{3} p_i + p_k - p_1) + \ldots + w_1\sum_{i=1}^{k} p_i + \ldots + w_n\sum_{i=1}^{n} p_i$ (εναλλακτική λύση)

   Αρκεί να δείξουμε ότι (2) – (1) ≥ 0.

   Πράγματι:

   (2) – (1) =

   $w_kp_k + w_2(p_k + p_2) + w_3(\sum_{i=1}^{3} p_i + p_k - p_1) + \ldots + w_1\sum_{i=1}^{k} p_i + \ldots + w_n\sum_{i=1}^{n} p_i - (w_1p_1 + w_2(p_1 + p_2) + \ldots + w_n\sum_{i=1}^{n} p_i) =$
   $w_kp_k + w_2(p_k + p_2) + w_3(\sum_{i=1}^{3} p_i + p_k - p_1) + \ldots + w_1\sum_{i=1}^{k} p_i - (w_1p_1 + w_2(p_1 + p_2) + \ldots + w_k\sum_{i=1}^{k} p_i) =$
   $- w_k\sum_{i=1}^{k-1} p_i + w_2(p_k - p_1) + \ldots + w_{k-1}(p_k - p_1) + w_1\sum_{i=2}^{k} p_i =$
   $-w_kp_1 - w_k\sum_{i=2}^{k-1} p_i + (p_k - p_1)\sum_{i=2}^{k-1} w_i + w_1p_k + w_1\sum_{i=2}^{k-1} p_i =$
   $w_1p_k - w_kp_1 + \sum_{i=2}^{k-1}(-p_i w_k + p_k w_i - p_1 w_i + w_1 p_i)$

   Χρησιμοποιώντας την αρχική υπόθεση ότι $\frac{w_i}{p_i} \leq \frac{w_j}{p_j} \leftrightarrow i > j$, έχουμε:
   $\frac{w_1}{p_1} \geq \frac{w_k}{p_k} \leftrightarrow w_1p_k \geq w_kp_1 \leftrightarrow w_1p_k - w_kp_1 \geq 0,$
   $\frac{w_i}{p_i} \geq \frac{w_k}{p_k} \leftrightarrow w_ip_k \geq w_kp_i \leftrightarrow w_ip_k - w_kp_i \geq 0$ και
   $\frac{w_1}{p_1} \geq \frac{w_i}{p_i} \leftrightarrow w_1p_i \geq w_ip_1 \leftrightarrow w_1p_i - w_ip_1 \geq 0.$

   Άρα η άπληστη λύση είναι και βέλτιστη.

Άσκηση 5: Ελάχιστη Διαταραχή Ακολουθίας

1. Χωρίς βλάβη της γενικότητας θεωρούμε ότι το μέγιστο στοιχείο είναι το τελευταίο. Έστω οι ακολουθίες που αποτελούν λύση του προβλήματος $s_1 = a_1, a_2, \ldots, a_n$ (με ελάχιστο στοιχείο min = a_i, για κάποιο $i < k$ και μέγιστο το a_n) και $s_2 = a_1, a_2, \ldots, a_{k-1}, a_n, a_{k+1}, \ldots, a_k$ (στην οποία διατηρείται η διάταξη όλων των στοιχείων εκτός από την εναλλαγή του a_n με το a_k). Τότε, αφαιρώντας τη διαταραχή της s_1 από τη διαταραχή της s_2, έχουμε $(a_n - \text{min})(n - k) - \sum_{i=k}^{n}(a_i - \text{min}) > 0$, καθώς όλες οι διαφορές a_i – min έχουν μικρότερη τιμή από την ποσότητα a_n – min, αφού $a_n > a_i$, για κάθε i. Ομοίως, εάν τελευταίο στοιχείο ήταν το ελάχιστο. Άρα, το τελευταίο στοιχείο της βέλτιστης ακολουθίας πρέπει να είναι είτε το μέγιστο είτε το ελάχιστο.

2. Το πρόβλημα μπορεί να λυθεί χρησιμοποιώντας την τεχνική του δυναμικού προγραμματισμού, δεδομένου ότι το τελευταίο στοιχείο της ζητούμενης ακολουθίας β* είναι είτε το πρώτο είτε το τελευταίο του στιγμιότυπου εισόδου, ορίζοντας την παρακάτω συνάρτηση διαταραχής D:

$$D(i,j) = \begin{cases} x_j - x_i + \min\{D([i,j-1]), D([i+1,j])\}, & j-i>1 \\ x_j - x_i, & j-i=1 \\ 0, & \text{else} \end{cases}$$

ορίσματα της οποίας είναι οι δείκτες αρχής και τέλους μιας υπο-ακολουθίας της ταξινομημένης ακολουθίας που προκύπτει από το στιγμιότυπο εισόδου του προβλήματος.

Από τη συνάρτηση D προκύπτει ο εξής ψευδοκώδικας, χρησιμοποιώντας memoization:

```
D(i, j):
    if j – 1 < 1 return 0

    if F(i, j) return F(i, j)

    if j – i == 1
        F(i, j) = xj – xi
        return F(i, j)

    F(i, j) =       xj – xi + min{D[i, j – 1], D[i + 1, j]}
    return F(i, j)
```

Η ελάχιστη διαταραχή δίνεται από την κλήση της D(1, n).

