ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΕΧΝΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

2^{Η} ΣΕΙΡΑ ΓΡΑΠΤΩΝ ΑΣΚΗΣΕΩΝ

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Καμπουγέρης Χαράλαμπος
ΑΜ: 03120098
ΜΑΘΗΜΑ: Αλγόριθμοι και Πολυπλοκότητα
ΑΚ. ΕΤΟΣ: 2023-2024 ΕΞΑΜΗΝΟ: 7^{Ο}

Άσκηση 1: Πολύχρωμος Πεζόδρομος

Περιγραφή Αλγορίθμου

Η ιδέα πίσω από την ανάπτυξη του αλγορίθμου είναι η εξής: Δεδομένης της ζητούμενης ακολουθίας χρωμάτων, βρίσκουμε διαστήματα των οποίων τα άκρα έχουν χρωματιστεί με το ίδιο χρώμα x και κανένα στοιχείο (πλάκα) μέσα στο διάστημα δεν έχει χρωματιστεί με το x. Αυτό συμβαίνει, διότι δεν έχει νόημα να χρωματίσουμε σε δύο διαφορετικά «βαψίματα», διαστήματα που θα πάρουν το ίδιο χρώμα, τα χρωματίζουμε με τη μία ώστε να γλιτώσουμε ένα «βάψιμο». Εύκολα διαπιστώνουμε ότι αν έχουμε k τέτοια διαστήματα, ο αριθμός των «βαψιμάτων» που θα κάνουμε είναι $c = N - k$ (1), όπου N ο αριθμός των πλακών.

Με βάση αυτή τη λογική, έχουμε ανάγει το πρόβλημα σε υπολογισμό διαστημάτων και ψάχνουμε το μέγιστο αριθμό τέτοιων διαστημάτων, ώστε με βάση την (1) να ελαχιστοποιηθεί ο αριθμός των «βαψιμάτων» που απαιτούνται και να καταλήξουμε στη βέλτιστη λύση. Θα χρησιμοποιήσουμε δυναμικό προγραμματισμό στον αριθμό διαστημάτων. Έστω $d[i][j]$ ο μέγιστος αριθμός διαστημάτων θεωρώντας σαν αρχή την πλάκα i και σαν τέλος την πλάκα j.

Για τον υπολογισμό της αναδρομικής σχέσης έχουμε:

- Για να βρούμε το μέγιστο αριθμό διαστημάτων με άκρα τις πλάκες i, j αντίστοιχα, σα βέλτιστη λύση θα κρατήσουμε είτε την τρέχουσα υπολογισμένη, είτε θα ψάξουμε στο διάστημα $i + 1,\ j - 1$, αφού εκεί θα περιέχονται (αν υπάρχουν) τα υπόλοιπα διαστήματα. Έτσι θα έχουμε:

$$d[i][j] = max\ (d[i][j], 1 + d[i+1][j-1])$$

- Αν τα δύο άκρα i, j δεν έχουν κοινό χρώμα, τότε πρέπει να εξετάσουμε όλα τα πιθανά άκρα k μεταξύ των i, j, για να δούμε αν μπορούν να σχηματιστούν διαστήματα μεταξύ των άκρων i, k και k, j. Έτσι θα έχουμε:

$$d[i][j] = max_{i<k<j}\ (d[i][j], d[i][k]\ + d[k][j])$$

Το αποτέλεσμα που θα επιστραφεί εκτελώντας δυναμικό προγραμματισμό είναι το $N - d[0][[N-1]$.

<u>Ανάλυση Πολυπλοκότητας</u>

Ο αλγόριθμος εκτελεί ένα πέρασμα πάνω στον πίνακα εισόδου μέσω του δείκτη i. Για κάθε i διατρέχει άλλη μια φορά τον πίνακα μέσω του δείκτη j, αφού ψάχνει για όλα τα ζεύγη i, j με $i < j$. Τέλος, για κάθε i, j χρειάζεται έλεγχος για τα k μέσω άλλου ενός περάσματος στον πίνακα. Συνεπώς η πολυπλοκότητα του αλγορίθμου είναι $O(n^3)$.

