Ελάχιστο Συνδετικό Δέντρο

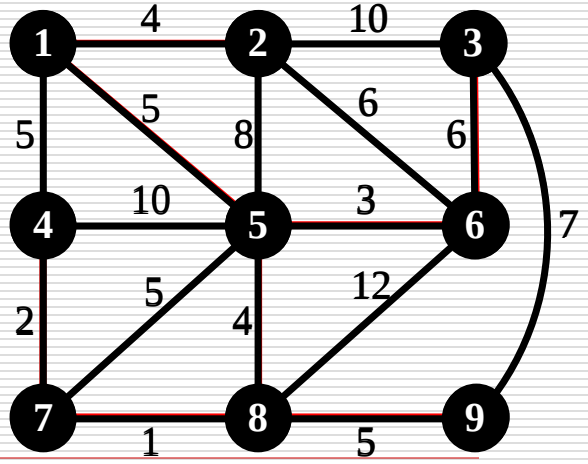
Δημήτρης Φωτάκης, Αριστείδης Παγουρτζής,
Δώρα Σούλιου, Παναγιώτης Γροντάς

Σχολή Ηλεκτρολόγων Μηχανικών
και Μηχανικών Υπολογιστών

Εθνικό Μετσόβιο Πολυτεχνείο

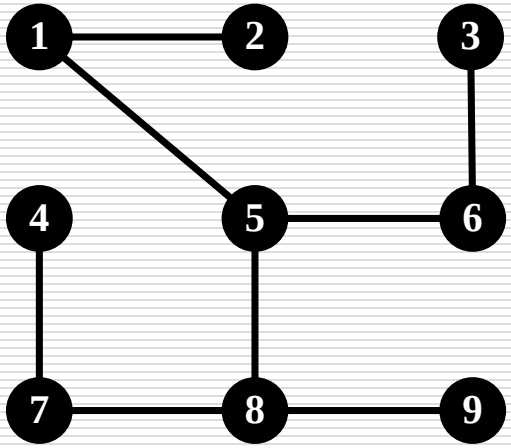
Ελάχιστο Συνδετικό Δέντρο (MST)

- **Είσοδος:** Συνεκτικό μη-κατευθ. $G(V, E, w)$ με βάρη $w: E \mapsto \mathbb{R}^+$
  - Βάρος υπογραφήματος $T(V, E_T): w(T) = \sum_{\{e \in E_T\}} w(e)$
- **Έξοδος:** 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.



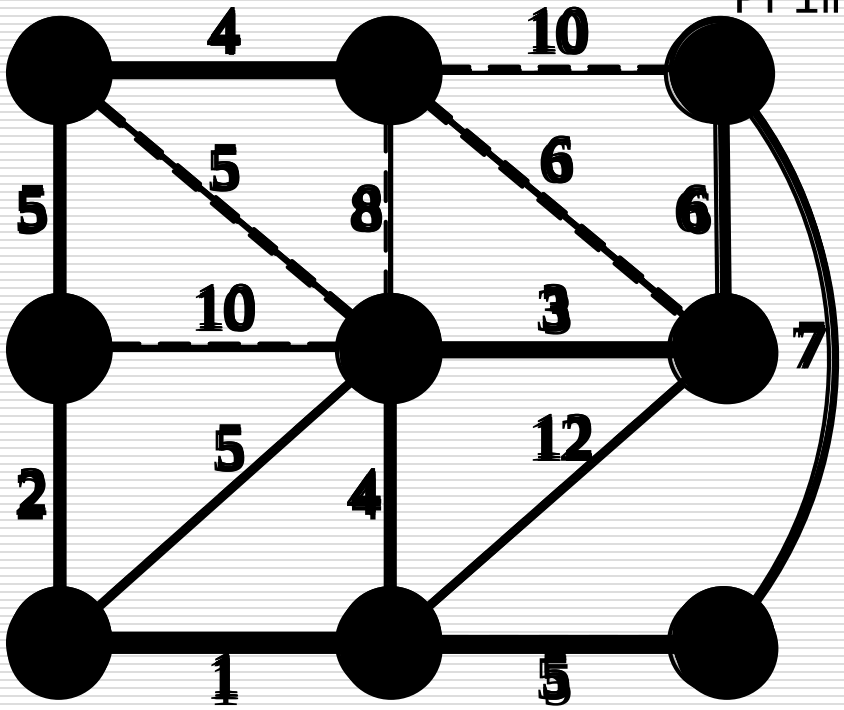
Χρήσιμες ιδιότητες δέντρων

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



Αλγόριθμος Prim (1957) & Jarnik (1930)

- Παράδειγμα




```
Prim(G):
    S = {s} #s can be any vertex
    Δ = {}  #MST
    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
```

- **Πολυπλοκότητα**: O(nm)
- **Μπορούμε καλύτερα?**

Αλγόριθμος Prim με Heap

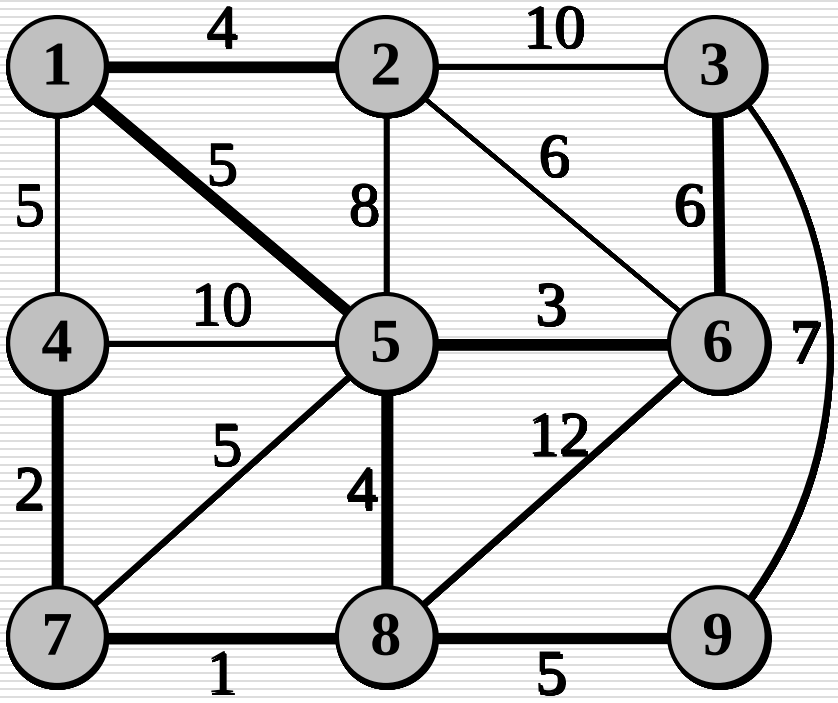
```
Prim(G):
    S={s}, Δ={}, H={}
    for v ∈ V-S:
        if (s,v) ∈ E:
            v.key = w_sv,  v.edge = (s,v)
        else:
            v.key = +∞,  v.edge = null
        H.insert(v)
    while H ≠ ∅:
      v* = H.deleteMin()
      S = S ∪ {v*}, Δ = Δ ∪ v*.edge
