Διάλεξη 24: Ελάχιστα Γεννητορικά Δένδρα Αλγόριθμος Kruskal

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

*Ο αλγόριθμος του **Kruskal** για εύρεση ΕΓΔ σε γράφους*

Παράδειγμα Εκτέλεσης

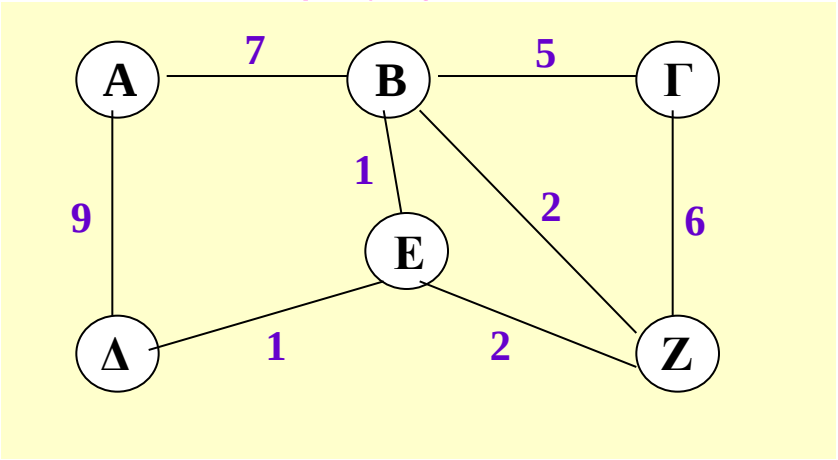
Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Ο αλγόριθμος του Kruskal

- Ακόμα ένας άπληστος (greedy) αλγόριθμος που υπολογίζει το Ελάχιστο Γεννητορικό Δένδρο (ΕΓΔ).
- Ενώ ο αλγόριθμος του Prim επεξεργάζεται μια-μια τις ***κορυφές***, ο αλγόριθμος του Kruskal επεξεργάζεται μια-μια τις ***ακμές*** του γράφου.
- Επίσης, ενώ σε κάθε βήμα του αλγόριθμου του Prim οι επιλεγμένες ακμές σχηματίζουν ένα **δένδρο**, στην περίπτωση του αλγόριθμου Kruskal, σχηματίζουν ένα **δάσος** (ένα σύνολο από δένδρα).
- **Κεντρική ιδέα.**
  - Αρχικά το δάσος T είναι άδειο.
  - Επεξεργαζόμαστε μια-μια τις ακμές, **σε αύξουσα σειρά βάρους.**
  - Αν η εισαγωγή της e στο T **δεν προκαλεί κύκλο**, τότε προσθέτουμε την e στο T, δηλαδή $T := T \cup \{e\}$.

