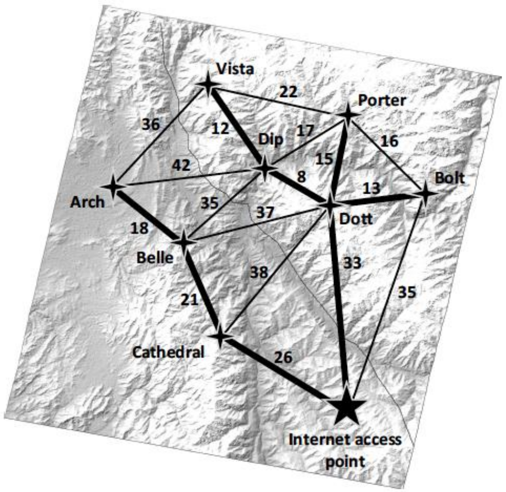
Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

Δέντρα επικάλυψης ελαχίστου κόστους

Saguaros, Saguaro National Monument, 1941. Ansel Adams. U.S. government image. U.S. National Archives and Records Administration.

Εφαρμογή: Σύνδεση ενός δικτύου

- Υποθέστε ότι μια χώρα έλαβε επιχορήγηση για να εγκαταστήσει μεγάλους πύργους Wi-Fi στο κέντρο κάθε ορεινού χωριού της.
- Τα καλώδια επικοινωνίας φτάνουν από το κύριο σημείο πρόσβασης στο Internet ως ένα ή περισσότερους πύργους σε χωριά και καλύπτουν επίσης τις αποστάσεις μεταξύ των πύργων.
- Το ζητούμενο είναι να συνδεθούν όλοι οι πύργοι και το σημείο πρόσβασης στο Internet όσο το δυνατόν **πιο οικονομικά** σε ότι αφορά το κόστος των καλωδίων επικοινωνίας.

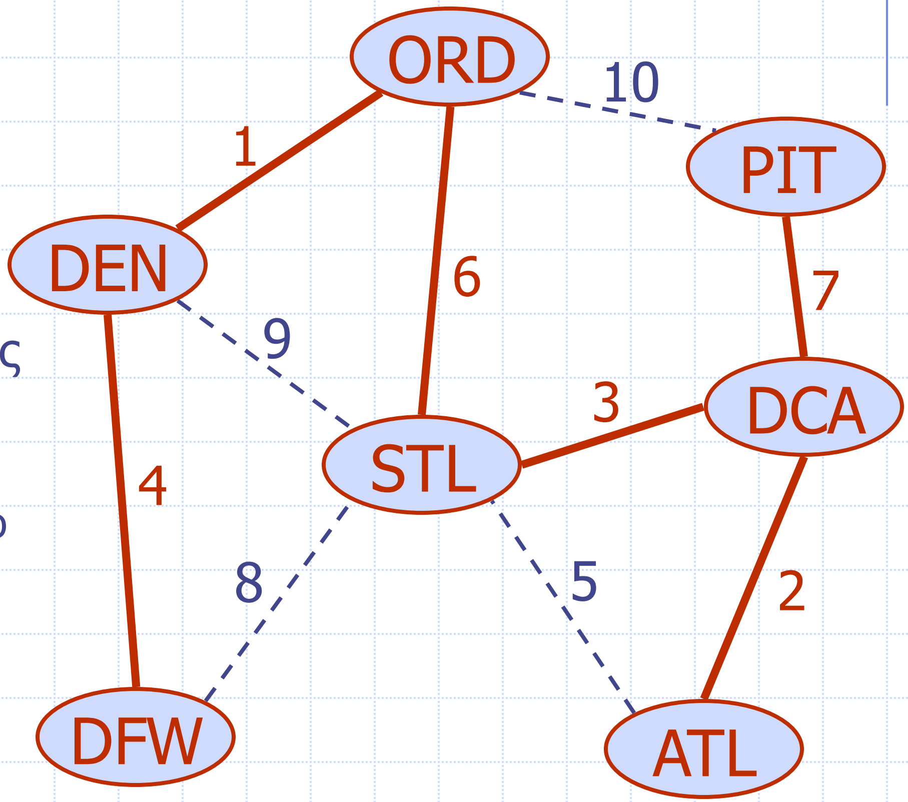


Εφαρμογή: Σύνδεση ενός δικτύου

- Μπορούμε να μοντελοποιήσουμε αυτό το πρόβλημα χρησιμοποιώντας ένα γράφο **G**, όπου κάθε κορυφή στον **G** είναι η θέση ενός πύργου Wi-Fi ή του σημείου πρόσβασης στο Internet και μία ακμή στον **G** είναι ένα πιθανό καλώδιο που θα μπορούσαμε να τοποθετήσουμε ανάμεσα σε δύο τέτοιες κορυφές.
- Σε κάθε ακμή στο **G** θα μπορούσαμε τότε να εκχωρούμε ένα βάρος που ισούται με το κόστος της τοποθέτησης το καλωδίου, που αναπαριστά αυτή η ακμή.
- Συνεπώς, μας ενδιαφέρει να βρούμε ένα συνεκτικό ακυκλικό υπογράφο του **G** που περιλαμβάνει όλες τις κορυφές του **G** και έχει ελάχιστο συνεκτικό κόστος.
- Με άλλα λόγια, χρησιμοποιώντας τη γλώσσα της θεωρίας γράφων, μας ενδιαφέρει να βρούμε ένα **δέντρο επικάλυψης ελάχιστου κόστους (minimum spanning tree - MST)** του **G**.