      for (v*,y)|y ∈ V-S:
       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

Πολυπλοκότητα:

- $O(m+n)$ το αρχικό **for**
- **Heap** περιέχει n κλειδιά
  - Λειτουργίες insert, delete $logn$
- Στο **while**:
  - Κάθε ακμή εξετάζεται ακριβώς μία φορά
  - 2 λειτουργίες του σωρού
  - $O(m\ logn)$
- Τελικά:
  - $\boldsymbol{O(m\ logn)}$

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 γραμμικά πχ. BFS.

$\Theta(mn)$

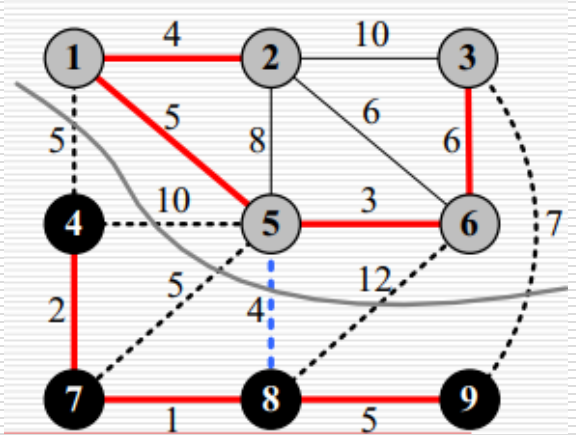
```
Kruskal(G):
    Δ ← {}, U ← V #n distinct vertices
    sort(E), i←1
    while |Δ|< |V| - 1 and i<=m do
       (x,y) ← E[i]
       if U.find(x) ≠ U.find(y):
          Δ ← Δ ∪ {(x,y)}
       U.union(x,y)
    return T
```

Υλοποίηση: κύκλος ελέγχεται με Union-Find.

$O(mlogm) = O(mlogn)$

Ορθότητα: Τομές – Ιδιότητα Τομής

- Τομή $(S, V\backslash S)$: διαμέριση V σε 2 σύνολα $S, V\backslash S$.
- Σύνολο τομής $\delta(S, V\backslash S)$:
  - ακμές με ένα άκρο στο S και άλλο άκρο στο , $V\backslash S$.
- Ιδιότητα Τομής / Ακμή επαύξησης:

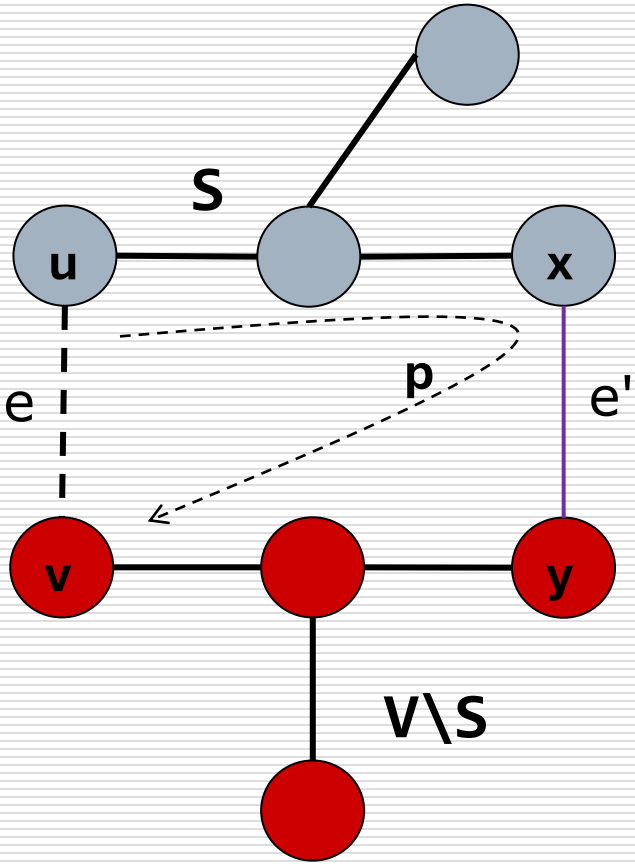


Έστω $S \subseteq V$ και $S \neq V$ και $e = \{u, v\}$ η ακμή ελαχίστου βάρους του $\delta(S, V\backslash S)$. Τότε κάθε MST θα περιέχει την e.

