ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Λύσεις 2ης Σειράς Γραπτών Ασκήσεων για το μάθημα "Αλγόριθμοι και Πολυπλοκότητα"

Δημήτριος Βασιλείου
03119830
7ο Εξάμηνο

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 $N - d[0][[N - 1]$.

Ανάλυση Πολυπλοκότητας

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 $O(n^3)$.

Απόδειξη Ορθότητας

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

Ο αλγόριθμος θα δουλεύει όπως ο KMP με μια τροποποίηση: Θα συμπληρώσουμε τον πίνακα $lps\ (longest\ prefix\ that\ is\ also\ a\ suffix)$ ώστε κάθε φορά που υπάρχει mismatch μεταξύ ενός χαρακτήρα του pattern και ενός του text, ο δείκτης που θα τρέχει στο pattern να γυρνάει στην θέση του μεγαλύτερου prefix που είναι και suffix, αρχίζοντας όμως το prefix από τον χαρακτήρα μετά το $'*'$. Ο κλασσικός KMP αλγόριθμος θα έψαχνε για prefix, αρχίζοντας από τον 1ο χαρακτήρα του pattern, εξαιτίας όμως της ύπαρξης των μπαλαντέρ $*$, μπορούμε να θεωρήσουμε ότι έχει γίνει match μεταξύ όλων των χαρακτήρων πριν το μπαλαντέρ.

Δηλαδή αν υποθέσουμε ότι ένα $*$ βρίσκεται στη θέση x του pattern, τότε όλοι οι χαρακτήρες μετά από το $*$ και μέχρι το επόμενο $*$ θα έχουν $lps \geq x + 1$. Για παράδειγμα, για το pattern $ABR*CADABR\ CAD*BRA$ ο πίνακας lps θα μοιάζει ως εξής:

i	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
text[i]	*A*	*B*	*R*	*	*C*	*A*	*D*	*A*	*B*	*R*	*C*	*A*	*D*	*	*B*	*R*	*A*
lps[i]	*0*	*0*	*0*	-	*5*	*5*	*5*	*5*	*5*	*5*	*5*	*6*	*7*	-	*15*	*15*	*15*

Επίσης, όταν ο αλγόριθμος «σαρώνει» το text και το pattern, αν στο pattern συναντάει $*$, τότε παγώνει ο δείκτης j του pattern, και όσο ισχύει ότι $pattern[j+1] \neq text[i]$ προχωράει μόνο ο δείκτης i του text. Αυτό, διότι το $*$ ταιριάζει με οποιαδήποτε συμβολοσειρά.

Στη συνέχεια παραθέτουμε ψευδοκώδικα για τον αλγόριθμο:

```
Repeat
    if pattern[j] ≠ '*'
        while text[i] = pattern[j]
            i++; j++;
        if text[i] ≠ pattern[j]
            j = lps[j – 1];
    if pattern[j] = '*'
        while text[i] ≠ pattern[j+1]
            i++;
        j++;
        while text[i] = pattern[j]
            i++; j++;
        if text[i] ≠ pattern[j]
            j = lps[j – 1];
```

Επειδή ο αλγόριθμος δουλεύει αρκετά παρόμοια με τον KMP, η πολυπλοκότητά του είναι $O(m+n)$ όπου m είναι το μέγεθος του pattern και n το μέγεθος του text.

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

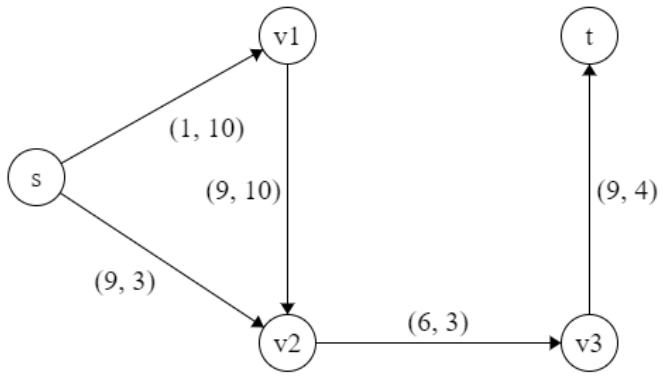
Για να σχηματίσουμε τον γράφο G' εκτελούμε ένα πέρασμα στο πλήθος των ακμών του αρχικού γράφου. Ο νέος γράφος G' έχει $n \cdot (k+1)$ κορυφές και $2 \cdot (k+1) \cdot m$ ακμές ($(k+1) \cdot m$, διότι έχουμε $(k+1)$ «αντίγραφα» του αρχικού γράφου και $(k+1) \cdot m$, διότι τόσες είναι οι ακμές μεταξύ 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: Σύντομα Μονοπάτια με Επαρκή Μεταφορική Ικανότητα