<u>Απόδειξη Ορθότητας</u>

Η ιδέα υπολογισμού διαστημάτων με κοινά άκρα, παρότι αποφεύγει κάποιες επιπλέον αναζητήσεις οι οποίες γίνονται στην brute-force λύση δεν επηρεάζει το αποτέλεσμα, διότι δεν έχει νόημα να χρωματίσουμε σε δύο διαφορετικά «βαψίματα» διαστήματα που θα πάρουν το ίδιο χρώμα, τα χρωματίζουμε με τη μία ώστε να γλιτώσουμε ένα «βάψιμο». Ο αλγόριθμος είναι εξαντλητικός, καθώς δοκιμάζει όλα τα ζεύγη i, j με $i < j$, για αρχή και τέλος ενός διαστήματος.

Άσκηση 2: String matching

1) Για να αντιστοιχίσουμε το μοτίβο p με το κείμενο t, μπορούμε να υπολογίσουμε τους πυρήνες όλων των προθεμάτων του pt και να τους συγκρίνουμε με το p. Εάν ο πυρήνας ενός προθέματος του pt είναι ίσος με το p, τότε γνωρίζουμε ότι αυτό το πρόθεμα του pt ταιριάζει με το μοτίβο p. Αυτό συμβαίνει επειδή ο πυρήνας μιας συμβολοσειράς είναι το μεγαλύτερο πρόθεμα που είναι ταυτόχρονα πρόθεμα και επίθημα της συμβολοσειράς. Επομένως, εάν ο πυρήνας ενός προθέματος του pt είναι ίσος με το p, τότε αυτό το πρόθεμα πρέπει να είναι ένα πρόθεμα του p και ένα επίθημα του pt, πράγμα που σημαίνει ότι πρέπει να ταιριάζει με το μοτίβο p.

2) Ορίζουμε το u ως τον k−πυρήνα της συμβολοσειράς v, όπου k ≥ 0 και v≠ ϵ, αν : u ≺ v, το μήκος του u είναι το πολύ k και το u είναι η μεγαλύτερη συμβολοσειρά με αυτήν την ιδιότητα. Για να δείξουμε ότι ο πυρήνας k− είναι καλά ορισμένος, πρέπει να δείξουμε ότι υπάρχει το πολύ ένας k− πυρήνας για κάθε συμβολοσειρά v και k. Μπορούμε να το αποδείξουμε αυτό με αντίφαση.
Ας υποθέσουμε ότι υπήρχαν δύο πυρήνες k− για μια συμβολοσειρά v, ας πούμε u και w. Τότε τόσο το u όσο και το w θα ήταν προθέματα του v, και τα δύο θα είχαν μήκος το πολύ k, και τα δύο θα ήταν μέγιστα σε σχέση με αυτές τις ιδιότητες. Αυτό θα σήμαινε ότι το u και το w θα ήταν ίσα, κάτι που έρχεται σε αντίθεση με την υπόθεση μας ότι είναι διαφορετικά.

Επομένως, μπορεί να υπάρχει το πολύ ένας k− πυρήνας για κάθε συμβολοσειρά v και k.
Για να υπολογίσουμε τον k− πυρήνα μιας συμβολοσειράς v για ένα δεδομένο k, μπορούμε να χρησιμοποιήσουμε τον ακόλουθο αλγόριθμο:

ΠΕΡΙΓΡΑΦΗ ΑΛΓΟΡΙΘΜΟΥ

- Αρχικοποιούμε ένα κενό σύνολο S για να αποθηκεύουμε όλα τα προθέματα του v που έχουν το πολύ k χαρακτήρες και είναι προθέματα του v.
- Αρχικοποιούμε το u στην κενή συμβολοσειρά.
- Όσο το u δεν είναι ο k-core του v:
- Προσθήκη του u στο S
- Αν u = v, τότε το u είναι ο πυρήνας k και μπορούμε να σταματήσουμε.
- Εάν το μήκος του u είναι k, τότε το u είναι ο πυρήνας k και σταματάμε.
- Έστω c ο πρώτος χαρακτήρας του v που δεν είναι στο u.
- Τότε u = u + c.

