Data structures and algorithms

Tutorial 8 - Part 2

Amr Keleg

Faculty of Engineering, Ain Shams University

April 17, 2020

Contact: amr_mohamed@live.com

Outline

Outline

- Priority Queue is an extension of queue with following properties.
- Every item has a priority associated with it.
- An element with high priority is dequeued before an element with low priority.
- `http://www.cplusplus.com/reference/queue/priority_queue/`
- We will study one of the ways to implement the priority queue later in this course (Heaps).

```cpp
#include <queue>

int main(){
  // Maximum value first
  priority_queue<int> q;
  q.push(3);
  q.push(10);
  q.push(-3);

  while(!q.empty()){
    cout<<q.top()<<endl;
    q.pop();
  }
}
```

The output is:

10

3

-3

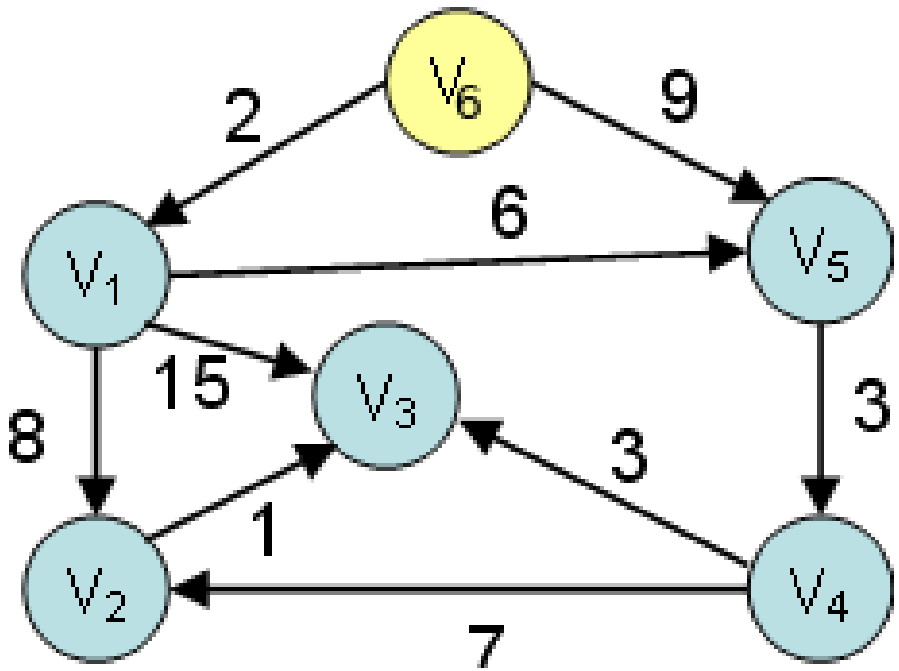
```cpp
#include <queue>

int main(){
  // How to get the maximum for a pair?
  priority_queue<pair<int, int> > q;
  q.push({3, 10});
  q.push({10, 1});
  q.push({-3, 100});
  q.push({10, 10});
  q.push({3, 1});
  while(!q.empty()){
    pair<int, int> p = q.top();
    q.pop();
    cout<<p.first<<" "<< p.second<<endl;
  }
}
```

The output is:
10 10
10 1
3 10
3 1
-3 100

```cpp
#include <queue>

int main(){
  // Minimum value on top
  priority_queue<pair<int, int>,
                 vector<pair<int, int> >,
                 greater<pair<int, int> > > q;

  q.push({3, 10});
  q.push({10, 1});
  q.push({-3, 100});
  q.push({10, 10});
  q.push({3, 1});
  while(!q.empty()){
    pair<int, int> p = q.top();
    q.pop();
    cout<<p.first<<" "<< p.second<<endl;
  }
}
```