Η e ονομάζεται **ακμή επαύξησης.**

Απόδειξη ιδιότητας τομής

- Έστω L ένα MST που δεν περιέχει $e = \{u, v\}$.
- Αφού L MST:
  - υπάρχει μονοπάτι p μεταξύ $u - v$ στο L
  - $e' = \{x, y\}$ του p με $x \in \boldsymbol{S}$ και $y \in \boldsymbol{V \backslash S}$
- Ανταλλάζουμε e' με e.
  - $T = L \setminus \{e'\} \cup \{e\}$
  - $w(T) = w(L) - w(e') + w(e) < W(L)$
  - T **συνεκτικό**
    - ∀ μονοπάτι με e' reroute σε e
  - (V, T) **ακυκλικό**:
    - Ο μόνος κύκλος θα περιείχε (e, e')
- Άρα: T είναι MST



Ορθότητα Prim - Kruskal

Prim

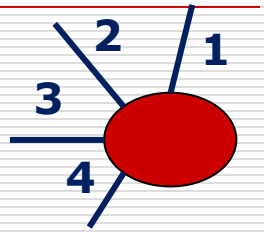
- $\{v, p[v]\}$ αποτελεί **ακμή επαύξησης** εξ΄ ορισμού:
  - Διασχίζει τομή (S, V \ S).
  - Ελάχιστου βάρους μεταξύ ακμών του δ(S, V \ S).

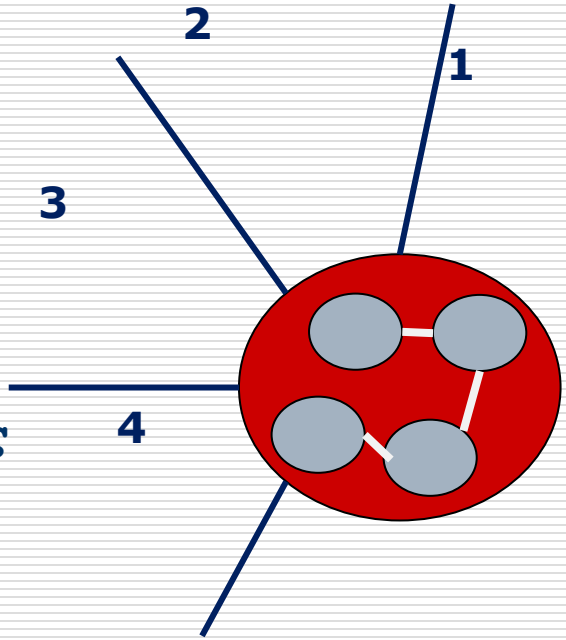
Kruskal

- 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.

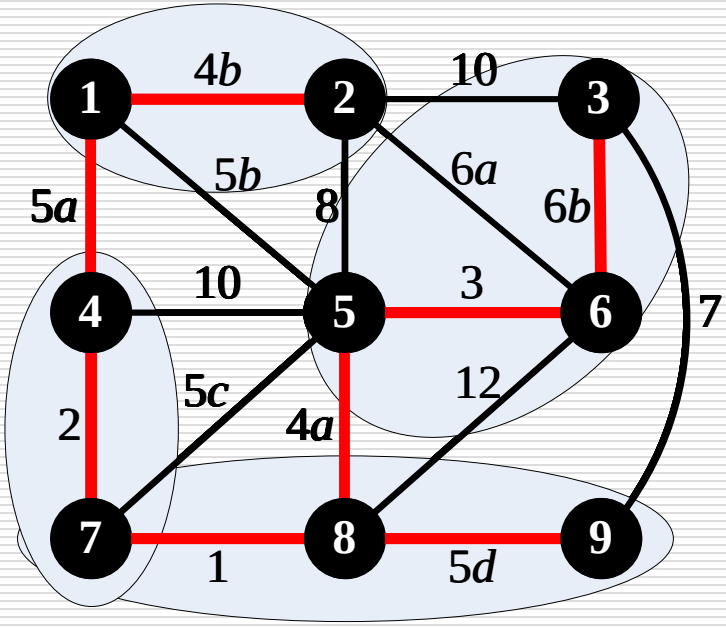
Αλγόριθμος Borůvka (1926)

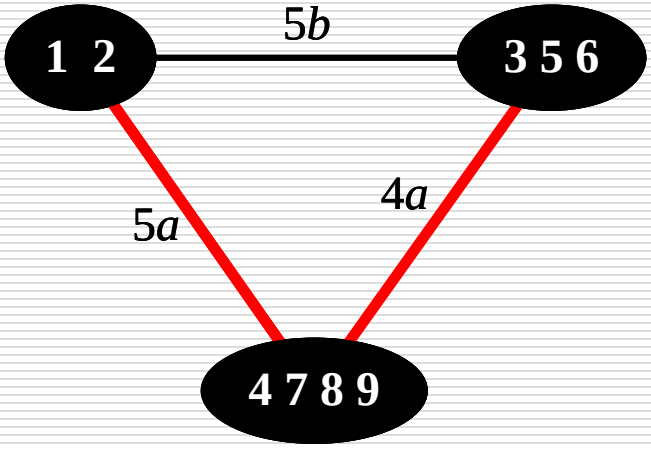
- Παρατήρηση από ιδιότητα τομής:
  - Η μικρότερη ακμή κάθε κορυφής θα ανήκει στο MST
  - επειδή είναι ακμή επαύξησης για την τομή $(\{v\}, V\{v\})$
- Κορυφή = συνδεδεμένη συνιστώσα
- Αλγόριθμος:
  - **Κάθε κορυφή συνδεδεμένη συνιστώσα**
  - **Όσο $|ΣΣ| > 1$:**
    - **Προσθήκη ακμής ελαχίστου βάρους από μία συνιστώσα σε μια άλλη στο MST (αν δεν υπάρχει ήδη)**
    - **Σύμπτυξη ΣΣ**
  - **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 = $O(\log n)$.