Δέντρα επικάλυψης ελαχίστου κόστους

- Υπογράφος επικάλυψης
  - Υπογράφος ενός γράφου G που περιέχει όλες τις κορυφές του G.
- Δέντρο επικάλυψης
  - Υπογράφος επικάλυψης που είναι ένα (ελεύθερο) δέντρο.
- Δέντρο επικάλυψης ελαχίστου κόστους (MST)
  - Δέντρο επικάλυψης ενός σταθμισμένου γράφου με ελάχιστο συνολικό βάρος ακμών.
- Εφαρμογές
  - Δίκτυα επικοινωνίας.
  - Δίκτυα μεταφοράς.



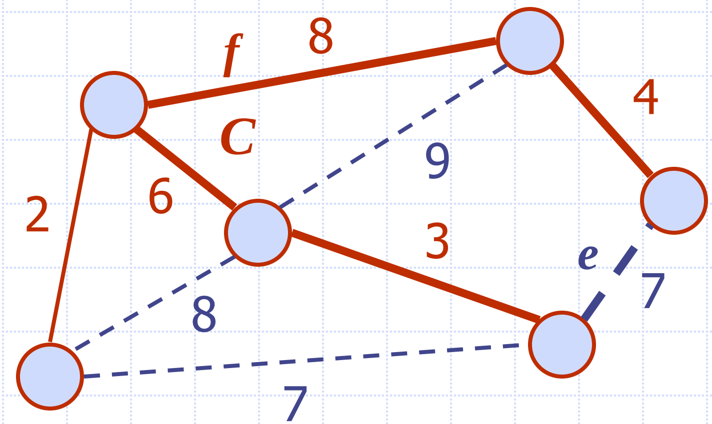
Ιδιότητα κύκλου

Ιδιότητα κύκλου:

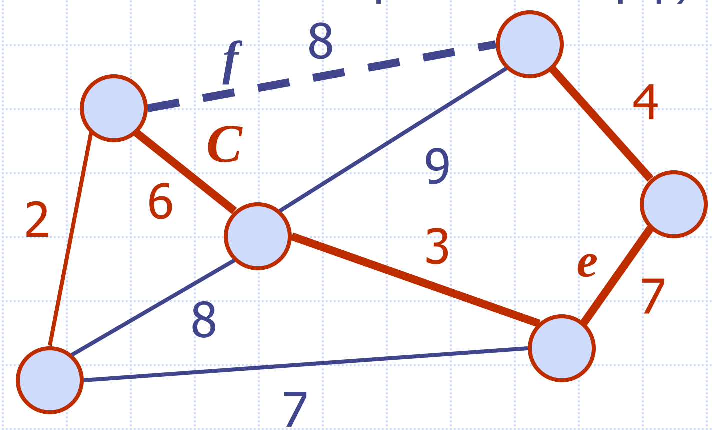
- Έστω T ένα **δέντρο επικάλυψης ελάχιστου κόστους** ενός σταθμισμένου γράφου G.
- Έστω e μία ακμή του G που δεν υπάρχει στο T και έστω C ο κύκλος που δημιουργείται αν συμπεριληφθεί το e στο T.
- Για κάθε ακμή f στο C, ισχύει ότι $weight(f) \leq weight(e)$

Απόδειξη:

- Εις άτοπον απαγωγή.
- Εάν $weight(f) > weight(e)$ μπορούμε να δημιουργήσουμε ένα δέντρο επικάλυψης μικρότερου κόστους αντικαθιστώντας την e με την f.



Η αντικατάσταση της f με την e δημιουργεί ένα καλύτερο δέντρο επικάλυψης



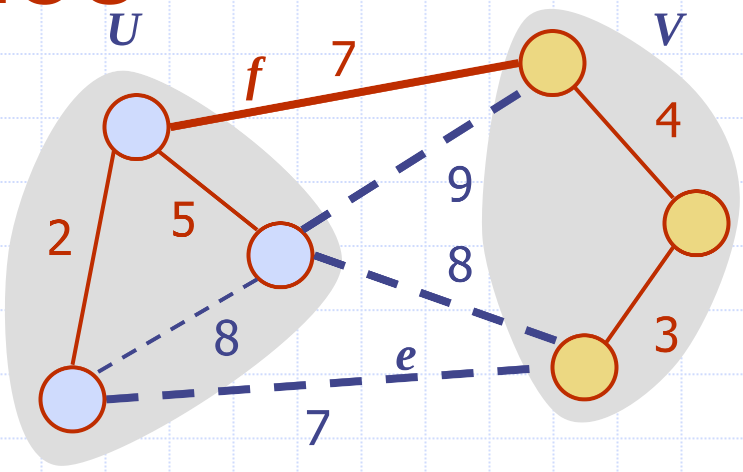
Ιδιότητα διαμερισμού

Ιδιότητα τμημάτων:

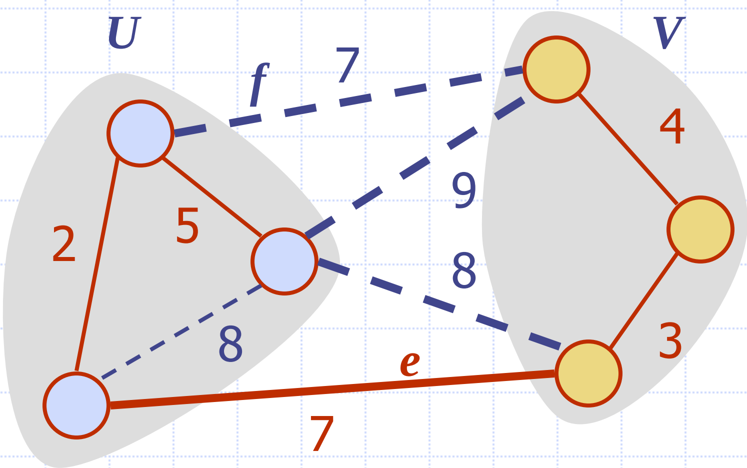
- Θεωρήστε διαμέριση των κορυφών του G στα υποσύνολα U και V.
- Έστω e μια ακμή με ελάχιστο βάρος μεταξύ των τμημάτων.
- Θα υπάρχει ένα δέντρο επικάλυψης ελαχίστου κόστους του G που θα περιέχει την ακμή e.

Απόδειξη:

- Έστω T ένα MST του G.
- Εάν το T δεν περιέχει την e, έστω C ο κύκλος που δημιουργείται από την e και το T και έστω f μία ακμή του C μεταξύ των τμημάτων U και V.
- Λόγω της ιδιότητας κύκλου,
  $$weight(f) \leq weight(e)$$
- Οπότε, $weight(f) = weight(e)$
- Έχουμε ένα ακόμη MST αντικαθιστώντας την f με την e.



Η αντικατάσταση της f με την e δημιουργεί ένα ακόμη MST.



Αλγόριθμος Prim-Jarnik

- Παρόμοιος με τον αλγόριθμο του Dijkstra.
- Επιλέγουμε μία τυχαία κορυφή s και αναπτύσσουμε το MST ως ένα σύννεφο κορυφών, ξεκινώντας από την s.
- Αποθηκεύουμε για κάθε κορυφή v την ετικέτα $d(v)$ που αντιπροσωπεύει το ελάχιστο βάρος της ακμής που συνδέει την v με κάποια κορυφή που ήδη έχει ενταχθεί στο σύννεφο.
- Σε κάθε βήμα:
  - Προσθέτουμε στο σύννεφο την κορυφή u εκτός σύννεφου με την μικρότερη ετικέτα απόστασης.
  - Ενημερώνουμε τις ετικέτες των γειτονικών κορυφών του u.

Ψευδό-κώδικας Prim-Jarnik

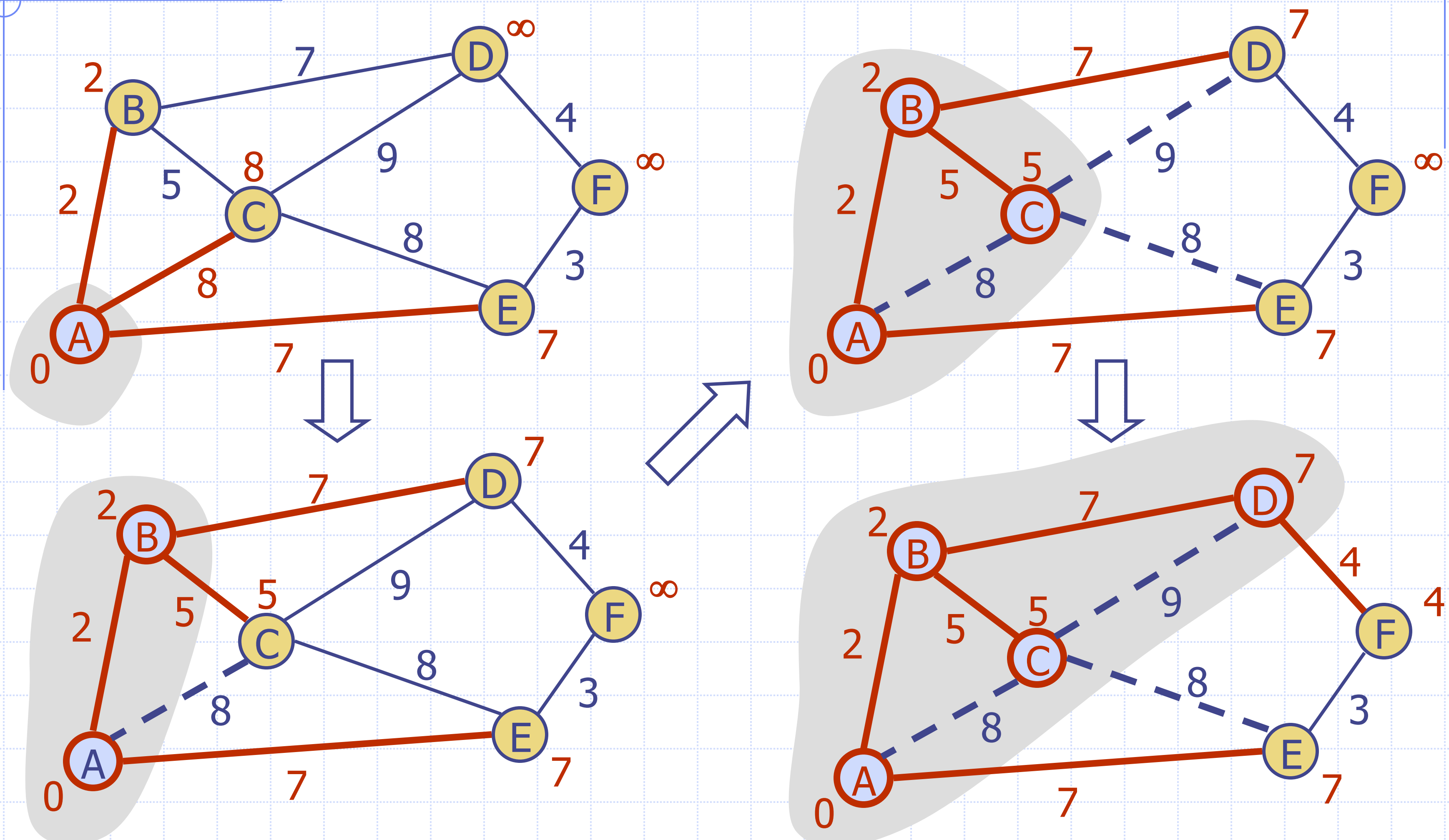
Algorithm PrimJarníkMST(G):

