Ελάχιστο Συνδετικό Δέντρο

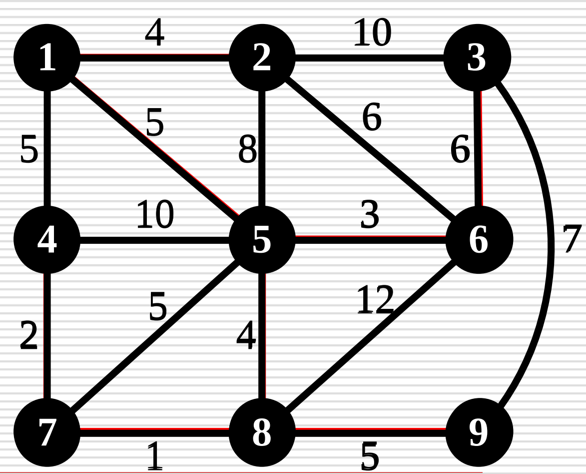
Δημήτρης Φωτάκης, Αριστείδης Παγουρτζής,
Δώρα Σούλιου, Παναγιώτης Γροντάς

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

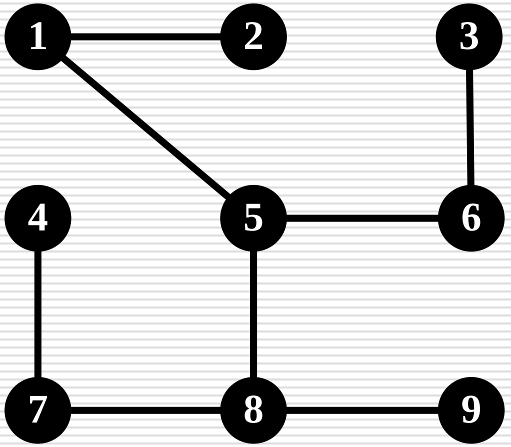
Ελάχιστο Συνδετικό Δέντρο (MST)

- Συνεκτικό μη-κατευθ. G(V, E, w) με βάρη $w : E \mapsto \mathbb{R}_{>0}$
  - Βάρος υπογραφήματος $T(V, E_T)$: $w(T) = \sum_{e \in E_T} w(e)$
- Ζητούμενο: ελάχιστου βάρους **συνεκτικό** υπογράφημα που καλύπτει όλες τις κορυφές.
  - Συνεκτικό (εξ' ορισμού) + ακυκλικό (ελάχιστο) $\Rightarrow$ Δέντρο.
  - Minimum Spanning Tree (MST, ΕΣΔ).
- Πρόβλημα συνδυαστ. βελτιστοποίησης με πολλές και σημαντικές εφαρμογές.
  - Σχεδιασμός συνδετικού δικτύου (οδικού, τηλεπ/κου, ηλεκτρικού) με ελάχιστο κόστος.



