Definitions

Definition 1 (Greedy algorithm). A greedy algorithm is one which always makes the choice that looks best at the moment—the *locally optimal* choice—in order to find the best globally optimal solution. Greedy algorithms do not always yield optimal solutions, but for many problems they do.

Definition 2 (Shortest path). A shortest path from vertex *s* to vertex *t* is a directed path from *s* to *t* with the property that no other such path has a lower total edge weight.

Definition 3 (Spanning tree). A spanning tree of the graph *G* is a subgraph of *G* which is a tree and connects all of the vertices in *G*.

Definition 4 (Minimum spanning tree). A minimum spanning tree *T* of *G* is a spanning tree of *G* with the property that the sum of the weights of every edge in *T* is smaller than the sum of the weights in any other spanning tree of *G*.

Dijkstra’s Algorithm

Dijkstra’s algorithm finds the shortest path between two given vertices in a weighted graph, assuming that the graph’s edge weights are non-negative. The running time of the algorithm is \(O(E \log V + V \log V)\) when the graph is implemented using adjacency lists. With a special transformation (use of Fibonacci heaps) this can be reduced to \(O(E + V \log V)\), which is the fastest version of this algorithm. The pseudo-code for the algorithm is given below.

Pseudocode

Dijkstra(G, s)

1. for each vertex \(v \in V_G\)
2. \(\text{dist}[v] = \infty\)
3. \(\text{parent}[v] = \text{NIL}\)
4. \(\text{dist}[s] = 0\)
5. \(Q = V_G\)
6. while \(Q \neq \emptyset\)
7. \(u = \text{Extract-Min}(Q)\)
8. for each vertex \(v \in G.\text{Adj}[u]\)
9. if \(\text{dist}[v] > \text{dist}[u] + w(u, v)\)
10. \(\text{dist}[v] = \text{dist}[u] + w(u, v)\)
11. \(\text{parent}[v] = u\)

Edge-Weighted DAGs (Directed Acyclic Graphs)

The algorithm for shortest path on edge weighted DAGs is simpler and faster than Dijkstra’s algorithm. However, instead of considering vertices by priority of their distance estimates, we consider the vertices of the DAG in a topological order. (Why must a DAG always have a topological order?) Then we just relax each vertex in the topological ordering. Running time: \(O(|V| + |E|)\).
**Prim’s Algorithm**

Prims algorithm finds a minimum spanning tree for a connected weighted graph. The greedy algorithm can be summarized in the following way:

- Initialize a tree with a single vertex, chosen arbitrarily from the graph.
- Grow the tree by one edge: of the edges that connect the tree to vertices not yet in the tree, find the minimum-weight edge, and transfer it to the tree.
- Repeat the previous step (until all vertices are in the tree).

**Dijkstra Questions**

**Problem 1.** Find the shortest path between vertices $E$ and $G$ in the graph provided.

**Problem 2.** Explain why Dijkstra’s algorithm is a greedy algorithm.

**Problem 3.** Does Dijkstra’s Algorithm work with negative weights? Why or why not?

**Problem 4.** True or false: Dijkstra’s algorithm will not terminate if run on a graph with negative edge weights.

**Problem 5.** True or false: The shortest path algorithm in an edge weighted DAG works even with negative edge weights.

**Problem 6.** How could you modify Dijkstra’s algorithm to find all shortest paths?

**Problem 7.** How could you modify Dijkstra’s algorithm to stop once it’s found the shortest path to a particular node?

**Problem 8.** Explain the running time of Dijkstra’s algorithm.

**Problem 9.** True or false: If we double the weights of all the edges in a graph, then Dijkstra’s algorithm will produce the same shortest path.

**Problem 10.** Say we are given a graph $G$ where all edges are positively weighted. Construct graph $G'$ where for all edges $e$ with weight $w(e)$ and endpoints $u$ and $v$, we replace $e$ with $w(e)$ edges of weight 1 in $G'$, such that the path from $u$ to $v$ in $G'$ consists of $w(e) - 1$ middle nodes. How could you use this method to find the shortest path between two vertices in $G$? What problem do you see with this approach?

**MST Questions**

**Problem 11.** Create the MST of the graph provided using Prims Algorithm.

**Problem 12.** Can a graph have more than one MST? Explain.

**Problem 13.** Explain why Prim’s algorithm is a greedy algorithm.

**Problem 14.** Does Prim’s Algorithm work with negative weights? Why or why not?

**Problem 15.** Say we have some MST, $T$, in a positively weighted graph $G$. Construct a graph $G'$ where for any weight $w(e)$ for edge $e$ in $G$, we have weights $(w(e))^2$ in $G'$. Does $T$ still remain an MST in $G'$? Prove your answer. Now if $G$ also had negative weights, would your answer change from the previous part? Prove your answer.