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

Αλγόριθμοι και Πολυπλοκότητα

Διδάσκοντες: Δημήτρης Φωτάκης, Δώρα Σούλιου, Θανάσης Λιανέας

1η Σειρά Προγραμματιστικών Ασκήσεων - Σχέδιο Λύσεων

Άσκηση 1: Ενοικίαση Αυτοκινήτου

Μία πρώτη παρατήρηση-κλειδί που είναι απαραίτητη για την κατανόηση και την επίλυση της άσκησης είναι ότι ο (βέλτιστος) χρόνος στον οποίο ένα όχημα μπορεί να πραγματοποιήσει τη διαδρομή από την πόλη Α στην πόλη Β εξαρτάται μόνο από την χωρητικότητά του σε καύσιμα. Επιπλέον στις ενδιάμεσες πόλεις-σταθμούς το όχημα μπορεί να γεμίσει πάλι την χωρητικότητά του και έτσι μας ενδιαφέρουν μόνο οι χρόνοι στους οποίους ένα όχημα μπορεί να διασχίσει τις αποστάσεις από τον ένα σταθμό στον άλλο.

Ξεκινάμε υπολογίζοντας τις αποστάσεις αυτές. Επειδή οι θέσεις των σταθμών **δεν** δίνονται ταξινομημένες, πρέπει πρώτα να ταξινομήσουμε τον πίνακα position σε αύξουσα σειρά, με χρήση mergesort/quicksort σε $O(K \log K)$ (όπως θα δούμε στη συνέχεια η πολυπλοκότητα υπολογισμού του βέλτιστου οχήματος ξεπερνάει το $O(K \log K)$ και επομένως η ταξινόμηση αυτή δεν επηρεάζει την πολυπλοκότητα της λύσης μας). Έπειτα, ο υπολογισμός των $(K+1)$-αποστάσεων ανάμεσα σε αυτούς τους σταθμούς γίνεται γραμμικά ως προς K από την σχέση:

$$\text{distance}[i] = \begin{cases} \text{position}[0] & , i = 0 \\ \text{position}[i] - \text{position}[i-1] & , 0 < i < K \\ S - \text{position}[K-1] & , i = K \end{cases}$$

όπου στον πίνακα distance έχουμε συμπεριλάβει και τις αποστάσεις από την πόλη Α (θέση 0) στον πρώτο σταθμό και από τον τελευταίο σταθμό στην πόλη Β (θέση S).

Στην συνέχεια πρέπει να αναλύσουμε τον τρόπο με τον οποίο υπολογίζεται ο βέλτιστος χρόνος στον οποίο ένα όχημα χωρητικότητας C διανύει μία απόσταση D. Ας συμβολίσουμε με $\text{time}(C, D)$ αυτήν την ποσότητα. Από την εκφώνηση κάθε όχημα έχει δύο λειτουργίες και μπορεί να κινείται "αργά" καλύπτοντας απόσταση 1 σε χρόνο T_s και χρησιμοποιώντας καύσιμα C_s είτε "γρήγορα" καλύπτοντας απόσταση 1 σε χρόνο $T_f < T_s$ αλλά χρησιμοποιώντας περισσότερα καύσιμα $C_f > C_s$. Επομένως, τα ελάχιστα καύσιμα που μπορεί να χρησιμοποιήσει για να διανύσει απόσταση D είναι $D \cdot C_s$ αν κινείται αργά σε όλη τη διαδρομή (κάνοντας τον χειρότερο χρόνο $D \cdot T_s$), ενώ ο βέλτιστος χρόνος είναι $D \cdot T_f$ δεδομένου ότι έχει τουλάχιστον $D \cdot C_f$ χωρητικότητα σε καύσιμα. Στην ενδιάμεση περίπτωση, επειδή θέλουμε να ελαχιστοποιήσουμε τον χρόνο και δεν μας νοιάζει να εξοικονομήσουμε καύσιμα (αφού στους σταθμούς γεμίζουμε όσο θέλουμε), θα πρέπει να κινηθούμε όσο μπορούμε στη γρήγορη λειτουργία και έπειτα να καλύψουμε την υπόλοιπη απόσταση με την αργή λειτουργία.