The output is:
-3 100
3 1
3 10
10 1
10 10

Outline


V6
V1
V5
V3
V2
V4
2
9
6
15
8
3
3
1
7

```
vector<int> shortest_distance(int src,
  vector<vector<pair<int,int> > > adj_list){

  // Create queue
  priority_queue<pair<int, pair<int, int> >,
            vector<pair<int, pair<int, int>> >,
            greater<pair<int, pair<int, int> > >nodes_q;

  // Create d and p arrays
  int n = adj_list.length();
  vector<int> d(n, INF);
  vector<int> p(n, -1);

  nodes_q.push({0, {src, src}});

  ...
}
```

```
while(!nodes_q.empty()){
  pair<int, pair<int, int>> p = nodes_q.top();
  nodes_q.pop();
  int cur_node = p.second.first;
  int cur_prev_node = p.second.second;
  int cur_dis = p.first;

  if (d[cur_node] != INF)
    continue;

  d[cur_node] = cur_dis;
  p[cur_node] = cur_prev_node;

  // Add the nodes connected to current one
```

```
        // Add the nodes connected to current one
        for (int i=0;
             i< adj_list[cur_node].size();
             i++)
        {
          int next_node = adj_list[cur_node][i].first;
          int weight = adj_list[cur_node][i].second;
          if (d[next_node] !=INF)
            continue;
          nodes_q.push({cur_dis + weight,
                  {next_node, cur_node}});
        }
    }

    return d;
}
```

- The shortest distance from 6 to 1 is: 2
- The shortest distance from 6 to 2 is: 10
- The shortest distance from 6 to 3 is: 11
- The shortest distance from 6 to 4 is: 11
- The shortest distance from 6 to 5 is: 8
- The shortest distance from 6 to 6 is: 0

Outline

How can we print the path using the p vector?

```
void print_path(int src, int des, vector<int> p);
```

```
void print_path(int src, int des, vector<int> p){
  stack<int> path_nodes;
  int node = des;

  path_nodes.push(node);

  while(p[node] != node){
    node = p[node];
    path_nodes.push(node);
  }

  while(!path_nodes.empty()){
    cout<<path_nodes.top()<<endl;
    path_nodes.pop();
  }
}
```

Outline

Compelxity??

```
1   DIJKSTRA(G, w, s) ▷ Graph, weights, start vertex
2       for each vertex v in V[G] do
3           d[v] ←∞
4           π[v] ←NIL
5       d[s] ←0
6       Q ←BUILD-PRIORITY-QUEUE(V[G])
7       ▷ Q is V[G] - K
8       while Q is not empty do
9           u = EXTRACT-MIN(Q)
10          for each vertex v in Adj[u]
11              RELAX(u, v, w)      // DECREASE_KEY
```

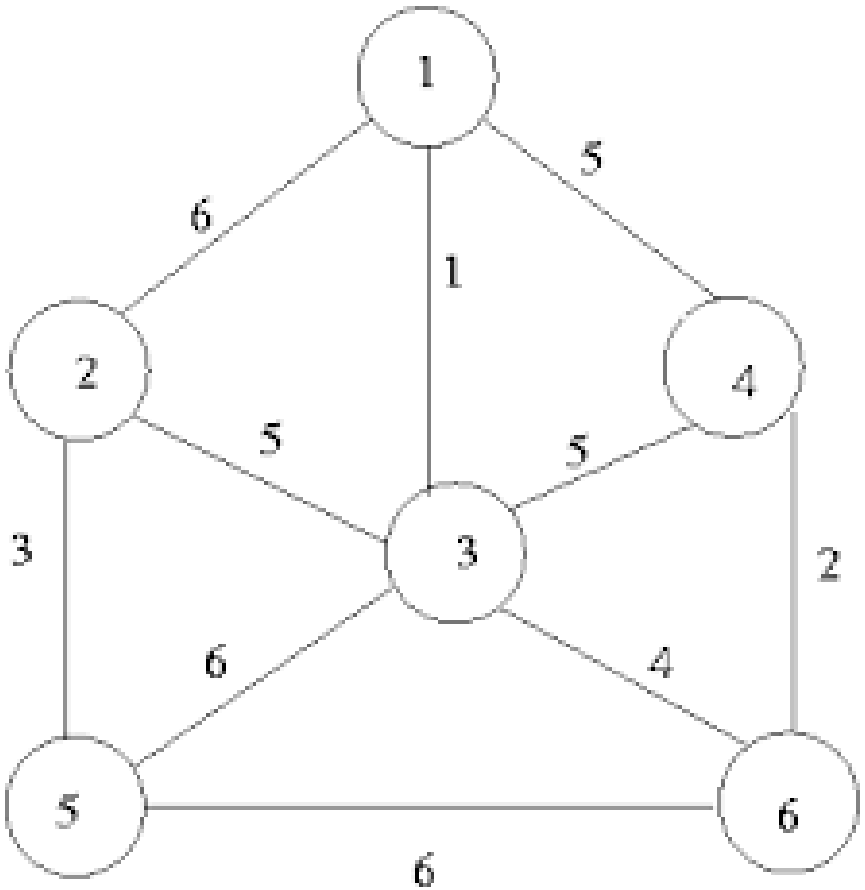
O(V LogV + E LogV)

