Βραχύτερα Μονοπάτια σε Γράφους

(CLR, κεφάλαιο 25)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

Ο αλγόριθμος των Bellman-Ford

Ο αλγόριθμος του Dijkstra

Βραχύτερα Μονοπάτια σε Γράφους

- Γενίκευση της αναζήτησης κατά βάθος ή πλάτος σε γράφους με βάρη.
- Με δεδομένο ένα κατευθυνόμενο γράφο με βάρη G=(V,E) και μια συνάρτηση βαρών w: E→R, θέλουμε να βρούμε μονοπάτια με το ελάχιστο δυνατό βάρος.
- Υπενθύμιση: Το βάρος w(p) ενός μονοπατιού p δίνεται ως εξής:

$$\text{Αν} \quad p = v_o \rightarrow v_1 \rightarrow \ldots \rightarrow v_k$$

$$\text{τότε} \quad w(p) = \sum_{i=1}^{k} w(v_{i-1}, v_i)$$

- **Ορισμός:** ***Βραχύτερο μονοπάτι*** μεταξύ ενός συνόλου από μονοπάτια είναι το μονοπάτι με το ελάχιστο βάρος.

Η δομή της βέλτιστης λύσης

Λήμμα 1

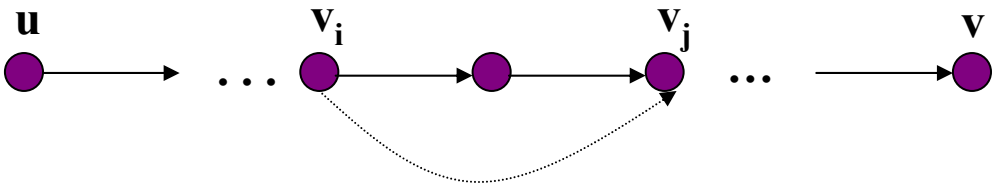
Έστω κατευθυνόμενος γράφος με βάρη G=(V,E) και έστω

$$p = u \rightarrow v_1 \rightarrow \ldots \rightarrow v_k \rightarrow v$$

το βραχύτερο μονοπάτι μεταξύ των κόμβων u και v.

Τότε κάθε υπομονοπάτι του p είναι βραχύτερο.

Απόδειξη



- Ας υποθέσουμε ότι το πιο πάνω είναι το βραχύτερο μονοπάτι μεταξύ των κόμβων u και v και ότι το υπομονοπάτι $v_i \Rightarrow v_j$ *δεν είναι βραχύτερο* ανάμεσα στα μονοπάτια που συνδέουν τους κόμβους p και s.
- Τότε αν αντικαταστήσουμε το βραχύτερο τέτοιο υπομονοπάτι στο αρχικό μονοπάτι θα παίρναμε ένα βραχύτερο μονοπάτι μεταξύ των κόμβων u και v. Αντίφαση!

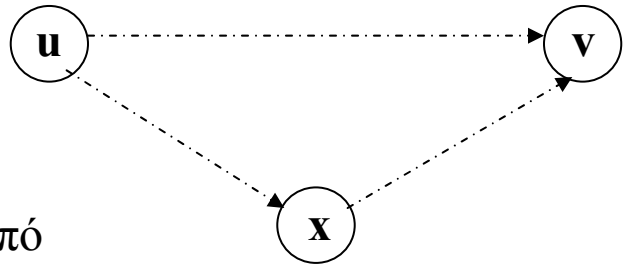
Η δομή της βέλτιστης λύσης

Συμβολισμός: Θα γράφουμε $\delta(u,v)$ για το βάρος του βραχύτερου μονοπατιού από την κορυφή u στην κορυφή v.