Παράδειγμα Εκτέλεσης

Γράφος G

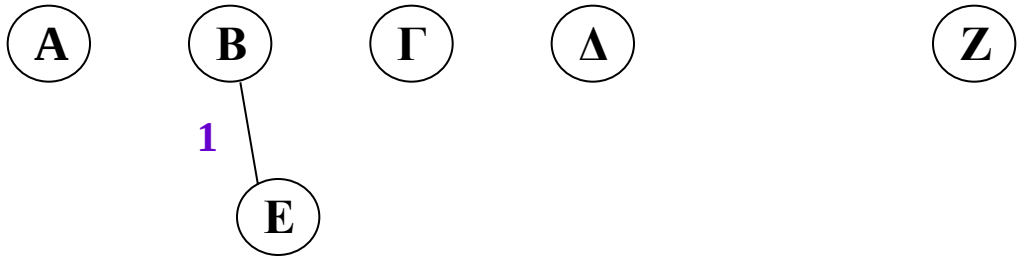


Ταξινομημένες Ακμές

Β-Ε = 1
Δ-Ε = 1
Β-Ζ = 2
Ε-Ζ = 2
Β-Γ = 5
Γ-Ζ = 6
Α-Β = 7
Α-Δ = 9

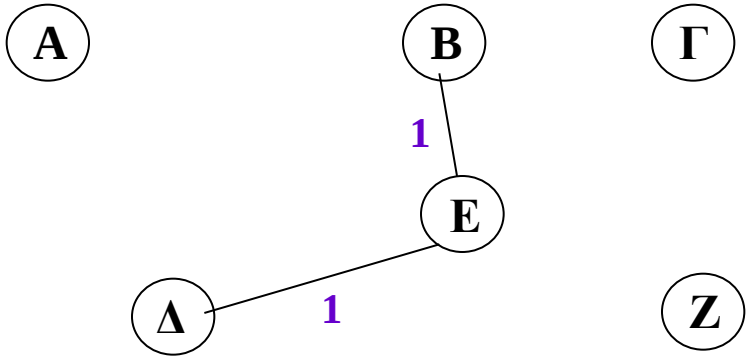
Αρχική Κατάσταση

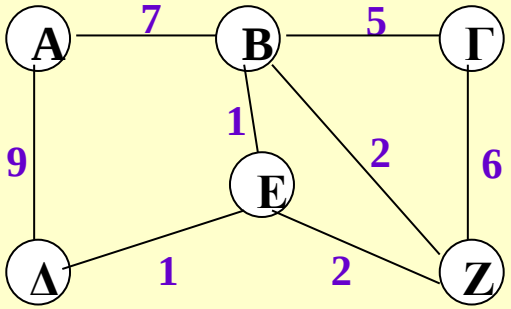
Μετά από επιλογή της πρώτης ακμής (Β,Ε)



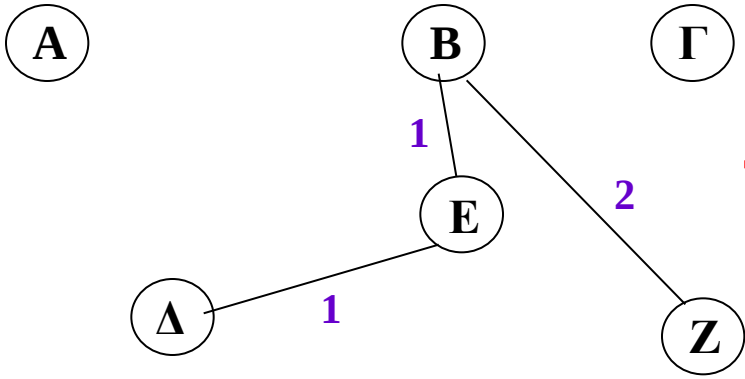
Παράδειγμα Εκτέλεσης

Μετά από επιλογή της **δεύτερης** ακμής (Δ,Ε)





Μετά από επιλογή της **τρίτης** ακμής (Β,Ζ)



