Διάλεξη 32: Ελάχιστα Γεννητορικά Δέντρα Ο Αλγόριθμος Kruskal

Στην ενότητα αυτή θα μελετηθούν τα εξής επιμέρους θέματα:

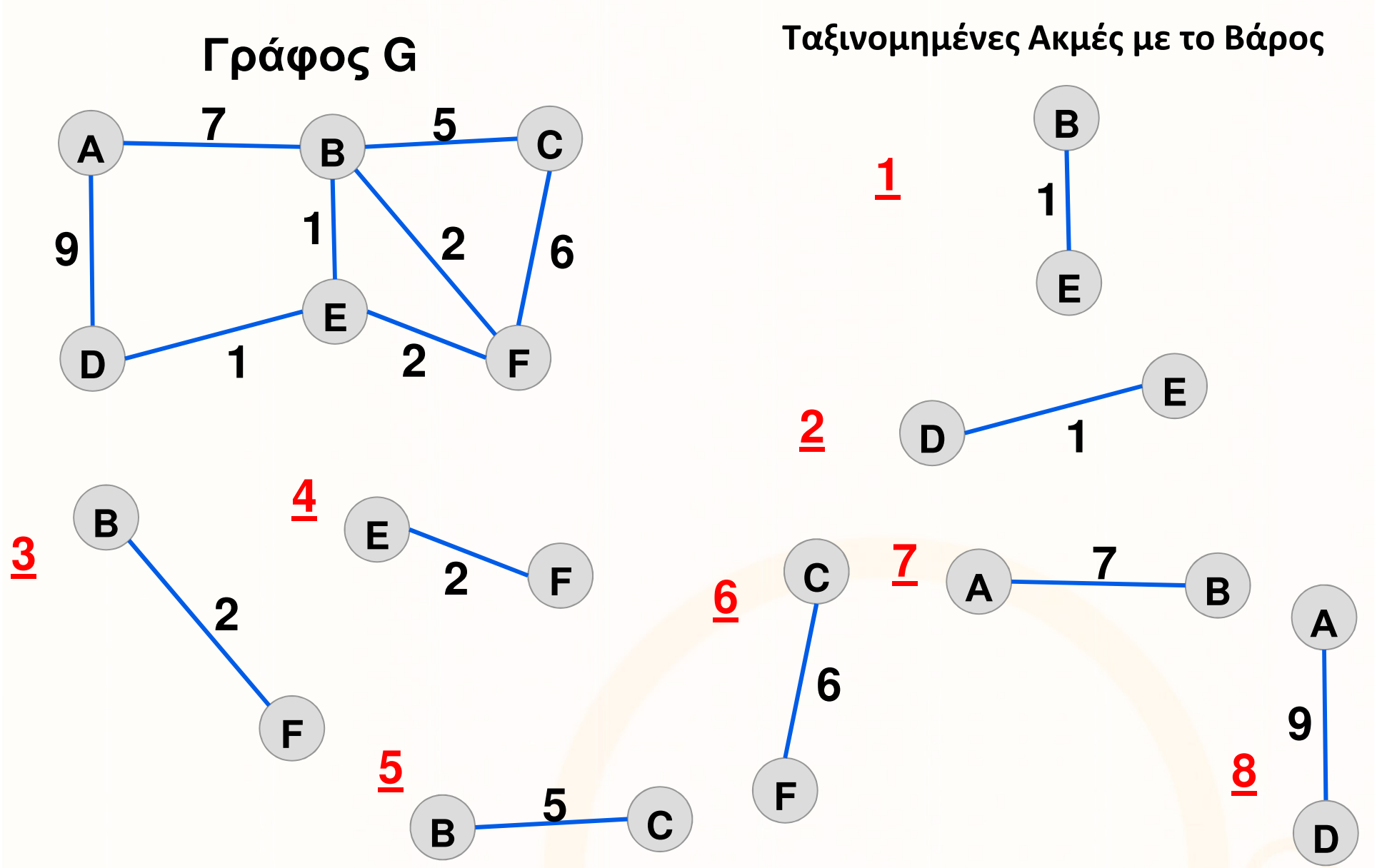
- Ο αλγόριθμος του Kruskal για εύρεση ΕΓΔ σε γράφους
- Παράδειγμα Εκτέλεσης

