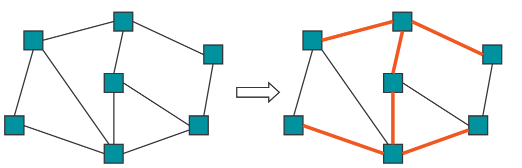
Minimum Spanning Tree

CSE 326
Data Structures
Lecture 18

Reading

- Chapter 9.5

Spanning Tree in a Graph

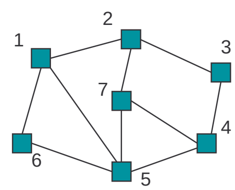


Vertex = router
Edge = link between routers

Spanning tree
- Connects all the vertices
- No cycles

Undirected Graph

- G = (V,E)
  - V is a set of vertices (or nodes)
  - E is a set of unordered pairs of vertices



V = {1,2,3,4,5,6,7}
E = {{1,2},{1,6},{1,5},{2,7},{2,3}, {3,4},{4,7},{4,5},{5,6}}

2 and 3 are adjacent
2 is incident to edge {2,3}

Spanning Tree Problem

- Input: An undirected graph G = (V,E). G is connected.
- Output: T contained in E such that
  - (V,T) is a connected graph
  - (V,T) has no cycles

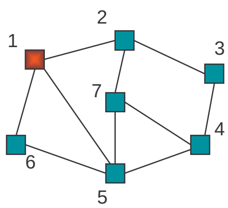
Spanning Tree Algorithm

```
ST(i: vertex)
   mark i;
   for each j adjacent to i do
      if j is unmarked then
         Add {i,j} to T;
         ST(j);
end{ST}
```

```
Main
T := empty set;
ST(1);
end{Main}
```

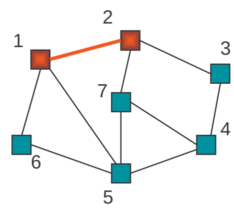
Example of Depth First Search

ST(1)



Example Step 2

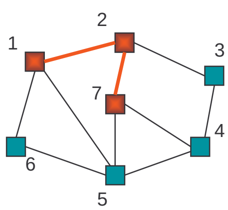
ST(1)
ST(2)



{1,2}

Example Step 3

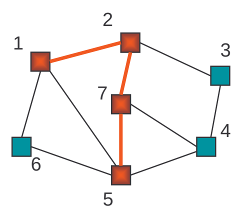
ST(1)
ST(2)
ST(7)



{1,2} {2,7}

Example Step 4

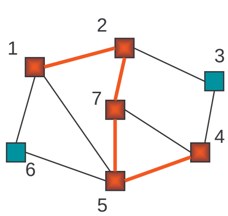
ST(1)
ST(2)
ST(7)
ST(5)



{1,2} {2,7} {7,5}

Example Step 5

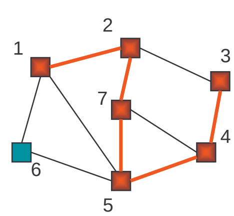
ST(1)
ST(2)
ST(7)
ST(5)
ST(4)



{1,2} {2,7} {7,5} {5,4}

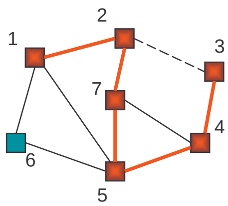
Example Step 6

ST(1)
ST(2)
ST(7)
ST(5)
ST(4)
ST(3)



{1,2} {2,7} {7,5} {5,4} {4,3}

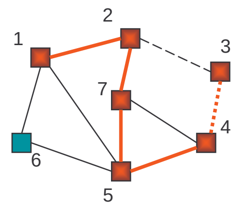
Example Step 7



ST(1)
ST(2)
ST(7)
ST(5)
ST(4)
ST(3)

{1,2} {2,7} {7,5} {5,4} {4,3}

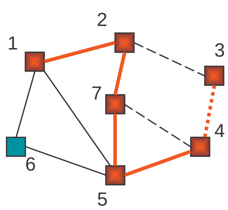
Example Step 8



ST(1)
ST(2)
ST(7)
ST(5)
ST(4)

