Γράφοι (συνέχεια)

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

Ο αλγόριθμος Dijkstra για εύρεση βραχυτέρων μονοπατιών

Τα μονοπάτια Euler

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

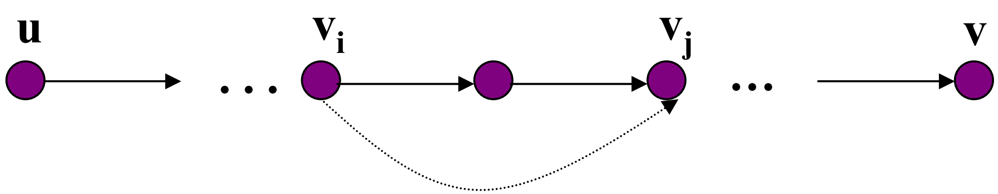
Έστω κατευθυνόμενος γράφος με βάρη G=(V,E) και έστω

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

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!

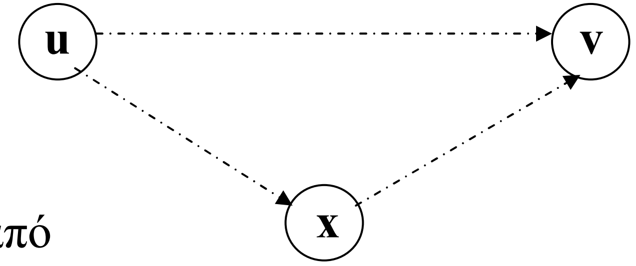
Η δομή της βέλτιστης λύσης

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

Λήμμα 1 (Τριγωνική ανισότητα)

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

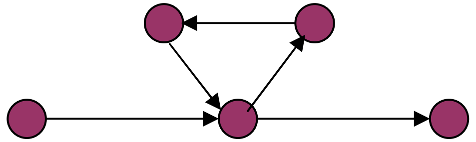
Απόδειξη



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 του Dijkstra

- Δουλεύει όταν όλα τα βάρη είναι μη-αρνητικά.
- Έστω ότι θέλουμε να βρούμε τα βραχύτερα μονοπάτια από κάποιο κόμβο A 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 B του γράφου φυλάει την ανά πάσα στιγμή μικρότερη απόσταση του κόμβου B από τον κόμβο A την οποία γνωρίζει.
- Αρχικά $S=\emptyset$ και $d[|V|] = \infty$.
- Ο αλγόριθμος επανελημένα διαλέγει την κοντινότερη κορυφή B προς τον A, που δεν έχει μέχρι στιγμής επεξεργασθεί (δηλαδή $B \in V-S$), και ελέγχει αν για οποιοδήποτε γείτονα Γ της B χρήση της ακμής (B,Γ) μπορεί να δημιουργήσει βραχύτερο μονοπάτι προς την Γ.