Outline

A MST for a connected undirected graph:

- is a subset of the edges of a connected, edge-weighted undirected graph.
- connects all the vertices together, without any cycles.
- achieves the minimum possible total edge weight.

Outline


1
2
3
4
5
6
6
1
5
5
5
3
2
6
4
6

The minimum spanning tree has total weight=15
The edges(u,v,w) are:

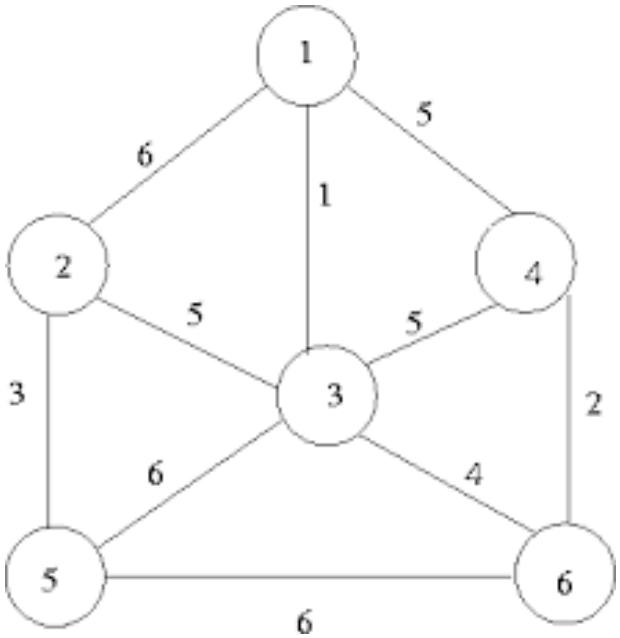
- 2, 5, 3
- 2, 3, 5
- 1, 3, 1
- 3, 6, 4
- 4, 6, 2

```
int compute_MST_cost_PRIM(
  vector<vector<pair<int,int> > > adj_list){

  int n = adj_list.size();
  vector<bool> vis(n, false);

  // start at a random node
  vis[0] = true;

  // The queue sorts the weights ascendingly
  // w, v
  priority_queue<pair<int,int>,
            vector<pair<int, int>>,
            greater<pair<int,int>> >q;

  // Push the edges connected to the start node
  for(int i=0 ;i< adj_list[0].size(); i++){
    q.push({adj_list[0][i].second,
        adj_list[0][i].first});
  }
```