- Πολλοί σύγχρονοι αλγόριθμοι (σχεδόν) γραμμικού χρόνου βασίζονται σε ιδέα Borůvka.

Σύνοψη

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]

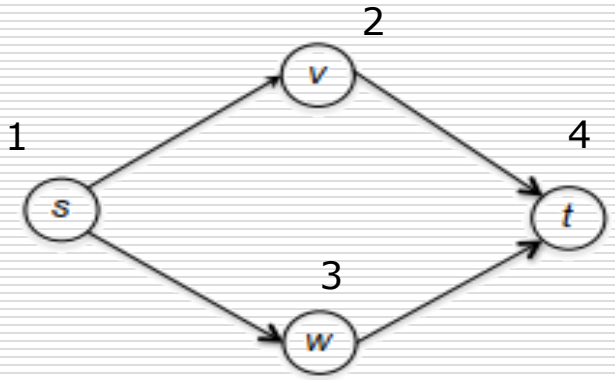
Holy grail. $O(m)$.

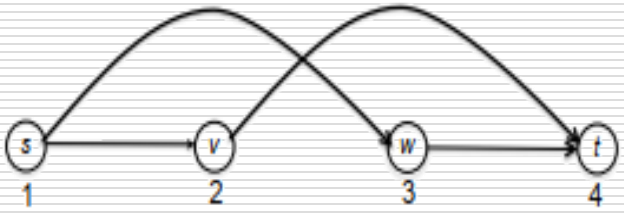
Notable.

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

DAGs (Directed Acyclic Graphs)

- DAG (Directed Acyclic Graph)
  - Source: κορυφή με $in - degree = 0$
  - Sink: κορυφή $out - degree = 0$
- Αντιστοιχεί σε σχέση μερικής διάταξης:
  - Ακμή (u, v) ⇔ δ(u) ≤ δ(v) (δηλ. u «προηγείται» v).
- Εφαρμογές DAG
  - Σειρά υπολογισμού αριθμητικών παραστάσεων σε compilers.
  - Προγραμματισμός εργασιών σε σύνθετα έργα με προτεραιότητες.
  - Επίλυση γραμμικών συστημάτων ανισώσεων (περιορισμών)

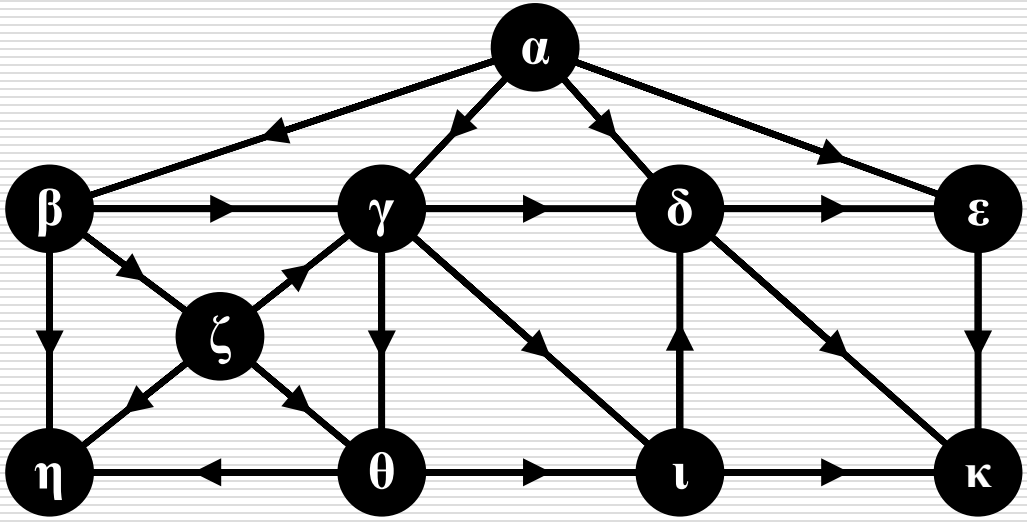




Τοπολογική Ταξινόμηση

- Τοπολογική διάταξη **ανν** γράφημα ακυκλικό (DAG).
  - **Ευθύ:**
    - Ύπαρξη κύκλου ασύμβατο με διάταξη
  - **Αντίστροφο:**
    - DAG: ύπαρξη sources
    - Διάταξη: Διαδοχική Αφαίρεση **sources** του DAG
- **DFS** ελέγχει για ύπαρξη κύκλων και υπολογίζει «σειρά» κορυφών **συμβατή** με μερική διάταξη του DAG
- Τοπολογική ταξινόμηση: Κορυφές σε φθίνουσα σειρά χρόνων αναχώρησης του DFS, δηλ. $f[v_1] > f]v_2] > \cdots > f[v_n]$
  - Χρόνος $\Theta(n+m)$.

