Βραχύτερα Μονοπάτια σε Γράφους

(CLR, κεφάλαιο 25)

Στην ενότητα αυτή θα μελετηθούν τα εξής θέματα:

Βραχύτερα Μονοπάτια για όλα τα Ζεύγη

Λύση Δυναμικού Προγραμματισμού

Ο αλγόριθμος των Floyd-Warshal

Βραχύτερα Μονοπάτια

- Με δεδομένο ένα κατευθυνόμενο γράφο με βάρη G=(V,E), και μια συνάρτηση βαρών w: E→R, θέλουμε να βρούμε,

  για κάθε ζεύγος κορυφών (u, v) το βάρος του ελάχιστου μονοπατιού μεταξύ των δύο κορυφών στον G.

- Δηλαδή, αν |V|=n, θέλουμε να υπολογίσουμε ένα πίνακα n×n ο οποίος να περιέχει όλα τα δ(u,v).

Λύση 1

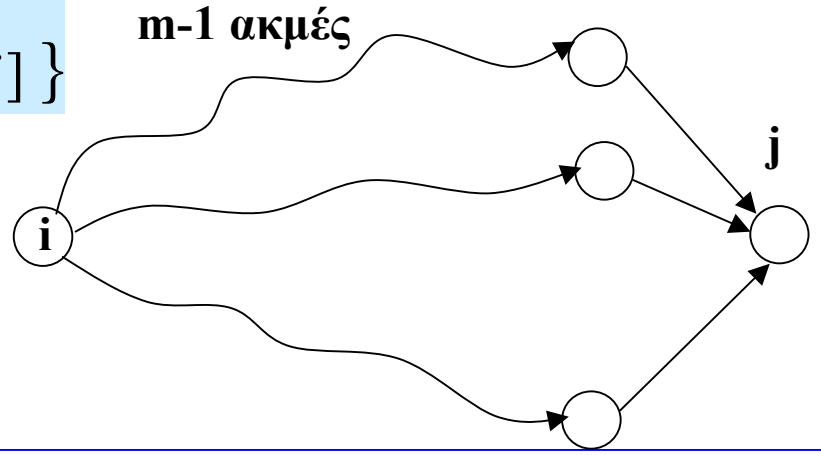
- Τρέξε τον αλγόριθμο Bellman-Ford, n φορές, μια φορά για κάθε κορυφή.
- Πολυπλοκότητα λύσης 1: $n \cdot \Theta(n \cdot |E|) = \Theta(n^2 \cdot |E|)$

Λύση Δυναμικού Προγραμματισμού

- Παριστάνουμε το γράφο με τον πίνακα βαρών W, όπου W[i,i]=0, για κάθε i.
- *Συμβολισμός:* d[i,j,m] συμβολίζει το βάρος του βραχύτερου μονοπατιού από την κορυφή i στην κορυφή j το οποίο χρησιμοποιεί ≤m ακμές.
- Αναδρομικός ορισμός της βέλτιστης λύσης:

$$d[i,j,0] = \begin{cases} 0, & \text{αν } i = j \\ \infty, & \text{αν } i \neq j \end{cases}$$

$$d[i,j,m] = \min_k \{ d[i,k,m-1] + w[k,j] \}$$



Λύση Δυναμικού Προγραμματισμού

- Έστω D_m ο πίνακας που περιέχει στη θέση $D_m[i,j]$ την τιμή d[i,j,m].
- Η πιο κάτω διαδικασία υπολογίζει τον πίνακα D_m, υποθέτοντας γνώση των πινάκων D_{m-1} και W.

```
distance(m){

for(i=1; i≤n; i++)
   for(j=1; j≤n; j++)
      for (k=1; k≤n; k++)
          d[i,j,m]= min(d[i,j,m-1],
                        d[i,k,m-1] + w[k,j])
}
```

- Χρονική Πολυπλοκότητα της distance: $\Theta(n^3)$

Λύση Δυναμικού Προγραμματισμού

- Δεδομένου ότι δεν υπάρχουν κύκλοι αρνητικού βάρους, λύση στο πρόβλημα δίνεται από τις τιμές του πίνακα D_{n-1}.
- Άρα θα πρέπει να εφαρμόσουμε τη διαδικασία distance, n-1 φορές.
- Σημείωση: Ο πίνακας D_0 μπορεί να υπολογιστεί εύκολα…
- Συνολική χρονική πολυπλοκότητα: $(n-1)\cdot\Theta(n^3)=\Theta(n^4)$.
- Όχι καλύτερη από την πολυπλοκότητα της λύσης 1.

