

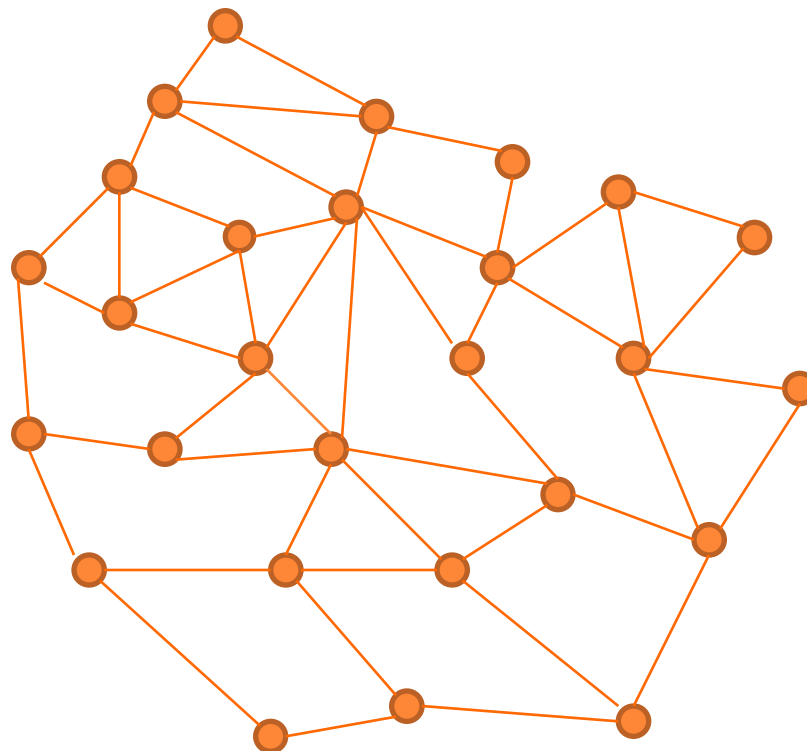
MINIMALNO VPETO DREVO

(minimum spanning tree)

- Primov algoritem
- **Kruskalov algoritem**

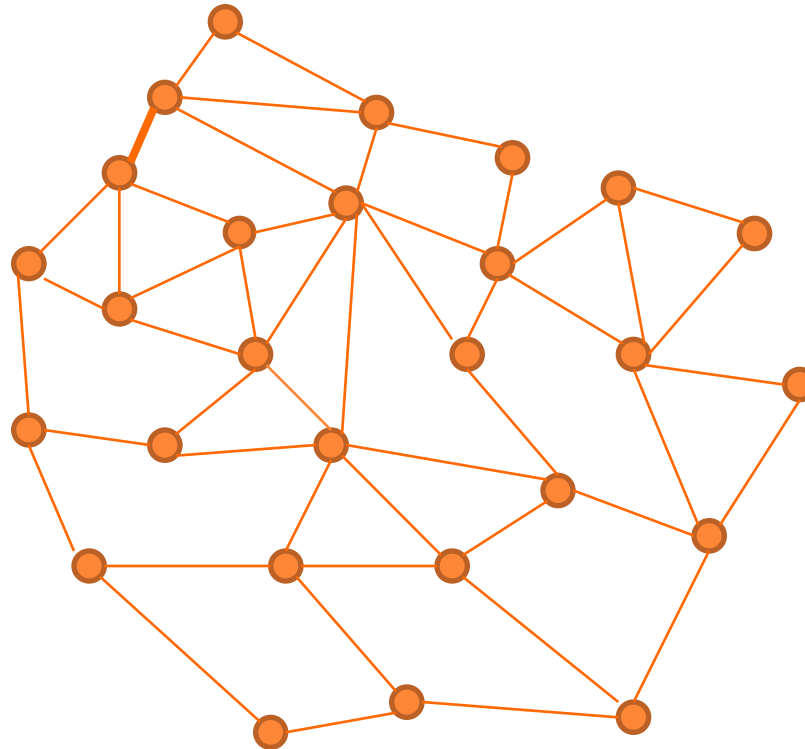
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



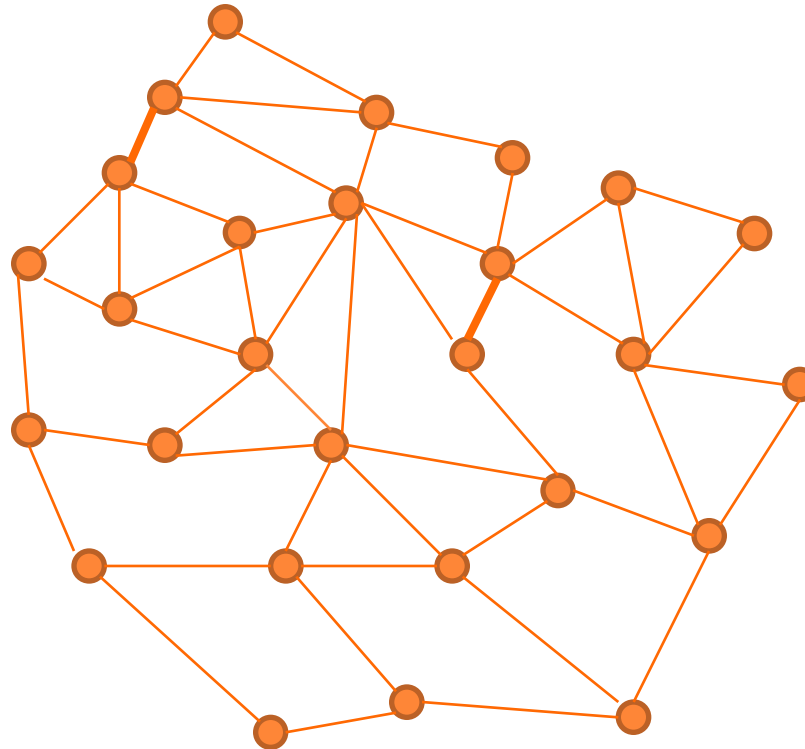
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



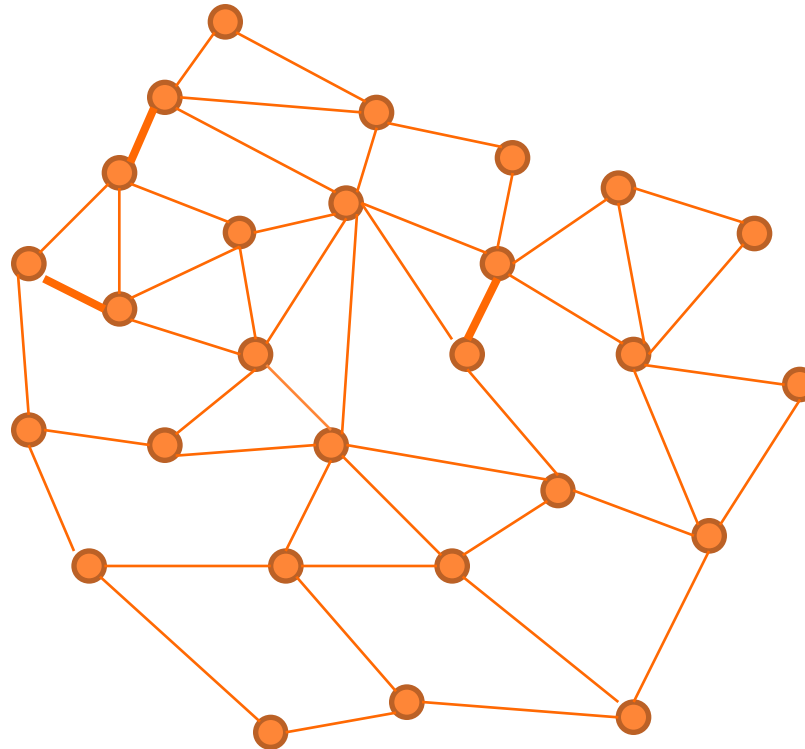
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