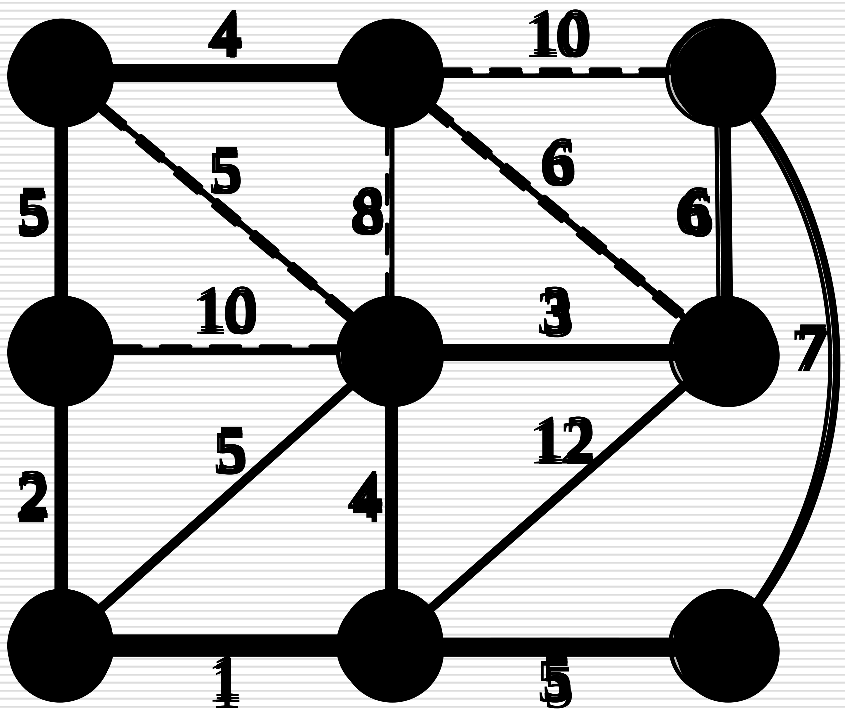
Δέντρα: Βασικές Ιδιότητες

- Δέντρο: συνεκτικό και ακυκλικό γράφημα.
- Για κάθε απλό μη-κατευθ. γράφημα T(V, E), τα παρακάτω είναι ισοδύναμα:
  - T δέντρο.
  - Κάθε ζευγάρι κορυφών ενώνεται με μοναδικό μονοπάτι.
  - T συνεκτικό και |E| = |V| − 1.
  - T ακυκλικό και |E| = |V| − 1.
  - T ελαχιστικά συνεκτικό.
  - T μεγιστικά ακυκλικό.



Αλγόριθμος Prim (1957) & Jarnik (1930)

- Παράδειγμα




```
Prim(G,s):
    S = {s}
    Δ = {}
    R = {(x,y)|x ∈ S, y ∉ S}
    while R ≠ ∅ :
        (v*,y*) = min_w{R}
        S = S ∪ {v*}
        Δ = Δ ∪ {(v*,y*)}
        R = {(x,y)|x ∈ S, y ∉ S}
    return Δ
```

- **Αφελής Υλοποίηση**:
  - O(nm)

Αλγόριθμος Prim με Heap

```
Prim(G,s):
    S={ }, Δ={}, H={(s,0,null)}
    for v ∈ V-{s}:
        v.key = +∞
        v.edge = Null
    while H ≠ ∅:
      v* = H.deleteMin()
      S = S ∪ {v*}
      for (v*,y)|y ∈ V-S:
        if w_{v*y} < y.key: H.delete(y)
        y.key = w_{v*y}, y.edge = (v*,y)
        H.insert(y)
     return Δ
```

Θ(n)

H περιέχει max n κορυφές

Ο(logn)

Μία φορά ανά ακμή

Θ(m)

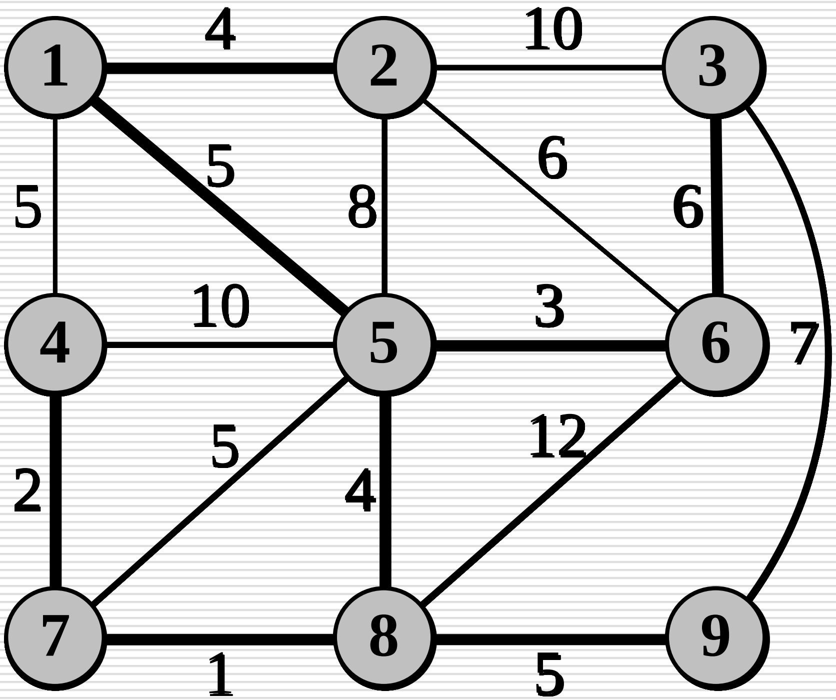
Θ(logn)