Αυτός ο αλγόριθμος λειτουργεί επαναλαμβάνοντας όλα τα προθέματα του v που έχουν το πολύ k χαρακτήρες και είναι προθέματα του v. Διατηρεί ένα σύνολο S όλων των προθεμάτων που έχει συναντήσει μέχρι τώρα. Αν συναντήσει ένα πρόθεμα u που είναι ίσο με v, τότε το u είναι ο k-core και ο αλγόριθμος τερματίζεται. Εάν το μήκος του u είναι k, τότε το u είναι επίσης ο πυρήνας k και ο αλγόριθμος τερματίζεται. Διαφορετικά, ο αλγόριθμος προσθέτει το u στο S και προσθέτει τον επόμενο χαρακτήρα του v στο u για να πάρει το επόμενο πρόθεμα που πρέπει να ληφθεί υπόψη.

Αυτός ο αλγόριθμος είναι εγγυημένο ότι θα τερματιστεί επειδή σημειώνει πάντα πρόοδο. Είναι επίσης αποτελεσματικό επειδή αποθηκεύει μόνο έναν σταθερό αριθμό προθεμάτων στη μνήμη.

Άσκηση 3: Συντομότερα Μονοπάτια με Συντομεύσεις Ενδιάμεσων Ακμών

1)

Περιγραφή Αλγορίθμου

- Αρχικά εκτελούμε τον $Dijkstra$ μία φορά έχοντας ως εναρκτήρια κορυφή την s. Θεωρούμε ως $D_s[u]$ την απόσταση οποιαδήποτε κορυφής u από την s.
- Σχηματίζουμε τον αντίστροφο γράφο (αλλάζοντας την φορά των ακμών) και εκτελούμε τον $Dijkstra$ μία φορά έχοντας ως εναρκτήρια κορυφή την t. Θεωρούμε ως $D_t[u]$ την απόσταση οποιαδήποτε κορυφής u από την t.
- Με αυτόν τον τρόπο, για δύο κορυφές u, v που ανήκουν στο μονοπάτι $s - t$ και συνδέονται με ακμή, θα ισχύει ότι $D_s[t] = D_s[u] + D_t[v] + w(u, v)$. Για να υπολογίσουμε την απόσταση $s - t$ μηδενίζοντας το μήκος της ακμής u, v, απλώς θέτουμε στην παραπάνω σχέση $w(u, v) = 0$. Δηλαδή, θα είναι $D_{0,s}[t] = D_s[u] + D_t[v]$.
- Για κάθε ακμή (u, v) του γράφου υπολογίζουμε την ποσότητα $D_s[u] + D_t[v]$ η οποία αντιστοιχεί στο κόστος ενός μονοπατιού $s - t$ με μηδενισμένη την ακμή (u, v). Επιστρέφουμε την ελάχιστη από αυτές τις ποσότητες.

Ανάλυση Πολυπλοκότητας

Εκτελούμε δύο φορές τον αλγόριθμο $Dijkstra$ και έπειτα κάνουμε ένα πέρασμα από όλες της ακμές του γράφου. Συνεπώς η πολυπλοκότητα είναι $O(mlogn + m) = O(mlogn)$.