{1,2} {2,7} {7,5} {5,4} {4,3}

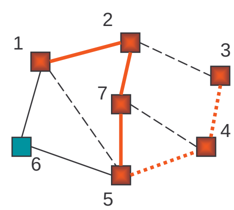
Example Step 9



ST(1)
ST(2)
ST(7)
ST(5)
ST(4)

{1,2} {2,7} {7,5} {5,4} {4,3}

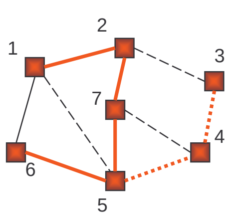
Example Step 10



ST(1)
ST(2)
ST(7)
ST(5)

{1,2} {2,7} {7,5} {5,4} {4,3}

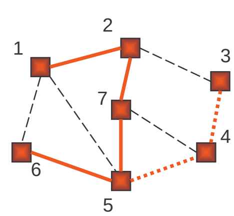
Example Step 11



ST(1)
ST(2)
ST(7)
ST(5)
ST(6)

{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Example Step 12

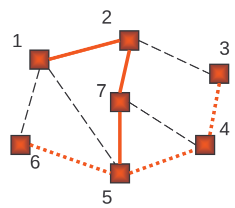


ST(1)
ST(2)
ST(7)
ST(5)
ST(6)

{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Example Step 13

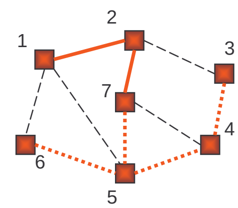
ST(1)
ST(2)
ST(7)
ST(5)



{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Example Step 14

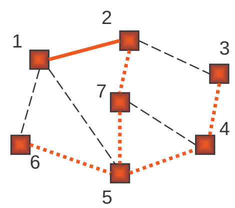
ST(1)
ST(2)
ST(7)



{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Example Step 15

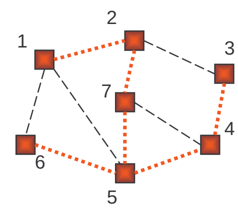
ST(1)
ST(2)



{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Example Step 16

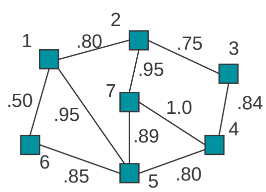
ST(1)



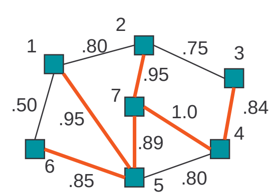
{1,2} {2,7} {7,5} {5,4} {4,3} {5,6}

Best Spanning Tree

- Each edge has the probability that it won't fail
- Find the spanning tree that is least likely to fail



Example of a Spanning Tree



Probability of success = .85 x .95 x .89 x .95 x 1.0 x .84
= .5735

Minimum Spanning Tree Problem

- Input: Undirected Graph G = (V,E) and a cost function C from E to the reals. C(e) is the cost of edge e.
- Output: A spanning tree T with minimum total cost. That is: T that minimizes

$$C(T) = \sum_{e \in T} C(e)$$

Reducing Best to Minimum

Let P(e) be the probability that an edge doesn't fail.
Define:

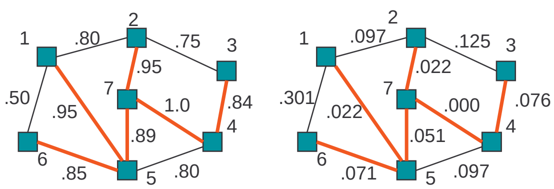
$$C(e) = -\log_{10}(P(e))$$

Minimizing $\sum_{e \in T} C(e)$

is equivalent to maximizing $\prod_{e \in T} P(e)$

because $\prod_{e \in T} P(e) = \prod_{e \in T} 10^{-C(e)} = 10^{-\sum_{e \in T} C(e)}$