Διδάσκων: Παναγιώτης Ανδρέου

Ο αλγόριθμος του Kruskal

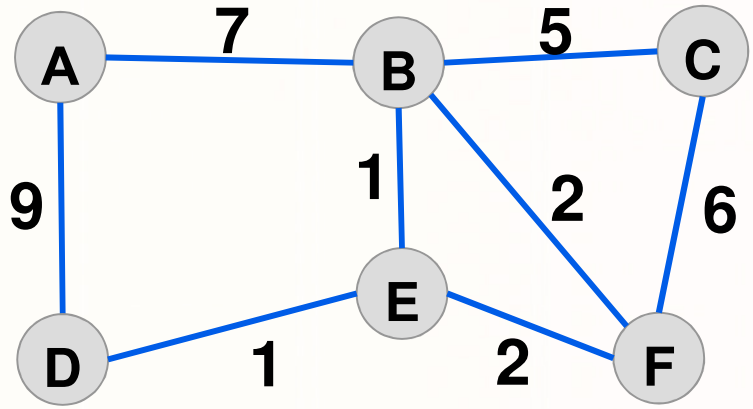
- Ακόμα ένας άπληστος (greedy) αλγόριθμος που υπολογίζει το Ελάχιστο Γεννητορικό Δένδρο (ΕΓΔ).
- Ενώ ο αλγόριθμος του Prim επεξεργάζεται μια-μια τις ***κορυφές***, ο αλγόριθμος του Kruskal επεξεργάζεται μια-μια τις ***ακμές*** του γράφου.
- Επίσης, ενώ σε κάθε βήμα του αλγόριθμου του Prim οι επιλεγμένες 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



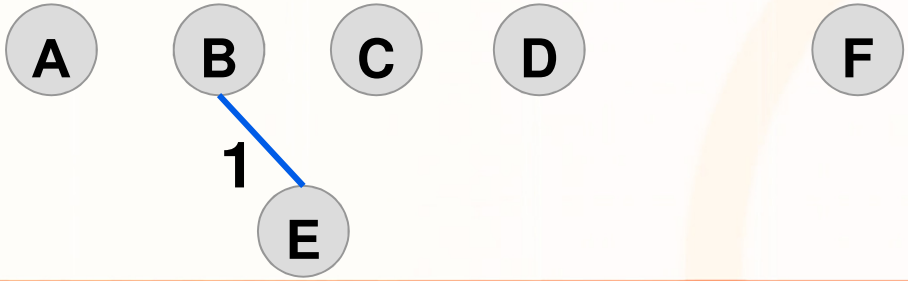
Παράδειγμα Εκτέλεσης

Γράφος G

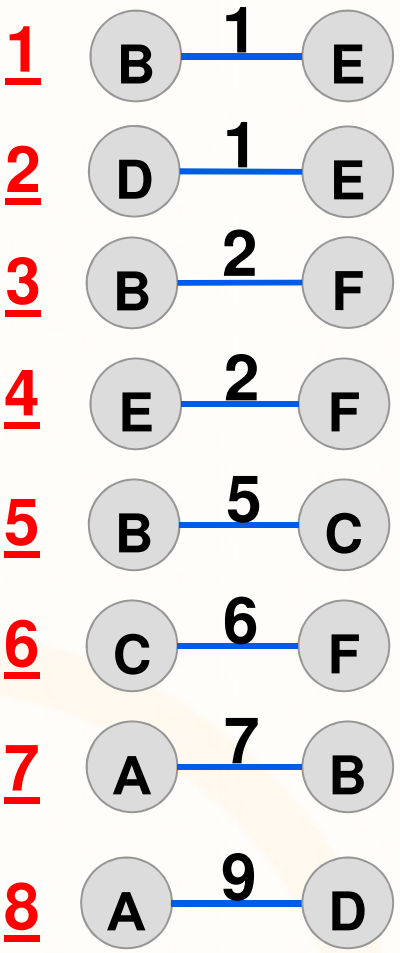


Αρχική Κατάσταση

Μετά από επιλογή της πρώτης ακμής (B,E)

