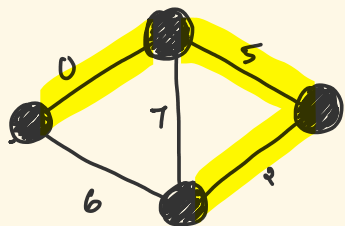# Minimum spanning trees

Sunday, October 29, 2023   6:40 AM

- Subset of edges which connects all vertices together (without cycles) while minimizing the total cost.

Prim's MST Algorithm

- Greedy algo that works well on dense graphs.
- Performs better than other algos on dense graphs.
- However, in case of a disconnected graph the algo has to be run on every single component.

→ Must be run on each component

- 2 implementations $O(E \times \log(E))$ &
  → $O(E \times \log(V))$
  faster

- We store all the edges to unvisited nodes in the priority queue.
- Priority queue will always return the min edge.

```cpp
14  struct compare{
15      bool operator()(pair<int, int> a, pair<int, int> b)
    { return a.second > b.second; }
16  };
17
18  int Prims(int src){
19      priority_queue<pair<int, int>, vector<pair<int,
    int>>, compare> pq;
20      pq.push({src, 0});
21
22      fill(visited, visited+n, false);
23      int mst_cost = 0;
24
25      while(!pq.empty()){
26          auto p = pq.top();
27          pq.pop();
28          int node = p.first;
29          int cost = p.second;
30
31          if(visited[node]) continue;
32          mst_cost += cost;
33          visited[node] = true;
34
35          // Iterate through all the adjacent nodes of
    the node
36          // push the adjacent nodes in the pq only if
    they are not visited yet
37          for(auto next : adj_list[node]){
38              int adj_node = next.first;
39              if(!visited[adj_node]) pq.push(next);
40          }
41      }
42      return mst_cost;
43  }
44
```
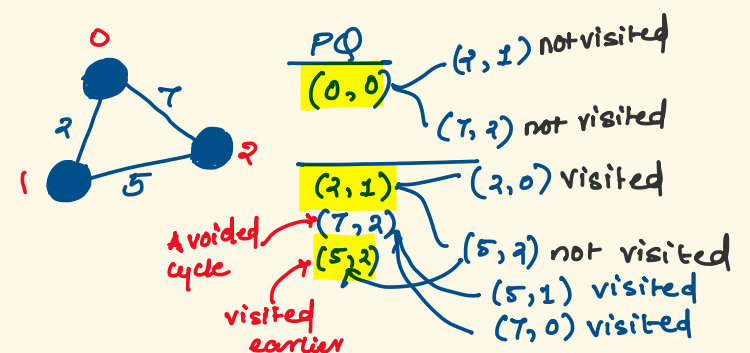
This step avoids cycles.

we always add edge as we need to greedily explore all of them

The reason we check visited twice is that we are adding every single edge to the pq.

∴ A node might hv already been visited with a smaller cost.
- We avoid including the other edge with larger cost.

PQ
(0,0)   (7, 1) not visited
         (7, 2) not visited
(2,1)   (2,0) visited
(7,2)
(5,2)   (5,2) not visited
Avoided (5,1) visited
cycle   (7,0) visited
visited
earlier

← for every node we just care about the lowest cost edge incident on it.
Not lowest cost path.

Kruskal's MST algorithm

- Sort all edges.
- Take edge with min cost.
- Repeat while discarding cycles.

```cpp
12
13  int collapsive_find(int a){
14      // finds parent of subset this node belongs to
15      while(parent[a] != a){
16          parent[a] = parent[parent[a]]; // collapsive find operation
17          a = parent[a];
18      }
19      // recursive
20      // if(parent[a] != a){
21      //     parent[a] = collapsive_find(parent[a]);
22      //     a = parent[a];
23      // }
24      return a;
25  }
26
27  void weighted_union(int a, int b){
28      int d = collapsive_find(a);
29      int e = collapsive_find(b);
30      parent[d] = parent[e]; // merge two subsets
31  }
32
33  int KruskalsMST(){
34      int a, b;
35      int cost, minCst = 0;
36      for(int i = 0; i < v; i++){
37          a = edges[i].second.first;
38          b = edges[i].second.second;
39          cost = edges[i].first;
40          if(collapsive_find(a) != collapsive_find(b)){ // check if we are forming a cycle
    (both nodes belong to same subset)
41              minCst += cost;
42              weighted_union(a, b);
43          }
44      }
45      return minCst;
46  }
```

we update the parent of the subset not the node itself