Example of Reduction



Best Spanning Tree Problem   Minimum Spanning Tree Problem

Minimum Spanning Tree

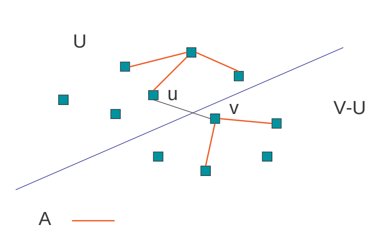
- Boruvka 1926
- Kruskal 1956
- Prim 1957 also by Jarnik 1930
- Karger, Klein, Tarjan 1995
  - Randomized linear time algorithm
  - Probably not practical, but very interesting

MST Optimality Principle

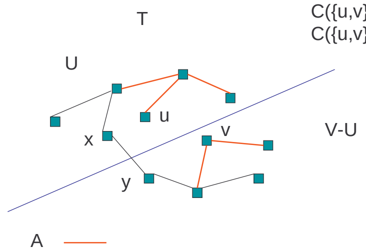
- G = (V,E) with costs C. G connected.
- Let (V,A) be a subgraph of G that is contained in a minimum spanning tree. Let U be a set such that no edge in A has one end in U and one end in V-U. Let C({u,v}) minimal and u in U and v in V-U. Let A' be A with {u,v} added. Then (V,A') is contained in a minimum spanning tree.

Proof of Optimality Principle

C({u,v}) is minimal

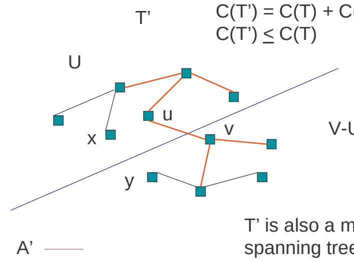


Proof of Optimality Principle



C({u,v}) is minimal
C({u,v}) ≤ C({x,y})

Proof of Optimality Principle



C(T') = C(T) + C({u,v}) - C({x,y})
C(T') ≤ C(T)

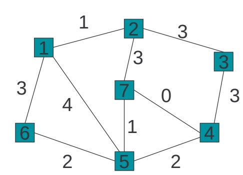
T' is also a minimum spanning tree

Kruskal's Greedy Algorithm

Sort the edges by increasing cost;
Initialize A to be empty;
For each edge e chosen in increasing order do
    if adding e does not form a cycle then
        add e to A

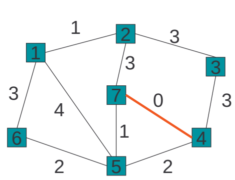
Invariant: A is always contained in some minimum spanning tree

Example of Kruskal 1



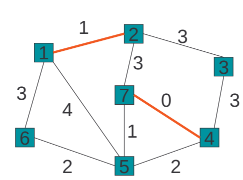
{7,4}	{2,1}	{7,5}	{5,6}	{5,4}	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
0	1	1	2	2	3	3	3	3	4

Example of Kruskal 2



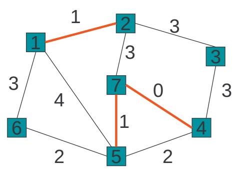
~~{7,4}~~	{2,1}	{7,5}	{5,6}	{5,4}	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	1	1	2	2	3	3	3	3	4

Example of Kruskal 2



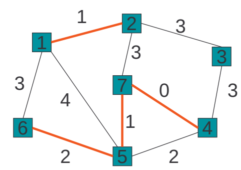
~~{7,4}~~	~~{2,1}~~	{7,5}	{5,6}	{5,4}	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	1	2	2	3	3	3	3	4

Example of Kruskal 3



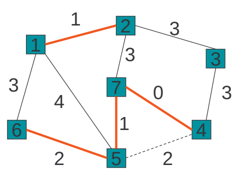
~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	{5,6}	{5,4}	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	2	2	3	3	3	3	4

Example of Kruskal 4



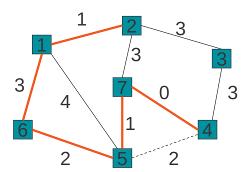
~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	~~{5,6}~~	{5,4}	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	~~2~~	2	3	3	3	3	4

Example of Kruskal 5



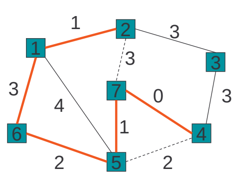
~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	~~{5,6}~~	~~{5,4}~~	{1,6}	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	~~2~~	~~2~~	3	3	3	3	4