1). Θεωρούμε τον ακόλουθο γράφο:



Εύκολα βλέπουμε ότι τα $s-t$ μονοπάτια στον γράφο αυτόν είναι δύο, το $p_1 = \{s, v_2, v_3, t\}$, καθώς και το $p_2 = \{s, v_1, v_2, v_3, t\}$. Βέλτιστο είναι το p_1 αφού $c(p_1)/b(p_1) = 24/3 = 8$, ενώ $c(p_2)/b(p_2) = 25/3 = 8{,}33$. Θεωρούμε δύο προθέματα των μονοπατιών p_1, p_2, το ${p_1}' = \{s, v_2\}$ και το ${p_2}' = \{s, v_1, v_2\}$, τα οποία καταλήγουν στον κόμβο v_2. Για αυτά ισχύει $c({p_1}')/b({p_1}') = 9/3 = 3$ και $c({p_2}')/b({p_2}') = 10/10 = 1$, δηλαδή το πρόθεμα ${p_2}'$ είναι βέλτιστο, ενώ το ${p_1}'$ δεν είναι. Αποδείξαμε λοιπόν με αυτό το αντιπαράδειγμα ότι μπορεί ένα $s-t$ μονοπάτι να είναι βέλτιστο, ωστόσο δεν ισχύει απαραίτητα το ίδιο για τα προθέματά του.

2). Η αναδρομική σχέση βασίζεται στον αλγόριθμο που περιγράφεται στο ερώτημα (3) και γίνεται στις ακμές του γραφήματος. Με T συμβολίζουμε τον βέλτιστο λόγο. Η σχέση είναι η ακόλουθη:

$$T(G(V,E),p) = min\begin{cases} \dfrac{c(p)}{b(p)} \\ T(G(V, E - \{e: b(e) < b(p)\}), \quad p') \end{cases}$$

όπου p' το τρέχον βέλτιστο μονοπάτι στο νέο γράφο, που προέκυψε με αφαίρεση των κατάλληλων ακμών. Η αναδρομική σχέση μας εξηγεί πως δεδομένου ενός τρέχοντος βέλτιστου μονοπατιού σε έναν γράφο $G(V,E)$, η βέλτιστη λύση προκύπτει ως το ελάχιστο μεταξύ του τρέχοντος βέλτιστου πηλίκου $\frac{c(p)}{b(p)}$, είτε με αναδρομικές κλήσεις στον γράφο που προκύπτει με αφαίρεση των ακμών $e: b(e) < b(p)$, δηλαδή όσων έχουν χωρητικότητα μικρότερη από το τρέχον βέλτιστο μονοπάτι.

3).

<u>Περιγραφή Αλγορίθμου:</u>

Η ιδέα για την υλοποίηση του αλγορίθμου βασίζεται στην εκτέλεση του αλγορίθμου του $Dijkstra$ για την εύρεση ελάχιστων μονοπατιών από τον κόμβο s προς τον t. Συγκεκριμένα:

- Εκτελούμε τον αλγόριθμο $Dijkstra$ στον γράφο $G(n,m)$ και βρίσκουμε τα ελάχιστα μονοπάτια με βάση το κόστος, από τον κόμβο s προς όλους τους υπόλοιπους. Μας ενδιαφέρει το ελάχιστο κόστος του $s-t$ μονοπατιού, έστω $\min\{c(s-t)\}$. Επίσης υπολογίζουμε και την μεταφορική ικανότητα του μονοπατιού, την $b(\min\{s-t\})$.
- Έχοντας υπολογίσει το ελάχιστο $s-t$ μονοπάτι, γνωρίζουμε ότι τα υπόλοιπα $s-t$ μονοπάτια έχουν κόστος μεγαλύτερο από $\min\{c(s-t)\}$. Συνεπώς, για να αποτελεί κάποιο άλλο μονοπάτι τη βέλτιστη λύση, πρέπει αναγκαστικά να έχει μεταφορική ικανότητα μεγαλύτερη από $b(\min\{s-t\})$.
- Η παραπάνω παρατήρηση οδηγεί στο συμπέρασμα ότι αν κάποιο άλλο $s-t$ μονοπάτι είναι βέλτιστο, αποκλείεται να χρησιμοποιεί ακμές που έχουν μεταφορική ικανότητα μικρότερη από $b(\min\{s-t\})$, διότι εφόσον έχει κόστος μεγαλύτερο από $\min\{c(s-t)\}$, τότε ο ζητούμενος λόγος θα είναι μεγαλύτερος από αυτόν του τρέχοντος βέλτιστου μονοπατιού, άτοπο. Οπότε, μπορούμε να διαγράψουμε από το γράφο τις ακμές με χωρητικότητα μικρότερη από $b(\min\{s-t\})$ και να συνεχίσουμε αναδρομικά την ίδια διαδικασία.
- Έχοντας διαγράψει τις απαραίτητες ακμές, εκτελούμε τον αλγόριθμο $Dijkstra$, βρίσκουμε το ελάχιστο $s-t$ μονοπάτι στο νέο γράφο, βρίσκουμε τη χωρητικότητα του ελάχιστου μονοπατιού, συγκρίνουμε τον λόγο με τον τρέχοντα βέλτιστο και αν ο νέος λόγος είναι μικρότερος, κρατάμε αυτόν σα βέλτιστο και συνεχίζουμε αναδρομικά, αλλιώς συνεχίζουμε αναδρομικά με τον «παλιό» βέλτιστο λόγο.
- Αν σε κάποιο βήμα ο αλγόριθμος υπολογίσει άπειρο κόστος βέλτιστου $s-t$ μονοπατιού, σημαίνει ότι δεν υπάρχει τέτοιο μονοπάτι, οπότε τερματίζει και επιστρέφει τον τρέχοντα βέλτιστο λόγο και το μονοπάτι που τον επιτυγχάνει. Στη χειρότερη περίπτωση, θα αφαιρεθούν όλες οι ακμές του αρχικού γράφου, οπότε ο αλγόριθμος θα τερματίσει μετά από m επαναλήψεις.

<u>Ανάλυση Πολυπλοκότητας</u>

Ο αλγόριθμος εκτελεί πολλές φορές τον αλγόριθμο του $Dijkstra$, στη χειρότερη περίπτωση m φορές. Επίσης, απαιτείται υπολογισμός της μεταφορικής ικανότητας κάθε ελάχιστου μονοπατιού ως προς το κόστος. Η διαδικασία αυτή, μπορεί να ενσωματωθεί στην υλοποίηση του $Dijkstra$ με απλές τροποποιήσεις οι οποίες δεν αλλάζουν την συνολική πολυπλοκότητα του $Dijkstra$. Συνεπώς, η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(m \cdot mlogn) = O(m^2 logn)$.

Απόδειξη Ορθότητας

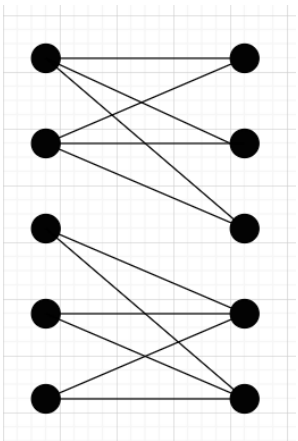
Η ορθότητα του αλγορίθμου έγκειται στην παρατήρηση που διατυπώσαμε προηγουμένως, στην περιγραφή του αλγορίθμου. Ένα ελάχιστο ως προς το κόστος $s-t$ μονοπάτι, έστω p κόστους $c(p)$ έχει μια ακμή ελάχιστης χωρητικότητας, έστω e_{min}. Αυτό που μας ενδιαφέρει είναι ο ελάχιστος λόγος c/b μεταξύ όλων των $s-t$ μονοπατιών. Το p έχει λόγο $c(p)/b(p)$. Ξέρουμε ότι το p έχει ελάχιστο κόστος, συνεπώς όλα τα υπόλοιπα $s-t$ μονοπάτια έχουν κόστος μεγαλύτερο από $c(p)$. Για να έχουν «ελπίδα» αυτά να είναι βέλτιστα, πρέπει να έχουν χωρητικότητα μεγαλύτερη από $b(p)$, συνεπώς αποκλείεται να χρησιμοποιούν ακμές χωρητικότητας μικρότερης από $b(p)$, διότι αν τις χρησιμοποιούσαν, δεν θα ήταν βέλτιστα. Άρα, μπορούμε να διαγράψουμε αυτές τις ακμές χωρίς να χάσουμε περιπτώσεις υποψήφιων βέλτιστων μονοπατιών. Συνεχίζοντας αναδρομικά στο νέο γράφο με λιγότερες ακμές, ο αλγόριθμος επιστρέφει σωστό αποτέλεσμα, δηλαδή τον ελάχιστο λόγο και το μονοπάτι στον οποίο επιτυγχάνεται, αφού κάθε φορά κρατάει και ενημερώνει το βέλτιστο αποτέλεσμα.