Λήμμα 2 (Τριγωνική ανισότητα)

Για κάθε τριάδα κορυφών u, v, x, $\delta(u,v) \leq \delta(u,x) + \delta(x,v)$

Απόδειξη

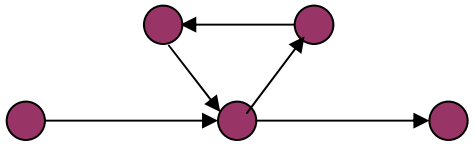


Το ελάχιστο μονοπάτι από τον u στον v δεν είναι μακρύτερο από οποιοδήποτε άλλο μονοπάτι από τον u στο v – εν προκειμένω, το μονοπάτι που παίρνει πρώτα το ελάχιστο μονοπάτι από τον u στον κόμβο x και στη συνέχεια από τον x στον v.

□

□

- Τι συμβαίνει σε γράφους με αρνητικά βάρη;
- Τι συμβαίνει αν υπάρχει κύκλος με αρνητικό βάρος;



Αλγόριθμος Bellman-Ford

- Ο αλγόριθμος βρίσκει τα βάρη των ελάχιστων μονοπατιών από μια δεδομένη πηγή s προς κάθε κορυφή $v \in V$.

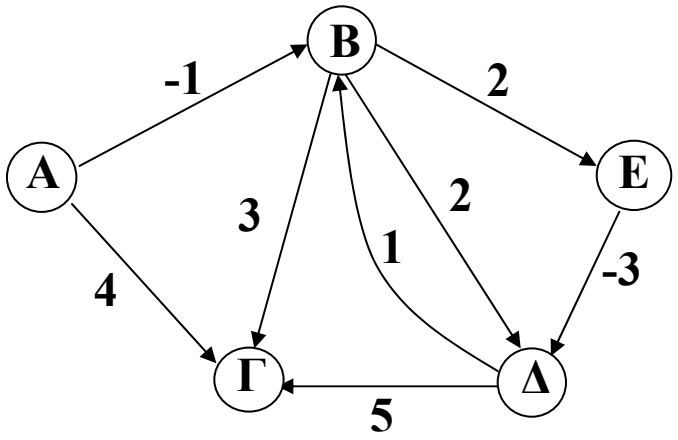
```
for all v∈V
  d[v] = ∞;
d[s]=0;

for (i=1; i<|V|; i++)
   for all edges (u,v)∈E
      if (d[v] > d[u]+w(u,v))
         d[v]= d[u] + w(u,v);

for all edges (u,v)∈E
   if (d[v] > d[u]+ w(u,v))
       NO SOLUTION
```

Εξήγηση του αλγόριθμου

1. Στην πρώτη φάση του αλγόριθμου αποδίδονται οι αρχικές τιμές στα d.
2. Για |V| - 1 φορές γίνεται 'χαλάρωση' της κάθε ακμής, δηλαδή μικραίνει η απόσταση της κάθε ακμής από την πηγή s.
3. Τελικά γίνεται έλεγχος κατά πόσο οι τιμές που έχουν δοθεί είναι πράγματι λύσεις (αυτό συμβαίνει εφόσον δεν υπάρχει κύκλος με αρνητικό κόστος).

- Χρόνος Εκτέλεσης: O(|V|·|E|)

Παράδειγμα



Έστω ότι ακμές τυγχάνουν επεξεργασία με την εξής σειρά: (Α,Β), (Α,Γ), (Β,Γ), (Β,Δ), (Δ,Γ), (Ε,Δ), (Β,Ε).

φάση/κορυφή	Α	Β	Γ	Δ	Ε
αρχικές τιμές	0	∞	∞	∞	∞
χαλάρωση 1	0	-1	~~4~~ 2	1	1
χαλάρωση 2	0	-1	2	-2	1

Οι τιμές για τα d που παίρνουμε σε κάθε φάση χαλάρωσης, καθώς και η ταχύτητα σύγκλισης (αριθμός χαλαρώσεων) είναι συνάρτηση της σειράς επεξεργασίας των ακμών.

Ορθότητα του Αλγόριθμου

Δείχνουμε ότι d[v] =δ(s,v) μετά από |V|-1 χαλαρώσεις. Αρχικά έχουμε το πιο κάτω λήμμα.