Ταξινομημένες Ακμές

Β-Ε = 1

Δ-Ε = 1

Β-Ζ = 2

Ε-Ζ = 2

Β-Γ = 5

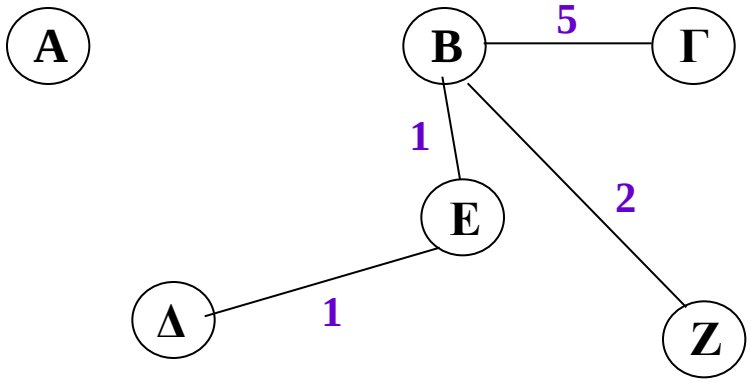
Γ-Ζ = 6

Α-Β = 7

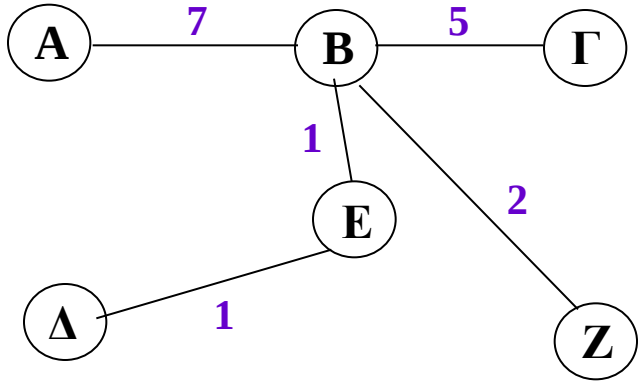
Α-Δ = 9

Παράδειγμα Εκτέλεσης

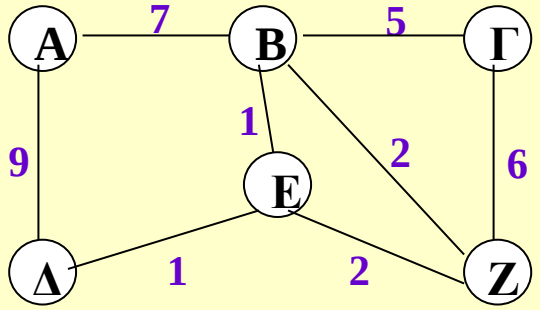
Μετά από επιλογή της <u>τέταρτης</u> ακμής (Β,Γ)



Μετά από επιλογή της <u>πέμπτης</u> ακμής (Α,Β)



<u>Γράφος G</u>



Ταξινομημένες Ακμές

Β-Ε = 1

Δ-Ε = 1

Β-Ζ = 2

~~**Ε-Ζ = 2**~~ **(Κύκλος)**

Β-Γ = 5

~~**Γ-Ζ = 6**~~ **(Κύκλος)**

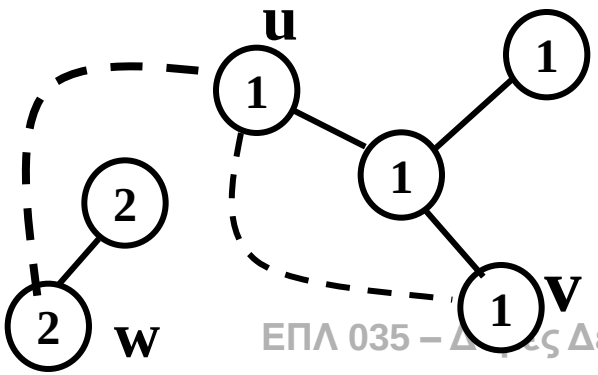
Α-Β = 7

Τέλος

Α-Δ = 9

Αποδοτική Εξάλειψη Κύκλων

- Η ταξινόμηση των Άκμων είναι απλή, δηλαδή ταξινομούμε μια φορά όλες τις ακμές με κάποιο αλγόριθμο ταξινόμησης.
- **Βασικό Πρόβλημα:** Το πρόβλημα που απομένει είναι πως θα βρίσκουμε αποδοτικά εάν μια ακμή μπορεί να δημιουργήσει κύκλο
- **Λύση**
  - Θα χρησιμοποιήσουμε ένα πίνακα **TID[n]** (TreeID) ο οποίος μας υποδεικνύει για κάθε κορυφή v σε πιο δένδρο ανήκει η v.
  - **Π.χ. εάν** θέλω να **προσθέσω** μια ακμή **(u,v)** και u & v ανήκουν στο ίδιο δένδρο **(TID[u]==TID[v]),** τότε αυτή η ακμή θα δημιουργήσει κύκλο.
  - **Επομένως δε θα προσθέσω την (u,v)**

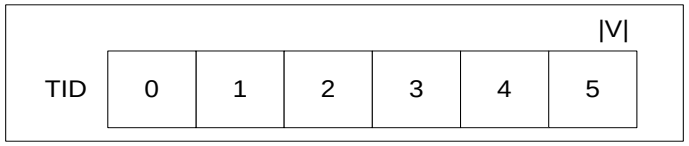


• Το (u,v) θα δημιουργήσει κύκλο γιατί και τα δυο ανήκουν στο ίδιο TID (i.e., 1)