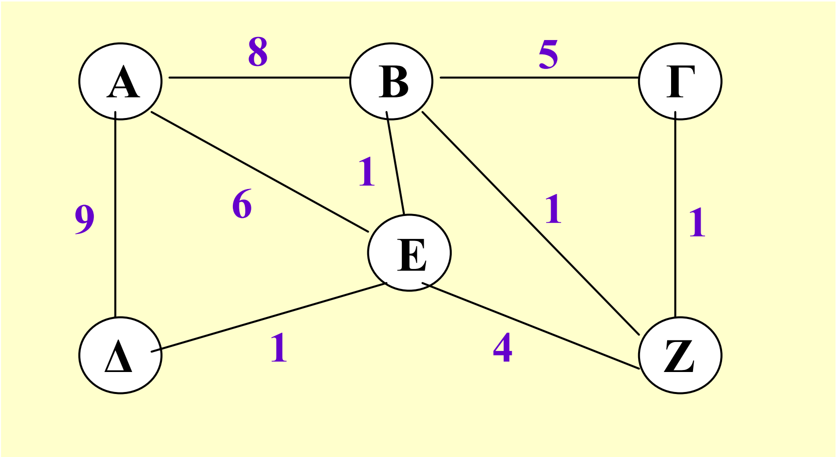
Αλγόριθμος του Dijkstra

- 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);
}
```

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.

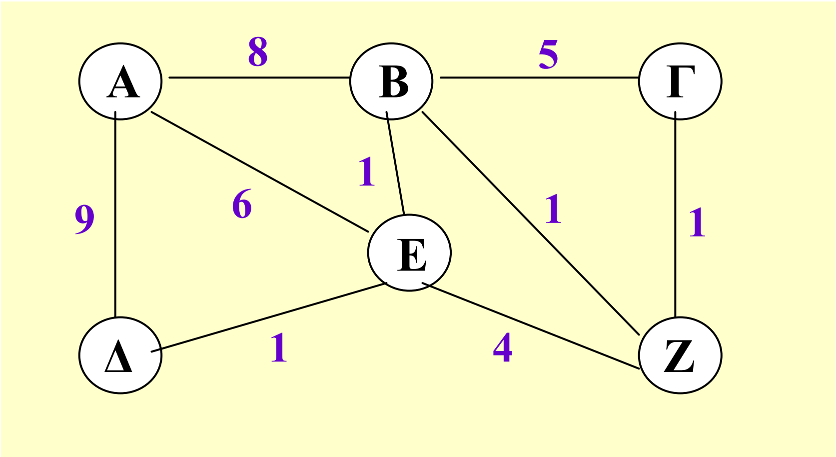
- Σε αυτή την περίπτωση, πως μπορούμε με τον τερματισμό του αλγόριθμου να κατασκευάσουμε από τον πίνακα P το μέγιστο μονοπάτι από τον κόμβο εκκίνησης προς κάποιον άλλο κόμβο X;

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] = w;
  }
```

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

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)

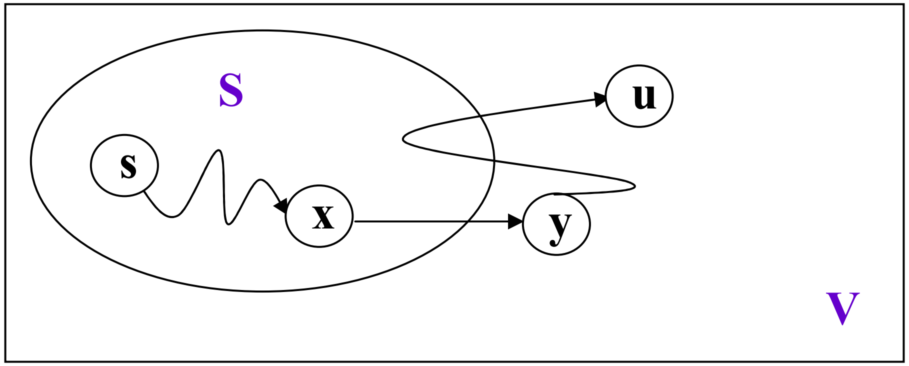
Απόδειξη ορθότητας

Θεώρημα 1 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 2) 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:

$d[u]$	$> \delta(s,u)$	(αρχική υπόθεση)
	$= \delta(s,y) + \delta(y,u)$	(δομή βέλτιστης λύσης)
	$= d[y] + \delta(y,u)$	($d[y] = \delta(s,y)$)
	$\geq d[y]$	(τα κόστα είναι ≥ 0)

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

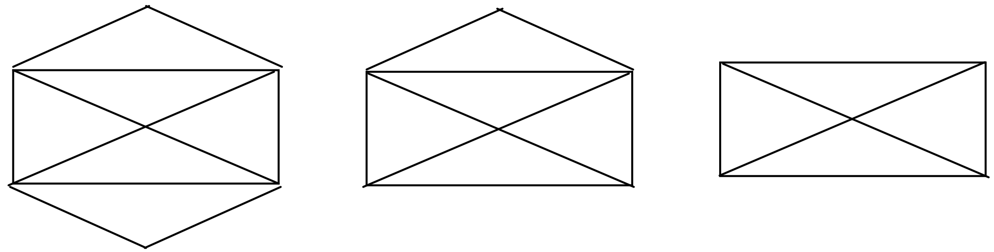
□

<u>Λήμμα 2</u> Καθ'όλη τη διάρκεια του αλγόριθμου $d[u] \geq \delta(s,u)$.
Η απόδειξη είναι παρόμοια με αυτή του Λήμματος 1.