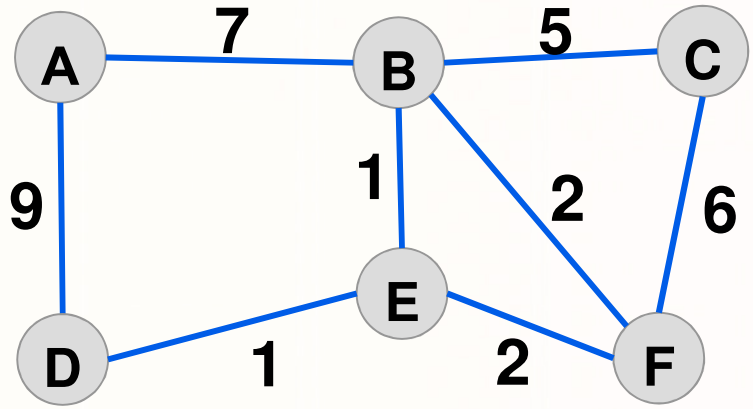


Ταξινομημένες Ακμές

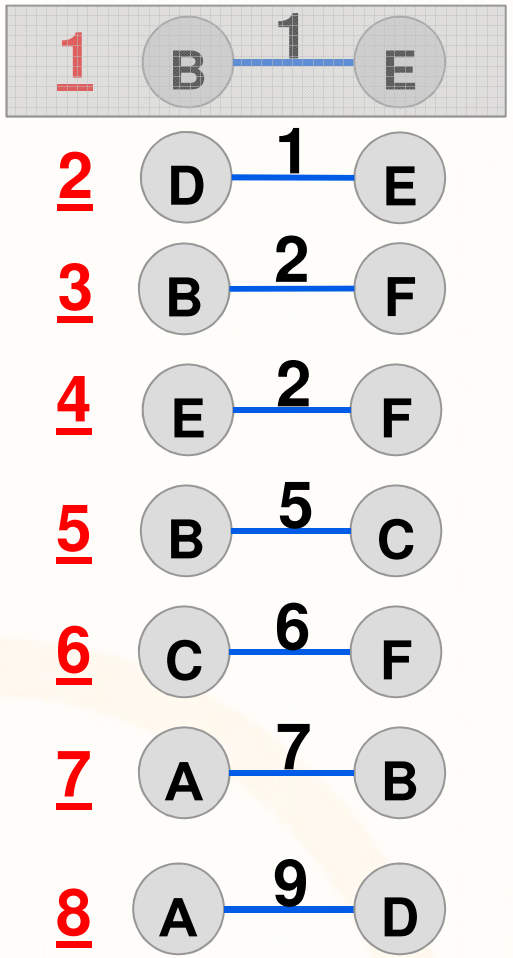


Παράδειγμα Εκτέλεσης

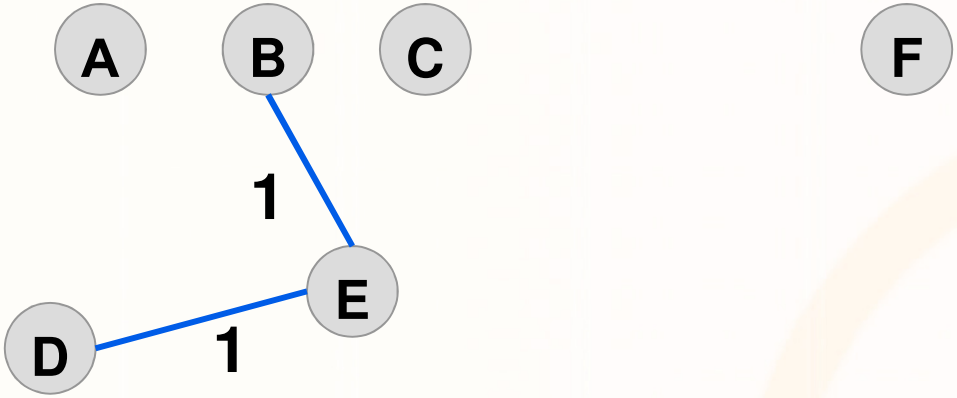
Γράφος G



Ταξινομημένες Ακμές