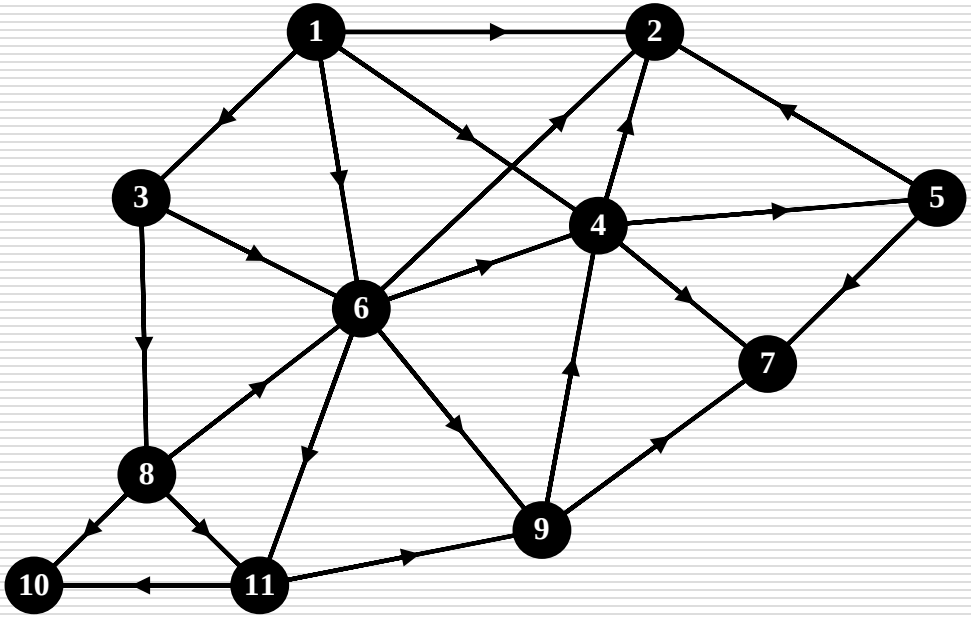
Παράδειγμα 1



v	d	f
α	1	**20**
β	2	**19**
γ	3	**16**
δ	4	**9**
ε	5	**8**
ζ	17	**18**
η	11	**12**
θ	10	**15**
ι	13	**14**
κ	6	**7**

Τοπολογική διάταξη:
α, β, ζ, γ, θ, ι, η, δ, ε, κ

Παράδειγμα 2 - άσκηση



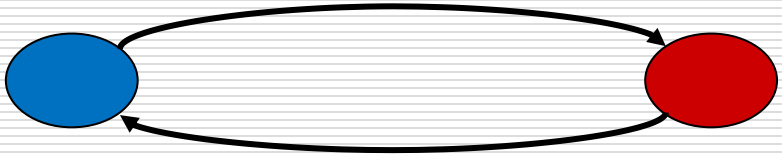
Τοπολογική διάταξη:
1, 3, 8, 6, 11, 10,
9, 4, 5, 7, 2

Τοπολογική Ταξινόμηση: Ορθότητα

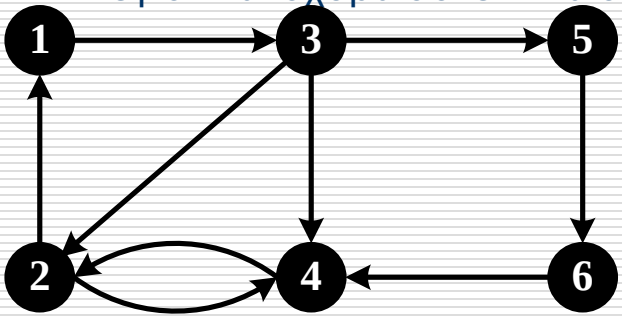
- Έστω DAG $G(V, E)$. Θδο $\forall (u, v) \in E,\ f[u] > f[v]$.
  - Εξερεύνηση (u, v) σημαίνει $u = Y$ και $v \in \{A, E, Y\}$:
  - **v = A**
    - v απόγονος της u στο DFS-δάσος.
    - Άρα $f[u] > f[v]$, γιατί πρώτα τίθεται f[v] και μετά f[u].
  - **v = E**
    - εξερεύνηση της v ολοκληρώθηκε πριν ολοκληρωθεί εξερεύνηση u, άρα $f[u] > f[v]$.
  - **v = Y**
    - αποκλείεται γιατί σημαίνει πίσω ακμή δηλ. κύκλο.

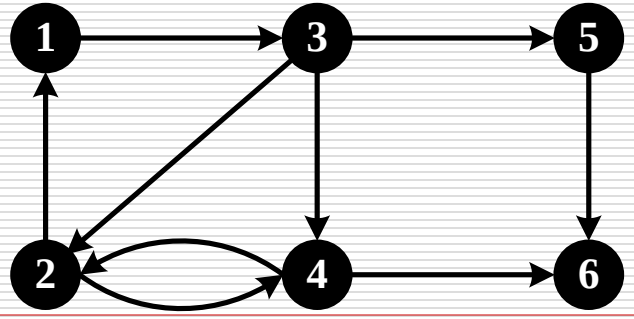
Ισχυρά Συνεκτικές Συνιστώσες

- Κατευθυνόμενο γράφημα $G(V, E)$ **ισχυρά συνεκτικό** αν $\forall u, v \in V$, υπάρχουν $u \rightsquigarrow v$ και $v \rightsquigarrow u$ μονοπάτια.
