Minimum spanning tree

Algorithms and variant

beOI Training

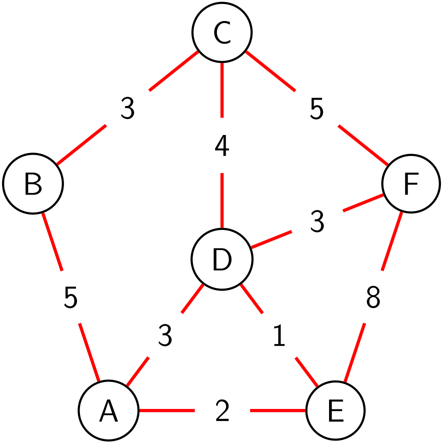
OLYMPIADE BELGE D'INFORMATIQUE
BELGISCHE INFORMATICA-OLYMPIADE

February 22, 2025

Table of Contents

Motivating problem

Several neighbourhoods:



Connect all of them in the least total weight

Spanning tree

A tree connecting all nodes in a graph.
Tree: no cycles.
A graph can have multiple spanning trees.

Spanning tree

A tree connecting all nodes in a graph.
Tree: no cycles.
A graph can have multiple spanning trees.

If a graph has v nodes, how many edges do all its spanning trees have?

Spanning tree

A tree connecting all nodes in a graph.
Tree: no cycles.
A graph can have multiple spanning trees.

If a graph has v nodes, how many edges do all its spanning trees have?
Answer: $v - 1$

Minimum spanning tree

A spanning tree with the minimum total weight.

Minimum spanning tree

A spanning tree with the minimum total weight.

Minimum spanning tree

A spanning tree with the minimum total weight.

There can be several minimum spanning trees (mst).
Total weight is unique (minimum).

Table of Contents

The algorithm

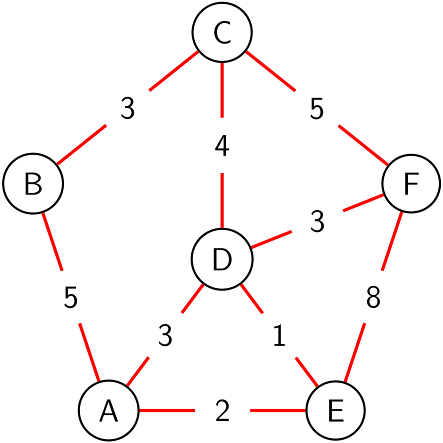
1. Sort al the edges in non-decreasing order of their weight
2. Pick the smallest edge
   Does adding it form a cycle in the ST?
   - No: include it
   - Yes: discard it
3. Repeat until ...

The algorithm

1. Sort al the edges in non-decreasing order of their weight
2. Pick the smallest edge
   Does adding it form a cycle in the ST?
   - No: include it
   - Yes: discard it
3. Repeat until ... the tree contains $v - 1$ edges

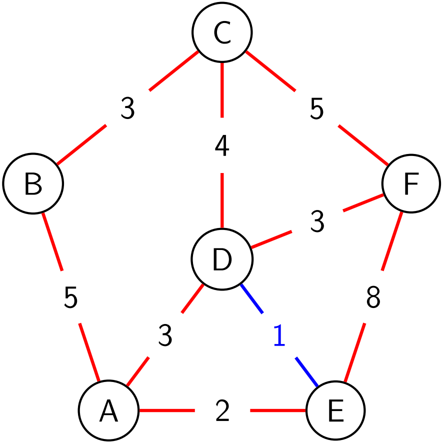
Example

Network:



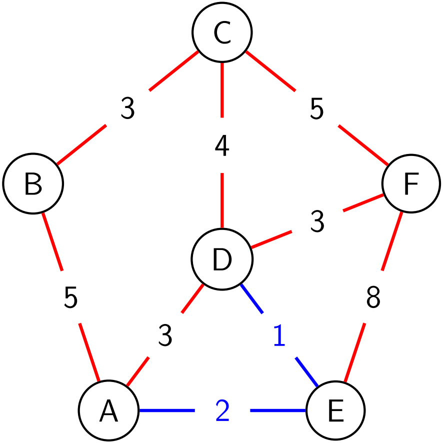
Example

Network:



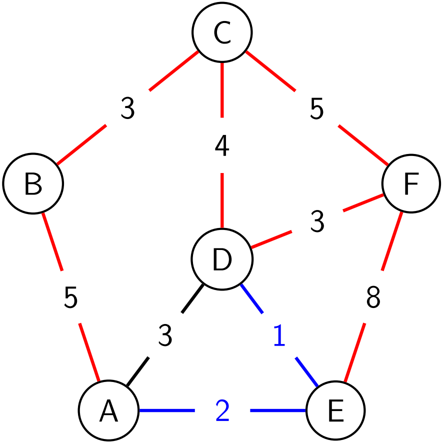
Example

Network:



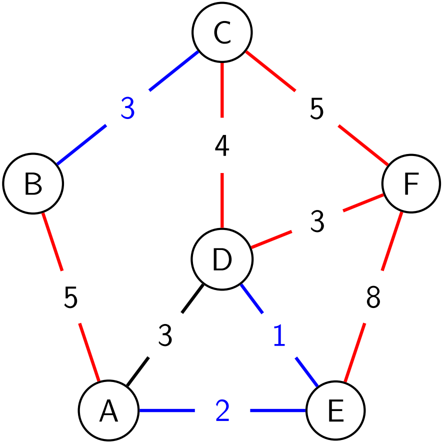
Example

Network:



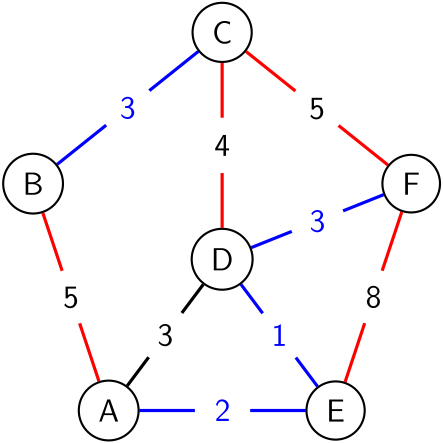
Example

Network:



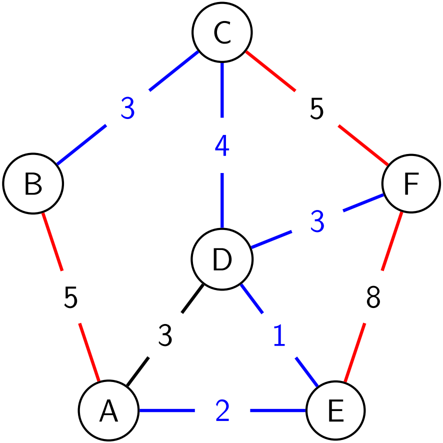
Example

Network:



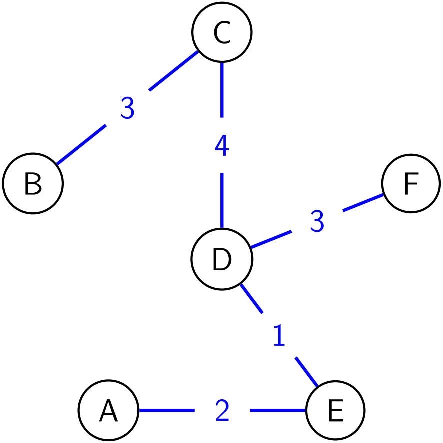
Example

Network:



Example

MST:



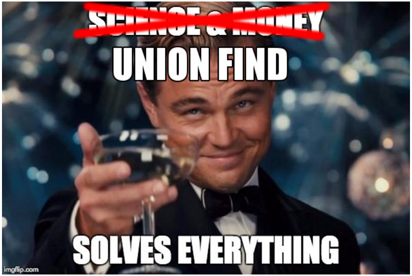
Cost: 13

Check for cycles

Any ideas?