Θ(logn)

Πολυπλοκότητα: Θ((n+m)logn = Θ(mlog(n))

Αλγόριθμος Kruskal (1956)

- Παράδειγμα



Αλγόριθμος Kruskal

MST-Kruskal$(G(V, E, w))$

Ταξινόμησε ακμές σε αύξουσα σειρά βάρους, $w(e_1) \leq \cdots \leq w(e_m)$.
$\Delta \leftarrow \emptyset$; $i \leftarrow 1$;
while $|\Delta| < |V| - 1$ **and** $i \leq m$ **do**
  if $\Delta \cup \{e_i\}$ δεν έχει κύκλο **then**
    $\Delta \leftarrow \Delta \cup \{e_i\}$;
  $i \leftarrow i + 1$;

Υλοποίηση: κύκλος ελέγχεται με BFS. Θ(mn)

```
Kruskal(G,s):
    Δ = {}, U = V
    E = sort(e ∈ E) in increasing order of weight
    for (x,y) ∈ E:
        if U.find(x) ≠ U.find(y) : Δ = Δ ∪ {(x,y)}
        U.union(x,y)
    return T
```

Υλοποίηση: κύκλος ελέγχεται με Union-Find. Θ(mlogm)

Τομές – Ιδιότητα Τομής

- Τομή (S, V \ S): διαμέριση V σε 2 σύνολα S, V \ S.
- Σύνολο τομής δ(S, V \ S): ακμές ένα άκρο στο S και άλλο άκρο στο V \ S.
- Ιδιότητα Τομής:

Έστω $S \neq \emptyset$ και $S \neq V$.

Έστω e = {u,v} η ακμή ελαχίστου βάρους του δ(S, V \ S).

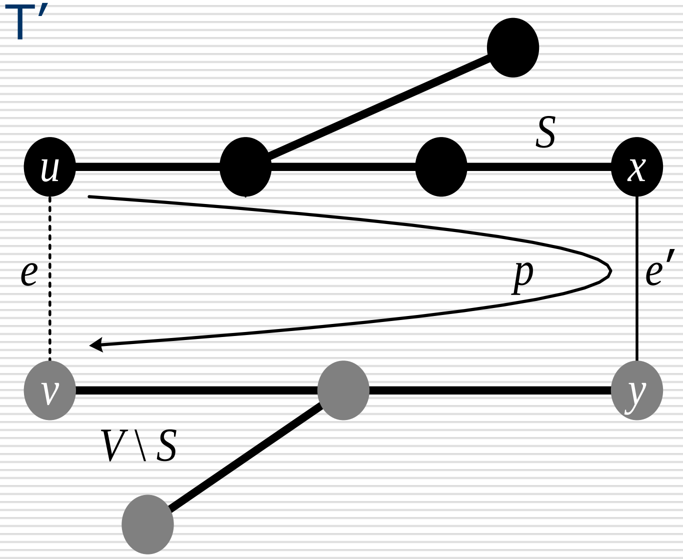
Τότε κάθε ΕΣΔ θα περιέχει την e.