Άσκηση 5: Αγορές Προϊόντων από Συγκεκριμένα Καταστήματα

1).

Περιγραφή Αλγορίθμου

Αρχικά θα αναπαραστήσουμε το πρόβλημα με ένα διμερές γράφημα. Το ένα ανεξάρτητο σύνολο θα είναι τα καταστήματα S και το άλλο θα είναι τα προϊόντα P. Οι συσχετίσεις που μας ενδιαφέρουν είναι το γεγονός ότι καταστήματα πουλάνε προϊόντα, συνεπώς θα υπάρχουν ακμές μόνο από το S στο P και κάθε τέτοια ακμή (έστω (s_i, p_j)) δηλώνει ότι το κατάστημα s_i πουλάει το προϊόν p_j. Προφανώς, ένα κατάστημα μπορεί να πουλάει πολλά προϊόντα οπότε κάθε s_i μπορεί να συνδέεται με περισσότερους από έναν κόμβους του P και αντίστοιχα κάθε προϊόν μπορεί να πωλείται σε περισσότερα του ενός καταστήματα, συνεπώς σε κάθε p_j μπορούν να καταλήγουν πολλές ακμές από το S. Μια αναπαράσταση για το πρόβλημα μπορεί να είναι η ακόλουθη:



Σημειώνουμε ότι το S είναι το αριστερό σύνολο και το P το δεξιό.

Εφόσον ψάχνουμε ένα υποσύνολο του S από το οποίο μπορούμε να προμηθευτούμε όλα τα προϊόντα αγοράζοντας ένα προϊόν από κάθε κατάστημα, ουσιαστικά αναζητούμε κορυφές του S και ακμές που έχουν τις εξής ιδιότητες:

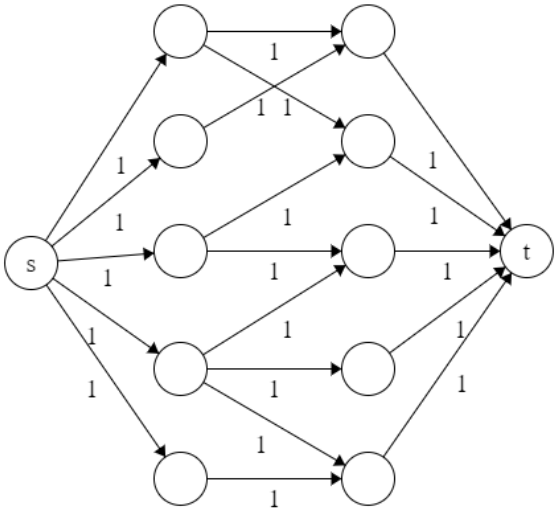
- ➔ Από κάθε κορυφή να «φεύγει» ακριβώς μία ακμή, καθώς δεν μπορούμε να προμηθευτούμε πάνω από ένα προϊόντα από κάποιο κατάστημα. Δηλαδή δεν θέλουμε οι ακμές που θα βρούμε και ξεκινάνε από κορυφές του S να έχουν κοινό άκρο στο S.
- ➔ Το σύνολο των ακμών που θα βρούμε πρέπει να περιλαμβάνει όλες τις κορυφές του P, διότι πρέπει να προμηθευτούμε όλα τα προϊόντα. Κι εδώ μας ενδιαφέρει οι ακμές που καταλήγουν

σε κορυφές του P να μην έχουν κοινό άκρο στο P, επειδή δεν μας ενδιαφέρει να αγοράσουμε κάποιο προϊόν από δύο διαφορετικά καταστήματα.