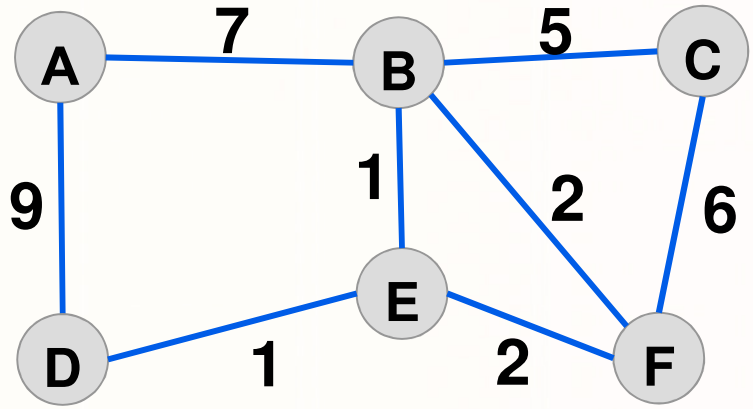


Μετά από επιλογή της πρώτης ακμής (D,E)

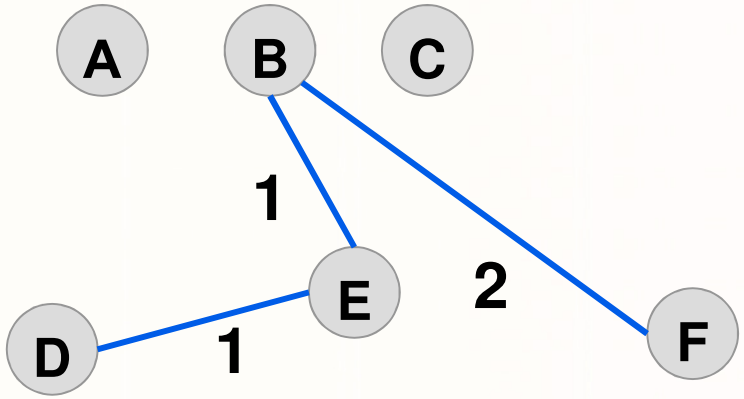


Παράδειγμα Εκτέλεσης

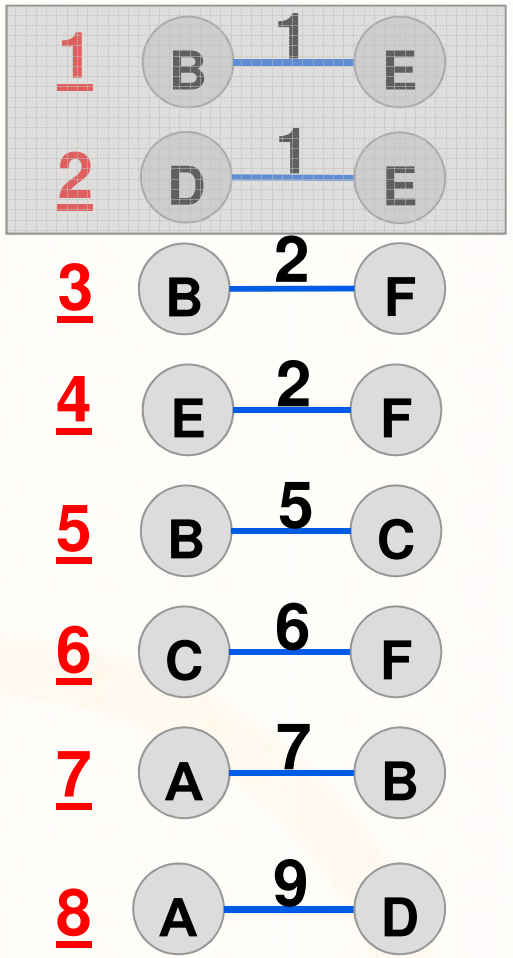
Γράφος G



Μετά από επιλογή της τρίτης ακμής (B,F)