  - Για κάθε ζευγάρι κορυφών ισχυρά συνεκτικού γραφήματος, υπάρχει κυκλική διαδρομή που τις περιλαμβάνει.



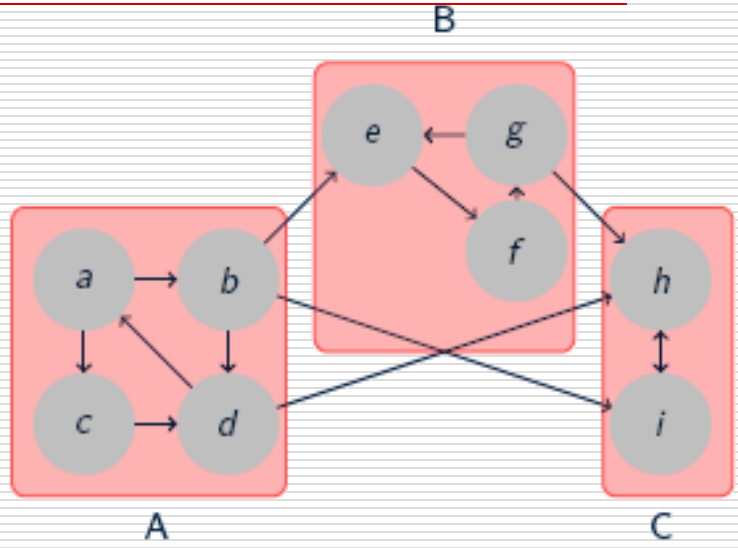
- Αν ένα κατευθυνόμενο γράφημα δεν είναι ισχυρά συνεκτικό, διαμερίζεται σε ισχυρά συνεκτικές συνιστώσες:
  - Μεγιστικά ισχυρά συνεκτικά υπογραφήματα.

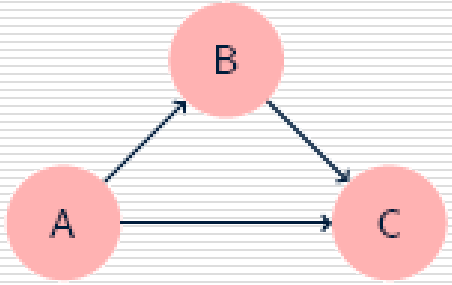




(Μετα)-Γράφημα Ισχυρά Συνεκτικών Συνιστωσών (ΙΣΣ)

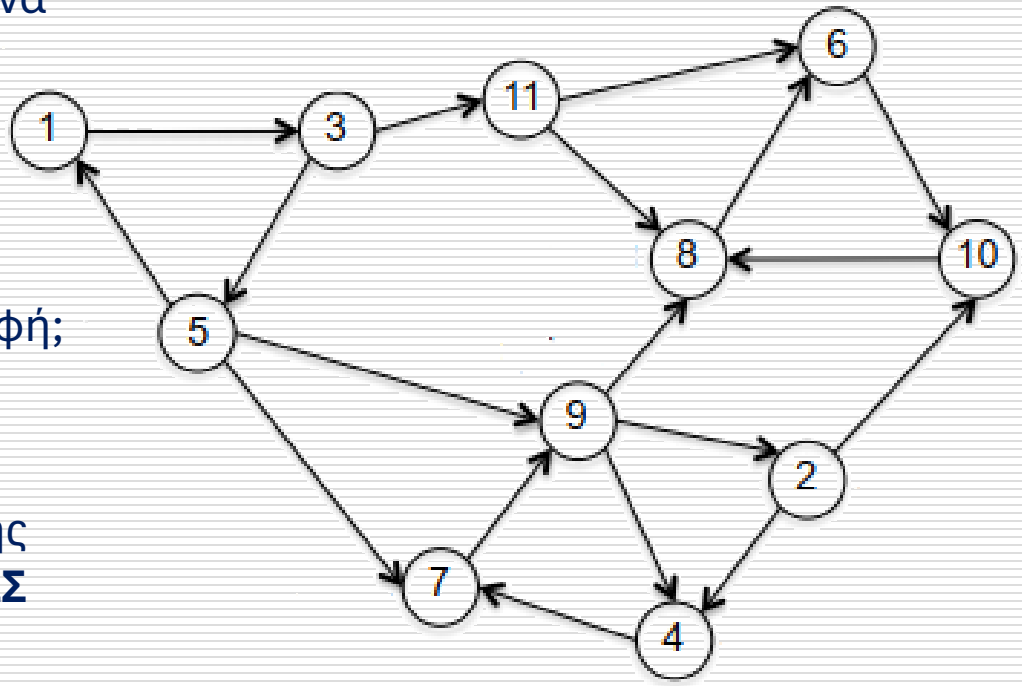
- (Κατευθυνόμενο) γράφημα ισχυρά συνεκτικών συνιστωσών (ΙΣΣ):
  - Κορυφή για κάθε ΙΣΣ
  - Ακμή (X, Y) από ΙΣΣ X προς ΙΣΣ Y ανν (αρχικά) υπάρχει τουλάχιστον 1 ακμή (v, u) για $v \in X$ και $u \in Y$.
- Γράφημα ΙΣΣ είναι **ακυκλικό** (DAG- γιατί?) και **μπορεί να ταξινομηθεί τοπολογικά.**
- ΙΣΣ με DFS?