- Η e ονομάζεται **ακμή επαύξησης**

Απόδειξη ιδιότητας τομής

- Έστω T′ ένα MST που δεν περιέχει e = {u, v}.
- Αφού T′ MST:
  - υπάρχει μονοπάτι p μεταξύ u – v στο T′
  - e′= {u′, v′} του p με
    - u′ ∈ S και v′ ∈ V-S
- Ανταλλάζουμε e με e′.
  - T = T′ ∪ {e} \ {e′} και

    $w(T) = w(T') + w(e) - w(e') \leq w(T')$
  - (V,T) συνεκτικό
    - μονοπάτι με e′ → μονοπάτι με e
  - (V,T) ακυκλικό:
    - Ο μόνος κύκλος περιέχει e, e′
- T είναι MST



Ορθότητα Prim - Kruskal

Prim

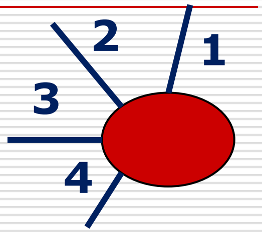
- {v, p[v]} αποτελεί **ακμή επαύξησης** εξ᾽ ορισμού:
  - Διασχίζει τομή (S, V \ S).
  - Ελάχιστου βάρους μεταξύ ακμών του δ(S, V \ S).

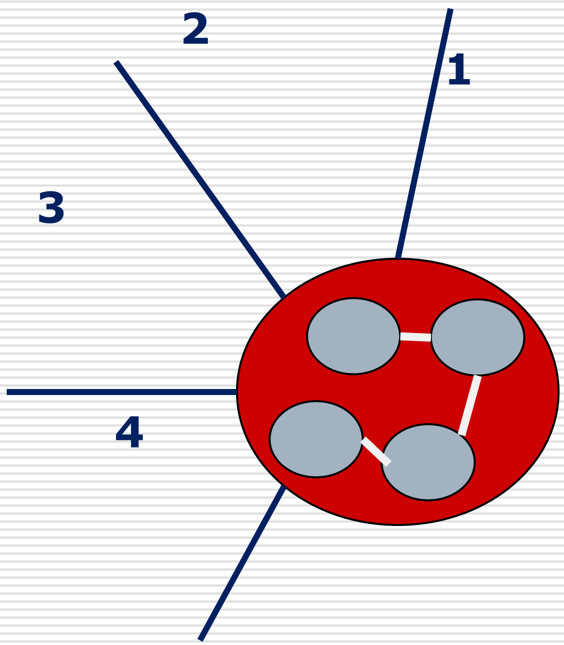
Kruskal

- Αν e_i προστεθεί τότε **ακμή επαύξησης** για Δ:
  - Όχι κύκλος, άρα $e_i \in$ δ(S, V \ S).
  - Αύξουσα σειρά βάρους: e_i ελάχιστου βάρους (πρώτη που ελέγχεται) από όσες ακμές διασχίζουν συγκεκριμένη τομή.

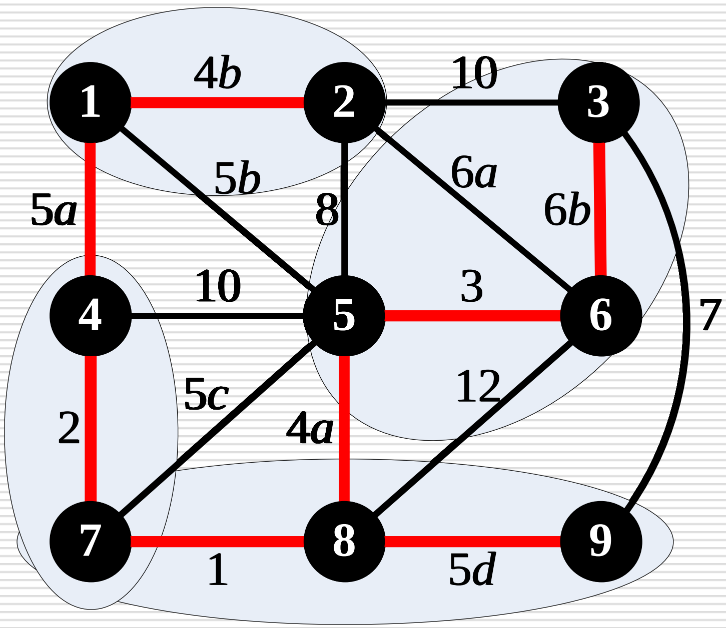
Αλγόριθμος Borůvka (1926)