Απόδειξη Ορθότητας

Πρέπει να δείξουμε ότι ο αλγόριθμος επιστρέφει σωστό αποτέλεσμα, δηλαδή την απόσταση $s - t$ με μηδενισμένη μία ακμή του $s - t$ μονοπατιού. Οι ποσότητες $D_s[u] + D_t[v]$ που υπολογίζουμε, δίνουν για κάθε ακμή (u, v), το ελάχιστο κόστος να πάμε από την s στην t, διασχίζοντας την ακμή (u, v) με μηδενικό κόστος. Εφόσον εξετάζουμε όλες τις ακμές, αυτό που στα αλήθεια κάνουμε είναι ότι εξετάζουμε όλα τα μονοπάτια από την s στην t και δοκιμάζουμε να μηδενίσουμε όλες τις ακμές που αποτελούν αυτά τα μονοπάτια. Η εξαντλητικότητα του αλγορίθμου λοιπόν, συνεπάγεται και την ορθότητά του.

2)

Περιγραφή Αλγορίθμου

Για την λύση του προβλήματος θα σχηματίσουμε έναν νέο γράφο G' με $(k + 1)$ επίπεδα και συνολικά $n \cdot (k + 1)$ κορυφές. Ως $G'[u][i]$ συμβολίζουμε την κορυφή u στο i-οστό επίπεδο, δηλαδή αυτό στο οποίο έχουμε φτάσει στην u με μηδενισμένες ακριβώς i προηγούμενες ακμές. Διατρέχουμε όλες τις ακμές του αρχικού γράφου και για κάθε ακμή (u, v) με κόστος c, σχηματίζουμε σε κάθε επίπεδο του G' τις εξής ακμές:

- $G'[u][i] - G'[v][i]$ με κόστος c. Σε κάθε επίπεδο του G', πρέπει να υπάρχει η ακμή (u, v) όπως ακριβώς είναι και στον αρχικό γράφο.
- $G'[u][i] - G'[v][i + 1]$ με μηδενικό κόστος. Έχοντας φτάσει στην κορυφή u με μηδενισμένες i ακμές, μηδενίζουμε την ακμή (u, v), ενώνοντας την κορυφή u του i-οστού επιπέδου με την κορυφή v του $(i + 1)$-οστού.

Στη συνέχεια εφαρμόζουμε τον αλγόριθμο του $Dijkstra$ στον G', με αρχική κορυφή την s (στην πραγματικότητα την $G'[s][0]$), και ψάχνουμε το ελάχιστο μονοπάτι μεταξύ αυτών προς την $G'[t][i]$ για οποιοδήποτε i από 0 έως k.

Ανάλυση Πολυπλοκότητας

Για να σχηματίσουμε τον γράφο G' εκτελούμε ένα πέρασμα στο πλήθος των ακμών του αρχικού γράφου. Ο νέος γράφος G' έχει $n \cdot (k + 1)$ κορυφές και $2 \cdot (k + 1) \cdot m$ ακμές ($(k + 1) \cdot m$, διότι έχουμε $(k + 1)$ «αντίγραφα» του αρχικού γράφου και $(k + 1) \cdot m$, διότι τόσες είναι οι ακμές μεταξύ των δύο διαδοχικών επιπέδων). Τέλος, ψάχνουμε το ελάχιστο μονοπάτι από την s προς τις κορυφές $G'[s][i]$. Συνεπώς, η συνολική πολυπλοκότητα του αλγορίθμου είναι

$O(m + 2(k + 1)mlog[n(k + 1)] + 2(k + 1)m) = O\big(km \cdot log(nk)\big)$.

Απόδειξη Ορθότητας

Στον νέο γράφο G' κάθε κορυφή έρχεται σε διαφορετικές «εκδόσεις», ανάλογα με τον αριθμό των ακμών που έχουμε μηδενίσει για να φτάσουμε σε αυτήν την κορυφή. Στον νέο γράφο δεν υπάρχει μόνο ένα μονοπάτι από μια κορυφή u σε μια κορυφή v του αρχικού, αλλά όλα τα δυνατά μονοπάτια όσον αφορά μηδενισμό το πολύ k ακμών στο μονοπάτι. Επιλέγοντας το ελάχιστο από αυτά, βρίσκουμε τελικά το ζητούμενο.