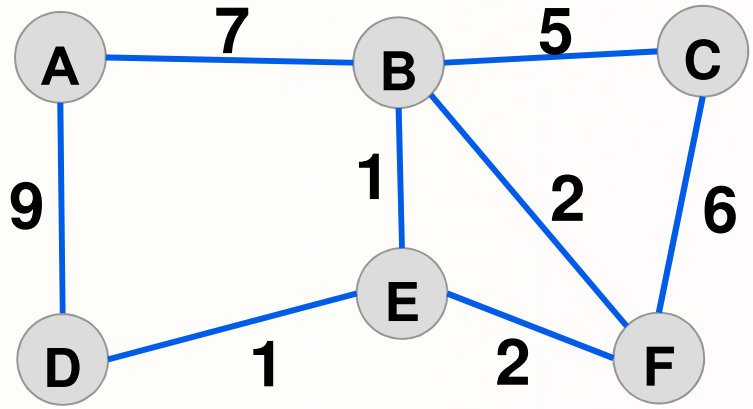


Ταξινομημένες Ακμές

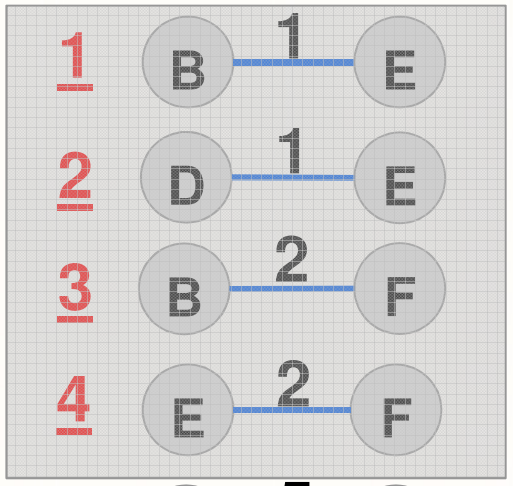


Παράδειγμα Εκτέλεσης

Γράφος G



Ταξινομημένες Ακμές



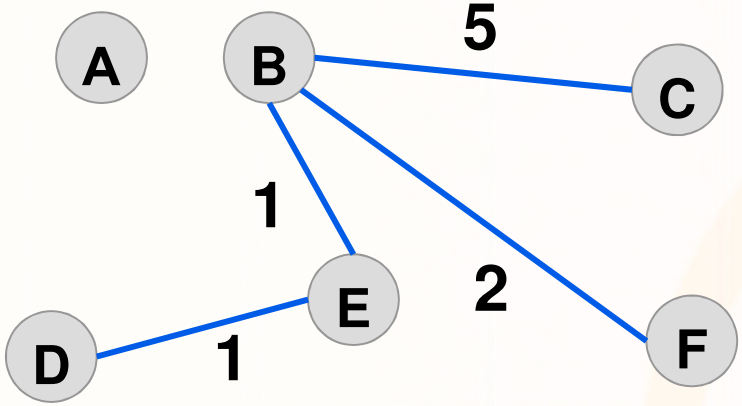
5 B 5 C

6 C 6 F

7 A 7 B

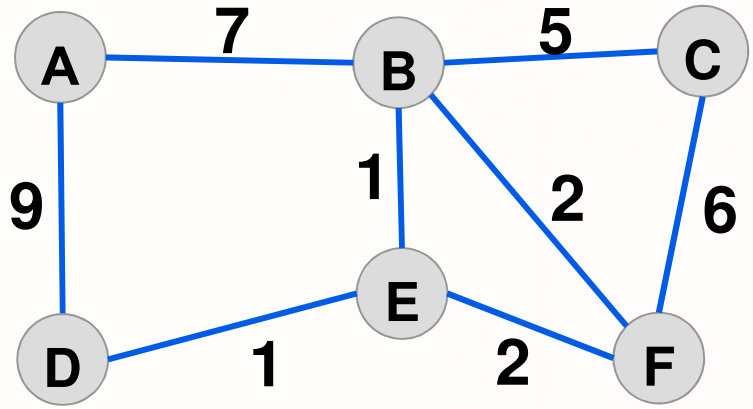
8 A 9 D

Μετά από επιλογή της τέταρτης ακμής (B,C)