Από τα παραπάνω, προκύπτει ότι:

$$\text{time}(C, D) = \begin{cases} \infty & \text{αν } C < D \cdot C_s \\ D \cdot T_s - (C - D \cdot C_s)\frac{T_s - T_f}{C_f - C_s} & \text{αν } D \cdot C_s \leq C < D \cdot C_f \\ D \cdot T_f & \text{αν } C \geq D \cdot C_f \end{cases}$$

Η παραπάνω σχέση μας λέει ότι αν ένα όχημα έχει χωρητικότητα $C < D \cdot C_s$ τότε δεν μπορεί να καλύψει απόσταση D ακόμα και να πηγαίνει μόνο αργά, άρα θα χρειαστεί άπειρο χρόνο. Αντιθέτως οχήματα με χωρητικότητα $C \geq D \cdot C_f$ μπορούν να καλύψουν όλη την απόσταση κινούμενα γρήγορα και έτσι πετυχαίνουν τον βέλτιστο εφικτό χρόνο $D \cdot T_f$. Στις ενδιάμεσες περιπτώσεις, ένα όχημα θα κινηθεί γρήγορα για να καλύψει απόσταση x και αργά για να καλύψει την υπόλοιπη απόσταση $D - x$. Σε αυτήν την περίπτωση θα χρειαστεί καύσιμα $f = x \cdot C_f + (D - x) \cdot C_s$. Όπως είπαμε ψάχνουμε τον βέλτιστο χρόνο, άρα θα χρησιμοποιήσουμε όλα τα καύσιμά μας. Άρα, από $f = C$ προσδιορίζουμε το x και έπειτα τον χρόνο $\text{time}(C, D) = x \cdot T_f + (D - x) \cdot T_s$.

Για τον συνολικό βέλτιστο χρόνο που χρειάζεται ένα όχημα χωρητικότητας C για να μεταβεί από την πόλη Α στην πόλη Β είναι εύκολο να δούμε ότι απλά πρέπει να αθροίσουμε τους χρόνους που απαιτούνται για όλες τις ενδιάμεσες αποστάσεις, δηλαδή:

$$\text{opt} - \text{time}(C) = \sum_{i=0}^{K} \text{time}(C, \text{distance}[i])$$

Παρατηρούμε επίσης ότι ο υπολογισμός του $\text{time}(C, D)$ γίνεται σε χρόνο $O(1)$ και επομένως ο συνολικός χρόνος που απαιτείται για να μεταβεί ένα όχημα χωρητικότητας C από την πόλη Α στην πόλη Β υπολογίζεται σε χρόνο $O(K)$. Έχοντας ολοκληρώσει την παραπάνω ανάλυση, μπορούμε πλέον να λύσουμε το πρόβλημα.

Λύση 1: $O(KN)$

Η πιο απλή λύση για το πρόβλημα είναι να εξετάσουμε για κάθε όχημα i χωρητικότητας $\text{capacity}[i]$ αν ισχύει ότι $\text{opt} - \text{time}(\text{capacity}[i]) \leq T$. Σε αυτήν την περίπτωση, το όχημά μας είναι feasible, δηλαδή μπορεί να πραγματοποιήσει το ταξίδι από την πόλη Α στην πόλη Β σε χρόνο το πολύ T. Δεν μένει παρά να διαλέξουμε το φθηνότερο από τα feasible οχήματα (-1 αν αυτό δεν υπάρχει). Επειδή έχουμε N οχήματα και ο υπολογισμός του opt – time χρειάζεται χρόνο $O(K)$, η συνολική πολυπλοκότητα της λύσης είναι $O(NK)$ και αρκεί για να περάσει μόνο τα μικρά testcases λόγο του χρονικού περιορισμού.

