**Dijkstra’s Algorithm:**

Initialize S = {s} and d(s) = 0;

While S V:

Select a node v S with at least one edge from S for which

;

Add v to S, record d(v);

Endwhile

**Kruskal’s Algorithm (G, W):**

Sort edges weights so that

T

For each (u make a set containing singleton u;

For i = l to m:

(u, v) =

If (u and v are in different sets):

T

Merge the sets containing u and v

Endif

Endfor

Return T

**Prim’s Algorithm (G, W):**

While S

Select a node v with at least one edge from S for which

Add v to S;

Add to T;

Endwhile

Return T

Proof of **Dijkstra’s Algorithm:**

**Proposition**: Given a graph G(V, E, L) with no negative edge, for any node v selected by the criteria , the quantity d(u) + corresponds to an s - v path, which is the shortest s - v path

**Proof: We will use mathematical induction for the proof.**

(Base case): The statement is clearly true when v = s, if we set d(s) = 0 and = 0 for arbitrary node i. That means the shortest path from s to s is null and its length is 0, and it is true per definition.

图示

描述已自动生成(induction step): Let S be a set of vertices that have been visited. We assume for any node u S we have d(u) being the shortest path length between s and u, and the corresponding shortest path is known (Note: initially S = {s} and the hypothesis is true). In each algorithm run, we would pick the node v, where .

We compare the greedy path with a length of d(u) + and another arbitrary s – v path (generalization of a generic particular). Since the path connects

s S and v V – S, must contain an edge where x S and y V – S (Note: it is possible that x = u or y = v, but the equalities do not both hold at the same time because it would directly lead to d(u) + correspond to the shortest path). In per selection criterion, we have . Also, the graph has no negative-weighted edge, hence d() 0. Adding the term to the right-hand side we will get . Or (. Since is an arbitrary s – v path, it indicates that is shorter than or has the same length as any valid s – v path. Note that can be constructed by appending the edge to the path corresponding to d(u), which exists per induction hypothesis.

(conclusion): Therefore, for the node v selected by criteria , the shortest path s – v has a length of .

Proof of **Kruskal’s Algorithm:**

**Lemma:** Assume that all edge costs are distinct (will be relaxed later). Let S be any subset of vertices that is neither empty nor equal to all of V, and let edge e = (v, w) be the minimum cost edge with one end in S and the other end in V – S. Then every minimum spanning tree contains the edge s.

Proof: argue that every edge selected by the greedy algorithm is at least as good as any other edge.

**We use prove by contradiction.**

图示, 工程绘图

描述已自动生成Proof: Consider the scenario shown in the figure, where the greedy algorithm picked edge e = (v, w), where v S and w V – S. We assume that the MST of the graph is T and it does not contain e. Per definition of MST, all nodes must be connected through it, including v and w. Since e T, there must exist another path in T that connects v, w. Since v S and w V – S, the path must contain an edge , where and . Note that . Let (replace with e in the MST). Obviously, per greedy selection criterion (also because we assumed that all edges have distinct weights). It is then clear that w() < w(T). We also argue that all nodes in G remained connected, since all connections in T that need to pass can now be redirected through . Specifically, let be the path that connects v, w, it follows that v, v’ are connected and w, w’ are connected. So, it is feasible to connect v’, w’ in T via in T’. We also claim that T’ contains no cycle. Note that T is an MST, and it contains no cycle per definition. It indicates that P is the unique path that connects v, w. After removing , no path can connect v, w anymore. After adding e, we can only create a unique connection between v, w. Since e is the only added edge in T’, the other parts of T’ remained the same as T and contain no cycle either. Taken together, T’ contains no cycle.

(conclusion): To summarize, we have shown that , T’ connects all nodes in G and T’ contains no cycle. In this case T’ is a spanning tree that has a lower cost than T, which contradicts with the assumption that T is an MST. Therefore, any MST for G must contain e.

Proposition: **Kruskal’s algorithm produces an MST of G.**

Proof: According to the proven cut property, each added edge by the Kruskal’s algorithm must be contained in an MST of G. We also claim that when the algorithm terminates, we have added sufficient amount of edges to form a spanning tree. This is true because the algorithm only terminates when all nodes are connected.

Proposition: **Kruskal’s algorithm is correct on graphs containing equally-weighted edges.**

Proof: In a special case where all edges have the same weight, any spanning tree will be an MST, since Kruskal’s algorithm identifies a spanning tree, it also identifies an MST. On the other hand, we have at least two edges that have different weights. Let the minimum difference between any two differently weighted edge be d. We can make an alternative graph G’, which has the identical node and edge set as G, but with its edge-weight ties broken by adding positive, arbitrarily small perturbations e. For example, consider 4 edges with weights 1.9, 2, 2, 2, then d = 0.1. We can set e = 0.01, and break the ties of the equally-weighted edges by reassigning their weights into 2(+0e), 2.01(+1e), 2.02(+2e). Note that we have at most |V| equally weighted edges, then we can set e < d / |V|^2 to satisfy the condition. We then run the Kruskal’s algorithm on G’ and denote the result as T’. since G’ contains no equally weighted edge, T’ is an MST of G’. Also denote that the true MST of G be T. Denote the weight of T’ evaluated on the graph G be . That is, the total weight after removing all the added perturbations. There are three cases to discuss:

1. When < . This is impossible since it contradicts with the assumption that T is the MST of G
2. When = . Then T’ is also an MST of G, so Kruskal’s algorithm is correct.
3. When > . In this case, - < d. Otherwise - d, we will have Simplify the inequation we get . Also note that since we only add positive perturbations to form G’. Taken together, we have . It indicates that T will be an MST of G’, contradicting with the fact that T’ is an MST of G’. Since d is defined as the smallest difference between any two differently weighted edges, T’ and T are identical.

In all possible cases, T’, and T are identical, hence the Kruskal’s algorithm correctly identifies an MST on graphs containing equally weighed edges.