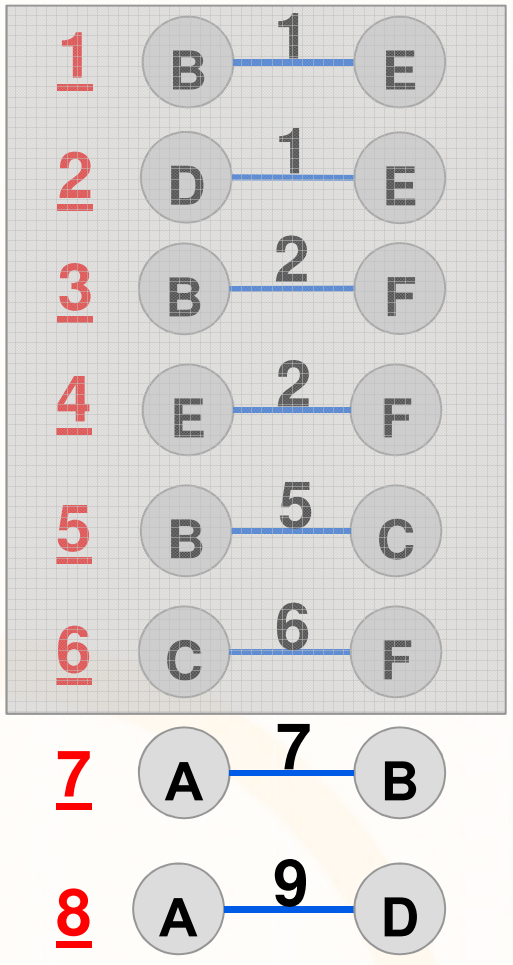


Παράδειγμα Εκτέλεσης

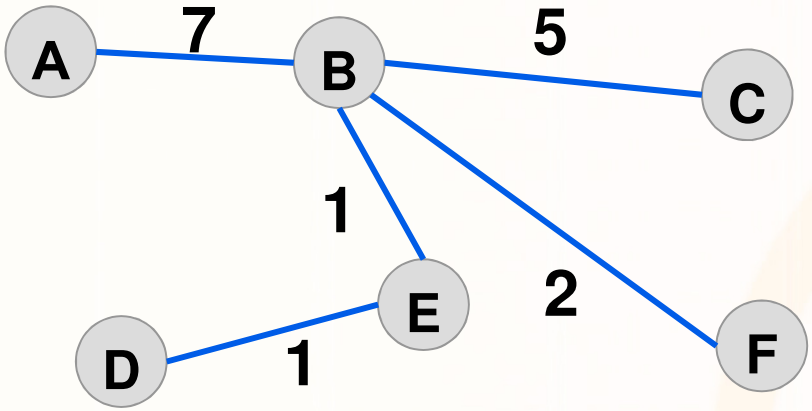
Γράφος G



Ταξινομημένες Ακμές

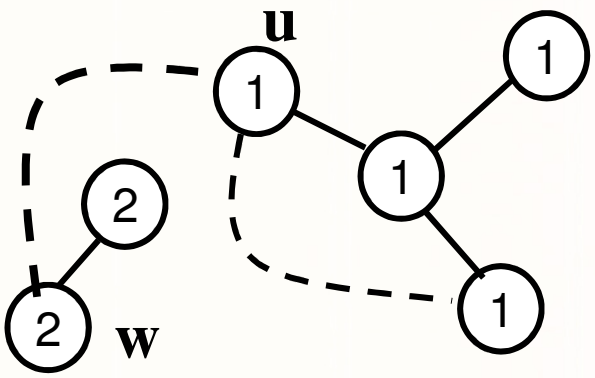


Μετά από επιλογή της τέταρτης ακμής (B,C)



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)**

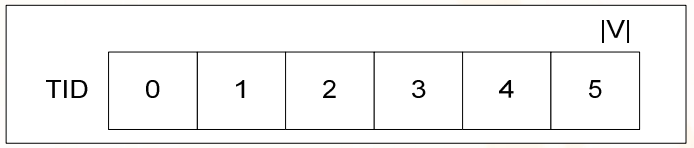


- **Το (u,v) θα δημιουργήσει κύκλο γιατί και τα δυο ανήκουν στο ίδιο TID (i.e., 1)**
- **Το (u,w) δεν θα δημιουργήσει κύκλο γιατί οι δυο κόμβο ανήκουν σε διαφορετικά TID (i.e., 1 και 2)**