  - Για κάθε κορυφή BFS
  - $O(nm)$
  - Μπορούμε καλύτερα?
    - **Ναι** σε $O(n+m)$ με 2 DFS



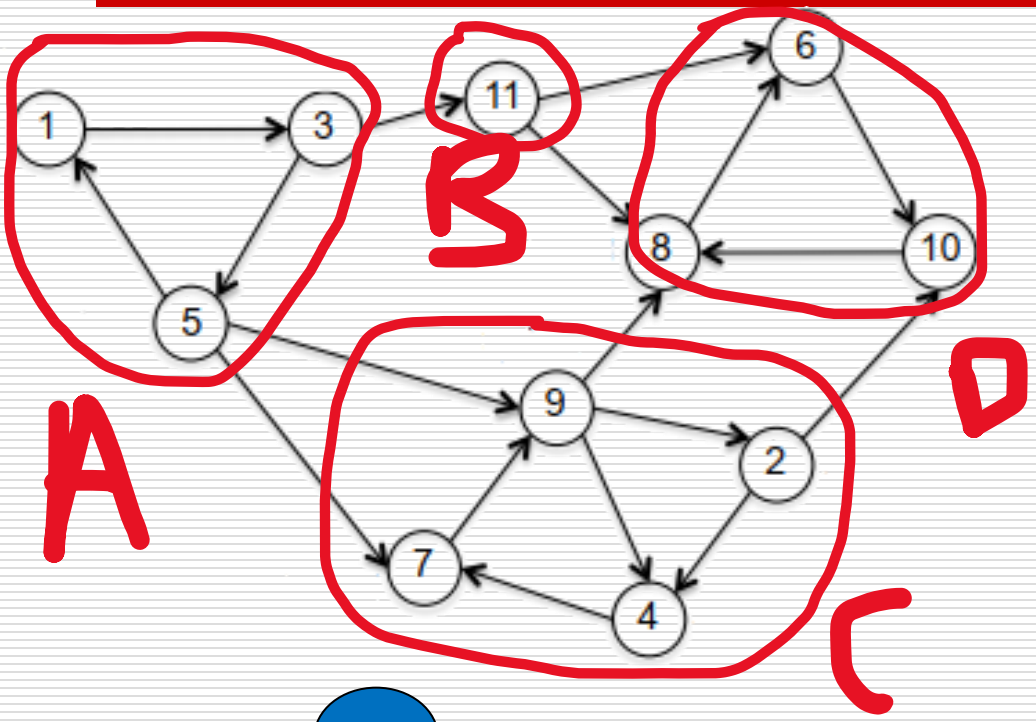


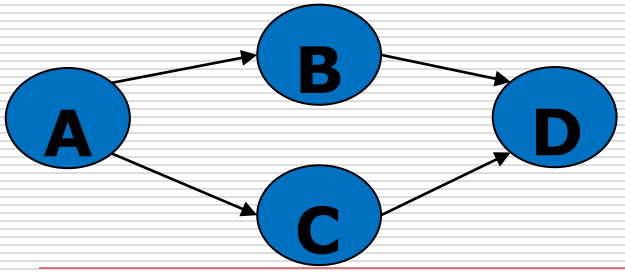
Γραμμικός Υπολογισμός Ισχυρά Συνεκτικών Συνιστωσών

- **Διαίσθηση:**
  - Θέλουμε κάποια κορυφή που να ανήκει σε sink ΙΣΣ (γιατί?)
- **Ιδέα:**
  - Υπολογισμός τοπολογικής διάταξης (χωρίς πίσω ακμές)
  - Εκκίνηση από τελευταία κορυφή;
- **Πρόβλημα:**
  - **Δεν** είναι υποχρεωτικό ότι η τελευταία κορυφή τοπολογικής διάταξης **θα ανήκει σε sink ΙΣΣ**



Υπολογισμός Ισχυρά Συνεκτικών Συνιστωσών





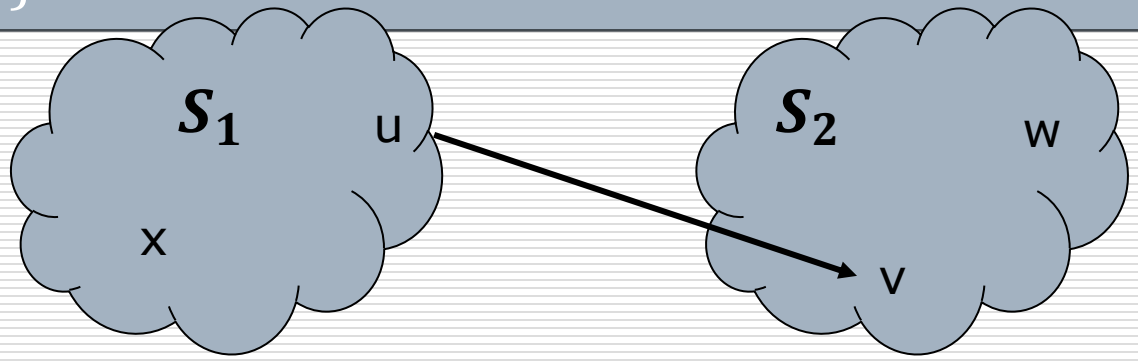
(μία) Τοπολογική Ταξινόμηση:

- 1
- 3
- 11
- 5
- 7
- 9
- 2
- 10
- 8
- 6
- *4 (ανήκει C και όχι D) ΠΡΟΒΛΗΜΑ*

Υπολογισμός Ισχυρά Συνεκτικών Συνιστωσών

- Όμως ισχύει το *ανάστροφο*: Το πρώτο στοιχείο **οποιασδήποτε** τοπολογικής ταξινόμησης ανήκει στο source ΙΣΣ.