Άσκηση 4: Ταξίδι σε Περίοδο Ενεργειακής Κρίσης

1)
Για κάθε κόμβο i, θεωρώ ότι έχω την τιμή g(i), η οποία αντιπροσωπεύει πόση βενζίνη έχω στο ντεπόζιτο όταν βρεθώ σε εκείνον. Για τον εκάστοτε κόμβο i, θα ψάξω να βρώ τον κόμβο j, για τον οποίο να ισχύει j > i και c[j] < c[i] ή dij > B. Στη περίπτωση που βρώ κάποιον κόμβο j για τον οποίο ισχύει c[j] < c[i] και dij ≤ B, Τότε θα γεμίσω το ντεπόζιτο μου τόσο όσο χρειάζεται ώστε να φτάσω σε εκείνη τη πόλη, δηλαδή θα αγοράσω dij-g(i) λίτρα βενζίνης, και θα συνεχίσω τον αλγόριθμο από εκείνον τον κόμβο j, μηδενίζοντας το ντεπόζιτο g[j]=0. Αν Δεν βρώ κάποιον κόμβο με μικρότερη τιμή σε ακτίνα B, τότε θα αγοράσω βενζίνη μέχρι να γεμίσει το ντεπόζιτο, δηλαδή θα αγοράσω B - g(i) λίτρα, και προχωρήσω στον επόμενο i+1 κόμβο. Στη χειρότερη περιπτωση όλες οι πόλεις θα έχουν γνησίως αύξουσα τιμή βενζίνης και άρα θα γίνουν n^2 επαναλήψεις άμα η χωρητικότητα του ντεποζίτου είναι πολύ μεγάλη. Άρα η πολυπλοκότητα αυτής της greedy λύσης θα είναι :
Τελική πολυπλοκότητα : $O(n^2)$

2)
Θα κάνω ένα BFS στο γράφο G και θα εφαρμόζω σταδιακά τον αλγόριθμο του πρώτου ερωτήματος. Για κάθε νέο κόμβο που θα συναντώ θα υπολογίζω το κόστος για να πάω από τον S στον κόμβο αυτό μέσω ενός συγκεκριμένου μονοπατιού που θα σχηματιστεί από τους πατεράδες των προηγούμενων κόμβων. Αν για ένα κόμβο v εχει υπολογιστεί προηγουμένως ένα κόστος διαδρομής s-v και τον ξανασυναντήσουμε μέσω μία ακμής u→v τότε θα υπολογίσουμε ξανά μία διαδρομή s-v με καινούργιο πατέρα του v τον u μόνο αν το κόστος που έχουμε υπολογίσει μέχρι στιγμής για να πάμε στον u είναι μικρότερο από που έχουμε υπολογίσει για να πάμε στον v. Έπειτα, θα ανανεώσουμε την τιμή του v με το μικρότερο κόστος. Στη χειρότερη περίπτωση, για κάθε ακμή του γράφου θα χρειαστεί να υπολογίσουμε μία καινούργια διαδρομή η οποία θα κοστίζει $O(n^2)$. Άρα η συνολική πολυπλοκότητα θα ανάγεται σε :
Τελική πολυπλοκότητα : $O(E*V^2)$

Άσκηση 5: Παιχνίδια Εξουσίας

Α)
Χρησιμοποιούμε μια παραλλαγή του αλγορίθμου μέγιστης ροής, όπως ο αλγόριθμος Ford-Fulkerson.