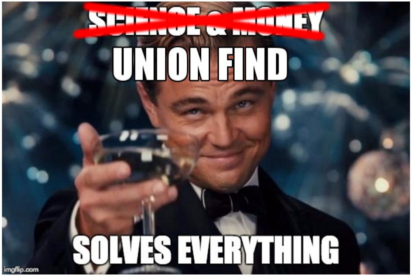
Check for cycles

Any ideas?



Check for cycles

Any ideas?



When you encounter an edge

- check for cycle by finding the parents of source/target vertex
- union the two nodes

Code

```
void Kruskal() {
  UF uf(n);
    sort(G.begin(), G.end());
    for(int i=0;i<m;i++) {
      int x = G[i].second.first;
      int y = G[i].second.second;
        if(!uf.sameSet(x, y)) {
            M.push_back(G[i]);
            total += G[i].first;
            uf.merge(x, y);
        }
    }
}
```

src/kruskal.cpp

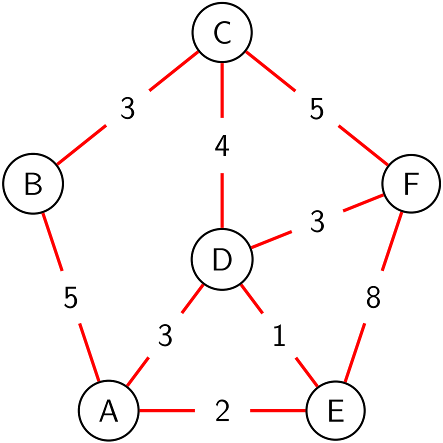
Table of Contents

The algorithm

1. Associate a value for each node
   - 0 for the initial node (arbitrary)
   - ∞ for the rest
2. Take the node with the smallest value that hasn't been added yet
   2.1 Set the value of all adjacent nodes to the minimum of the current value and the edge.
3. Repeat **2** until all nodes have been added

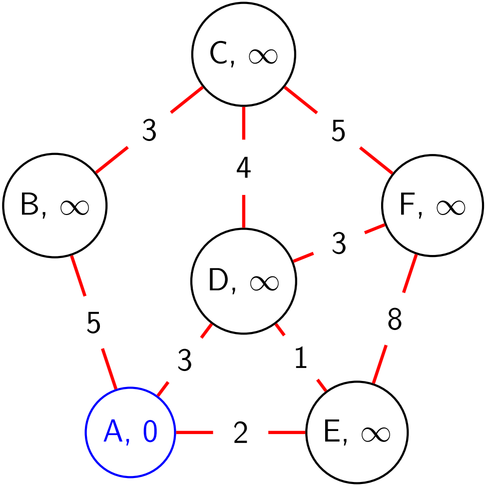
Example

Network:



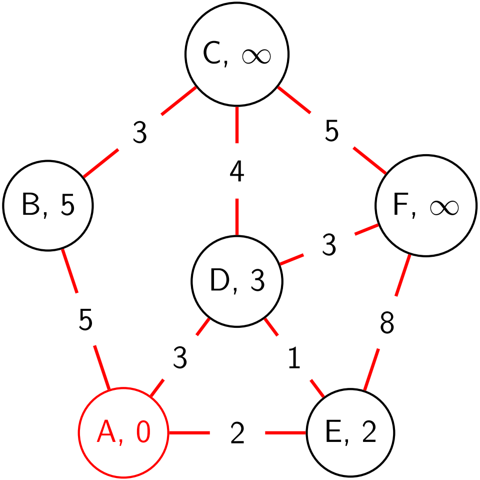
Example

Network with values:



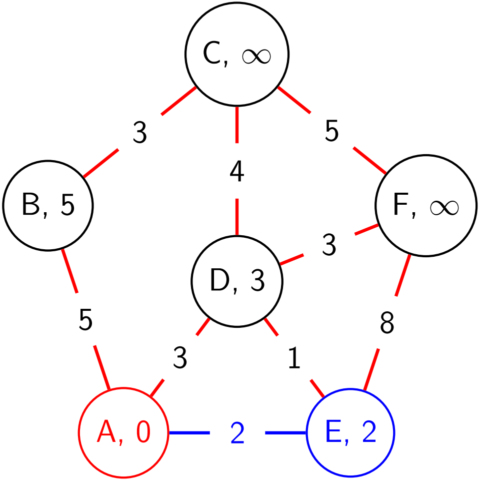
Example

Network with values:



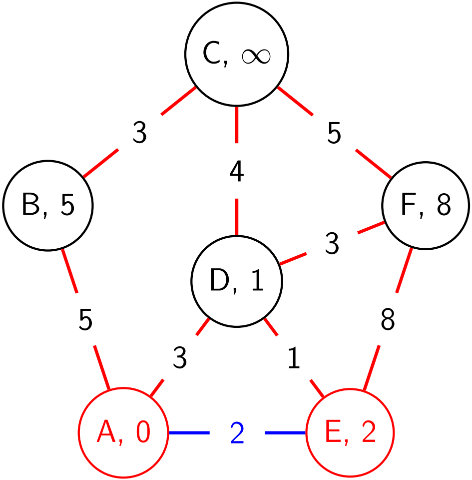
Example

Network with values:



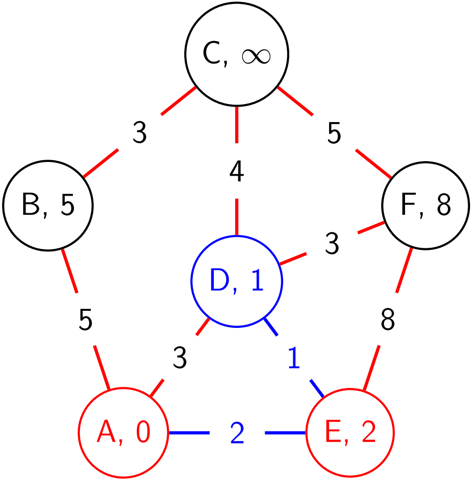
Example

Network with values:



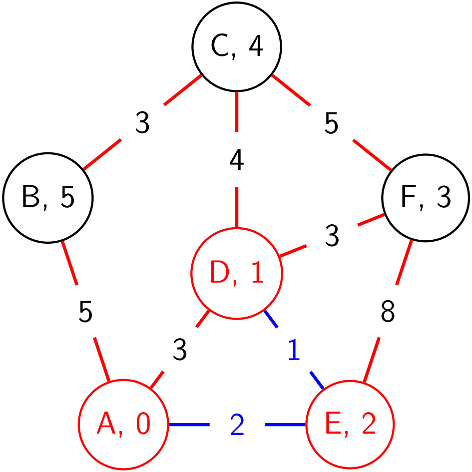
Example

Network with values:



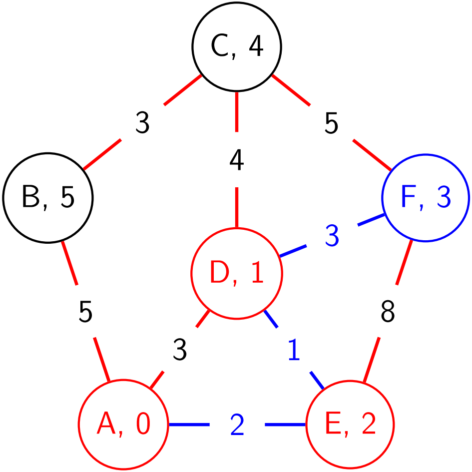
Example

Network with values:



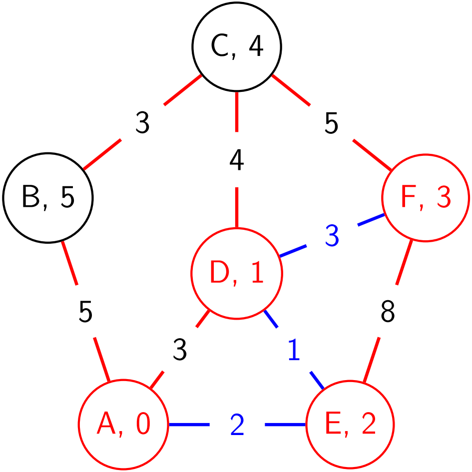
Example

Network with values:



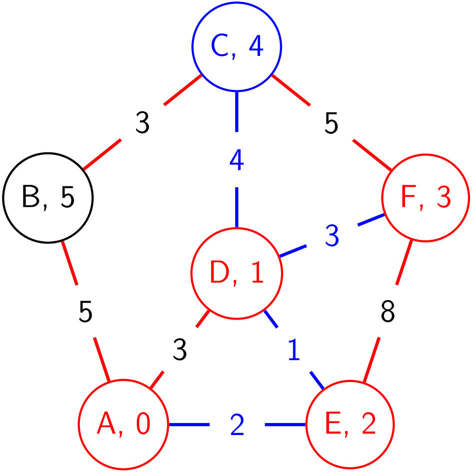
Example

Network with values:



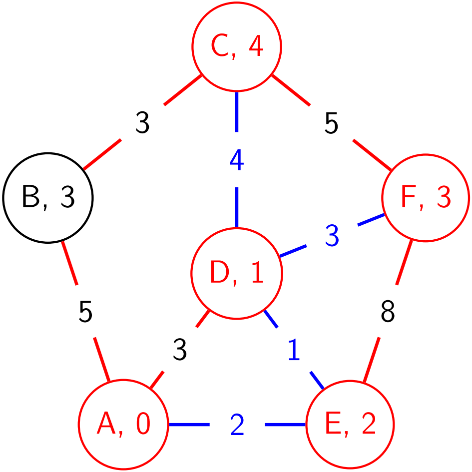
Example

Network with values:



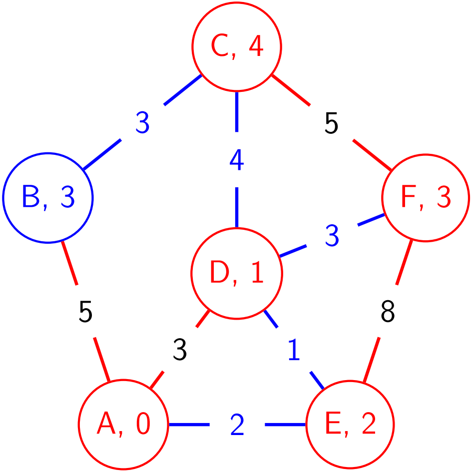
Example

Network with values:



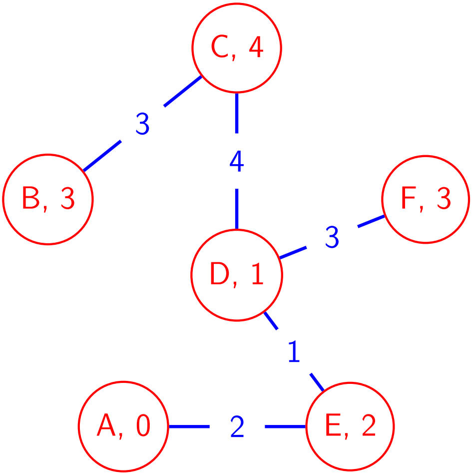
Example

Network with values:



Example

MST:



Cost: 13

Find minimum

Possibilities:

- Linear search every time: $O(v)$ per node $\rightarrow O(v^2)$
- Use a heap (priority queue): $O(vlog(v))$

Find minimum

Possibilities:

- Linear search every time: $O(v)$ per node $\rightarrow O(v^2)$
- Use a heap (priority queue): $O(v log(v))$

This algorithm reminds me of someone?

Find minimum

Possibilities:

- Linear search every time: $O(v)$ per node $\rightarrow O(v^2)$
- Use a heap (priority queue): $O(v log(v))$

This algorithm reminds me of someone?