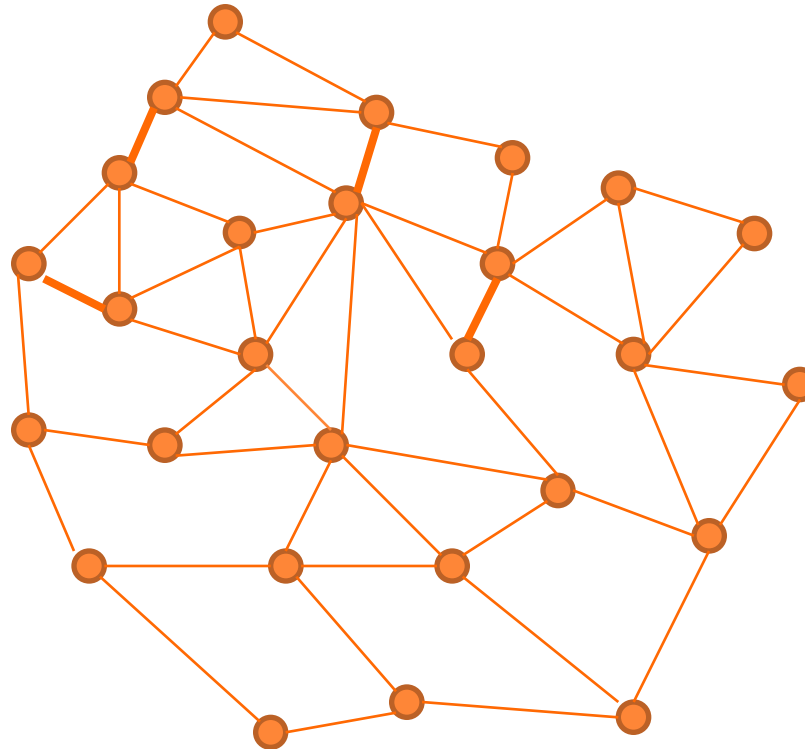
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



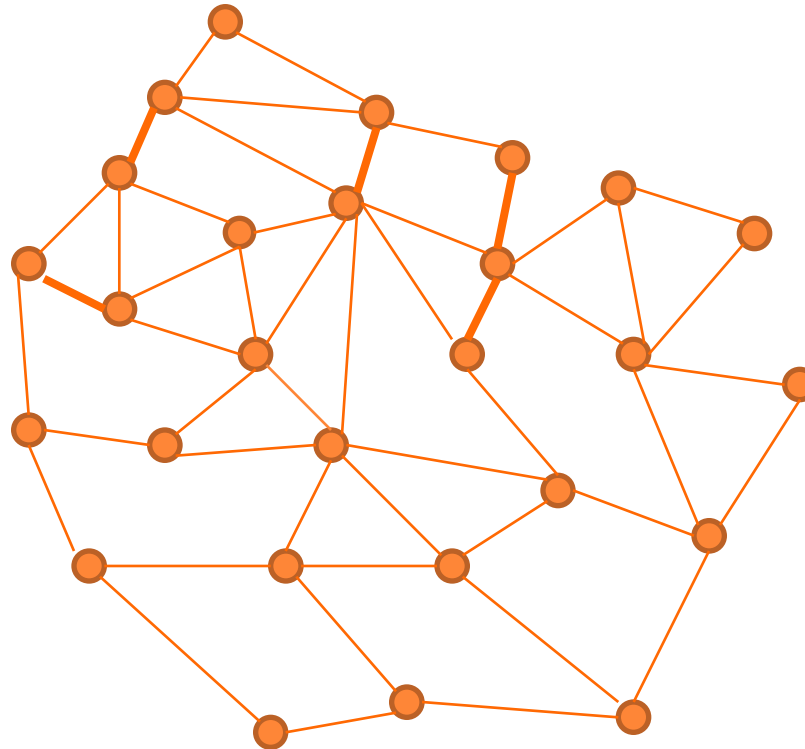
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



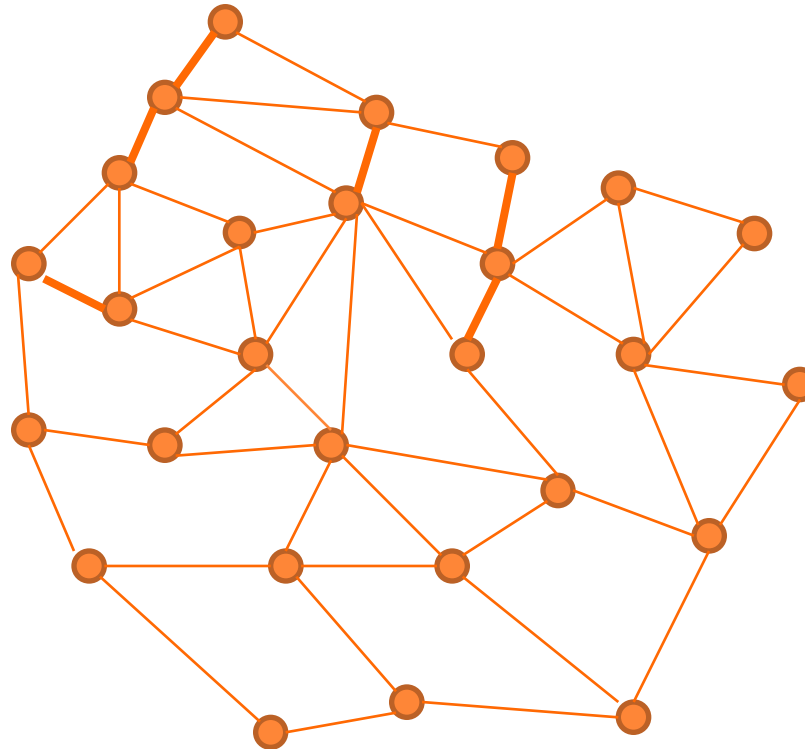
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