• Το (u,w) δεν θα δημιουργήσει κύκλο γιατί οι δυο κόμβο ανήκουν σε διαφορετικά TID (i.e., 1 και 2)

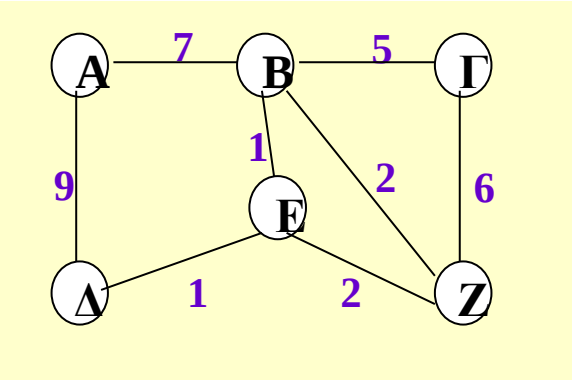
Υλοποίηση του Αλγορίθμου Kruskal

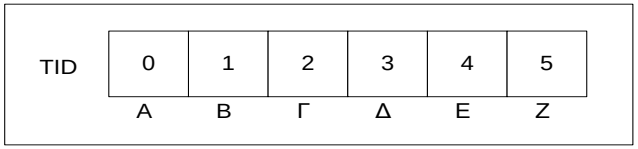
```
Kruskal(graph G(V,E)){
   Tree = {};  // Το ΕΓΔ: Ένα σύνολο ακμών (αρχικά κενό)
   TID[|V|] = {} // Πίνακας που κρατά το TreeID του κάθε κόμβου
   Count = 0;     // Μετρητής που κρατά από πόσες κορυφές πέρασα

   sortEdges(E);  // Ταξινόμηση Άκμων σε χρόνο O(|E|.log|E|)

   // Δημιουργούμε ένα δάσος από δένδρα μεγέθους 1
   for (i=0; i<|V|; i++)          // χρόνο Θ(|V|)
      TID[i] = i;

   for (i=0; i<|E|; i++) {        // χρόνο O(|E|)
      // ανάκτηση επόμενης (μικρότερης) ακμής
      (u,v) = nextEdge();         // χρόνο Θ(1)

      If (TID[u] != TID[v]) {     // Αν ανήκουν σε διαφορετικά δένδρα
          Tree = Tree ∪ {(u,v)};  // Προσθήκη Ακμής

          // Μετρούμε ποιος από τους u,v εμφανίζεται περισσότερο στο TID
          // και επιστρέφουμε τον μεγαλύτερο σαν TreeID x
          x = occurence(TID, u, v);   // χρόνο O(|V|)

          // Ανάθεση TreeID x σε όλους που έχουν  TID[u] ή TID[v]
          count = merge(TID[u],TID[v],x);
      }
      if (count == |V|) break;
    }
}
```



Συνολικός χρόνος: O(|E|.log|E| + |V| + |E|*|V|)

