Definition of Cut

A cut can be described as a partition that separates a graph into two distinct subsets in graph theory.

Formal Definition : - For a connected graph $G(V, E)$ a Cut $C = (S_1, S_2)$ is defined as partition that divides the vertex set V into 2 disjoint set S_1 and S_2.

We also need to define Cut Set

A cut set is defined for a particular cut. For a cut $C(S_1, S_2)$ in graph $G(V, E)$ It is defined as set edges that have one endpoint in S_1 and other in S_2.

Cut-set for $C(S_1, S_2)$ in graph $G(V, E)$ = $\{(j, k) \in E | j \in S_1, j \in S_2\}$

Example

some picture

Cut Property

If an edge in the cut set has the least edge weight or cost among all other edges in the cut set, the edge should be included in the minimal spanning tree, according to the cut property.

Formal: Suppose edges X is any subset of Edges of a MST of G (V,E). Pick any subset of nodes S for which does not cross between S and V-S and let e be the lightest edge in cut set of C(S,V-S) . Then X ∪ {e} is part of some MST.

Proof

- Edges X are part of some MST T. Now if the new edge also happens to be part of T then there is nothing to prove. So lets assume $e \notin T$. Since there can be multiple MST of a graph now we are making a new MST T'(changing only one of Ts edge) containing $X \cup \{e\}$.
- Now for making T' let add edge e to T . Since T was already connected adding an edge will result in cycle so by property that removing edge from from a graph doesn't disconnect a graph we will remove edge e' between $S \text{ and } V - S$. So by Properties
  - A tree on n nodes has n − 1 edges
  - Any connected, undirected graph G = (V, E) with |E|=|V| − 1 is a tree.

  T' is also a tree.
- Comparing T and T'

$$weight(T') = weight(T) + w(e) - w(e')$$

  Now since we have selected e with minimum edge weight between $S, V - S$ $\implies w(e) \leq w(e') \implies weight(T') \leq weight(T)$. Since T is an MST $weight(T') = weight(T)$. Hence Proved T' is also an MST.

Getting MST from cut property

Using cut property we can have many diff greedy mst algorithm.

As cut property always gives us the greedy step

You can have as many type sub-structuring as you where each substructure method will give different algorithm .

Trick with substructure is how to select the set S.

Prims algorithm

Getting Prims algorithm from Cut property

Greedy Step

Select the lightest edge between set set S and V-S

Restructure Property

Take endpoint of the edge (selected in greedy step) which is in set V-S. Add this vertex into S and remove from V-S.

- We start with an empty set S
- So initial value of S = {\phi}, V-S = V
- We add a source node to S
- Now, S = {Source}, V-S = All nodes other then source
- We select the lightest edge from cut-set of C(S,V-S).(Greedy step)
- Take the endpoint of edge which was in V-S. Add it into S and remove from V-S. (substructure step)
- Do the above step until S includes all the vertices

Time analysis

Using the Formula for time complexity = O(Finding and Executing Optimum step) * O(Restructuring) * O(No. of restructuring required)

O(Finding and Executing Optimum step)) = log V (using priority queue).

O(Restructuring) = 1

O(No. of restructuring required) = O(E)

Final time complexity = O(ElogV)

Kruskal's Algorithm

1. Start with a empty graph
2. Repeatedly add the next lightest edge that doesn’t produce a cycle.

Basically what we are doing here is we are constructing a tree edge by edge by selecting the best locally possible edge at each step (greedily).

Here

We initially have all vertexes as singleton sets

Greedy step

Here greedy step is smallest edge which does not make a cycle (Edge between 2 disjoint sets)

Restructure step

- Take the endpoints of the edge selected in greedy step

- Find the set of in which first endpoint is present and find set in which the second endpoint is present.
- Do the union of these 2 sets.

Time analysis

Final algorithm uses $|V|$ `makeset` , $2|E|$ `find` , $|V|-1$ `union` and 1 `sort` operations. Assuming we know all this complexities if we calculate final complexity $O(|E|+|V|\log|V|)$.