<u>Περιγραφή Αλγορίθμου</u>

- Δημιουργούμε έναν κόμβο πηγής 'S' και συνδέουμε με κάθε κόμβο ιππότη μια ακμή με χωρητικότητα ίση με τον αριθμό των καστρών που ο ιππότης μπορεί να επιβλέπει (c_i)
- Συνδέουμε κάθε κόμβο κάστρου με τους σχετικούς ιππότες του στη λίστα 'K_j' με ακμή χωρητικότητας 1.
- Δημιουργούμε έναν κόμβο εκροής 'T' και συνδέουμε με κάθε κόμβο κάστρου μια ακμή με χωρητικότητα ίση με 1

Εφαρμογή του Αλγορίθμου Μέγιστης Ροής:

- Εφαρμόζουμε έναν αλγόριθμο μέγιστης ροής (όπως ο Ford-Fulkerson) για να βρούμε τη μέγιστη ροή στο δίκτυο από το 'S' στο 'T'.

Πολυπλοκότητα

Ο χρόνος πολυπλοκότητας του αλγορίθμου Ford-Fulkerson εξαρτάται από την επιλογή των διαδρομών που αυξάνονται. Στη χειρότερη περίπτωση, μπορεί να είναι O(E^2 *log(C)), όπου V είναι ο αριθμός των κορυφών και E είναι ο αριθμός των ακμών. (E=πλήθος ακμών, C=άθρ. χωρητικ. ακμών που φεύγουν από πηγή)

Β)

Για να πείσει ο Πρωθυπουργός τον Βασιλιά ότι ο αλγόριθμος έχει εκτελεστεί σωστά και ότι δεν υπάρχει ικανοποιητική ανάθεση, μπορεί να χρησιμοποιήσει έναν απλό λόγο βασισμένο στο θεώρημα μέγιστης ροής - ελάχιστης διασποράς (max-flow min-cut theorem).

Το θεώρημα μέγιστης ροής - ελάχιστης διασποράς δηλώνει ότι η μέγιστη ροή σε ένα δίκτυο είναι ίση με την ελάχιστη χωρητικότητα ενός διασποράς στο δίκτυο. Στο πλαίσιο αυτού του προβλήματος, το δίκτυο ροής έχει έναν κόμβο πηγής 'S' και έναν κόμβο εκροής 'T'. Η διασπορά είναι μια διαίρεση των κόμβων σε δύο σύνολα, ένα που περιλαμβάνει το 'S' και ένα που περιλαμβάνει το 'T'. Η ελάχιστη χωρητικότητα της διασποράς αντιπροσωπεύει τον μέγιστο αριθμό των καστέλων που δεν μπορούν να ανατεθούν σε κανέναν ιππότη.

Ορισμός της Τομής:

Λαμβάνουμε υπόψη τη τομή που χωρίζει τους ιππότες (κόμβους που συνδέονται με το 'S') από τα κάστρα (κόμβους που συνδέονται με το 'T').

Υπολογισμός Χωρητικότητας Τομής:

Υπολογίζουμε τη χωρητικότητα αυτής της τομής. Αυτό είναι το άθροισμα των χωρητικοτήτων των ακμών που πηγαίνουν από το σύνολο των ιπποτών στο σύνολο των κάστρων στη τομή.

Σύγκριση με τη Μέγιστη Ροή:

Συγκρίνουμε την υπολογισμένη χωρητικότητα διασποράς με την τιμή της μέγιστης ροής που προκύπτει από τον αλγόριθμο.

Επιχείρημα:

Αν η χωρητικότητα της διασποράς είναι ίση με τη μέγιστη ροή, σημαίνει ότι όλα τα κάστρα καλύπτονται από τους ιππότες, και ο αλγόριθμος έχει βρει μια ικανοποιητική ανάθεση.

Αν η χωρητικότητα της διασποράς είναι μικρότερη από τη μέγιστη ροή, σημαίνει ότι υπάρχουν κάποια κάστρα που δεν μπορούν να ανατεθούν σε κανέναν ιππότη, και ο αλγόριθμος είναι σωστός που συμπεραίνει ότι δεν υπάρχει ικανοποιητική ανάθεση.