Input: A weighted connected graph G with n vertices and m edges

Output: A minimum spanning tree T for G

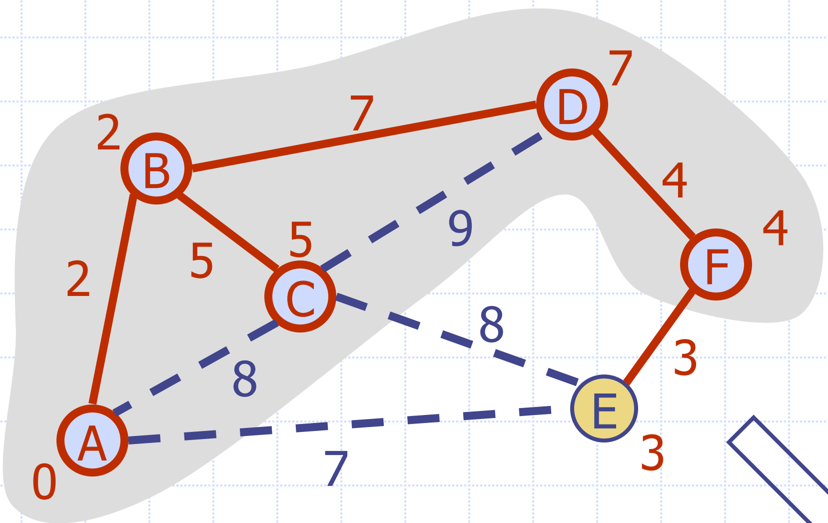
```
Pick any vertex v of G
D[v] ← 0
for each vertex u ≠ v do
    D[u] ← +∞
Initialize T ← ∅.
Initialize a priority queue Q with an item ((u, null), D[u]) for each vertex u,
where (u, null) is the element and D[u] is the key.
while Q is not empty do
    (u, e) ← Q.removeMin()
    Add vertex u and edge e to T.
    for each vertex z adjacent to u such that z is in Q do
        // perform the relaxation procedure on edge (u, z)
        if w((u, z)) < D[z] then
            D[z] ← w((u, z))
            Change to (z, (u, z)) the element of vertex z in Q.
            Change to D[z] the key of vertex z in Q.
return the tree T
```

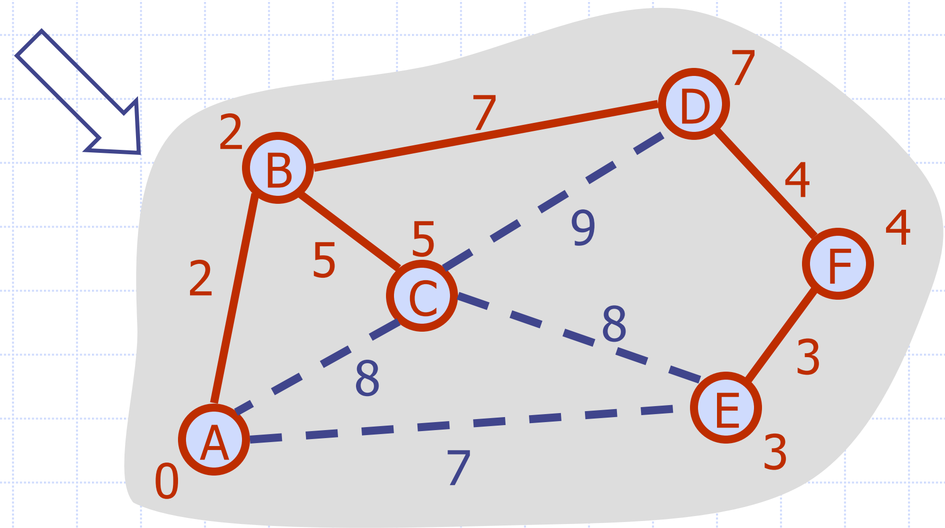
Παράδειγμα



D ∞
7
2 B
4
9
8
5 C F ∞
2
8
3
8
A E
0 7 7
D 7
7
2 B
4
9
5
5 C F ∞
2
8
8 3
A E
0 7 7
D 7
7
2 B
4
5 9
2 5 C F ∞
8
8 3
A E
0 7 7
D 7
7
2 B
4
5 9 4
2 5 C F
8
8 3
A E
0 7 7