Υλοποίηση του Αλγορίθμου Kruskal

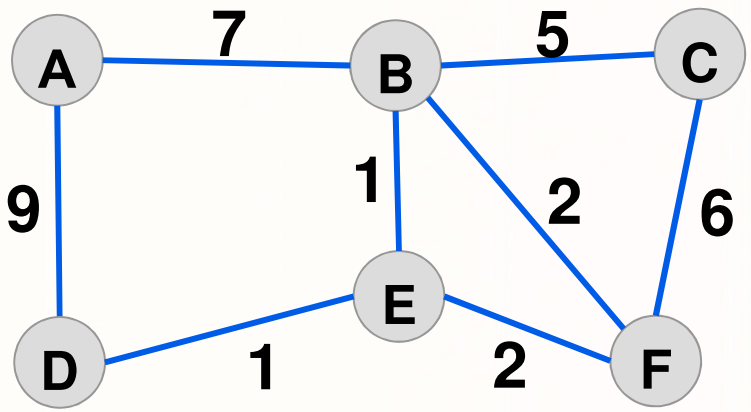
```
Kruskal(graph G(V,E)){
   Tree = {};   // Το ΕΓΔ: Ένα σύνολο ακμών (αρχικά κενό)
   TID[|V|] = {} // Πίνακας που κρατά το TreeID του κάθε κόμβου
   Count = 0;     // Μετρητής που κρατά από πόσες κορυφές πέρασα

   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

Γράφος G



Ταξινομημένες Ακμές

{ B-E=1, D-E=1, B-F=2, E-F=2, B-C=5, C-F=6, A-B=7, A-D=9 }

TID	0	1	2	3	4	5
	A	B	C	D	E	F

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

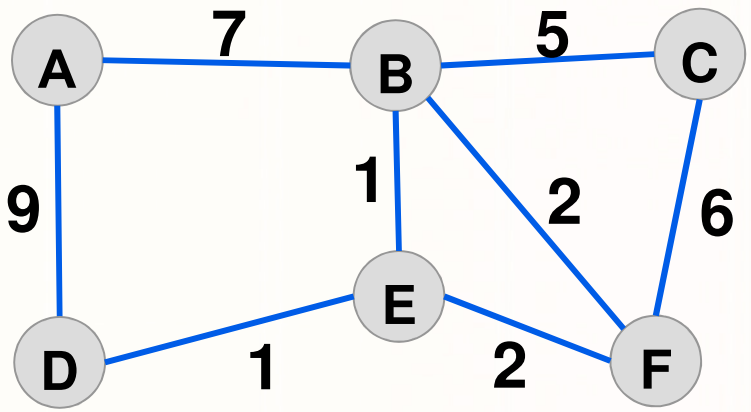
TID	0	1	2	3	1	5
	A	B	C	D	E	F

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

TID	0	1	2	1	1	5
	A	B	C	D	E	F

Εκτέλεση της Υλοποίησης Kruskal

Γράφος G



Ταξινομημένες Ακμές

{ B-E=1, D-E=1, B-F=2, E-F=2, B-C=5, C-F=6, A-B=7, A-D=9 }

TID	0	1	2	1	1	5
	A	B	C	D	E	F

7. Nextedge => (B,F)
8. TID[B] != TID[F]? => YES,Επομένως Tree = {(B,E),(D,E)} ∪ {(B,F)};
9. Merge(TID[B],TID[F],1);

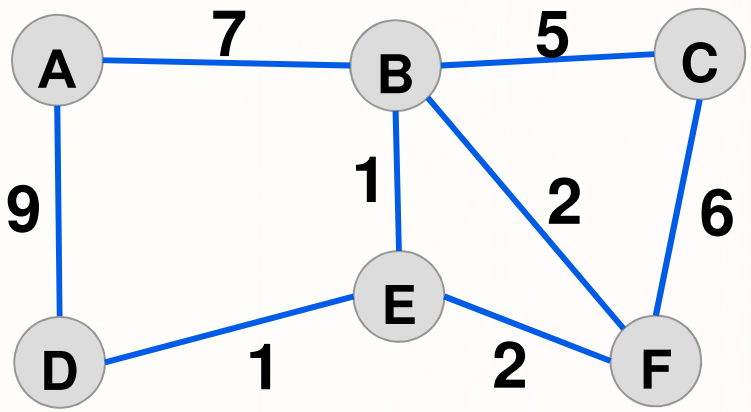
TID	0	1	2	1	1	1
	A	B	C	D	E	F

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

TID	0	1	2	1	1	1
	A	B	C	D	E	F

Εκτέλεση της Υλοποίησης Kruskal

Γράφος G



Ταξινομημένες Ακμές

{ B-E=1, D-E=1, B-F=2, E-F=2,
