Minimum Spanning Tree

Floris Kint

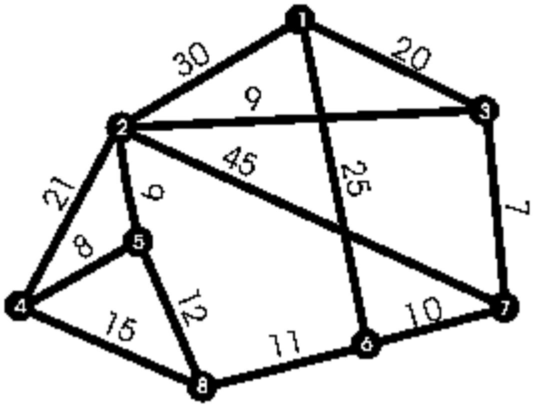
Introduction

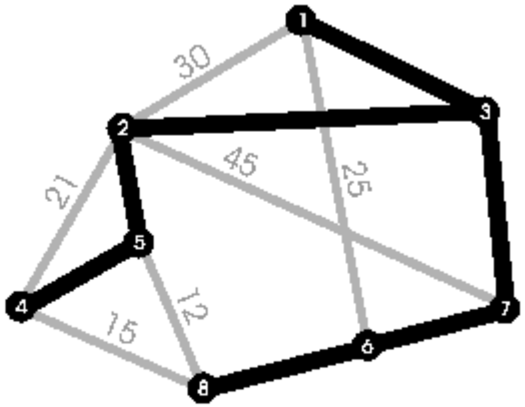
Spanning tree: contains all vertices of G

Minimal spanning tree: lowest total weight

Example problem

Cheapest subset of roads so every city can be reached from every other city.





Prim’s algorithm

Start with tree containing 1 node

Iteratively add closest node not in tree

Priority queue

$O(E \log(V))$

```
set<pair<int, int> > prim(int start_node){
  priority_queue<node> pq;
  pq.push(node(start_node, 0, start_node));
  set<int> tree;
  set<pair<int, int> > edges;
  while(!pq.empty()){
    node t = pq.top();
    pq.pop();
    if(tree.find(t.index) != tree.end())
      continue;
    tree.insert(t.index);
    if(t.index != t.edge_start)
      edges.insert(make_pair(t.index, t.edge_start));
    for(map<int, int>::iterator it = neighbours[t.index].
        begin(); it != neighbours[t.index].end(); ++it){
      node n(it->first, it->second, t.index);
      pq.push(n);
    }
  }
  return edges;
}
```

Kruskal’s algorithm

Start: every node separate tree

Iteratively add shortest edge if it connects two different trees

Union-Find

O(E log(V))

```
vector<edge> kruskal(){
  for(int i = 0; i < N; ++i){
    rank[i] = 1;
    parent[i] = i;
  }
  sort(edges.begin(), edges.end());
  vector<edge> mst_edges;
  for(vector<edge>::iterator it = edges.begin(); it !=
      edges.end(); ++it){
    if(is_same_tree(it->from, it->to)){
      continue;
    }
    merge(it->from, it->to);
    mst_edges.push_back(*it);
  }
  return mst_edges;
}
```

Union-Find

find_parent()

is_same_tree()

merge()

Exercises

- http://uva.onlinejudge.org/external/5/544.html
- http://uva.onlinejudge.org/external/117/p11710.pdf (Try to use both algorithms)
- http://uva.onlinejudge.org/external/101/p10147.pdf