Λήμμα 3 Πάντοτε $d[v] \geq \delta(s,v)$.

Απόδειξη:

- Αρχικά αληθής.
- Υποθέτουμε (για να φθάσουμε σε αντίφαση) ότι υπάρχουν κορυφές για τις οποίες δεν ισχύει η πρόταση του λήμματος.

  Έστω v η κορυφή για την οποία συμβαίνει για πρώτη φορά ότι $d[v] < \delta(s,v)$.

  Έστω u η κορυφή που προκάλεσε αλλαγή της d[v]: $d[v] = d[u] + w(u,v)$.

  Τότε

  $$\begin{aligned} d[v] &< \delta(s,v) \\ &\leq \delta(s,u) + \delta(u,v) \\ &\leq \delta(s,u) + w(u,v) \\ &\leq d[u] + w(u,v) \end{aligned}$$

  το οποίο αποτελεί αντίφαση στο ότι $d[v] = d[u] + w(u,v)$. ■

Ορθότητα του Αλγόριθμου

Θεώρημα 1

Ο αλγόριθμος Bellman-Ford είναι ορθός, δηλ. μετά από |V|-1 χαλαρώσεις όλες οι τιμές του d είναι σωστές (εφόσον τα ελάχιστα μονοπάτια υπάρχουν).

Απόδειξη

- Έστω v μια κορυφή, και θεωρήστε το ελάχιστο μονοπάτι από την πηγή s στη v (υποθέτοντας ότι δεν περνά κύκλος αρνητικού κόστους από κάποια κορυφή πάνω σε κάποιο μονοπάτι από την s στη v):

$$s \to v_1 \to \ldots \to v_k \to v$$

- Αρχικά, d[s]=0, το οποίο είναι ορθό και δεν μεταβάλλεται στη συνέχεια (ο κώδικάς μπορεί μόνο να μειώσει το d, ενώ από το Λήμμα 3, πάντοτε $d[s] \geq \delta(s,s) = 0$.)

- Μετά από την πρώτη χαλάρωση (πρώτο πέρασμα μέσα από τις ακμές), η τιμή $d[v_1]$ είναι ορθή και δεν μεταβάλλεται στη συνέχεια, διότι:

Ορθότητα του Αλγόριθμου

- Όπως ήδη δείξαμε, η τιμή της d[s] είναι ορθή και, από τη δομή της βέλτιστης λύσης, η ελάχιστη απόσταση από το s στο v_1 είναι ίση με $w(s, v_1)$. Το πρώτο πέρασμα θέτει

  $$d[v_1] = d[s] + w(s, v_1)$$

  που είναι η σωστή απάντηση, και η τιμή αυτή δεν μεταβάλλεται στη συνέχεια, γιατί, γράφοντας $d'[v_1]$ για τη τιμή που παίρνει η $d[v_1]$ σε οποιαδήποτε επόμενη στιγμή της εκτέλεσης, τότε

  $$d'[v_1] \geq \delta(s, v_1)$$
  $$d'[v_1] \leq d[v_1]$$
  $$= \delta(s, v_1)$$

- Παρόμοια, ισχύει ότι, μετά από τη δεύτερη χαλάρωση, η τιμή $d[v_2]$ είναι ορθή και δεν μεταβάλλεται στη συνέχεια…

Ανάλυση

- Ο αλγόριθμος τερματίζει μετά από |V|-1 χαλαρώσεις διότι

  Αν δεν υπάρχουν κύκλοι με αρνητικό κόστος, κάθε ελάχιστο μονοπάτι είναι απλό και το ΄μακρύτερο΄ απλό μονοπάτι έχει μήκος |V|-1.

  Έτσι αν δεν υπάρχει κύκλος αρνητικού κόστους, τότε όλα τα d συγκλίνουν στη σωστή τους τιμή μετά από |V|-1 χαλαρώσεις.

  Ισοδύναμα, αν μια τιμή d αποτύχει να συγκλίνει μετά από |V|-1 χαλαρώσεις, τότε υπάρχει κύκλος αρνητικού βάρους.