Λύση 2: $O(K \log N)$

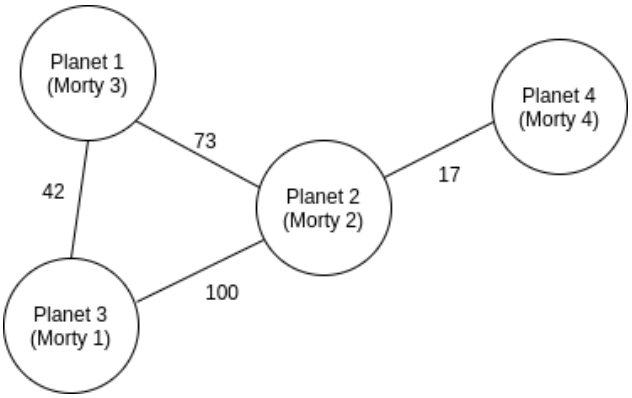
Ο λόγος για τον οποίο μπορούμε να βελτιώσουμε την πολυπλοκότητα της λύσης μας, είναι ότι δεν έχουμε αξιοποιήσει την ακόλουθη ιδιότητα του προβλήματος: η συνάρτηση $\text{opt} - \text{time}(C)$ είναι φθίνουσα ως προς C. Αυτό σημαίνει ότι αν ένα αμάξι χωρητικότητας C δεν μπορεί να κάνει το ταξίδι σε χρόνο το πολύ T τότε κανένα άλλο όχημα με χωρητικότητα μικρότερη του C δεν θα μπορεί. Αντίστοιχα αν ένα αμάξι χωρητικότητας C μπορεί να πραγματοποιήσει το ταξίδι σε χρόνο το πολύ T τότε και κάθε άλλο όχημα με μεγαλύτερη χωρητικότητα θα μπορεί.

Η ιδιότητα αυτή μας επιτρέπει να κάνουμε **δυαδική αναζήτηση** και να βελτιώσουμε την πολυπλοκότητα του αλγορίθμου μας. Ταξινομούμε τα οχήματα (τιμές - χωρητικότητες) σε αύξουσα σειρά χωρητικότητας

σε χρόνο $O(N \log N)$ με κάποιον καλό αλγόριθμο ταξινόμησης. Έπειτα κάνουμε δυαδική αναζήτηση πάνω στις χωρητικότητες των οχημάτων μέχρι να βρούμε το όχημα ελάχιστης χωρητικότητας που μπορεί να ολοκληρώσει το ταξίδι από την πόλη Α στην πόλη Β σε χρόνο το πολύ Τ - αυτό είναι το πρώτο feasible όχημα. Ξεκινάμε από το όχημα $N/2$, και αν είναι feasible εξετάζουμε το όχημα $N/4$, διαφορετικά το όχημα $3N/4$ κοκ. Συνολικά θα εξετάσουμε $\log N$ οχήματα ο έλεγχος για το αν είναι feasible χρειάζεται χρόνο $O(K)$. Έτσι σε χρόνο $O(K \log N)$ μπορέσαμε να προσδιορίσουμε τα feasible οχήματα και έπειτα διαλέγουμε αυτό με την ελάχιστη τιμή, σε γραμμικό ως προς N χρόνο.

Άσκηση 2: Τηλεμεταφορές

Εκ πρώτης όψεως, ακόμα και το αν μπορούν οι Mortys να γυρίσουν στους πλανήτες τους χρησιμοποιώντας όλα τα υπάρχοντα portals φαίνεται δύσκολο. Θα μεταφράσουμε το πρόβλημα σε ένα πρόβλημα βελτιστοποίησης χρησιμοποιόντας έννοιες με τις οποίες έχουμε οικειότητα. Συγκεκριμένα, παρατηρούμε πως τα ανοιχτά portals και οι πλανήτες επάγουν ένα γράφημα στο οποίο οι πλανήτες είναι οι κορυφές και το portal ανάμεσα σε δύο πλανήτες είναι μια μη-κατευθυνόμενη ακμή ανάμεσα σε αυτούς τους πλανήτες, με βάρος ίσο με το width του portal. Ας σχεδιάσουμε τον γράφο του παραδείγματος της εκφώνησης:



Έστω ότι ο Morty X (ο Morty του πλανήτη X) έχει βρεθεί στον πλανήτη Y. Εύκολα παρατηρούμε πως ο Morty X δεν θα μπορεί να επιστρέψει στον πλανήτη του αν οι κορυφές X και Y δεν συνδέονται στο γράφημα, δηλαδή οι X και Y θα πρέπει απαραίτητα να βρίσκονται στην ίδια συνεκτική συνιστώσα του γραφήματος. Μάλιστα, σε μια συνεκτική συνιστώσα του γραφήματος, όπου πάντα υπάρχει διαδρομή από κάθε κορυφή σε κάθε άλλη, μπορούμε να αποδείξουμε ότι υπάρχει κατάλληλη αλληλουχία μετακινήσεων ώστε να αναδιατάξουμε αυθαίρετα τους Mortys. Με άλλα λόγια, όλοι οι Mortys μπορούν να επιστρέψουν στους πλανήτες τους αν και μόνο αν για κάθε Morty, ο πλανήτης στον οποίο βρίσκεται και ο πλανήτης στον οποίο θέλει να πάει βρίσκονται στην ίδια συνεκτική συνιστώσα.

Το αρχικό πρόβλημα, πλέον, μεταφράζεται ως εξής: Διάλεξε κάποιες ακμές του γραφήματος (τα portals που θα χρησιμοποιήσουμε) ώστε αν ο Morty X έχει βρεθεί στον πλανήτη Y, οι X και Y να βρίσκονται στην ίδια συνεκτική συνιστώσα και παράλληλα προσπάθησε η ελαφρύτερη ακμή που χρησιμοποιήσες να είναι όσο πιο μεγάλη γίνεται.

Λύση 1: $O(NM)$

Εφόσον θέλουμε απλώς να χρησιμοποιήσουμε όσο μεγαλύτερες ακμές γίνεται μέχρι το γράφημα να αποκτήσει την ιδιότητα της συνδεσιμότητας όπως την περιγράψαμε παραπάνω, μπορούμε να ταξινομήσουμε τις ακμές ως προς τα βάρη τους, ξοδεύοντας $O(M \log M)$ και να αρχίσουμε να προσθέτουμε ακμές στο γράφημα ξεκινώντας από τις βαρύτερες ακμές, μέχρι να ισχύει η συνθήκη που θέλουμε.

Για να διατηρούμε τις συνεκτικές συνιστώσες, θα χρησιμοποιήσουμε την δομή Union-Find. Ξεκινώντας κάθε φόρα από την βαρύτερη ακμή που δεν έχουμε χρησιμοποιήσει, έστω $e = (x, y)$, θα κάνουμε unify τα άκρα της ακμής καθώς πλέον βρίσκονται στη ίδια συνεκτική συνιστώσα (union(x, y)). Τώρα που έχουμε εισαγάγει και την καινούργια ακμή στο γράφημα, ελέγχουμε αν ικανοποιείται η ιδιότητα που θέλουμε. Αν initial_location[x] είναι ο πλανήτης στον οποίο βρέθηκε ο Morty x τότε θα πρέπει να ισχύει:

$$\forall x \in \{1, \ldots, N\} : \text{find}(x) = find(\text{initial\_location}[x])$$

Αν η παραπάνω συνθήκη είναι αληθής, τότε μπορούμε να σταματήσουμε και η απάντηση θα είναι το βάρος της τελευταίας ακμής που προσθέσαμε.

Συνολικά σε όλες τις επαναλήψεις θα εκτελεστούν το πολύ M unions αφού κάθε ακμή θα εισαχθεί το πολύ μία φορά. Ακόμα, θα κάνουμε το πολύ M γύρους σε κάθε έναν από τους οποίους θα εκτελέσουμε στην χειρότερη περίπτωση $O(N)$ finds. Αν υλοποιήσουμε την δομή Union-Find με union by rank και path compression, τα unions γίνονται σε $O(1)$ και τα finds γίνονται πρακτικά σε $O(1)$. Τελικά, η συνολική πολυπλοκότητα θα είναι $O(M \log M + M + N \cdot M) = O(N \cdot M)$.