This handsome fellow:

Find minimum

Possibilities:

- Linear search every time: $O(v)$ per node $\rightarrow O(v^2)$
- Use a heap (priority queue): $O(vlog(v))$

This algorithm reminds me of someone?
This handsome fellow:

Hint: it's Dijkstra

Code

Inside main:

```cpp
// inside int main() --- assume the graph is stored in AdjList, pq is empty
taken.assign(V, 0);
process(0);
mst_cost = 0;
while (!pq.empty()) {
  ii front = pq.top(); pq.pop();
  u = -front.second, w = -front.first;
  if (!taken[u])
    mst_cost += w, process(u);
}
```

src/prim.cpp

Code

Process function:

```
void process(int vtx) {
  taken[vtx] = 1;
  for (int j = 0; j < AdjList[vtx].size(); j++) {
    ii v = AdjList[vtx][j];
    if (!taken[v.first])
      pq.push(ii(-v.second, -v.first));
  }
}
```

src/prim.cpp

Table of Contents

Maximum spanning tree

Spanning tree with maximum total weight.

Any ideas?

Maximum spanning tree

Spanning tree with maximum total weight.

Any ideas?
Solution:

- Compute the minimum spanning tree with opposite weights
- Use kruskal but sort the other way around

'Minimum' spanning subgraph

MST where some given edges have to be included in the result.

Any ideas?

'Minimum' spanning subgraph

MST where some given edges have to be included in the result.

Any ideas?
Solution: First add the necessary edges, then just continue running kruskal on the remaining edges until it's spanning

Minimum 'spanning forest'

A spanning forest (multiple trees) of K connected components, with the least total weight.

Any ideas?

Minimum 'spanning forest'

A spanning forest (multiple trees) of K connected components, with the least total weight.

Any ideas?
Solution: Run Kruskal until you have the required number of connected components.

Second best spanning tree

Literally what the title says.

Any ideas?

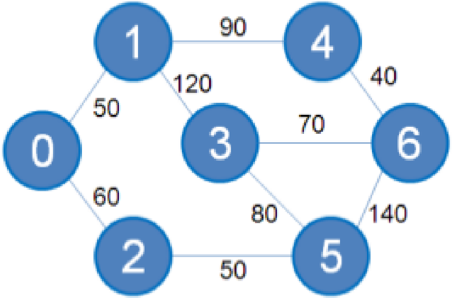
Second best spanning tree

Literally what the title says.

Any ideas?
Solution: Run Kruskal once to find the MST. For each edge in the MST, compute the MST without using this edge. Find the best of these.

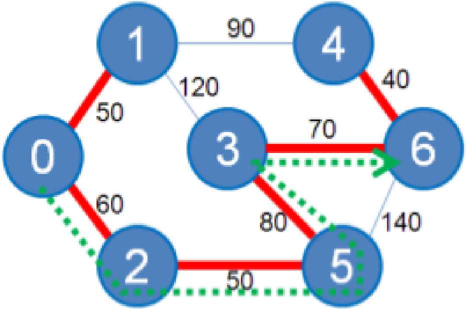
Minimax

Finding a path between two nodes that minimizes the maximum cost (= minimax) along the path.
Example: find the minimax for 1 and 4



Minimax

Finding a path between two nodes that minimizes the maximum cost (= minimax) along the path.
Example: find the minimax for 1 and 4



The minimax in this case is 80.

Minimax

Finding a path between two nodes that minimizes the maximum cost (= minimax) along the path.

How would you solve this?

Minimax

Finding a path between two nodes that minimizes the maximum cost (= minimax) along the path.

How would you solve this?
Solution: Compute the Minimum Spanning Tree and traverse it from source to target.

Minimax

Finding a path between two nodes that minimizes the maximum cost (= minimax) along the path.

How would you solve this?
Solution: Compute the Minimum Spanning Tree and traverse it from source to target.

Other (shorter) solution: use an adapted version of Floyd-Warshal, where instead of adding, you take the max.

Maximin

Finding a path between two nodes that *maximizes* the *minimum* cost ($=$ maximin) along the path.

Maximin

Finding a path between two nodes that *maximizes* the *minimum* cost (= maximin) along the path.

Analogous: Compute the *Maximum* Spanning Tree and traverse it from source to target.