  Επίσης ισχύει το αντίστροφο:

  αν υπάρχει κύκλος αρνητικού βάρους πάνω σε κάποιο μονοπάτι από την s στη v, τότε η d[v] αποτυγχάνει να συγκλίνει στη σωστή τιμή μετά από |V|-1 χαλαρώσεις.

Αλγόριθμος του Dijkstra

- Δουλεύει όταν όλα τα βάρη είναι μη-αρνητικά.
- Έστω ότι θέλουμε να βρούμε τα βραχύτερα μονοπάτια από κάποιο κόμβο s προς όλους τους υπόλοιπους κόμβους σε κάποιο γράφο G.
- Ο αλγόριθμος αυτός διατηρεί ένα πίνακα S από κορυφές όπου αποθηκεύει τις κορυφές του G, για τις οποίες το μήκος του βραχύτερου μονοπατιού έχει υπολογισθεί.
- Επίσης διατηρεί ένα πίνακα d όπου για κάθε κόμβο Β του γράφου φυλάει την ανά πάσα στιγμή μικρότερη απόσταση του κόμβου Β από τον κόμβο s την οποία γνωρίζει.
- Αρχικά $S=\varnothing$ και $d[|V|] = \infty$.
- Ο αλγόριθμος επανειλημμένα διαλέγει την κοντινότερη κορυφή Β προς τον s , που δεν έχει μέχρι στιγμής επεξεργασθεί (δηλαδή $B \in V-S$), και ελέγχει αν για οποιοδήποτε γείτονα Γ της Β χρήση της ακμής (Β,Γ) μπορεί να δημιουργήσει βραχύτερο μονοπάτι προς την Γ.

Αλγόριθμος του Dijkstra

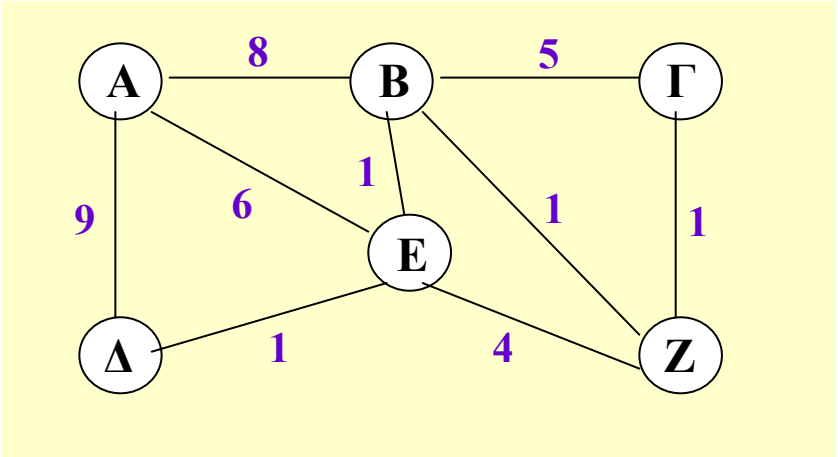
- Χρησιμοποιεί ουρά προτεραιότητας, Q, για αποθήκευση ακμών, όπου η προτεραιότητα δίνεται από το d[v].
- S είναι το σύνολο των κορυφών i για τα οποία d[i]=δ(s,i).

```
heap Q;
for all v∈V
    d[v]=∞;
d[s]=0;

S=∅;
Q=V;
while (Q ≠ ∅){
    u=DeleteMin(Q);
    S=S∪{u};
    για κάθε γείτονα v του u
        if d[v]>d[u]+w(u,v)
            d[v]=d[u]+w(u,v);
}
```

Παρατηρήσεις

1. Η χαλάρωση είναι η ίδια με τη χαλάρωση στον αλγόριθμο Bellman-Ford.
2. Κάθε χαλάρωση ισοδυναμεί με μείωση κλειδιού στη δομή δεδομένων που υλοποιεί την ουρά προτεραιότητας.
3. Ομοιότητα με τον αλγόριθμο του Prim;