Λύση 2: $((N + M) \log M)$

Για να βελτιώσουμε περαιτέρω την πολυπλοκότητα της λύσης μας, μπορούμε να παρατηρήσουμε ότι αν ένα σύνολο επιλεγμένων ακμών έχει την ιδιότητα που θέλουμε, τότε προσθέτοντας ακμές προφανώς η ιδιότητα συνεχίζει να ισχύει. Οπότε αν ξέρουμε πως με τις k βαρύτερες ακμές ο γράφος αποκτά την ζητούμενη ιδιότητα, τότε και με τις $k + 1$ βαρύτερες θα έχει αυτήν την ιδιότητα. Δηλαδή, χρειάζεται να αναζητήσουμε λύσεις που χρησιμοποιούν τις $k - 1$ βαρύτερες ακμές ή και λιγότερες.

Συγκεκριμένα, μπορούμε να ορίσουμε ένα κατηγόρημα $p(w)$ το οποίο αληθεύει αν και μόνο αν χρησιμοποιώντας τις ακμές που έχουν width μεγαλύτερο ή ίσο από w, ο γράφος αποκτά την ιδιότητα που θέλουμε. Η παρατήρηση που κάναμε σημαίνει πως αν για κάποιο width w ισχύει $p(w) = True$ τότε και για κάθε $w' \leq w$ θα ισχύει $p(w') = True$. Επομένως, αν ταξινομήσουμε τις ακμές ως προς το βάρος τους, η ακολουθία του p θα μοιάζει κάπως έτσι

Edges sorted by weight	1	2	$\cdots$	$k-1$	k	$k+1$	$\cdots$	$M-1$	M
$p(w[i])$	True	True	$\cdots$	True	True	False	$\cdots$	False	False

Δηλαδή θα υπάρχει μια ακμή, έστω η k-οστή, που θα είναι η μεγαλύτερη ακμή για την οποία θα ισχύει $p(w[k]) = True$. Η απάντηση που ψάχνουμε είναι ακριβώς η $w[k]$ καθώς είναι η βαρύτερη ακμή για την οποία αν χρησιμοποιήσουμε ακμές με $w \geq w[k]$ τότε ο γράφος που σχηματίζεται έχει την ιδιότητα που θέλουμε.

Λόγω της μονοτονίας του p, μπορούμε να αναζητήσουμε την ακμή k με δυαδική αναζήτηση, καθώς ξέρουμε πως αν για την k ισχύει $p(w[k]) = False$ τότε η απάντηση θα βρίσκεται στο εύρος $[1, k)$ ενώ αν $p(w[k]) = True$ τότε η απάντηση θα βρίσκεται στο εύρος $[k, M]$.

Αρκεί μόνο να βρούμε έναν τρόπο να υπολογίζουμε το κατηγόρημα $p(w)$ για κάθε w. Δηλαδή, θέλουμε να βρούμε αν ισχύει η ιδιότητα που μας ενδιαφέρει αν απο το γράφημα κρατήσουμε μόνο τις ακμές με $width \geq w$. Μπορούμε να χρησιμοποιήσουμε και πάλι μια δομή Union-Find, να κάνουμε union τα άκρα όλων των ακμών που έχουν $width \geq w$ και να ελέγξουμε αν η αρχική τοποθεσία και ο προορισμός κάθε Morty βρίσκονται στην ίδια συνεκτική συνιστώσα. Συνολικά θα χρειαστούν το πολύ $O(M)$ unions και $O(N)$ finds, τα οποία μπορούν να γίνουν σε $O(N + M)$

Για να βρούμε την τελική απάντηση θα πρέπει αρχικά να ταξινομήσουμε τα βάρη σε $O(M \log M)$ και μετά να κάνουμε την δυαδική αναζήτηση που θα έχει το πολύ $O(\log M)$ γύρους, σε κάθε έναν από τους οποίους υπολογίζουμε μια φορά το p το οποίο μας παίρνει $O(N + M)$. Συνολικά, η λύση μας έχει πολυπλοκότητα $O((N + M) \log M)$