- Πιο ισχυρά ισχύει το εξής θεώρημα:

Αν υπάρχει ακμή (u, v) μεταξύ διαφορετικών ΙΣΣ S_1 και S_2 με $u \in S_1$ και $v \in S_2$. Τότε $f[S_1] > f[S_2]$ όπου $f[S] = \max\{ f[v], v \in S\}$



Υπολογισμός Ισχυρά Συνεκτικών Συνιστωσών

- **Έστω $d[S_1] < d[S_2]$** – ανακαλύπτω πρώτα S_1 από κορυφή x.
- $\forall\, w \in S_2$: Υπάρχει μονοπάτι: $x \to u \to v \to w$ με όλες τις κορυφές ανεξερεύνητες.
- Άρα $f[x] > \max\{f[w] \Rightarrow f[S_1] > f[S_2]$
- **Αλλιώς $d[S_2] < d[S_1]$** – ανακαλύπτω πρώτα S_2 από κορυφή x.
- Εφόσον δεν υπάρχει μονοπάτι S_2 → S_1(DAG): S_1 ανεξερεύνητο
- Τότε $f[S_2] = f[y]$ και $f[S_1] > f[S_2]$

Υπολογισμός ΙΣΣ – Ανάστροφο γράφημα

- Ανάστροφο γράφημα G^T: ίδιες κορυφές, αντίστροφη φορά ακμών
  - Το sink του G - source στο G^T.
- Υπολογίζεται σε χρόνο $\boldsymbol{\Theta(n+m)}$
- Έχει τις **ίδιες** ισχυρά συνεκτικά συνιστώσες.
- Αν $f[S_1] > f[S_2]$ στο G, τότε δεν υπάρχει ακμή $S_1 \to S_2$ στο G^T.
  - contrapositive θεωρήματος
- **Συμπέρασμα**: Θα εγκλωβιστούμε στα components αν τα δούμε με τη σωστή σειρά

Αλγόριθμος Kosaraju (– Sharir)

Εύρεση ΙΣΣ σε κατευθυνόμενο $G(V, E)$:

1. **Υπολόγισε** τοπολογική διάταξη με DFS
   (αγνοώντας τις πίσω ακμές).
2. **Υπολόγισε** το ανάστροφο γράφημα G^T
   (αντιστροφή κατεύθυνσης ακμών, G και G^T έχουν ίδιες ΙΣΣ).
3. **Εφάρμοσε** DFS σε G^T με σειρά της τοπολογικής διάταξης στο Β1.
   I. Κάθε φορά που φτάνουμε σε αδιέξοδο: νέα ΙΣΣ.

Δύο DFS: $\Theta(n + m)$

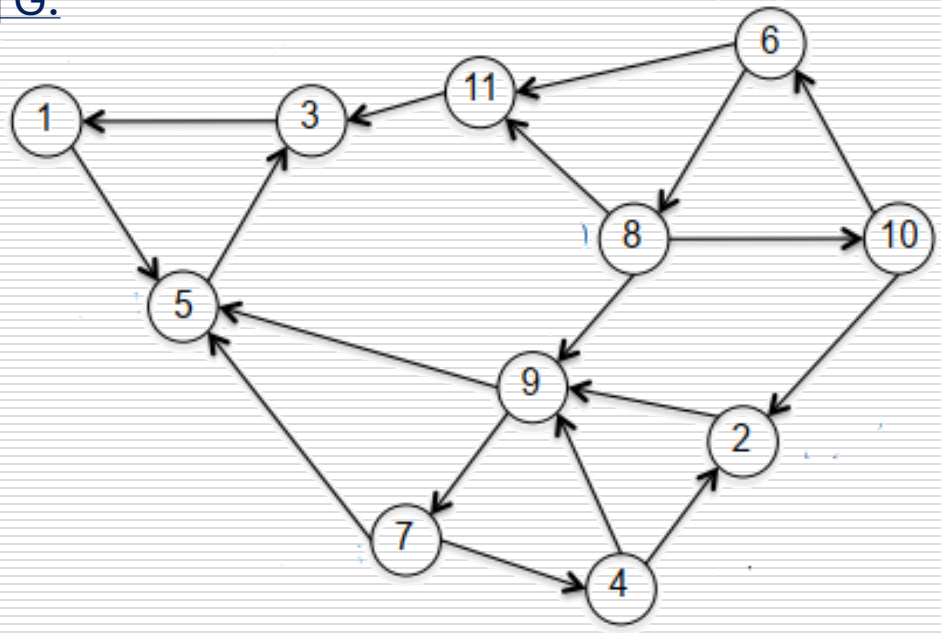
Βελτίωση: Tarjan με μία DFS.

Ολοκληρωμένο παράδειγμα

Τοπολογική Ταξινόμηση G:

- 1
- 3
- 11
- 5
- 7
- 9
- 2
- 10
- 8
- 6
- 4

Υπολογισμός G^T



DFS(G^T) με σειρά τοπολογικής

- 1 → {1,3,5}
- 3 → E
- 11 → {11}
- 5 → E
- 7 → {7,4,2,9}
- 9 → E
- 2 → E
- 10 → {10,8,6}
- 8 → E
- 6 → E
- 4 → E