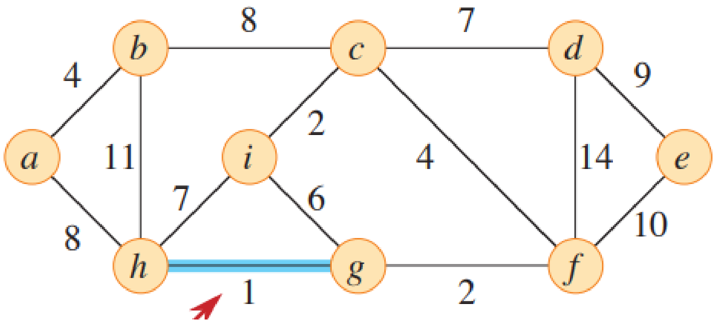
FALL 2023 CS431102 Algorithms
Bonus programming Assignment 4 – Union-Find Kruskal's MST
Student: Victor D. Lopez
ID: 110062426
Professor: JANG PING SHEU

1. How to execute

In the directory where 110062426_bonus4.cpp is located, please execute the following command to compile:

```
$ g++ -Wall -std=c++11 -o bonus4 110062426_bonus4.cpp
```

Then, simple execute *./bonus4* (or whatever you name the executable) in the current directory. This will output the input edges of the graph shown below, the edges belonging to the MST and minimum cost found:



2. Time complexity of Union-Find

```
void Union(int u, int v, std::vector<int>& parent, std::vector<int>& rank){
   int a = find(u, parent);
   int b = find(v, parent);
   if (a != b) {
      if (rank[a] < rank[b]) {
         parent[a] = b;
      }
      else if (rank[a] > rank[b]) {
         parent[b] = a;
      }
      else {
         parent[b] = a;
         rank[a] += 1;
      }
   }
}

int find(int v, std::vector<int>& parent){
   if(v == parent[v]){
      return v;
   }else{
      int res = find(parent[v], parent);
      parent[v] = res;
      return res;
   }
}


void makeSet(int v, std::vector<int>& parent, std::vector<int>& rank){
   parent[v] = v;
   rank[v] = 0;
}
```

From the book, the running time is of these three Disjoint-set operations is given by: $O((|V|+|E|)\alpha(| V |))$ where $)\alpha(| V |)$ is the inverse Ackermann function which has a very slow growing rate.

3. Time complexity of Kruskal’s algorithm

```
//Begin Kruskal’s algorithm to find MST
std::vector<int> parent(n);
std::vector<int> rank(n);

for(int i = 0 ; i < n; i++){
   makeSet(i, parent, rank);
};

//Sort edges by increasing weight
std::cout << "Input graph edges: " << std::endl;
prettyPrintEdge(edges);
std::cout << "--------------------------------" << '\n';
std::sort(edges.begin(), edges.end(),[](Edge * a, Edge * b){
            return a->w < b->w;});


//Kruskal’s
std::vector<Edge * > MST;

for(Edge * e: edges){
   if(find(e->u, parent) != find(e->v, parent)){
      MST.emplace_back(e);
      Union(e->u, e->v,parent, rank);
   }
       }

int cost = computeMSTcost(MST);
```

The running time is dominated by the sorting edges in increasing order of weight. The *std::sort()* function has a running time of *O(|E|log|E|)* which also corresponds to the running time of Kruskal’s algorithm to find an MST.

More specifically, we also have that lg |E| = *O(lg |V|)* since and then we can claim the running time of Kruskal’s algorithm is *O(|E|log|V|).*

Note that both for loops above have linear running time w.r.t to the number of edges *O(|E|)* (Edge selection) and number of vertices *O(|V|)* (MakeSet) and thus are dominated by the running time of the sorting call.

4. Step-by-step edge selection

Our algorithm outputs the following steps to select and discard edges:

```
`Edge: g --->  h with edge weight: 1 chosen
 Edge: f --->  g with edge weight: 2 chosen
 Edge: c --->  i with edge weight: 2 chosen
 Edge: a --->  b with edge weight: 4 chosen
 Edge: c --->  f with edge weight: 4 chosen
 Edge: g --->  i discarded due to cycle formation
 Edge: c --->  d with edge weight: 7 chosen
 Edge: h --->  i discarded due to cycle formation
 Edge: b --->  c with edge weight: 8 chosen
 Edge: a --->  h discarded due to cycle formation
 Edge: d --->  e with edge weight: 9 chosen
 Edge: e --->  f discarded due to cycle formation
 Edge: b --->  h discarded due to cycle formation
 Edge: d --->  f discarded due to cycle formation
```

5. References

[1] Chapter 21 Slides for FALL 2023 CS 431102 Design and Analysis of Algorithms

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein: Introduction to Algorithms, 4th Edition.