Παράδειγμα (συνέχεια)





Ανάλυση

- Έστω γράφος G με n κορυφές και m ακμές.
- Πράξεις γράφου
  - Περνάμε κυκλικά από όλες τις ακμές κάθε κορυφής εξετάζοντας την κάθε ακμή μία μόνο φορά.
- Πράξεις ετικέτας
  - Θέτουμε/ανακτούμε την απόσταση κάθε κορυφής z $O(\deg(z))$ φορές.
  - Η ανάθεση/ανάκτηση τιμής μίας ετικέτας απαιτεί χρόνο $O(1)$.
- Πράξεις ουράς προτεραιότητας
  - Κάθε κορυφή εισάγεται και αφαιρείται μία φορά από την ουρά προτεραιότητας, με κάθε εισαγωγή ή αφαίρεση να απαιτεί χρόνο $O(\log n)$.
  - Το κλειδί μιας κορυφής w στην ουρά προτεραιότητας τροποποιείται το πολύ $\deg(w)$ φορές, με κάθε αλλαγή κλειδιού να απαιτεί χρόνο $O(\log n)$.
- Ο αλγόριθμος Prim-Jarnik είναι χρόνου $O((n+m)\log n)$ όταν ο γράφος αναπαρίσταται με δομή λίστας γειτνίασης.
  - Θυμηθείτε ότι $\Sigma_v \deg(v) = 2m$
- Ο χρόνος εκτέλεσης είναι $O(m \log n)$ δεδομένου ότι ο γράφος είναι συνεκτικός.

Η προσέγγιση του Kruskal

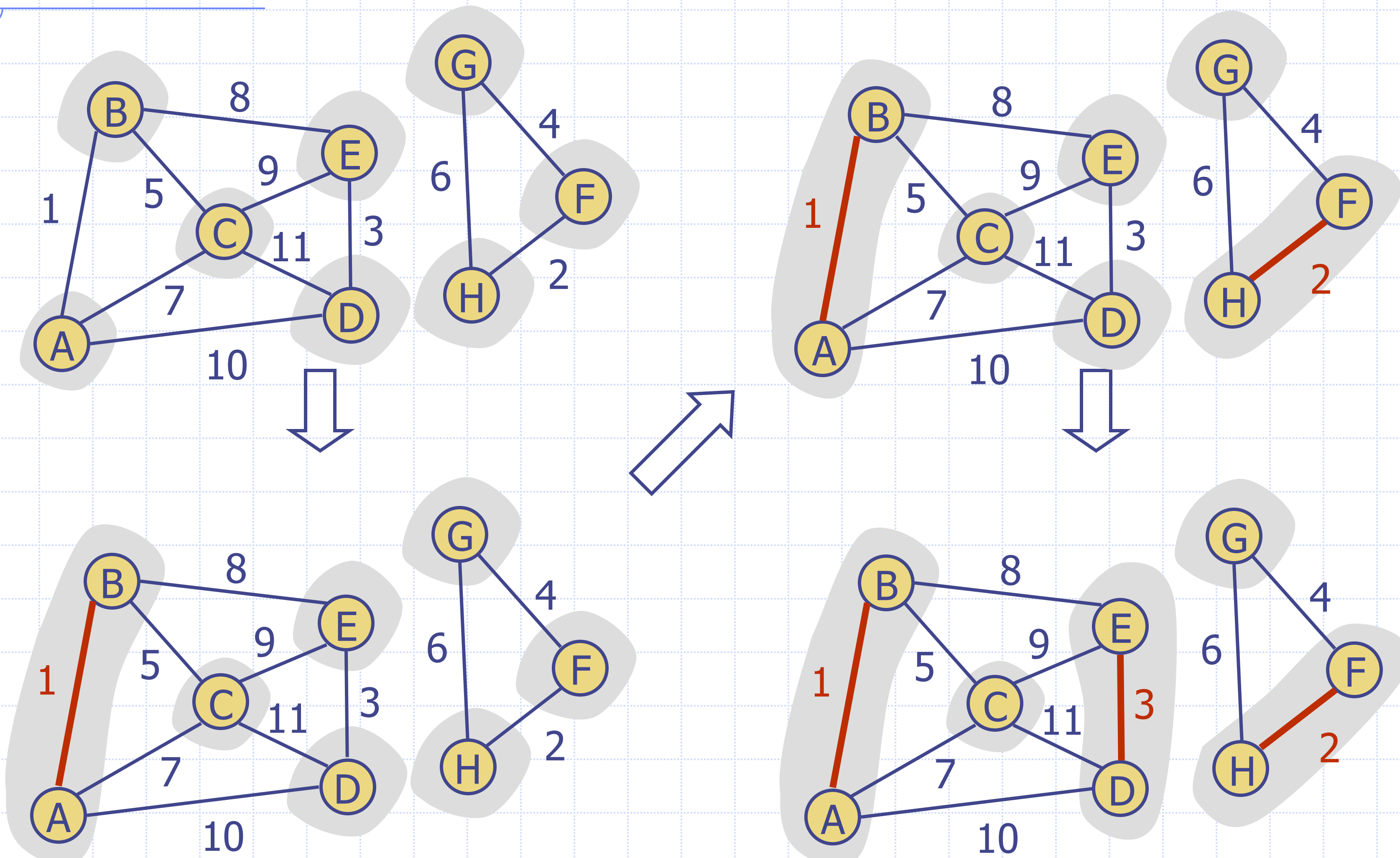
- Οι κορυφές διαμερίζονται σε συστάδες.
  - Αρχικά, συστάδες μίας κορυφής.
  - Διατήρηση ενός MST για κάθε συστάδα.
  - Συνένωση «πλησιέστερων» συστάδων και των MST τους.
- Μία ουρά προτεραιότητας αποθηκεύει τις ακμές εκτός συστάδων.
  - Κλειδί: βάρος.
  - Στοιχείο: ακμή.
- Στο τέλος του αλγορίθμου
  - Μία συστάδα και ένα MST