Με βάση τα παραπάνω, παρατηρούμε ότι το πρόβλημα που προσπαθούμε να λύσουμε είναι αυτό του μέγιστου ταιριάσματος, με τη διαφορά ότι πρέπει να συμπεριλαμβάνονται όλες οι κορυφές του ενός συνόλου (εδώ του P), γιατί πρέπει να προμηθευτούμε όλα τα προϊόντα.

Θα χρησιμοποιήσουμε την μέθοδο της αναγωγής, ανάγοντας το πρόβλημα σε αυτό της μέγιστης ροής-ελάχιστης τομής με τον εξής τρόπο:

- Τοποθετούμε δύο νέες κορυφές στο γράφημα, την s και την t και ενώνουμε την s με όλες τις κορυφές του S και την t με όλες τις κορυφές του P.
- Τοποθετούμε χωρητικότητα 1 σε όλες τις ακμές του γραφήματος.



- Αυτό μας εξασφαλίζει ότι οι ακμές μεταξύ των σημείων του S και του P δεν θα έχουν κοινά άκρα. Διότι αν υπήρχε κόμβος από τον οποίο «περνάει» η ροή και διέρχονται χβγ 2 ακμές τότε αφού εξέρχεται ροή μιας μονάδας προς το t, δεν θα ίσχυε η αρχή διατήρησης της ροής σε αυτόν τον κόμβο(*).
- Τρέχουμε τον αλγόριθμο $Ford-Fulkerson$ για την εύρεση μέγιστης ροής. Όταν τελειώσει η εκτέλεση του αλγόριθμου, θα ανατεθεί σε κάθε ακμή μία ροή, η οποία θα είναι είτε 0 είτε 1, αφού η χωρητικότητα κάθε ακμής είναι 1. Ελέγχουμε όλες τις ακμές και κρατάμε όσες έχουν χωρητικότητα 1, διότι από αυτές ουσιαστικά «ταξιδεύει» η ροή προς την κορυφή t.
- Με βάση την παρατήρηση (*) γνωρίζουμε ότι οι ακμές μέσω των οποίων «ταξιδεύει» η ροή και συνδέουν κορυφές από το S στο P δεν έχουν κοινά άκρα. Ωστόσο δεν είμαστε σίγουροι ότι όλες οι κορυφές τόσο του S όσο και του P ανήκουν σε κάποιες από τις ακμές.
- Κρατάμε τις κορυφές του S από τις οποίες περνούν ακμές και ορίζουμε το σύνολο $S' \subseteq S$ που αντιστοιχεί στο σύνολο καταστημάτων.
- Αυτό που μένει να ελέγξουμε είναι αν από τις κορυφές του S' εξέρχονται ακμές προς όλες τις κορυφές του P, ώστε να εξασφαλίσουμε πως προμηθευόμαστε όλα τα προϊόντα. Επομένως σχηματίζουμε το σύνολο $P' \subseteq P$ όπου οι κορυφές του P' περιλαμβάνουν όλες τις κορυφές του P στις οποίες φτάνει ακμή από τις κορυφές του S'. Αν $P' = P$ τότε μπορούμε να προμηθευτούμε όλα τα προϊόντα και ο αλγόριθμος επιστρέφει $true$, αλλιώς επιστρέφει $false$.

<u>Ανάλυση Πολυπλοκότητας</u>

Η δημιουργία των κορυφών s, t,η ένωσή τους με τις κορυφές των συνόλων S, P αντίστοιχα, και η τοποθέτηση χωρητικότητας σε όλες τις ακμές «κοστίζουν» χρόνο $O(m)$. Η εκτέλεση του αλγορίθμου $Ford-Fulkerson$ με $blocking-flow$ τεχνική θέλει χρόνο $O(nmlogn)$. Οι υπόλοιπες διαδικασίες που εκτελεί ο αλγόριθμος, δηλαδή η δημιουργία των συνόλων S', P' και οι υπόλοιποι έλεγχοι είναι

γραμμικοί ως προς το πλήθος των κορυφών και των ακμών. Συνεπώς η «βαριά» δουλειά του αλγορίθμου γίνεται στην εκτέλεση του $Ford - Fulkerson$. Τελικά η συνολική πολυπλοκότητα είναι $O(nmlogn)$.