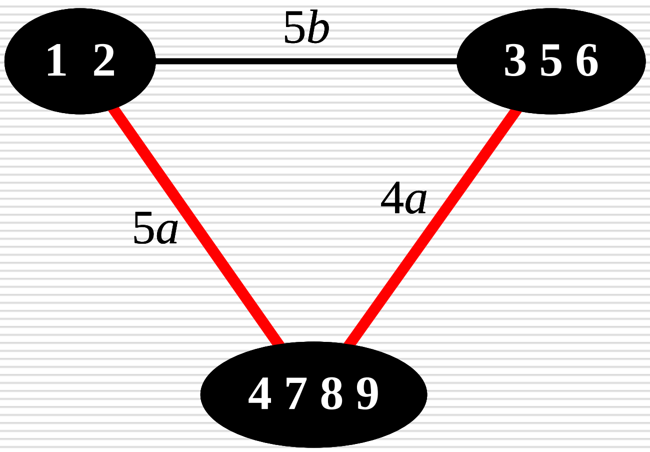
- Παρατήρηση από ιδιότητα τομής:
  - Η μικρότερη ακμή κάθε κορυφής θα ανήκει στο MST
  - επειδή είναι ακμή επαύξησης για την τομή ({v},V\{v})
- Κορυφή = συνδεδεμένη συνιστώσα
- Αλγόριθμος:
  - **Κάθε κορυφή συνδεδεμένη συνιστώσα**
  - **Όσο |ΣΣ| > 1:**
    - **Προσθήκη ακμή ελαχίστου βάρους από μία συνιστώσα σε μια άλλη στο MST (αν δεν υπάρχει ήδη)**
    - **Σύμπτυξη ΣΣ (BFS, DFS)**
- **Εύκολη παραλληλοποίηση**





Αλγόριθμος Borůvka: Παράδειγμα





Αλγόριθμος Borůvka

- (Ακολουθιακή) υλοποίηση σε $O(m \log n)$.
  - Κάθε φάση σε χρόνο $O(m)$ με δύο περάσματα των ακμών.
  - 1ο πέρασμα βρίσκει ελαφρύτερη ακμή κάθε συνιστώσας.
  - 2ο πέρασμα εντάσσει ελαφρύτερες ακμές στο ΕΣΔ και ενημερώνει / συμπτύσσει συνιστώσες (π.χ., με BFS/DFS).
  - Σε κάθε φάση, #συνιστωσών μειώνεται τουλάχιστον στο μισό.
    - #φάσεων = $O(\log n)$.
- Προσοχή στις ισοπαλίες (ολική διάταξη ακμών).
- Πολλοί σύγχρονοι αλγόριθμοι (σχεδόν) γραμμικού χρόνου βασίζονται σε ιδέα Borůvka.

Συζήτηση

Deterministic comparison based algorithms.

- $O(m \log n)$ [Jarník, Prim, Dijkstra, Kruskal, Boruvka]
- $O(m \log \log n)$. [Cheriton-Tarjan 1976, Yao 1975]
- $O(m\, \beta(m, n))$. [Fredman-Tarjan 1987]
- $O(m \log \beta(m, n))$. [Gabow-Galil-Spencer-Tarjan 1986]
- $O(m\, \alpha\, (m, n))$. [Chazelle 2000]

Holy grail. $O(m)$.

Notable.

- $O(m)$ randomized. [Karger-Klein-Tarjan 1995]
- $O(m)$ verification. [Dixon-Rauch-Tarjan 1992]