B-C=5, C-F=6, A-B=7, A-D=9 }

TID	0	1	2	1	1	1
	A	B	C	D	E	F

12. Nextedge => (B,C)
13. TID[B]!=TID[C]? => YES,Επομένως Tree = {(B,E),(D,E), (B,F)} ∪ {(B,C)};
14. Merge(TID[B],TID[C],1);

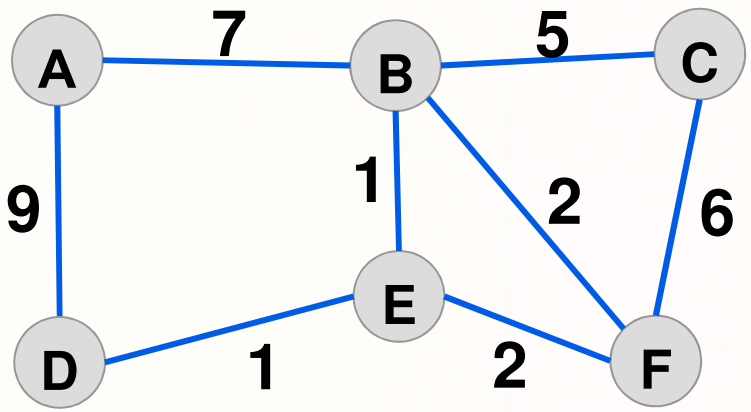
TID	0	1	1	1	1	1
	A	B	C	D	E	F

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

TID	0	1	1	1	1	1
	A	B	C	D	E	F

Εκτέλεση της Υλοποίησης Kruskal

Γράφος G



Ταξινομημένες Ακμές

{ B-E=1, D-E=1, B-F=2, E-F=2,
B-C=5, C-F=6, A-B=7, A-D=9 }

TID	0	1	1	1	1	1
	A	B	C	D	E	F

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

Επομένως Tree = {(Β,Ε),(Δ,Ε), (Β,Ζ), (Β,Γ)} ∪ {(Α,Β)};

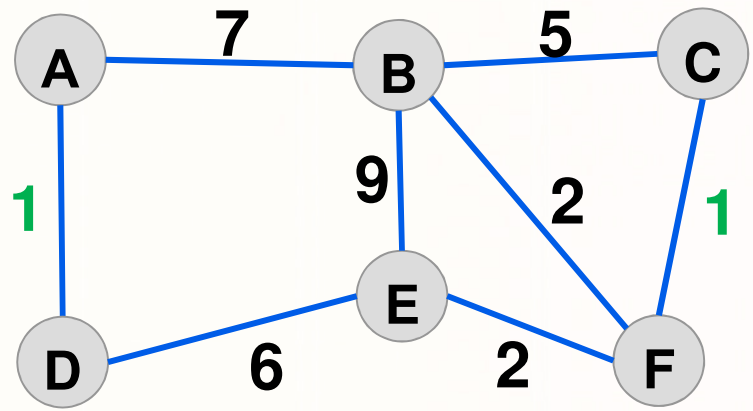
TID	1	1	1	1	1	1
	A	B	C	D	E	F

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

ΤΟ ΕΓΔ είναι το {(Β,Ε),(Δ,Ε), (Β,Ζ), (Β,Γ),(Α,Β)};

Παράδειγμα Εκτέλεσης 2

Γράφος G



Αρχική Κατάσταση

A B C D E F

Ταξινομημένες Ακμές

{ A-D=1, C-F=1, B-F=2, E-F=2,
B-C=5, D-E=6, A-B=7, B-E=9 }

0.

TID	0	1	2	3	4	5
	A	B	C	D	E	F

1.

TID	0	1	2	0	4	5
➔{A-D}	A	B	C	D	E	F

2.

TID	0	1	2	0	4	2
➔{C-F}	A	B	C	D	E	F

3.

TID	0	2	2	0	4	2
➔{B-F}	A	B	C	D	E	F

4.

TID	0	2	2	0	2	2
➔{E-F}	A	B	C	D	E	F

5. ➔{E-F} = τίποτα

6.

TID	2	2	2	2	2	2
➔{D-E}	A	B	C	D	E	F

7. break