<u>Απόδειξη Ορθότητας</u>

Από τα παραπάνω είναι σαφές ότι το πρόβλημα που προσπαθούμε να λύσουμε είναι αυτό του μέγιστου ταιριάσματος, με την διαφορά ότι θέλουμε να περιλαμβάνονται στις ακμές όλες οι κορυφές του ενός συνόλου, καθώς θέλουμε να προμηθευτούμε όλα τα προϊόντα. Ανάγουμε αυτό το πρόβλημα σε αυτό της μέγιστης ροής με δύο επιπλέον κορυφές $s - t$ που συνδέονται με όλες τις κορυφές των συνόλων S και P αντίστοιχα και τοποθετούμε χωρητικότητα 1 σε όλες τις ακμές. Η εκτέλεση του αλγορίθμου $Ford - Fulkerson$ θα βρει τη μέγιστη ροή αλλά αυτό που πραγματικά μας ενδιαφέρει είναι ότι κάθε ακμή θα λάβει ροή $0,1$ δηλαδή δεν χρησιμοποιείται ή χρησιμοποιείται στη μέγιστη ροή αντίστοιχα. Για τις ακμές που χρησιμοποιούνται στο «ταξίδι» της ροής ισχύει ο ακόλουθος ισχυρισμός:

- ➔ Οι ακμές αυτές δεν μπορούν να έχουν κοινά άκρα, κάτι το οποίο μας εξασφαλίζει ότι κάθε κορυφή του S συνδέεται με το πολύ μία κορυφή του P, ή αλλιώς, κάθε κατάστημα που θα βρούμε πουλάει ακριβώς ένα προϊόν.

Η απόδειξη είναι απλή: Έστω κορυφή $u \in P$ στην οποία καταλήγουν 2 ακμές από κορυφές του P, μετά την εκτέλεση του $Ford - Fulkerson$. Η ροή λοιπόν που εισέρχεται στην κορυφή ισούται με 2. Η κορυφή u όμως συνδέεται με την t και η ακμή (u,t) έχει ροή 1. Συνεπώς δεν ισχύει η αρχή διατήρησης της ροής για την κορυφή u και καταλήγουμε σε άτοπο.

Έχοντας αποδείξει τον ισχυρισμό και έχοντας εξασφαλίσει ότι η μέγιστη ροή «δίνει» το μέγιστο ταίριασμα αυτό που μένει είναι να ελέγξουμε αν όλες οι κορυφές του S' συνδέονται με ακμές του μέγιστου ταιριάσματος με όλες τις κορυφές του P, ώστε όλα τα προϊόντα να προμηθεύονται από όλα τα καταστήματα, πράγμα το οποίο κάνει ο αλγόριθμός μας.

2).

<u>Περιγραφή Αλγορίθμου</u>

Αντίστοιχα με το προηγούμενο ερώτημα, αναπαριστούμε το πρόβλημα με ένα διμερές γράφημα. Αναζητούμε ένα μέγιστο σύνολο προϊόντων και καταστημάτων $S' \cup P', S' \subseteq S, P' \subseteq P$ τέτοιο ώστε κανένα από τα προϊόντα του P'να μην πωλείται από τα καταστήματα του S'. Με άλλα λόγια, ψάχνουμε δύο υποσύνολα των δύο συνόλων S, P που αποτελούν το διμερές γράφημα και δεν υπάρχουν ακμές μεταξύ αυτών των δύο. Επειδή η ένωση των δύο αυτών υποσυνόλων πρέπει να είναι μέγιστη ως προς τον πληθικό αριθμό, αυτό που στα αλήθεια ψάχνουμε είναι το μέγιστο ανεξάρτητο σύνολο στο διμερές γράφημα.