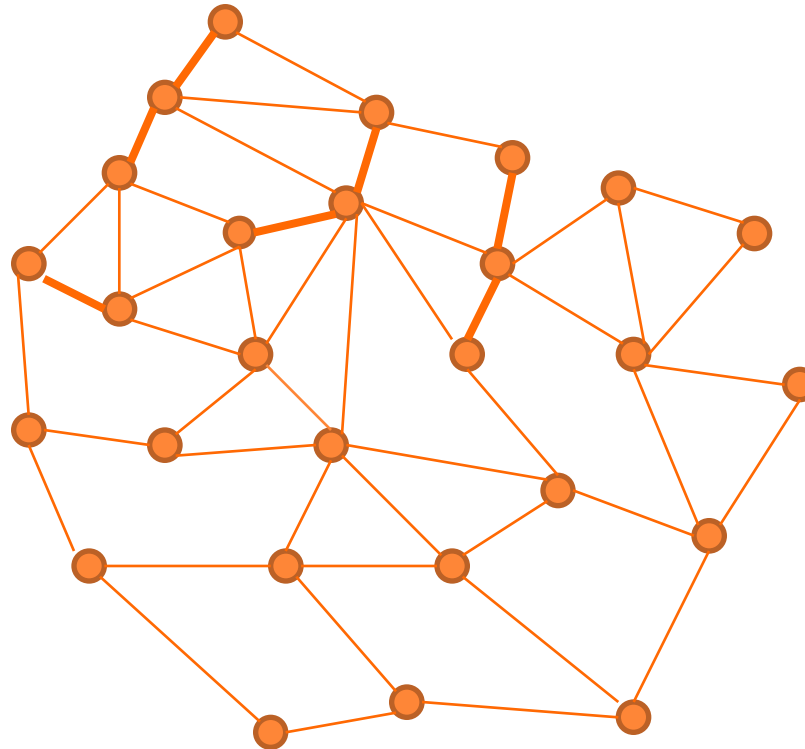
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



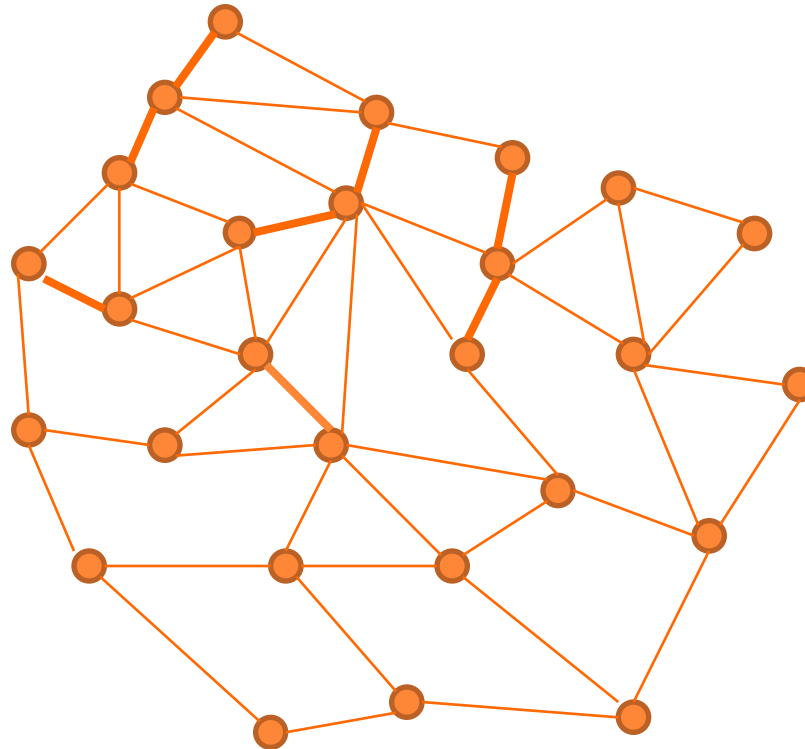
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



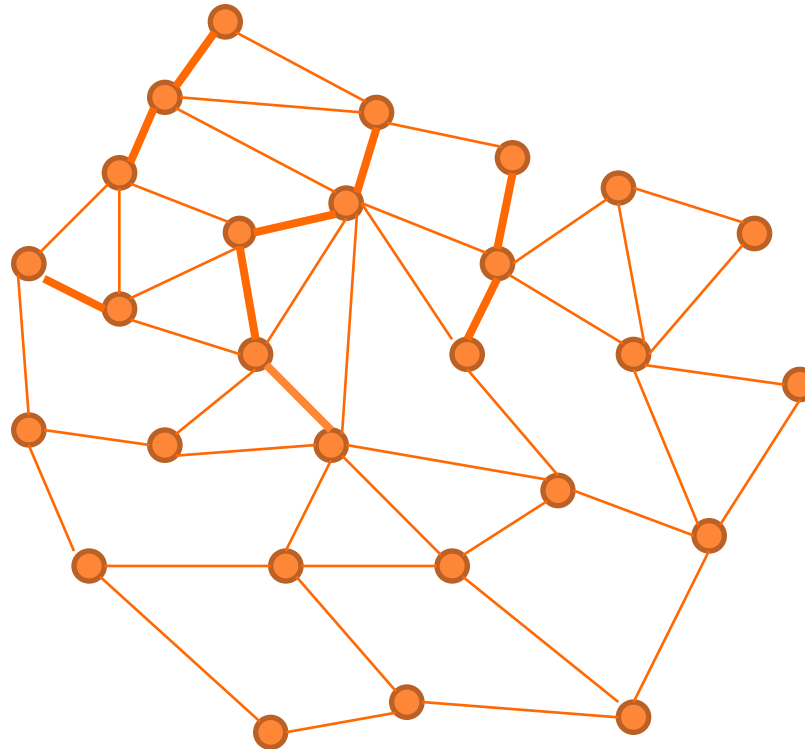
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



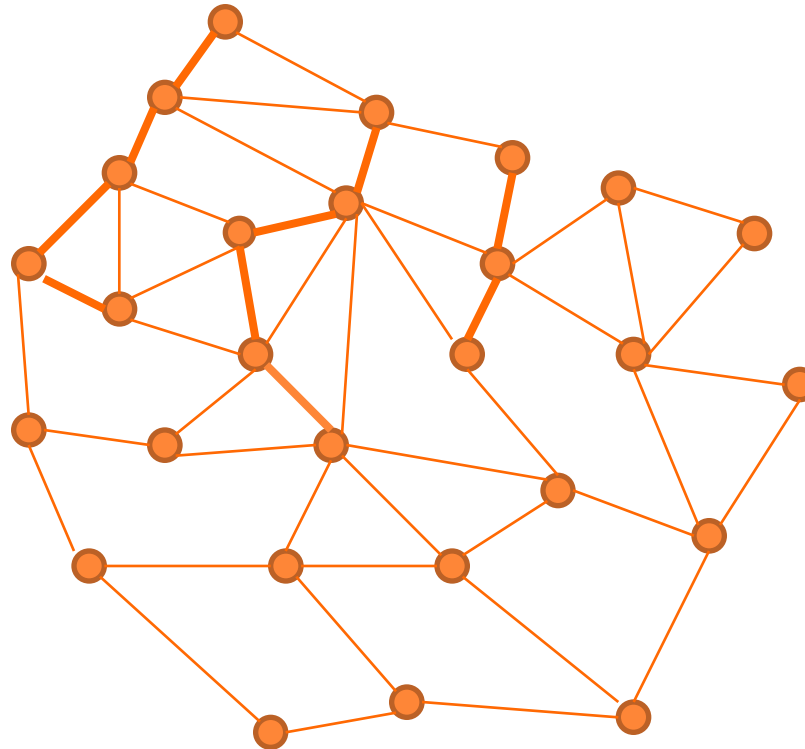
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