Γ)

Για να βρούμε τον ελάχιστο αριθμό ιπποτών που πρέπει να εξοριστούν για να σταματήσουν οι συγκρούσεις, μπορούμε να μοντελοποιήσουμε το πρόβλημα ως πρόβλημα μέγιστου ανεξάρτητου συνόλου σε ένα διμερές γράφημα. Τα δύο χωρίσματα αντιπροσωπεύουν τους Πράσινους και τους Βενετούς. Κάθε σύγκρουση μεταξύ ιπποτών είναι μια άκρη στο γράφημα. Εφαρμόζουμε έναν μέγιστο ανεξάρτητο αλγόριθμο συνόλου σε αυτό το διμερές γράφημα. Οι κορυφές στο ανεξάρτητο σύνολο αντιπροσωπεύουν τους ιππότες που πρέπει να εξοριστούν για να σταματήσουν οι συγκρούσεις. Αυτό θα δώσει στον Βασιλιά έναν αποτελεσματικό τρόπο επίλυσης συγκρούσεων, ελαχιστοποιώντας παράλληλα τον αριθμό των εξορισμένων ιπποτών.

Άσκηση 6: Ενοικίαση Αυτοκινήτων

Διαθέτοντας k ίδια αυτοκίνητα προς ενοικίαση και δεδομένου n προσφορών με κάθε προσφορά i να δεσμεύει ένα όχημα για το χρονικό διάστημα $[s_i, t_i)$ με αντίτιμο p_i θέλουμε να μεγιστοποιήσουμε το συνολικό ποσό που θα εισπράξουμε ως αντίτιμο. Η λύση του προβλήματος ανάγεται στην εύρεση μιας ροής ελαχίστου κόστους σε ένα κατάλληλο διαμορφωμένο δίκτυο G(V,E) που αναπαριστά τους περιορισμούς του προβλήματος. Το δίκτυο G κατασκευάζεται ως εξής:

1. Θεωρούμε μια αρχική κορυφή s, και μια καταληκτική κορυφή t και ένα σύνολο κορυφών v_1 που περιέχει μια κορυφή για κάθε προσφορά. Το σύνολο των κορυφών του δικτύου είναι V = n + 2.
2. Η κορυφή s συνδέεται με ακμή χωρητικότητας 1 με κάθε κορυφή $j \in V_1$.
3. Η κάθε κορυφή $j \in V1$ συνδέεται με ακμή μοναδιαίας χωρητικότητας και κόστους $-p_j$ με την κορυφή t και με κάθε άλλη κορυφή του $i \in V_1$ ώστε να ισχύει $t_j \leq s_i$ με $i \neq j$.
4. Η κορυφή t συνδέεται με την s με ακμή χωρητικότητας k.

Δηλαδή δημιουργούμε έναν γράφο με μέγιστη ροή s-t όσα και τα αυτοκίνητα που διαθέτουμε. Αν κάποια προσφορά j τελειώνει πριν ξεκινήσει κάποια άλλη i μπορεί να χρησιμοποιηθεί το ίδιο αμάξι με την προηγούμενη. Στον γράφο αυτά τα μη επικαλυπτόμενα χρονικά διαστήματα μεταξύ των προσφορών απεικονίζονται με κατευθυνόμενη ακμή $j - i$ κόστους p_j. Ο αλγόριθμος ελαχίστου κόστους θα επιλέξει τα k φθηνότερα μονοπάτια του κατευθυνόμενου ακυκλικού γράφου που δημιουργήσαμε. Η πολυπλοκότητα δημιουργίας

του γράφου είναι O(nlogn)(O(nlogn) για να αποφασίσουμε ποιες προσφορές δεν έχουν επικαλυπτόμενα χρονικά διαστήματα και O(n+m) για την κατασκευή του γράφου). Για τον αλγόριθμο Min Cost Flow η πολυπλοκότητα θα είναι $O(n^2 * m * \log(n * C))$ όπου C είναι η τιμή του μικρότερου κόστους μονοπατιού στο γράφημα.