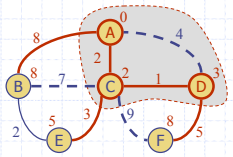
Το συντομότερο μονοπάτι

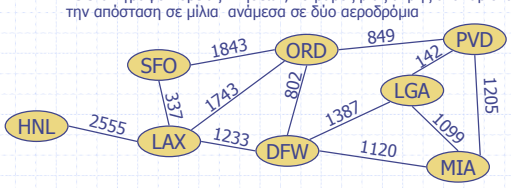


Κύρια σημεία για μελέτη

- Συντομότερο μονοπάτι
  - Ζυγισμένος γράφος
  - Το πρόβλημα του συντομότερου μονοπατιού
  - Ιδιότητες του συντομότερου μονοπατιού
- Ο αλγόριθμος του Dijkstra(§7.1.1)
  - Ο αλγόριθμος
  - Edge relaxation
  - Παράδειγμα
  - Ανάλυση

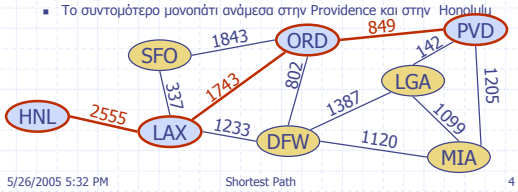
Ζυγισμένος γράφος

- Σε έναν ζυγισμένο γράφο, κάθε ακμή έχει μια σχετιζόμενη αριθμητική τιμή, που ονομάζεται βάρος της ακμής
- Τα βάρη των ακμών μπορεί να αναπαριστούν,αποστάσεις, κόστη, κλπ.
- παράδειγμα:
  - Σε έναν γράφο πορείας πτήσεων, το βάρος μιας ακμής αναπαριστά την απόσταση σε μίλια ανάμεσα σε δύο αεροδρόμια



Το πρόβλημα του συντομότερου μονοπατιού

- Δοθέντος ενός ζυγισμένου γράφου και δύο κόμβων *u* και *v*, θέλουμε να βρούμε το μονοπάτι του ελάχιστου συνολικού βάρους ανάμεσα στον *u* και στον *v*
- Εφαρμογές
  - Κρατήσεις πτήσεων
  - Πορείες οδήγησης
  - Δρομολόγηση πακέτων στο διαδίκτυο
- Παράδειγμα:
  - Το συντομότερο μονοπάτι ανάμεσα στην Providence και στην Honolulu



Ιδιότητες του συντομότερου μονοπατιού

Ιδιότητα 1:

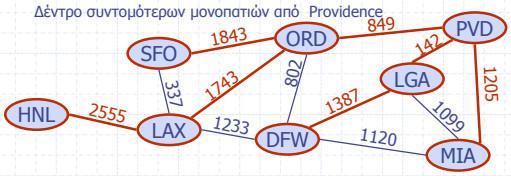
Ένα υπο-μονοπάτι ενός ελάχιστου μονοπατιού είναι από μόνο του ελάχιστο μονοπάτι

Ιδιότητα 2:

Υπάρχει ένα δέντρο ελάχιστων μονοπατιών από έναν αρχικό κόμβο σε όλους τους άλλους κόμβους

Παράδειγμα:

Δέντρο συντομότερων μονοπατιών από Providence



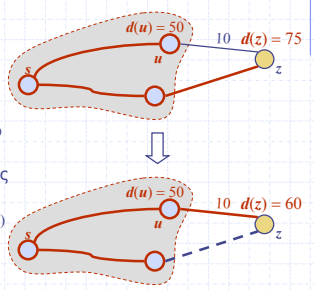
Ο αλγόριθμος του Dijkstra

- Η απόσταση ενός κόμβου v από έναν κόμβο s είναι το μήκος ενός συντομότερου μονοπατιού ανάμεσα στον s και στον v
- Ο αλγόριθμος Dijkstra υπολογίζει τις αποστάσεις όλων των κόμβων από έναν αρχικό κόμβο s
- Υποθέσεις:
  - Ο γράφος είναι συνδεδεμένος
  - Οι ακμές δεν είναι κατευθυνόμενες
  - Τα βάρη των ακμών είναι θετικά
- Δημιουργούμε ένα "σύννεφο"από κόμβους, ξεκινώντας με τον s και καλύπτοντας σταδιακά όλους τους κόμβους
- Αποθηκεύουμε με κάθε κόμβο v μια ετικέτα $d(v)$ που αναπαριστά την απόσταση του v από τον s στον υπογράφο στον υπογράφο που αποτελείται από το σύννεφο και τους προσκείμενους κόμβους
- Σε κάθε βήμα
  - Προσθέτουμε στο σύννεφο τον κόμβο u έξω από το σύννεφο με την μικρότερη ετικέτα απόστασης
  - Ενημερώνουμε τις ετικέτες των κόμβων που είναι προσκείμενες στον u