Ο αλγόριθμος Kruskal

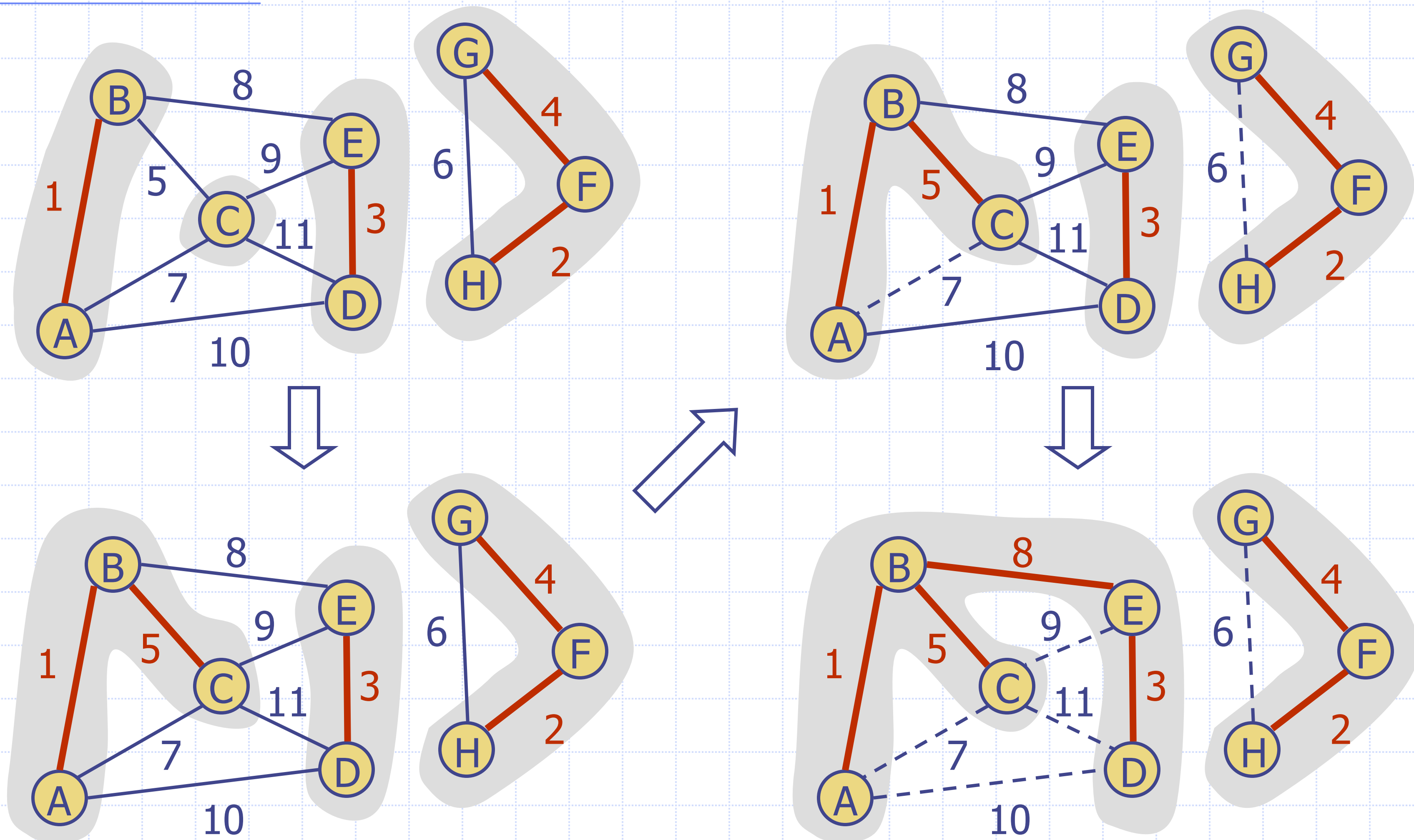
Algorithm KruskalMST(G):
    Input: A simple connected weighted graph G with n vertices and m edges
    Output: A minimum spanning tree T for G

```
for each vertex v in G do
    Define an elementary cluster C(v) ← {v}.
Let Q be a priority queue storing the edges in G, using edge weights as keys
T ← ∅        // T will ultimately contain the edges of the MST
while T has fewer than n − 1 edges do
    (u, v) ← Q.removeMin()
    Let C(v) be the cluster containing v
    Let C(u) be the cluster containing u
    if C(v) ≠ C(u) then
        Add edge (v, u) to T
        Merge C(v) and C(u) into one cluster, that is, union C(v) and C(u)
return tree T
```