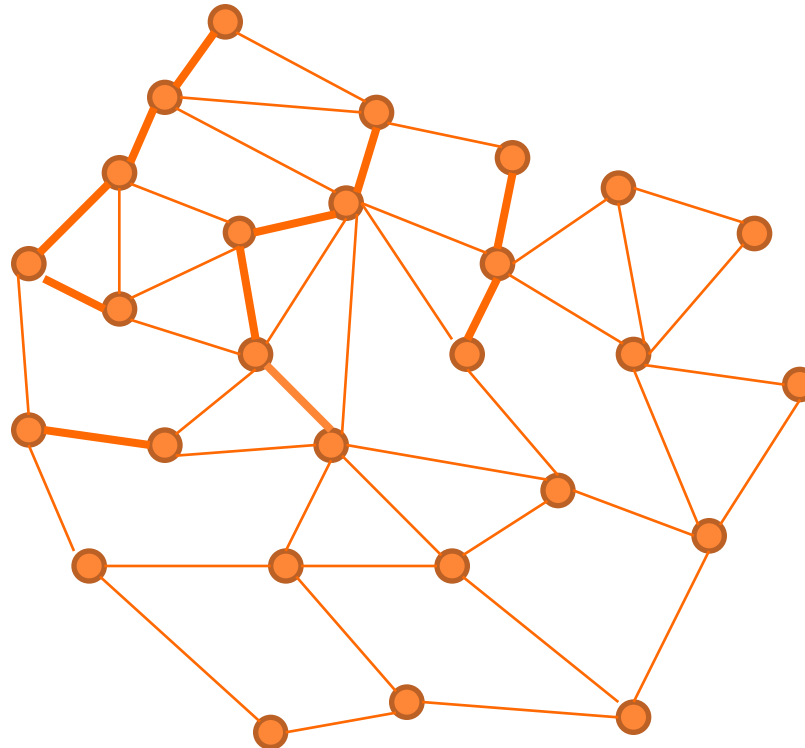
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



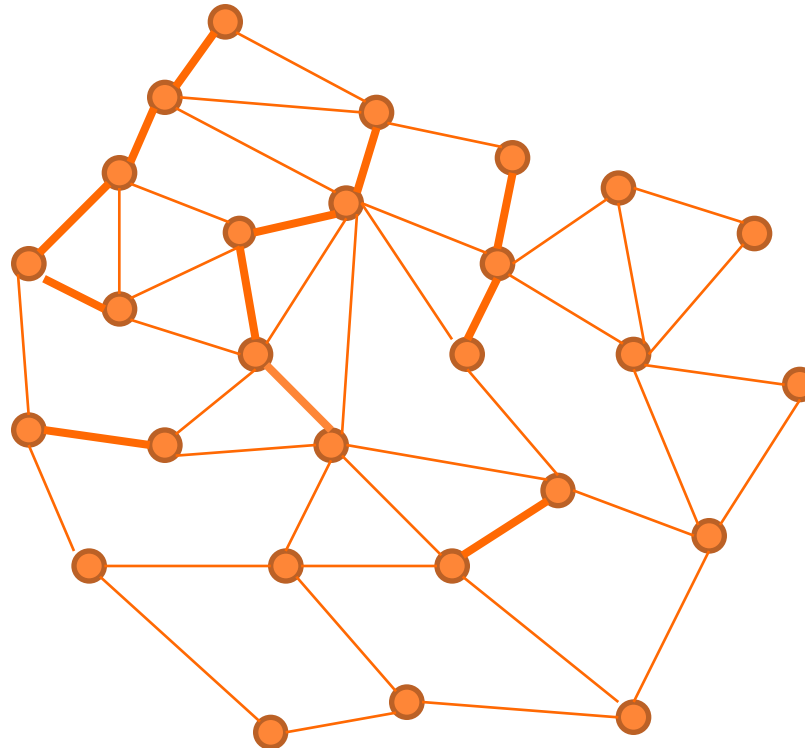
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



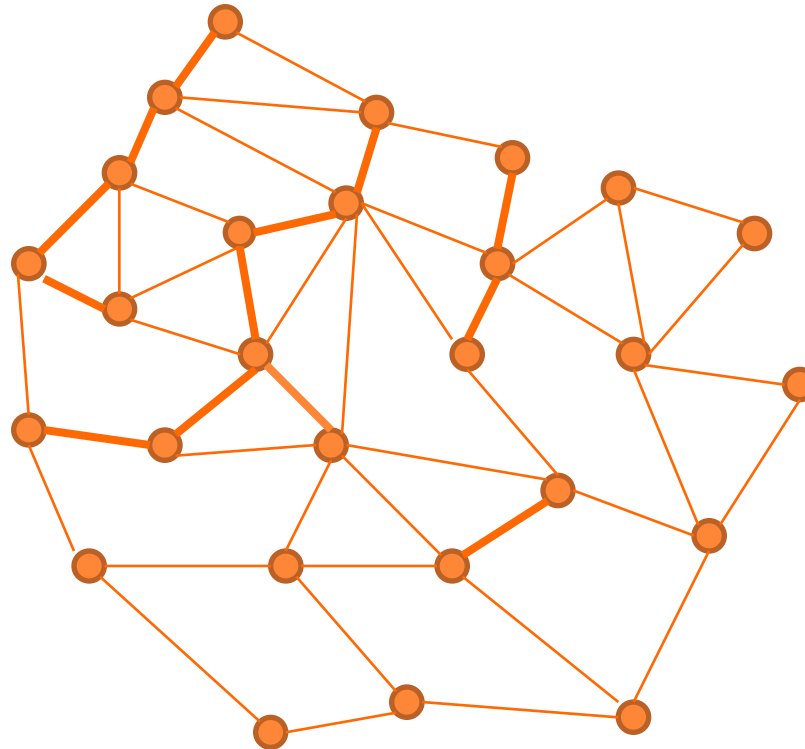
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



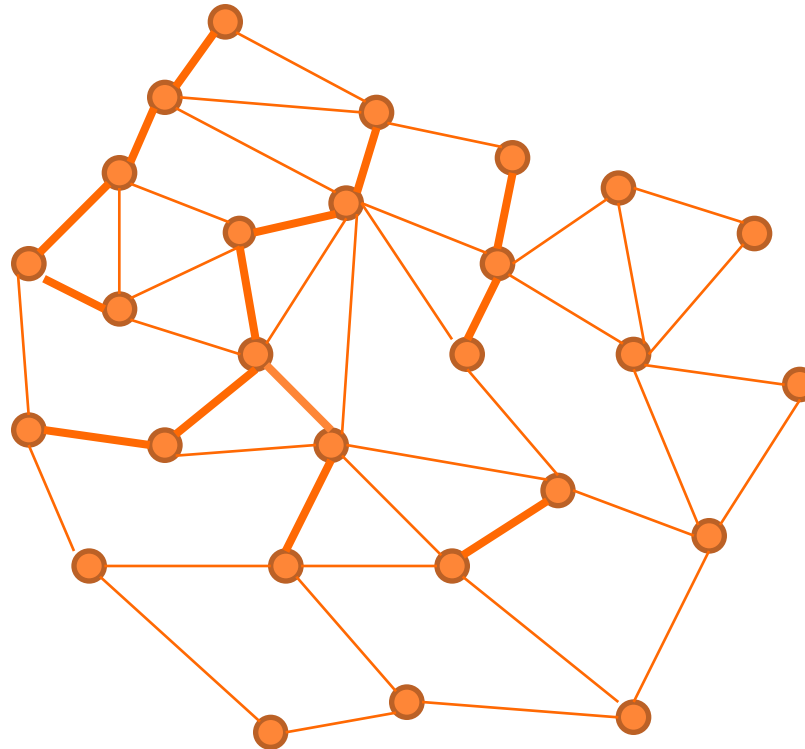
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