Σχέση του προβλήματος με πολλαπλασιασμό πινάκων

- Θυμηθείτε τον ορισμό του πολλαπλασιασμού πινάκων:

$$c_{ij} = \sum_{k} a_{ik} \cdot b_{kj}$$

- Συγκρίνετέ το με

$$d[i, j, m] = \min_k \{ d[i, k, m-1] + w[k, j] \}$$

- Η τελευταία εξίσωση μπορεί να θεωρηθεί σαν ένας *"γενικευμένος" πολλαπλασιασμός* των πινάκων D_{m-1} και W για να δώσουν τον πίνακα D_m, όπου οι πράξεις
  - × και + , του συνήθους πολλαπλασιασμού,
    έχουν αντικατασταθεί με τις πράξεις
  - + και **min** αντίστοιχα.

Σχέση του προβλήματος με πολλαπλασιασμό πινάκων

- Μπορούμε λοιπόν να γράψουμε

  $D_m = D_{m-1} \otimes W$

  πράξη "γενικευμένου" πολλαπλασιασμού.

- Το ουδέτερο στοιχείο του "γενικευμένου" πολλαπλασιασμού είναι ο πίνακας

$$I = \begin{pmatrix} 0 & \infty & \dots & \infty \\ \infty & 0 & \dots & \infty \\ \infty & \infty & \dots & \infty \\ & & & \\ \infty & \infty & \dots & 0 \end{pmatrix}$$

- Παρατηρούμε ότι $I=D_0$!!

Βελτίωση 1

- Έχουμε λοιπόν

$$D_1 = D_0 \otimes W = W$$
$$D_2 = D_1 \otimes W = W \otimes W = W^2$$
$$\ldots$$
$$D_{n-1} = D_{n-2} \otimes W = W^{n-2} \otimes W = W^{n-1}$$

- Άρα με βάση αυτή την ιδέα, υπολογίζουμε τις n-1 "γενικευμένες" δυνάμεις του πίνακα W, και επιστρέφουμε σαν αποτέλεσμα τον πίνακα W^{n-1}.

- Χρονική πολυπλοκότητα:

Βελτίωση 2

- Μετατρέπουμε τον πιο πάνω αλγόριθμο ως εξής:
- Αντί να υπολογίζουμε όλες τις "γενικευμένες" δυνάμεις υπολογίζουμε μόνο τις "γενικευμένες" δυνάμεις σε δύναμη του 2, δηλαδή, υποθέτοντας απλοποιητικά ότι το n-1 είναι δύναμη του 2:

$$W, W^2, W^4, \ldots, W^{2^{\lg(n-1)}} = W^{n-1}$$

χρησιμοποιώντας το γεγονός ότι

$$W^k \otimes W^k = W^{2k}$$

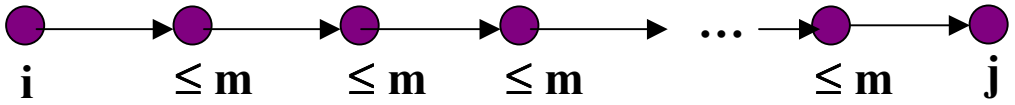
το οποίο ισχύει, αφού η πράξη $\otimes$ είναι προσεταιριστική (γιατί;).

- Χρονική Πολυπλοκότητα: $\Theta(\lg n)\cdot\Theta(n^3) = \Theta(\lg n \cdot n^3)$

πλήθος των "γενικευμένων τετραγωνισμών"

Ο αλγόριθμος των Floyd-Warshall

- Χρησιμοποιεί επίσης δυναμικό προγραμματισμό, αλλά είναι ταχύτερος κατά έναν συντελεστή lg n.
- Συμβολισμός: c[i,j,m] συμβολίζει το βάρος του βραχύτερου μονοπατιού από την κορυφή i στην κορυφή j με ενδιάμεσες κορυφές που ανήκουν στο σύνολο {1,2,…m}.