Παράδειγμα



S	d(A)	d(B)	d(Γ)	d(Δ)	d(E)	d(Z)
S=∅	0	∞	∞	∞	∞	∞
S={A}	0	8	∞	9	6	∞
S={A,E}	0	7	∞	7	6	10
S={A,E,B}	0	7	12	7	6	8
S={A,E,B,Δ}	0	7	12	7	6	8
S={A,E,B,Δ,Z}	0	7	9	7	6	8
S={A,E,B,Δ,Z, Γ}	0	7	9	7	6	8

Αλγόριθμος του Dijkstra – Εύρεση ΒΜ

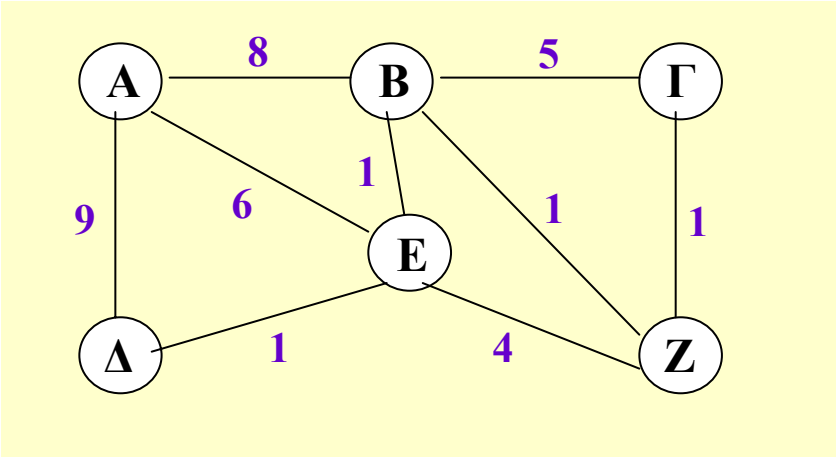
- Αν εκτός από το μήκος του μονοπατιού μας ενδιαφέρει και το ακριβές μονοπάτι (οι κόμβοι του) τότε θα πρέπει σε ένα πίνακα, έστω P, να φυλάγουμε και κορυφές ως εξής:
  - κάθε φορά που χρήση μιας κορυφής X διευκολύνει την εύρεση βραχύτερου μονοπατιού προς μια κορυφή Y, τότε φυλάσσουμε το όνομα της κορυφής: P[Y] = X.

- Σε αυτή την περίπτωση, πως μπορούμε με τον τερματισμό του αλγόριθμου να κατασκευάσουμε από τον πίνακα P το μέγιστο μονοπάτι από τον κόμβο εκκίνησης προς κάποιον άλλο κόμβο X;

Αλγόριθμος του Dijkstra με εύρεση ΒΜ

```
heap Q;
for all v∈V
        d[v]=∞; P[v]= '-';
d[s]=0;

S=∅;
Q=V;
while (Q ≠ ∅){
        u=DeleteMin(Q);
        S=S∪{u};
    για κάθε γείτονα v του u
                if d[v]>d[u]+w(u,v)
                        d[v]=d[u]+w(u,v);
                        P[v] = u;
   }
```

Παράδειγμα



S	P(A)	P(B)	P(Γ)	P(Δ)	P(E)	P(Z)
S=∅	-	-	-	-	-	-
S={A}	-	A	-	A	A	-
S={A,E}	-	E	-	E	A	E
S={A,E,B}	-	E	B	E	A	B
S={A,E,B,Δ}	-	E	B	E	A	B
S={A,E,B,Δ,Z}	-	E	Z	E	A	B
S={A,E,B,Δ,Z, Γ}	-	E	Z	E	A	B