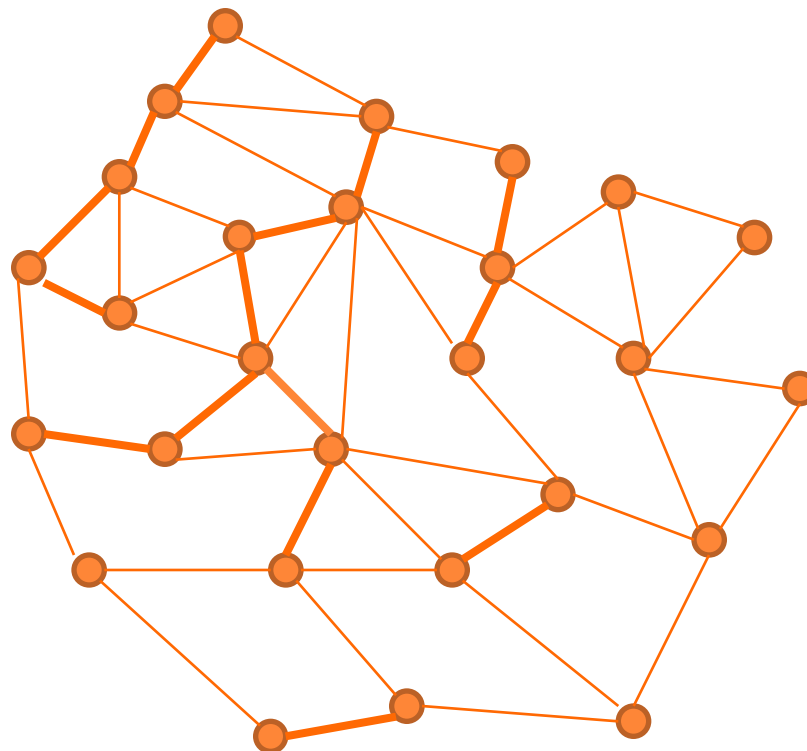
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



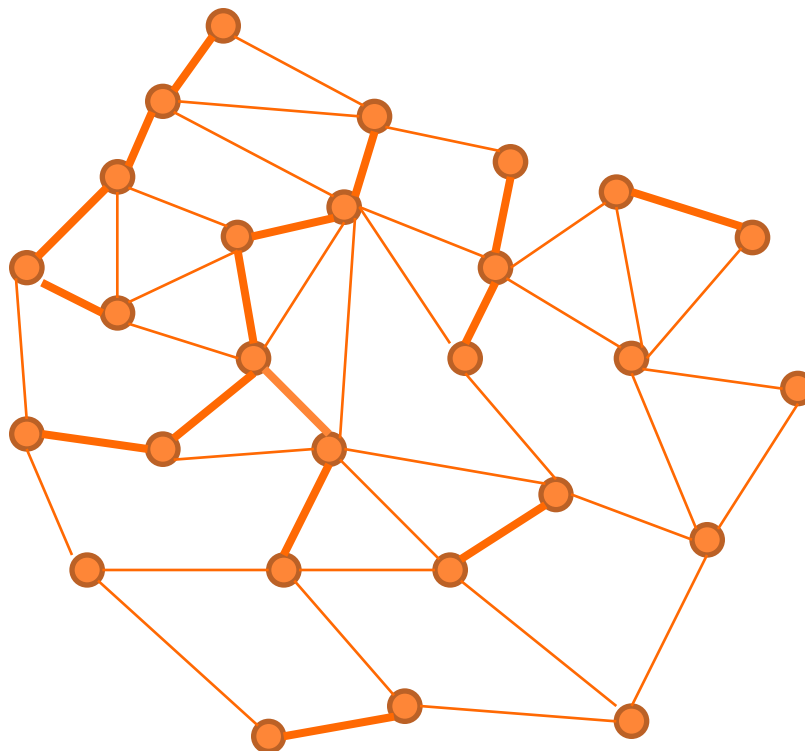
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



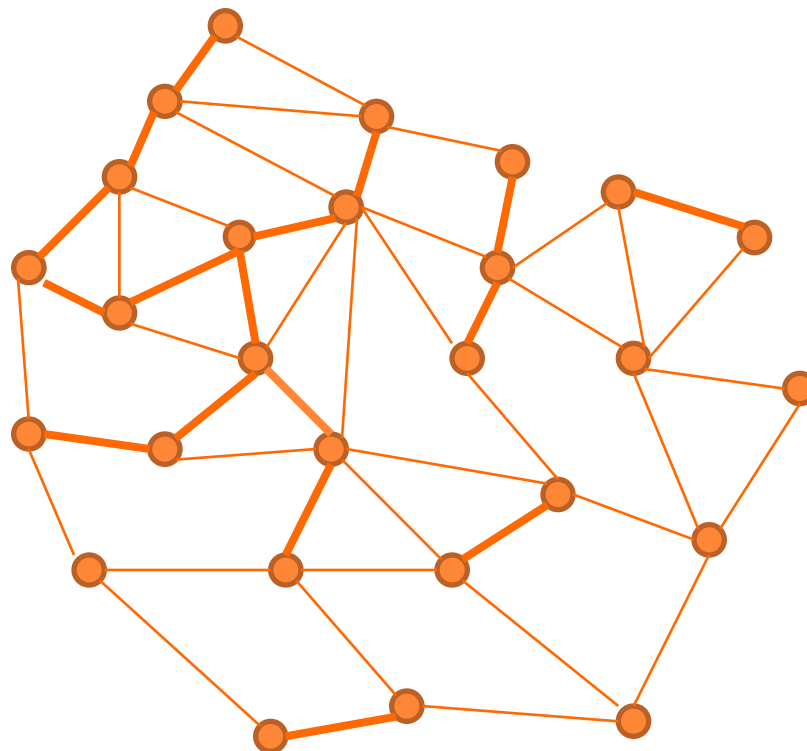
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