Εκτέλεση της Υλοποίησης Kruskal

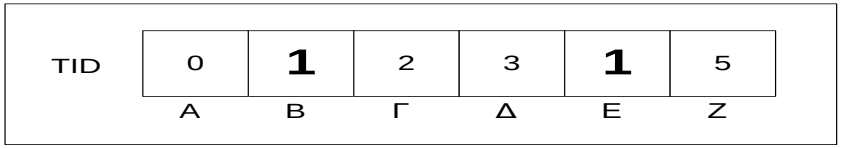




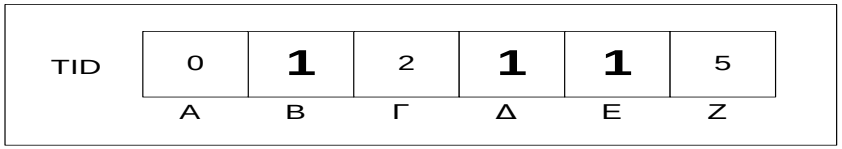
Ταξινομημένες Ακμές

{ B-E = 1, Δ-E = 1, B-Z = 2, E-Z = 2, B-Γ = 5, Γ-Z = 6, A-B = 7, A-Δ = 9 }

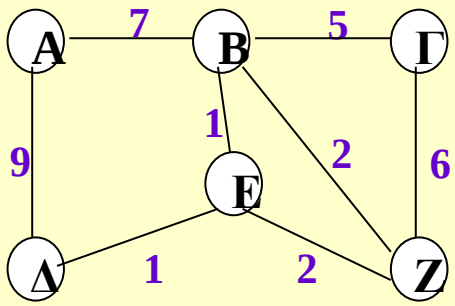
1. **Nextedge => (B,E)**
2. **TID[B] != TID[E]? => YES, Επομένως Tree = {} ∪ {(B,E)};**
3. **Merge(TID[B],TID[E],1);**



4. **Nextedge => (Δ,E)**
5. **TID[Δ] != TID[E]? => YES, Επομένως Tree={(B,E)} ∪ {(Δ,E)};**
6. **Merge(TID[Δ],TID[E],1);**



Εκτέλεση της Υλοποίησης Kruskal



TID	0	1	2	1	1	5
	A	B	Γ	Δ	E	Z

Ταξινομημένες Ακμές →

{ B-E = 1, Δ-E = 1, B-Z = 2, E-Z = 2, B-Γ = 5, Γ-Z = 6, A-B = 7, A-Δ = 9 }

7. Nextedge => (B,Z)

8. TID[B] != TID[Z]? => YES, Επομένως Tree = {(B,E),(Δ,E)} ∪ {(B,Z)};

9. Merge(TID[B],TID[Z],1);

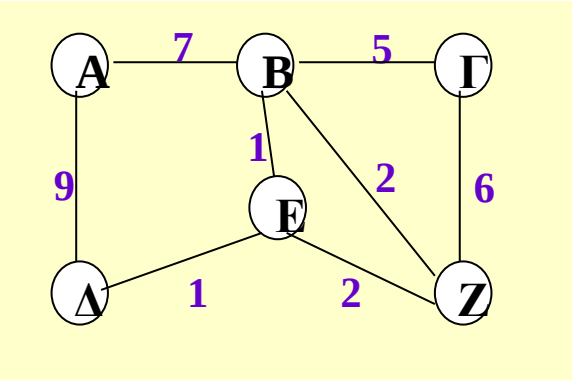
TID	0	1	2	1	1	1
	A	B	Γ	Δ	E	Z

10. Nextedge => (E,Z)

11. TID[E] != TID[Z]? => NO, Επομένως δεν χρησιμοποιούμε το (E,Z);

TID	0	1	2	1	1	1
	A	B	Γ	Δ	E	Z

Εκτέλεση της Υλοποίησης Kruskal



TID	0	1	2	1	1	1
	A	B	Γ	Δ	E	Z

Ταξινομημένες Ακμές

{ B-E = 1, Δ-E = 1, B-Z = 2, E-Z = 2, B-Γ = 5, Γ-Z = 6, A-B = 7, A-Δ = 9 }

12. **Nextedge => (B,Γ)**
13. **TID[B]!=TID[Γ]? => YES,** Επομένως Tree = {(B,E),(Δ,E), (B,Z)} ∪ {(B,Γ)};
14. **Merge(TID[B],TID[Γ],1);**

TID	0	1	1	1	1	1
	A	B	Γ	Δ	E	Z

15. **Nextedge => (Γ,Z)**
16. **TID[E] != TID[Z]? => NO,** Επομένως δεν χρησιμοποιούμε το (Γ,Z);

17. **Nextedge => (A,B)**
18. **TID[A] != TID[B]? => YES,**

TID	1	1	1	1	1	1
	A	B	Γ	Δ	E	Z

Επομένως Tree = {(B,E),(Δ,E), (B,Z), (B,Γ)} ∪ {(A,B)};

Εδώ βρήκαμε |V| vertices, επομένως διακόπτουμε την αναζήτηση.

ΤΟ ΕΓΔ είναι το Tree = {(B,E),(Δ,E), (B,Z), (B,Γ),(A,B)};