Εναλλακτικά, ακολουθώντας μια bottom – up προσέγγιση, η συνάρτηση D υπολογίζει τις n – 1 το πλήθος διαφορές διαδοχικών στοιχείων της ταξινομημένης σε αύξουσα σειρά ακολουθίας $x_1, \ldots, x_n$ και τις αποθηκεύει στη μνήμη. Έπειτα, υπολογίζει τις n – 2 το πλήθος διαφορές τελικών – αρχικών στοιχείων που προκύπτουν από τις τριάδες, τις προσθέτει στην ελάχιστη συμβατή διαφορά δυάδας που υπολογίστηκε στο προηγούμενο βήμα, τις αποθηκεύει και επαναλαμβάνει με όμοιο τρόπο (για τις τετράδες, πεντάδες κλπ.) μέχρι να υπολογιστεί η ελάχιστη συνολική διαταραχή D(β*).
Για τον υπολογισμό της ακολουθίας β*, μπορεί να αποθηκεύεται εκτός από τις τιμές και η διαδρομή που ακολουθήθηκε μέχρι τον υπολογισμό της D(β*).

Ο χρόνος εκτέλεσης του παραπάνω αλγορίθμου είναι nlogn για την ταξινόμηση του στιγμιότυπου εισόδου + $2\sum_{i=1}^{n-1} i = \frac{2n(n-1)}{2}$ = 2n(n – 1) υπολογισμοί (προσθέσεις, συγκρίσεις) = $O(n^2)$. Χρησιμοποιείται επίσης μνήμη $O(n^2)$ για την αποθήκευση των $\sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$ ενδιάμεσων αποτελεσμάτων.

Η ορθότητα του αλγορίθμου έγκειται στο γεγονός ότι εξετάζονται όλες οι περιπτώσεις και αποθηκεύονται σε κάθε βήμα οι βέλτιστες, καθώς πρόκειται για αλγόριθμο δυναμικού προγραμματισμού. Συγκεκριμένα η σχέση $D[1,n] = x_n - x_1 + \min\{D([1,n-1]), D([2,n])\}$ σε φυσική γλώσσα ερμηνεύεται ως «η ελάχιστη διαταραχή υπολογίζεται ως το άθροισμα της διαφοράς του μέγιστου από το ελάχιστο στοιχείο της δεδομένης ακολουθίας και της ελάχιστης διαταραχής της ακολουθίας που προκύπτει αφαιρώντας είτε το ελάχιστο είτε το μέγιστο».

Τέλος, ένα παράδειγμα εκτέλεσης του αλγορίθμου, χρησιμοποιώντας την ακολουθία α = (1, 3, 3, 3, 6, 6) της εκφώνησης, η οποία είναι ήδη ταξινομημένη:

i \ j	1	2	3	4	5	6
1	0	3 - 1 = 2	3 – 1 + min{2, 0} = 2	3 – 1 + min{2, 0} = 2 →	6 – 1 + min{2, 3} = 7	6 – 1 + min{7, 6} = 11
2		0	3 – 3 = 0 →	3 – 3 + min{0, 0} = 0 →	6 – 3 + min{0, 3} = 3 →	6 – 3 + min(3, 6} = 6
3			0	3 – 3 = 0 →	6 – 3 + min{0, 3} = 3 →	6 – 3 + min{3, 3} = 6
4				0	6 – 3 = 3	6 – 3 + min{0, 3} = 3
5					0	6 – 6 = 0
6						0

Στον παραπάνω πίνακα έχουν αποθηκευτεί οι ενδιάμεσοι υπολογισμοί, σύμφωνα με την περιγραφή του αλγορίθμου, ξεκινώντας από τις δυάδες (στοιχεία (1, 2), (2, 3), (3, 4), (4, 5), (5, 6)), συνεχίζοντας με τις τριάδες (στοιχεία (1, 3), (2, 4), (3, 5), (4, 6)) κοκ.
Τα βέλη υποδεικνύουν τη διαδρομή που οδηγεί στην ελάχιστη διαταραχή.
Οπότε, συμπεραίνουμε ότι η ελάχιστη διαταραχή έχει τιμή 11 (η τιμή στη θέση (1, 6)).

Επίσης, για την εύρεση μιας ακολουθίας που αποδίδει την ελάχιστη διαταραχή, αρκεί να ακολουθηθούν τα βέλη αντιστρόφως και να τοποθετηθεί το αντίστοιχο στοιχείο της στήλης ή γραμμής που εξαλείφεται στο τέλος της ακολουθίας.
Εν προκειμένω, για τον υπολογισμό της βέλτιστης λύσης το βέλος προέρχεται από το στοιχείο (2, 6), οπότε τοποθετούμε το πρώτο στοιχείο (1) στο τέλος. Συνεχίζοντας, το βέλος προέρχεται το στοιχείο (2, 5), οπότε τοποθετούμε το τελευταίο στοιχείο (6) στο τέλος και επαναλαμβάνουμε έως ότου απομένουν 2 στοιχεία, τα οποία τοποθετούνται στην αρχή με οποιαδήποτε σειρά, καθώς η σειρά των 2 πρώτων στοιχείων μιας ακολουθίας δεν επηρεάζει τη συνολική διαταραχή.