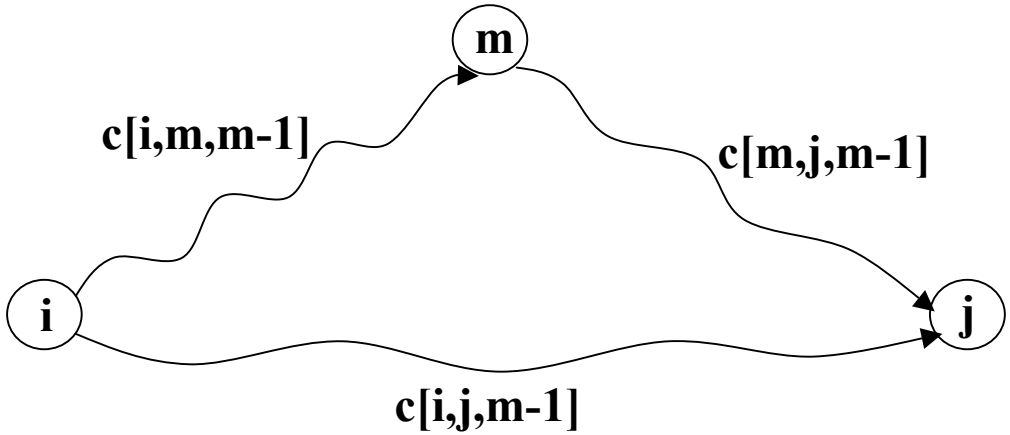


- Αναδρομικός ορισμός της βέλτιστης λύσης:

$$c[i,j,0] = w[i,j]$$
$$c[i,j,m] = \min(c[i,j,m-1], c[i,m,m-1] + c[m,j,m-1])$$

Ο αλγόριθμος των Floyd-Warshall

- Επεξήγηση του αναδρομικού βήματος:



- Παρατήρηση: σημαντική διαφορά από την προηγούμενη αναδρομική σχέση είναι ότι

  τώρα εξετάζεται μόνο μια καινούρια κορυφή πηγαίνοντας από το c[i,j,m-1] στο c[i,j,m], ενώ η προηγούμενη εξέταζε όλες τις κορυφές.

Ο αλγόριθμος των Floyd-Warshal

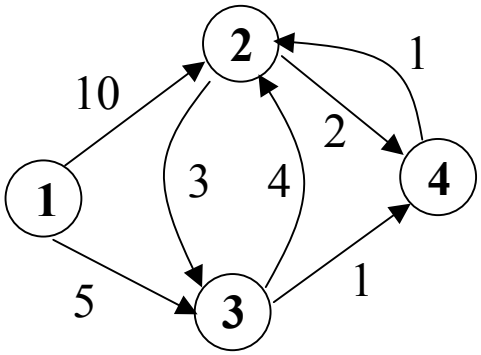
- Μπορούμε να υπολογίσουμε τα c[i,j,m] ως εξής:

```
cost(){

for all i,j c[i,j,0] = w(i,j);
for(m=1; m≤n; m++)
   for (i=1; i≤n; i++)
      for (j=1; j≤n; j++)
         c[i,j,m]= min(c[i,j,m-1],
                       c[i,m,m-1]+c[m,j,m-1])
}
```

- Χρονική πολυπλοκότητα:
- Για πυκνούς γράφους ($|E| = n^2$), ο αλγόριθμος των Floyd-Warshall δίνει όλα τα βραχύτατα μονοπάτια στον ίδιο χρόνο που ο αλγόριθμος των Bellman-Ford δίνει ένα μόνο βραχύτατο μονοπάτι.

Παράδειγμα



$$C[0] = \begin{pmatrix} 0 & 10 & 5 & \infty \\ \infty & 0 & 3 & 2 \\ \infty & 4 & 0 & 1 \\ \infty & 1 & \infty & 0 \end{pmatrix} \qquad C[1] = C[0] \qquad C[2] = \begin{pmatrix} 0 & 10 & 5 & 12 \\ \infty & 0 & 3 & 2 \\ \infty & 4 & 0 & 1 \\ \infty & 1 & 4 & 0 \end{pmatrix}$$

$$C[3] = \begin{pmatrix} 0 & 9 & 5 & 6 \\ \infty & 0 & 3 & 2 \\ \infty & 4 & 0 & 1 \\ \infty & 1 & 4 & 0 \end{pmatrix} \qquad C[4] = \begin{pmatrix} 0 & 7 & 5 & 6 \\ \infty & 0 & 3 & 2 \\ \infty & 2 & 0 & 1 \\ \infty & 1 & 4 & 0 \end{pmatrix}$$