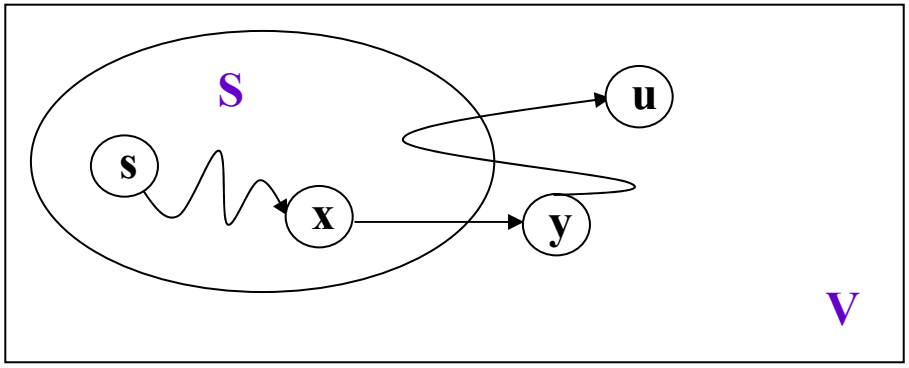
Απόδειξη ορθότητας

Θεώρημα 2 Όταν μια κορυφή u 'μπαίνει' στο S, τότε d[u] = δ(s,u).

Απόδειξη

- Υποθέτουμε, για να φτάσουμε σε αντίφαση, ότι η u είναι η πρώτη κορυφή η οποία, κατά την εκτέλεση του αλγόριθμου, μπαίνοντας στο S έχει d[u] μεγαλύτερο (προσέξτε Λήμμα 4) από το βάρος του βραχύτερου μονοπατιού μεταξύ της u και της s.

- Έστω y η `πρώτη΄ κορυφή στο V-S που ανήκει στο βραχύτερο μονοπάτι από την s στη u.



- Αφού η x μπήκε στο S πριν από την u, d[x]=δ(s,x).

- Επίσης, με την εισαγωγή του x στο S, ετέθει d[y]=d[x]+w(x,y), το οποίο είναι το κόστος του υπομονοπατιού στο σχήμα.

Απόδειξη ορθότητας

- Αφού το μονοπάτι από το s στο u, είναι βραχύτερο, τότε από τη δομή βέλτιστης λύσης συνεπάγεται ότι το υπομονοπάτι $s \Rightarrow x \rightarrow y$ από το s στο y είναι επίσης βραχύτερο. Άρα $d[y] = \delta(s,y)$.

- Έτσι:

$$
\begin{aligned}
d[u] &> \delta(s,u) && \text{(αρχική υπόθεση)} \\
&= \delta(s,y) + \delta(y,u) && \text{(δομή βέλτιστης λύσης)} \\
&= d[y] + \delta(y,u) && (d[y] = \delta(s,y)) \\
&\geq d[y] && \text{(τα κόστη είναι } \geq 0)
\end{aligned}
$$

- Αφού $d[u] > d[y]$, ο αλγόριθμος θα διάλεγε και θα εισήγαγε τη y στο S και όχι τη u. **Αντίφαση!**

□

Λήμμα 4 Καθ'όλη τη διάρκεια του αλγόριθμου $d[u] \geq \delta(s,u)$.
Η απόδειξη είναι παρόμοια με αυτή του Λήμματος 3.

- Χρόνος Εκτέλεσης: $|V| \cdot \log|V| + |E|$

Αλγόριθμος του Dijkstra - Υλοποίηση

```
heap Q;
for all v∈V
     d[v]=∞; P[v]= '-';
     Insert(Q, (v,d[v]));

d[s]=0; DecreaseKey(Q,s,0);

S=∅;
while (!IsEmpty(Q)){
     u=DeleteMin(Q);
     S=S∪{u};
     για κάθε v γείτονα του u
        if d[v]>d[u]+w(u,v)
            d[v]=d[u]+w(u,v);
            DecreaseKey(Q,v, d[v]);
            P[v] = u;
}
```

Μείωσε την τιμή του ζεύγους που αφορά το στοιχείο s στη σωρό Q έτσι ώστε να γίνει (s,0)