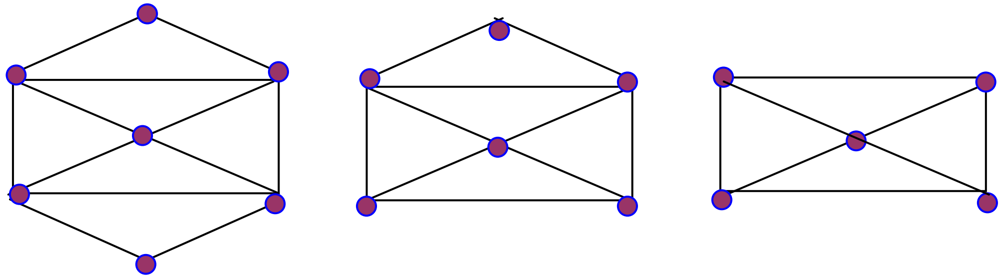
- Χρόνος Εκτέλεσης: $(|V| + |E|) \cdot \log|V|$

Μονοπάτια Euler

- (Παράδειγμα Εφαρμογής κατά βάθους διερεύνησης)
- Σπαζοκεφαλιά: μπορούμε να δημιουργήσουμε τα πιο κάτω σύνολα γραμμών ζωγραφίζοντας κάθε γραμμή ακριβώς μια φορά χωρίς να σηκώσουμε την πένα από το χαρτί;



- Το πρόβλημα μπορεί να μετατραπεί σε πρόβλημα γράφων: Κτίζουμε γράφο G του οποίου κόμβοι είναι τα σημεία όπου διασταυρώνονται γραμμές του σχεδίου και ακμή μεταξύ δύο κόμβων υπάρχει αν υπάρχει γραμμή μεταξύ των αντίστοιχων σημείων του σχεδίου.



Μονοπάτια Euler

- Το πρόβλημα τώρα είναι να διακριβώσουμε την ύπαρξη μονοπατιού που περνά από κάθε ακμή ακριβώς μια φορά. Τέτοια μονοπάτια ονομάζονται ***μονοπάτια Euler***, από το μαθηματικό Euler ο οποίος έλυσε το πρόβλημα το 1732.

- Ένας γράφος με n κόμβους έχει μονοπάτι Euler αν και μόνο αν
  1. είναι συνεκτικός,
  2. τουλάχιστον n-2 από τους κόμβους του έχουν

  Degree mod 2 = 0

  όπου Degree ενός κόμβου v είναι ο αριθμός των ακμών που ξεκινούν απ'αυτόν.

- Η δεύτερη ιδιότητα οφείλεται στο ότι για όλους εκτός από τον πρώτο και τον τελευταίο κόμβο κάθε φορά που το μονοπάτι 'μπαίνει' σε ένα κόμβο θα πρέπει και να 'βγει'.

- Πόσα μονοπάτια Euler έχει ένας γράφος;

Εύρεση Μονοπατιού Euler

- Πως μπορούμε να βρούμε μονοπάτια Euler; Υπάρχει αλγόριθμος βασισμένος σε DFS ο οποίος πετυχαίνει το στόχο σε χρόνο γραμμικό.
- Έστω ότι γνωρίζουμε την ύπαρξη μονοπατιού Euler σε ένα γράφο. Εφαρμόζουμε τα εξής βήματα:
  1. διαλέγουμε μια από τις κορυφές για τις οποίες Indegree≠Outdegree αν τέτοια κορυφή υπάρχει, διαφορετικά οποιαδήποτε άλλη κορυφή.
  2. Εφαρμόζουμε κατά βάθος διερεύνηση μέχρις ότου να μην μπορούμε να προχωρήσουμε. Έστω ότι παίρνουμε το μονοπάτι
     $$s = e_1 \dots e_n$$
  3. Βρίσκουμε την πρώτη ακμή e =(u,v) στο μονοπάτι s στην οποία ο αρχικός κόμβος, u, έχει παιδιά τα οποία δεν έχουν ήδη εξερευνηθεί και από εκεί ξεκινούμε DFS διερεύνηση στο μέρος του γράφου που παραμένει ανεξερεύνητο. Έστω ότι λαμβάνουμε το μονοπάτι s΄. Προσθέτουμε το s΄ στο s ως εξής:
     $$s = e_1 \dots e_{i-1} s' e_i \dots e_n$$
  4. Επαναλαμβάνουμε το βήμα 3 μέχρις ότου να εξερευνηθεί ολόκληρος ο γράφος και επιστρέφουμε το μονοπάτι s.