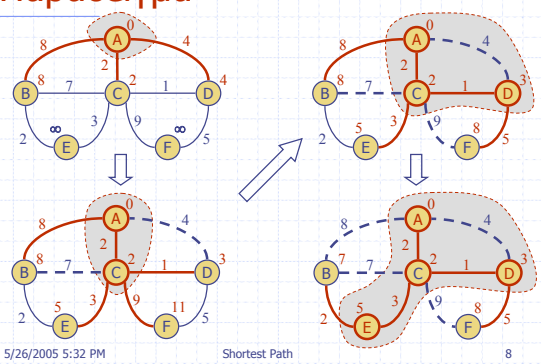
Edge Relaxation

- Θεωρείστε μια ακμή $e = (u,z)$ έτσι ώστε
  - u είναι ο κόμβος που προστέθηκε τελευταίος στο σύννεφο
  - Ο z δεν είναι στο σύννεφο
- Το relaxation μιας ακμής e ενημερώνει την τιμή $d(z)$ ως εξής

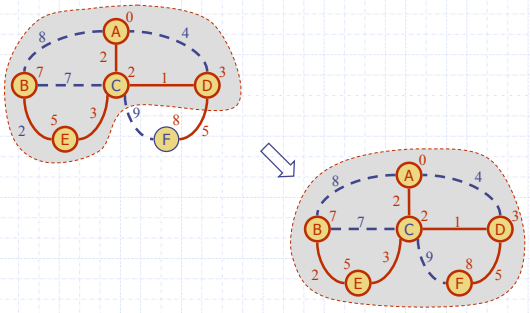
$d(z) \leftarrow \min(d(z), d(u) + weight(e))$



Παράδειγμα



Παράδειγμα (συν.)



Ο αλγόριθμος του Dijkstra

- Μια ουρά προτεραιότητας αποθηκεύει τους κόμβους έξω από το σύννεφο
  - Key: απόσταση
  - Element: κόμβος
- Locator-based μέθοδοι
  - *insert*(*k*,*e*) επιστρέφει έναν locator
  - *replaceKey*(*l*,*k*) αλλάζει το κλειδί ενός αντικειμένου
- Αποθηκεύουμε δύο ετικέτες για κάθε κόμβο:
  - απόσταση
  - Τον locator στην ουρά προτεραιότητας

```
Algorithm DijkstraDistances(G, s)
  Q ← new heap-based priority queue
  for all v ∈ G.vertices()
    if v = s
      setDistance(v, 0)
    else
      setDistance(v, ∞)
    l ← Q.insert(getDistance(v), v)
    setLocator(v,l)
  while ¬Q.isEmpty()
    u ← Q.removeMin()
    for all e ∈ G.incidentEdges(u)
      { relax edge e }
      z ← G.opposite(u,e)
      r ← getDistance(u) + weight(e)
      if r < getDistance(z)
        setDistance(z,r)
        Q.replaceKey(getLocator(z),r)
```


Ανάλυση

- Λειτουργίες του γράφου
  - Η μέθοδος incidentEdges καλείται μια φορά για κάθε κόμβο
- Λειτουργίες ετικετών
  - Θέτουμε και ανακτούμε τις ετικέτες απόστασης και locator ενός κόμβου z $O(\deg(z))$ φορές
  - Ανάθεση/ανάκτηση μιας ετικέτας παίρνει χρόνο $O(1)$
- Λειτουργίες ουρών προτεραιότητας
  - Κάθε κόμβος εισάγεται και απομακρύνεται μια φορά από την ουρά προτεραιότητας, και κάθε τέτοια πράξη παίρνει χρόνο $O(\log n)$
  - Το κλειδί ενός κόμβου στην ουρά προτεραιότητας τροποποιείται το πολύ $\deg(w)$ φορές, ενώ κάθε αλλαγή κλειδιού παίρνει χρόνο $O(\log n)$
- Ο αλγόριθμος του Dijkstra εκτελείται σε χρόνο $O((n + m) \log n)$ δεδομένου ότι ο γράφος αναπαρίσταται με την adjacency list δομή
  - Θυμηθείτε ότι $\sum_v \deg(v) = 2m$
- Ο χρόνος εκτέλεσης μπορεί επίσης να εκφραστεί ως $O(m \log n)$ δεδομένου ότι ο γράφος είναι συνδεδεμένος

Επέκταση

- Χρησιμοποιώντας το template method πρότυπο,μπορούμε να επεκτείνουμε τον αλγόριθμο του Dijkstra για να επιστρέφει ένα δέντρο των συντομότερων μονοπατιών από έναν αρχικό κόμβο σε όλους τους άλλους
- Αποθηκεύουμε για κάθε κόμβο μια Τρίτη ετικέτα
  - Την ακμή πατέρα στο δέντρο συντομότερων μονοπατιών
- Στο βήμα του edge relaxation step, ενημερώνουμε την ακμή πατέρα

```
Algorithm DijkstraShortestPathsTree(G, s)
  …
  for all v ∈ G.vertices()
    …
    setParent(v, ∅)
    …
  for all e ∈ G.incidentEdges(u)
    { relax edge e }
    z ← G.opposite(u,e)
    r ← getDistance(u) + weight(e)
    if r < getDistance(z)
      setDistance(z,r)
      setParent(z,e)
      Q.replaceKey(getLocator(z),r)
```