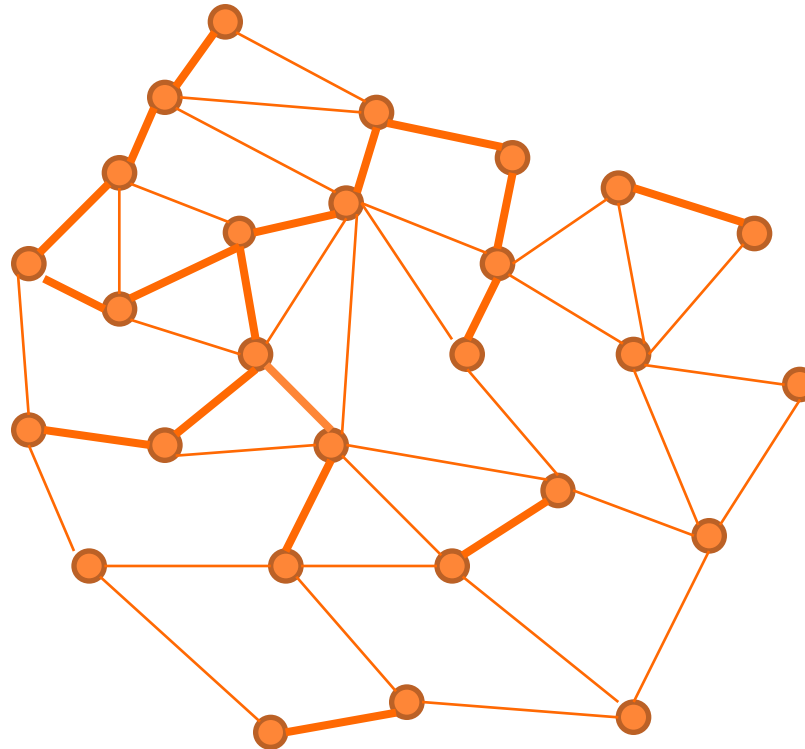
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



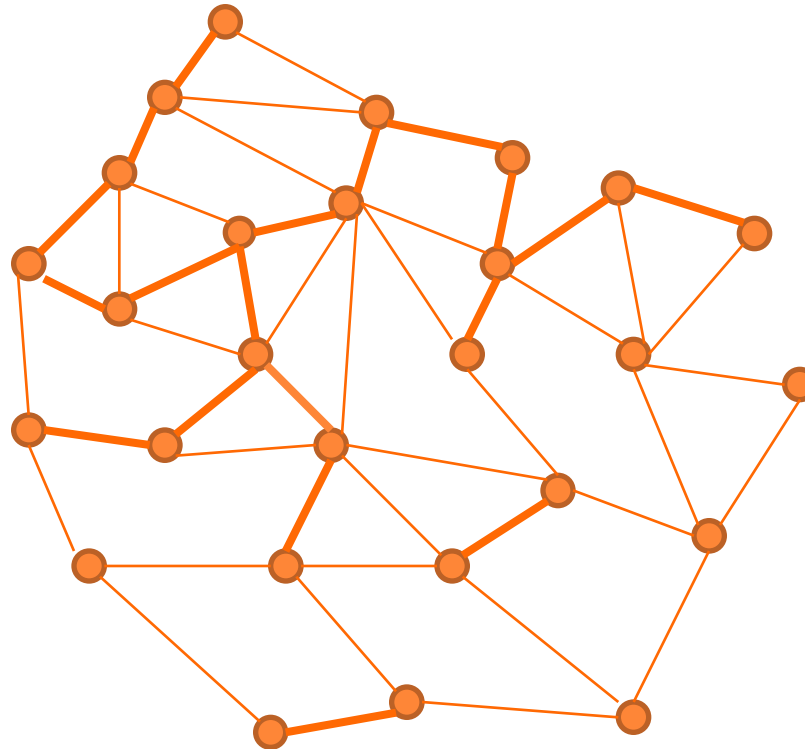
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



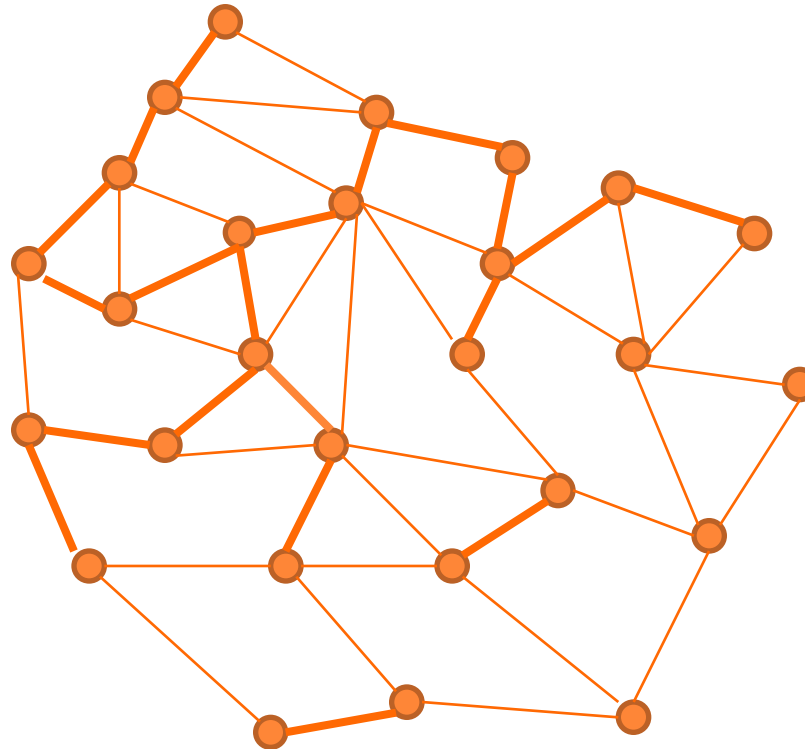
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



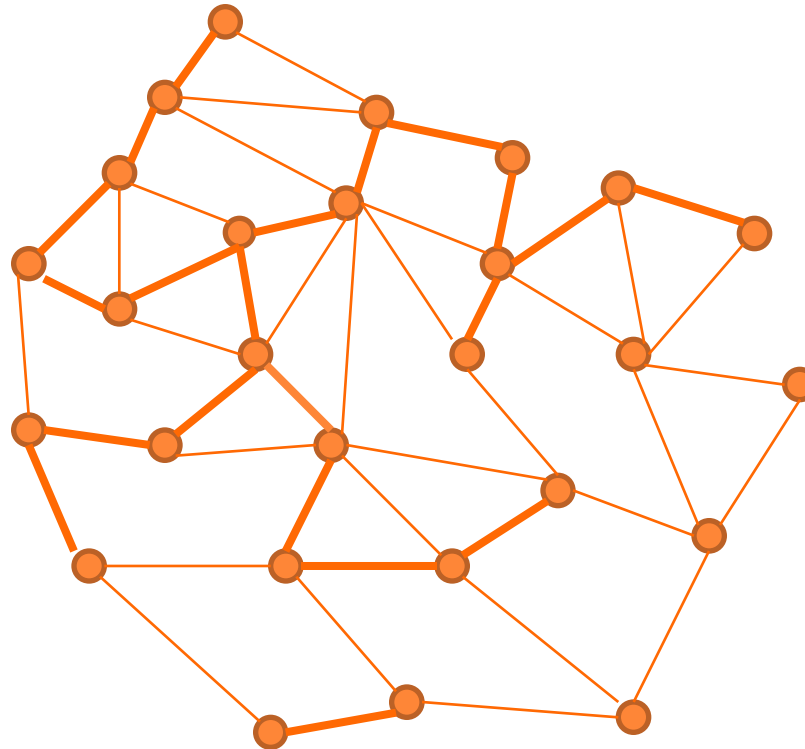
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