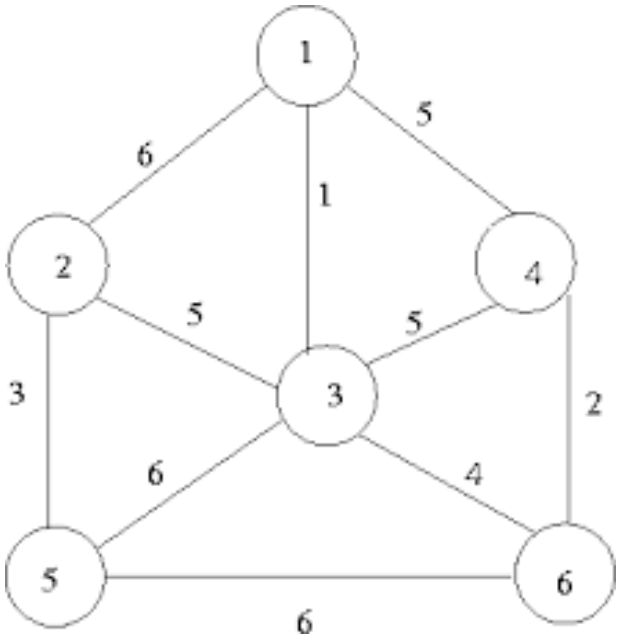
```
    long long cost = 0;
    while(!q.empty()){
      int w = q.top().first;
      int node = q.top().second;
      q.pop();

      if(vis[node])    continue;

      vis[node] = true;
      cost += w;

      for(int i=0; i< adj_list[node].size(); i++){
        q.pu sh({adj_list[node][i].second,
            adj_list[node][i].first});
      }

    }

    return cost;
}
```

Outline

- Let's check how it works.

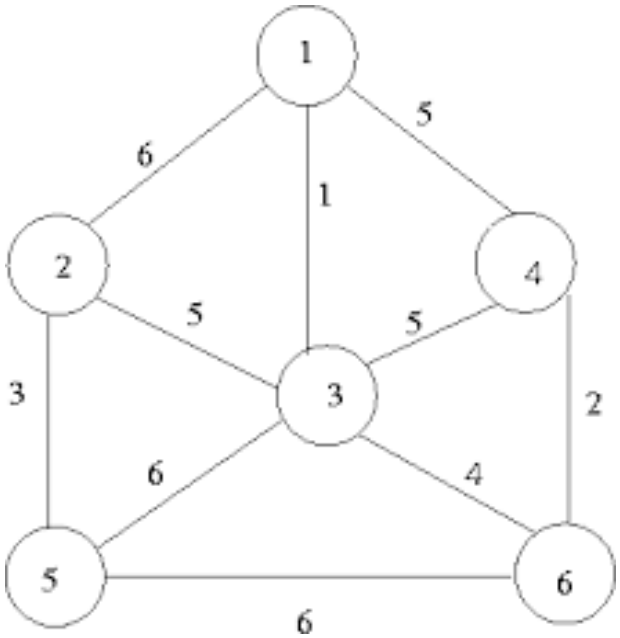


- Let's check how it works.



- Needs a way to detect if two nodes are connected or not. (Disjoint set or Union/Find data structure)

- Let's check how it works.



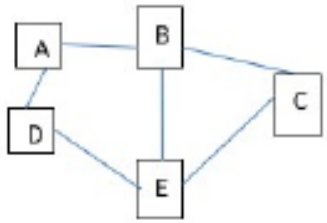
- Needs a way to detect if two nodes are connected or not. (Disjoint set or Union/Find data structure)
- Good Viz: `https://upload.wikimedia.org/wikipedia/commons/b/bb/KruskalDemo.gif`

Outline

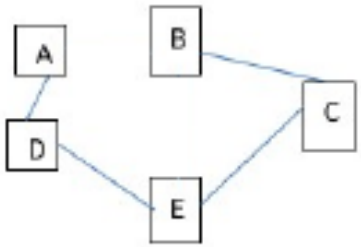
Q4. A spanning tree (ST) is a tree formed from a graph by

- Choosing one vertex to be the root
- Adding edges connecting the root vertex to the neighbor vertices, so they become children
- Doing the same step from the children till all nodes are connected to the tree

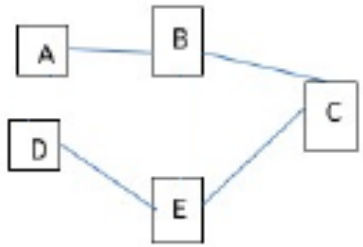
Create all possible spanning trees from the shown graph



A Spanning tree



A Spanning tree



Share your answer: `https://forms.gle/g5M38qiCT1gARcUZ7`

Feedback form: `https://forms.gle/gTJaQJPRrvGsHZFo6`