Παράδειγμα αλγορίθμου Kruskal



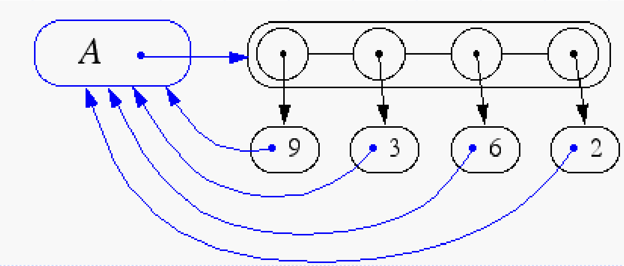
Παράδειγμα (συνέχεια)



Δομή δεδομένων για τον αλγόριθμο του Kruskal

- Ο αλγόριθμος διατηρεί ένα δάσος δέντρων.
- Μία ουρά προτεραιότητας εξάγει τις ακμές σε αύξουσα σειρά βάρους.
- Μία ακμή γίνεται αποδεκτή όταν συνδέει διαφορετικά δέντρα.
- Χρειαζόμαστε μία δομή δεδομένων που να διατηρεί μια διαμέριση, δλδ μία συλλογή ξένων συνόλων, με λειτουργίες:
  - makeSet(u): δημιουργία ενός συνόλου που αποτελείται από το u.
  - find(u): επιστροφή του συνόλου που περιέχει το u.
  - union(A, B): αντικατάσταση των συνόλων A και B με την ένωση τους.

Τμήματα βάση λίστας



- Κάθε σύνολο αποθηκεύεται σε μία ακολουθία.
- Κάθε στοιχείο έχει αναφορά προς το σύνολο.
  - η λειτουργία find(u) είναι χρόνου O(1) και επιστρέφει το σύνολο στο οποίο ανήκει το u.
  - στη λειτουργία union(A,B), μετακινούμε τα στοιχεία από το μικρότερο σύνολο στην ακολουθία του μεγαλύτερου συνόλου και ενημερώνουμε τις αναφορές τους.
  - ο χρόνος της λειτουργίας union(A,B) είναι min(|A|, |B|).
- Όταν ένα στοιχείο επεξεργάζεται, μεταφέρεται σε ένα σύνολο με τουλάχιστον διπλάσιο μέγεθος, οπότε κάθε στοιχείο επεξεργάζεται το πολύ log n φορές.

Υλοποίηση βάσει διαμέρισης

- Έστω γράφος G με n κορυφές και m ακμές.
- Για την υλοποίηση βάσει διαμέρισης του αλγόριθμου του Kruskal:
  - Οι συνενώσεις συστάδων πραγματοποιούνται ως λειτουργίες union.
  - Οι εντοπισμοί συστάδων πραγματοποιούνται ως λειτουργίες find.
- Χρόνος $O((n + m) \log n)$
  - Λειτουργίες ουράς προτεραιότητας: $O(m \log n)$.
  - Λειτουργίες ένωσης-εύρεσης: $O(n \log n)$.

Μία εναλλακτική υλοποίηση

Σε ορισμένες εφαρμογές, μπορεί να έχουμε στη διάθεσή μας τις ακμές ταξινομημένες κατά βάρος. Τότε, ο αλγόριθμος του Kruskal μπορεί να υλοποιηθεί ταχύτερα. Συγκεκριμένα, μπορούμε να υλοποιήσουμε την ουρά προτεραιότητας, Q, απλά ως μια ταξινομημένη λίστα. Αυτή η προσέγγιση μας επιτρέπει να εκτελούμε τις πράξεις removeMin σε σταθερό χρόνο.

Τότε, αντί να χρησιμοποιήσουμε μια απλή δομή δεδομένων διαμερισμού, που βασίζεται σε λίστα, μπορούμε να χρησιμοποιήσουμε τη δομή ένωσης-εύρεσης, που βασίζεται σε δέντρα (βλ. Κεφάλαιο 7). Αυτό σημαίνει ότι η ακολουθία O(m) πράξεων ένωσης-εύρεσης εκτελείται σε χρόνο O(m α(n)), όπου α(n) είναι η πολύ αργά αυξανόμενη αντίστροφη συνάρτηση της συνάρτησης Ackermann. Συνεπώς, προκύπτει το ακόλουθο θεώρημα.

Θεώρημα 15.5: Δεδομένου ενός απλού συνεκτικού σταθμισμένου γράφου G με n κορυφές και m ακμές, με τις ακμές ταξινομημένες κατά βάρος, μπορούμε να υλοποιήσουμε τον αλγόριθμο του Kruskal κατασκευής δέντρου επικάλυψης ελάχιστου κόστους για το G, σε χρόνο O(m α(n)).