Example of Kruskal 6



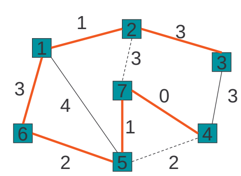
~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	~~{5,6}~~	~~{5,4}~~	~~{1,6}~~	{2,7}	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	~~2~~	~~2~~	~~3~~	3	3	3	4

Example of Kruskal 7



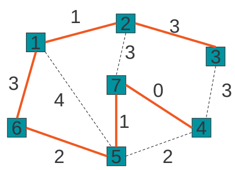
~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	~~{5,6}~~	~~{5,4}~~	~~{1,6}~~	~~{2,7}~~	{2,3}	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	~~2~~	~~2~~	~~3~~	~~3~~	3	3	4

Example of Kruskal 7



~~{7,4}~~	~~{2,1}~~	~~{7,5}~~	~~{5,6}~~	~~{5,4}~~	~~{1,6}~~	~~{2,7}~~	~~{2,3}~~	{3,4}	{1,5}
~~0~~	~~1~~	~~1~~	~~2~~	~~2~~	~~3~~	~~3~~	~~3~~	3	4

Example of Kruskal 8,9



~~{7,4}~~ ~~{2,1}~~ ~~{7,5}~~ ~~{5,6}~~ ~~{5,4}~~ ~~{1,6}~~ ~~{2,7}~~ ~~{2,3}~~ ~~{3,4}~~ ~~{1,5}~~
~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ ~~3~~ ~~3~~ ~~3~~ ~~3~~ ~~4~~

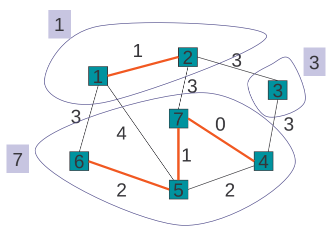
Data Structures for Kruskal

- Sorted edge list

  {7,4} {2,1} {7,5} {5,6} {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}
  0 1 1 2 2 3 3 3 3 4

- Disjoint Union / Find
  – Union(a,b) - union the disjoint sets named by a and b
  – Find(a) returns the name of the set containing a

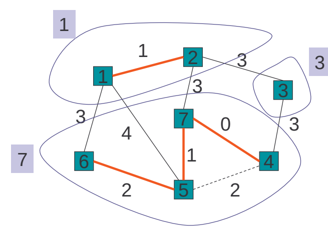
Example of DU/F 1



Find(5) = 7
Find(4) = 7

~~{7,4}~~ ~~{2,1}~~ ~~{7,5}~~ ~~{5,6}~~ {5,4} {1,6} {2,7} {2,3} {3,4} {1,5}
~~0~~ ~~1~~ ~~1~~ ~~2~~ 2 3 3 3 3 4

Example of DU/F 2

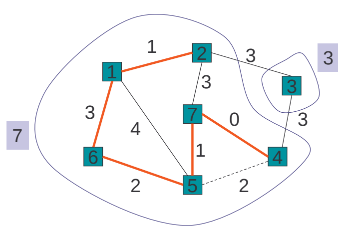


Find(1) = 1
Find(6) = 7

~~{7,4}~~ ~~{2,1}~~ ~~{7,5}~~ ~~{5,6}~~ ~~{5,4}~~ {1,6} {2,7} {2,3} {3,4} {1,5}
~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ 3 3 3 3 4

Example of DU/F 3

Union(1,7)



~~{7,4}~~ ~~{2,1}~~ ~~{7,5}~~ ~~{5,6}~~ ~~{5,4}~~ ~~{1,6}~~ {2,7} {2,3} {3,4} {1,5}
~~0~~ ~~1~~ ~~1~~ ~~2~~ ~~2~~ ~~3~~ 3 3 3 4

Kruskal’s Algorithm with DU / F

```
Sort the edges by increasing cost;
Initialize A to be empty;
for each edge {i,j} chosen in increasing order do
   u := Find(i);
   v := Find(j);
   if not(u = v) then
      add {i,j} to A;
      Union(u,v);
```