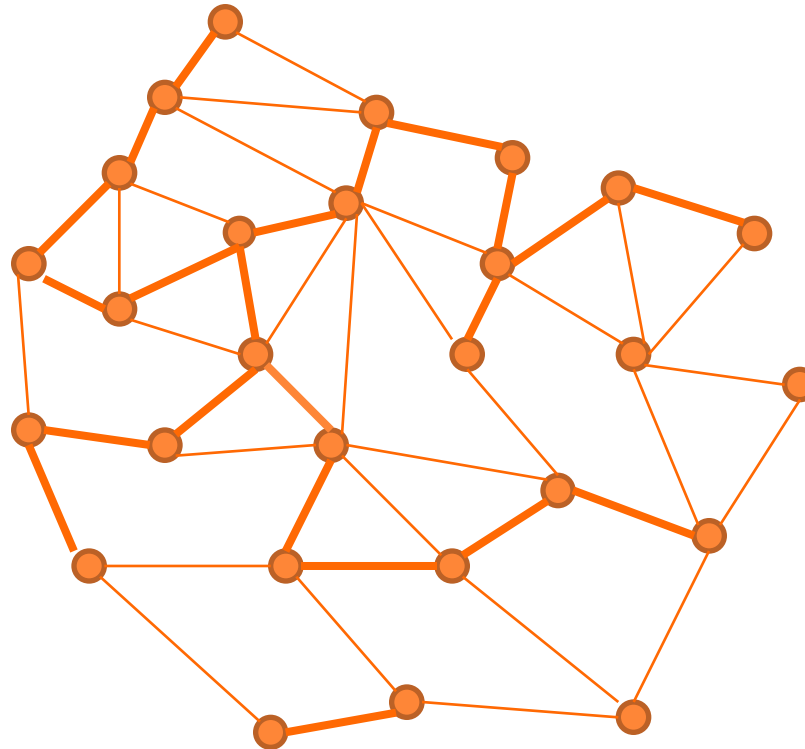
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



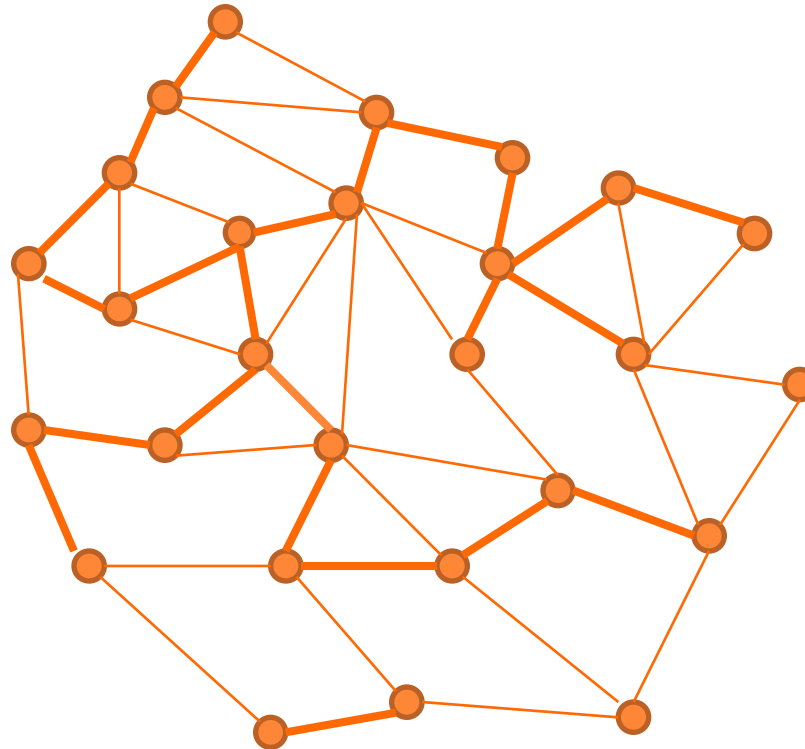
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



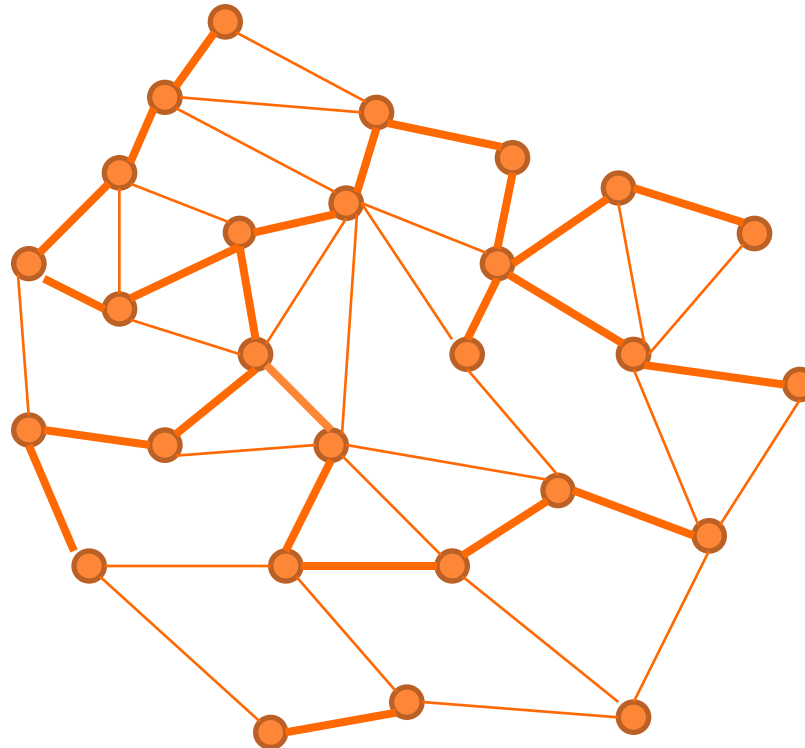
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



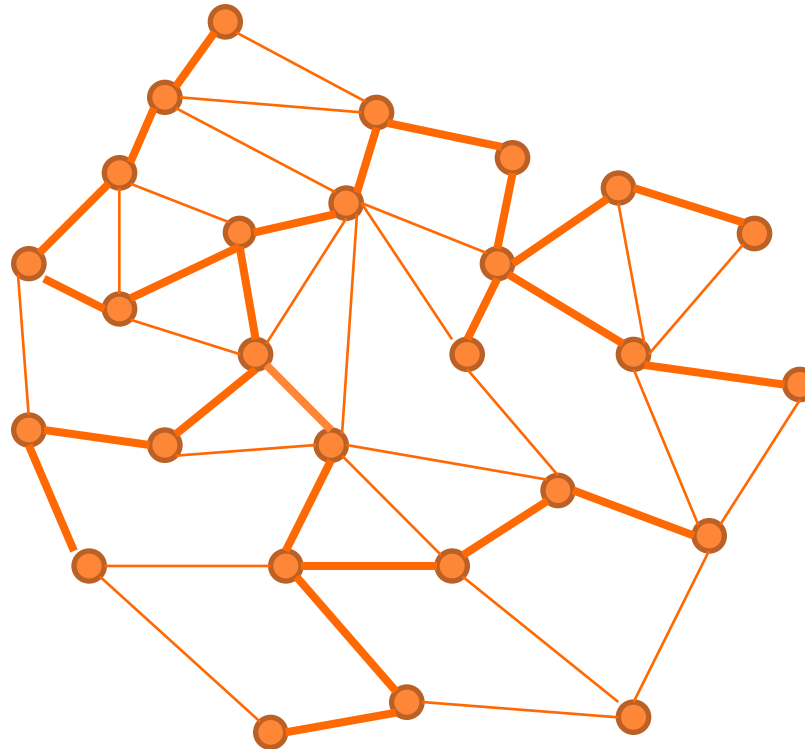
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