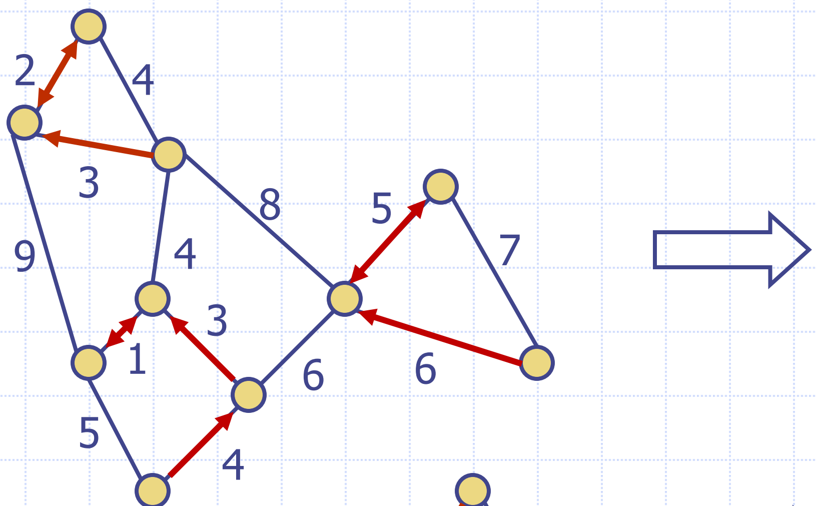
Ο αλγόριθμος του Borůvka

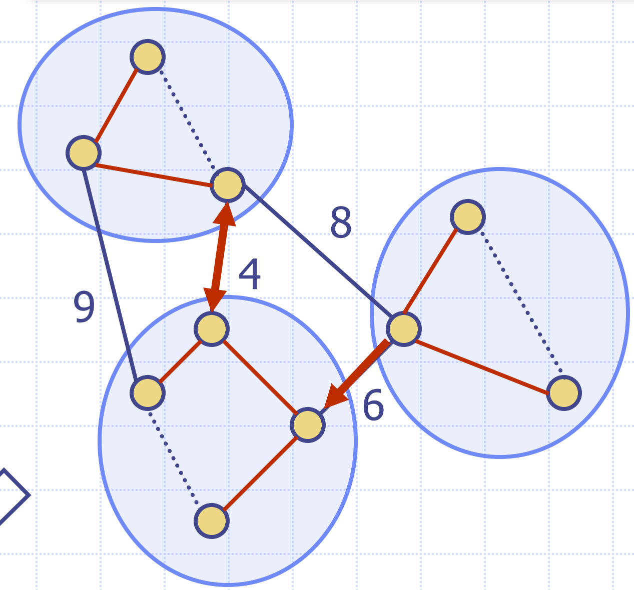
- Όπως ο αλγόριθμος Kruskal, ο αλγόριθμος του Borůvka αναπτύσσει πολλές συστάδες ταυτόχρονα και διατηρεί ένα δάσος T.
- Κάθε επανάληψη της while υποδιπλασιάζει τον αριθμό των συνδεδεμένων συνιστωσών του δάσους T.
- Είναι χρόνου $O(m \log n)$.

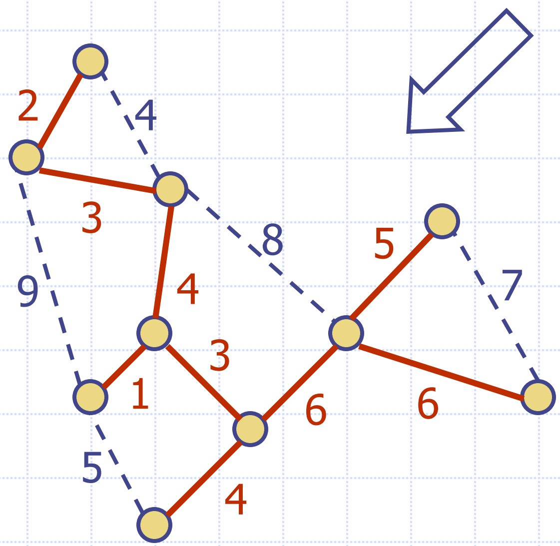
```
Algorithm BoruvkaMST(G)
    T ← V  {just the vertices of G}
  while T has fewer than n – 1 edges do
    for each connected component C in T do
      Let edge e be the smallest-weight edge from C to another component in T
      if e is not already in T then
        Add edge e to T
  return T
```

Παράδειγμα εκτέλεσης αλγόριθμου του Borůvka

Διαφάνεια του Matt Stallmann περιλαμβάνεται με άδεια.







- Αρχικά κάθε κορυφή είναι μόνη της στη δική της συστάδα.
- Στη συνέχεια, κάθε συστάδα εντοπίζει την ακμή που τη συνδέει με το μικρότερο κόστος με κάποια άλλη κορυφή. Αν η ίδια ακμή επιλέγεται από περισσότερες κορυφές οι συστάδες τους συνενώνονται, αλλιώς κάθε συστάδα επεκτείνεται με τη συστάδα της κορυφής στο άλλο άκρο της ακμής μικρότερου κόστους.