Θα χρησιμοποιήσουμε την μέθοδο της αναγωγής, ανάγοντας το πρόβλημα σε αυτό της μέγιστης ροής-ελάχιστης τομής με τον εξής τρόπο:

- Τοποθετούμε δύο νέες κορυφές στο γράφημα, την s και την t και ενώνουμε την s με όλες τις κορυφές του S και την t με όλες τις κορυφές του P.
- Τοποθετούμε χωρητικότητα 1 στις ακμές μεταξύ της s και των κορυφών του S και μεταξύ της t και των κορυφών του P. Επίσης τοποθετούμε άπειρη χωρητικότητα στις ακμές μεταξύ των συνόλων S, P.
- Σημειώνουμε ότι το γράφημα που έχουμε σχηματίσει είναι κατευθυνόμενο με τις ακμές να «κατευθύνονται» από την s προς την t.
- Εφαρμόζουμε τον αλγόριθμο $Ford - Fulkerson$ για την εύρεση μέγιστης ροής-ελάχιστης τομής. Έστω (X,Y) η ελάχιστη τομή με $s \in X, t \in Y$. Έστω επίσης $S' = S \cap X$ και $P' = P \cap Y$. Το σύνολο $S' \cup P'$ είναι ανεξάρτητο σύνολο, διότι:

-Δεν υπάρχουν ακμές μεταξύ των κορυφών του S', αφού αυτές ανήκουν στην ίδια «μεριά» του διμερούς γραφήματος.
-Δεν υπάρχουν ακμές μεταξύ των κορυφών του P' για τον ίδιο λόγο.
-Δεν υπάρχουν ακμές μεταξύ των κορυφών του S' και του P', διότι αυτές έχουν άπειρη χωρητικότητα, συνεπώς δεν λαμβάνονται υπόψιν κατά την εκτέλεση του αλγορίθμου για την εύρεση ελάχιστης τομής. Η τομή πρέπει να είναι ελάχιστη συνεπώς θα την διασχίζουν μόνο ακμές με μοναδιαία χωρητικότητα. Εδώ σημειώνουμε ότι επειδή το γράφημα είναι κατευθυνόμενο, στην χωρητικότητα της τομής λαμβάνονται υπόψιν μόνο οι ακμές που κατευθύνονται από το X προς το Y.

Ανάλυση Πολυπλοκότητας

Η τοποθέτηση των κορυφών s, t, η δημιουργία των νέων ακμών και η τοποθέτηση χωρητικοτήτων, κοστίζουν γραμμικό χρόνο ως προς τις ακμές του γραφήματος. Η εκτέλεση του αλγορίθμου $Ford-Fulkerson$ με $blocking-flow$ τεχνική θέλει χρόνο $O(nmlogn)$. Ο σχηματισμός των συνόλων S', P' και της ένωσης γίνεται σε γραμμικό χρόνο ως προς τις κορυφές. Άρα η συνολική πολυπλοκότητα του αλγορίθμου είναι $O(nmlogn)$.

Απόδειξη Ορθότητας

Μέχρι αυτό το σημείο δείξαμε ότι το σύνολο που θα σχηματιστεί μετά από την εκτέλεση του αλγορίθμου εύρεσης ελάχιστης τομής είναι ανεξάρτητο σύνολο, ωστόσο δεν έχουμε δείξει ότι αυτό είναι το μέγιστο δυνατό, όπως και ζητείται στο πρόβλημα.
Ο αλγόριθμος θα μας επιστρέψει μια s, t τομή ελάχιστης χωρητικότητας, (X, Y). Έχουμε βρει τα σύνολα S', P' και έχουμε δείξει ότι η ένωσή τους είναι ανεξάρτητο σύνολο. Η χωρητικότητα της τομής ισούται με τη χωρητικότητα των ακμών που διασχίζουν την τομή και έχουν κατεύθυνση από το σύνολο S προς το σύνολο T. Επειδή οι ακμές αυτές έχουν μοναδιαία χωρητικότητα, μπορούμε να πούμε ότι η χωρητικότητα της τομής ισούται με το πλήθος των ακμών που διασχίζουν την τομή και έχουν κατεύθυνση από το S προς το T. Το πλήθος αυτό είναι ίσο με
$|S - S'| + |P - P'| = |S| + |P| - |S' \cup P'|$. Επειδή η ποσότητα $|S| + |P|$ είναι σταθερή, για να είναι μέγιστο το ανεξάρτητο σύνολο $|S' \cup P'|$, πρέπει να είναι ελάχιστη η χωρητικότητα της τομής. Εφόσον έχουμε βρει μια ελάχιστη τομή με τον αλγόριθμο $Ford-Fulkerson$, συμπεραίνουμε ότι και το ανεξάρτητο σύνολο είναι μέγιστο, άρα ο αλγόριθμός μας είναι ορθός.