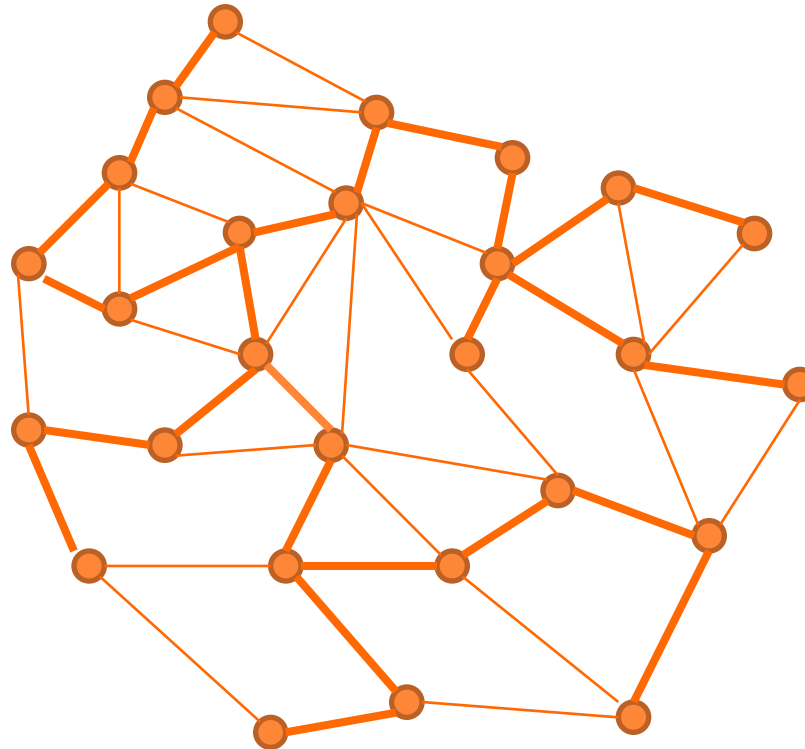
KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



KRUSKALOV ALGORITEM

- Gradi minimalni vpeti gozd, uporaben tudi za nepovezane grafe.
- NA začetku je vsako vozlišče svoje drevo. V enem koraku v gozd dodamo najkrajšo povezavo – združimo dve različni drevesi v eno (brez ciklov!)



KRUSKALOV ALGORITEM - IMPLEMENTACIJA

- na začetku je vsako vozlišče svoje drevo
- na začetku vse povezave damo v **prioritetno vrsto**
- v enem koraku najkrajšo povezavo dodamo v **minimalni vpeti gozd (MSF)**, če povezuje različni drevesi (število dreves se vsakič zmanjša za 1)
- drevo je množica vozlišč → ADT **disjunktne množice**

ADT disjoint sets omogočajo:

1. **MAKESET**: kreira množico (drevo) z enim vozliščem
2. **FIND**: iskanje, kateri množici (drevesu) pripada dano vozlišče
3. **UNION**: združitev dveh množic (dreves) v eno, ko dodamo povezavo

Kruskalov algoritem je **POŽREŠEN**, pa vseeno zagotavlja optimalno rešitev!



KRUSKALOV ALGORITEM - IMPLEMENTACIJA

ADT GRAPH spremenim \rightarrow **ADT KGRAPH:**

MAKENULL(G) naredi prazen graf G .

INSERT_VERTEX(v, G) doda vozlišče v v graf G .

INSERT_EDGE(v1, v2, G) doda povezavo $\langle v1, v2 \rangle$ v graf G .

FIRST_VERTEX(G) vrne prvo vozlišče v grafu G .

NEXT_VERTEX(v, G) vrne naslednje vozlišče danega vozlišča v po nekem vrstnem redu v grafu G .

***FIRST_EDGE(G)** vrne prvo povezavo v grafu G .

***NEXT_EDGE(e, G)** vrne naslednjo povezavo dane povezave e v grafu G po nekem vrstnem redu.

***ENDPOINTS(e, G, v1, v2)** vrne oba konca, $v1$ in $v2$, povezave $e = \langle v1, v2 \rangle$ v grafu G .

KRUSKALOV ALGORITEM - IMPLEMENTACIJA

Povezava poleg dolžin hrani še podatek o tem, ali je v MSF:

```
public class KEdge extends Edge {  
    KVertex v1, v2;  
    KEdge nextEdge;  
    boolean inForest ;  
} // class KEdge
```

← rezultat algoritma

```
public class Edge {  
    public Comparable evaluate ;  
} // class Edge
```

Vozlišče pa hrani svoj položaj v podmnožici, ki predstavlja eno MST v MSF:

```
DisjointSubset subset ;
```



KRUSKALOV ALGORITEM - IMPLEMENTACIJA

```
public void kruskal(KGraph g) {  
    KruskalVertex v1, v2 ;  
    DisjointSubset s1, s2 ; //dve disj. podmnozici – poddrevesi  
    // inicializacija disjunktih mnozic vozlisc  
    // ena mnozica je vpeto drevo:  
    DisjointSet dSet = new DisjointSetForest() ;  
  
    for (KruskalVertex t =(KruskalVertex)g.firstVertex(); t!= null;  
        t = (KruskalVertex) g.nextVertex(t))  
        t.subset = dSet.makeset(t) ;  
  
    // inicializacija prioritete vrste povezav  
    PriorityQueue q = new Heap() ; // urejena po evalve  
    KruskalEdge e ;  
    for (e = (KruskalEdge)g.firstEdge(); e!= null ;  
        e = (KruskalEdge)g.nextEdge(e)) {  
        q.insert(e) ;  
        e.inForest = false;  
    }  
}
```


KRUSKALOV ALGORITEM - IMPLEMENTACIJA

// zgradi minimalni vpeti gozd

// ce je graf povezan, zgradi minimalno vpeto drevo

while (!q.empty()) {

 e = (KruskalEdge) q.deleteMin();

 v1 = (KruskalVertex)g.endPoint1(e) ;

 v2 = (KruskalVertex)g.endPoint2(e) ;

// doloci poddrevesi obeh vozlisc

 s1 = dSet.find(v1.subset);

 s2 = dSet.find(v2.subset);

if (s1 != s2) {

// e je povezava med dvema razlicnimi vpetimi drevesi

 dSet.union(s1, s2) ;

 e.inForest = **true**;

 }

 } *// while*

} *// Kruskal*

POVZETEK

Lastnost MST zagotavlja, da **požrešni** algoritmi zadoščajo.

Primov algoritem (za **povezane grafe**): MST

1. Začnem z poljubnim vozliščem.
2. V enem koraku v MST dodam vozlišče z najkrajšo razdaljo do nekega vozlišča v MST.

V PQ so vozlišča.

Časovna zahtevnost: $O(m \log n)$

Kruskalov algoritem (za **(ne)povezane grafe**): MSF

1. Vsako vozlišče je svoje drevo.
2. V enem koraku z najkrajšo povezavo združim dve drevesi v enega.

V PQ so povezave; vozlišča so v DISJOINT SETS.

Časovna zahtevnost: $O(m \log m)$

