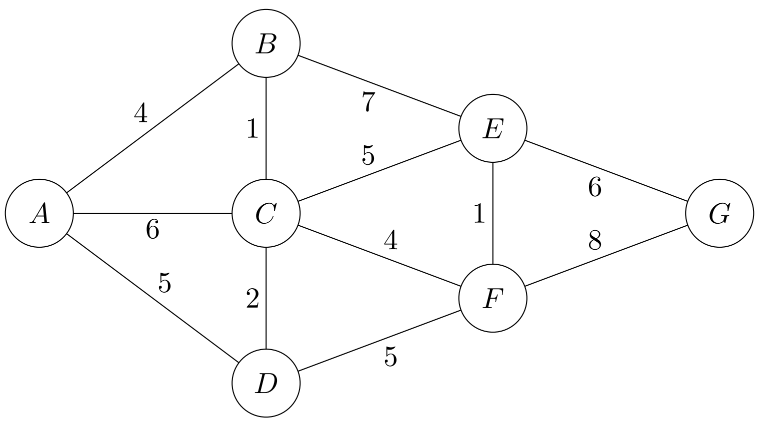
CSM 61B Fall 2017

Graph Algorithms

Mentoring 13: November 20, 2017

1 Networking

1.1 Suppose we want to design a telephone network connecting all the cities, labeled A to G, in a neighborhood. We'd like to do so at the least cost.



(a) In a graph with N vertices and M edges, how many edges form a minimum spanning tree?

(b) Will the new graph contain any cycles? Describe its structure.

(c) One algorithm to find a minimum spanning tree is Kruskal's algorithm.

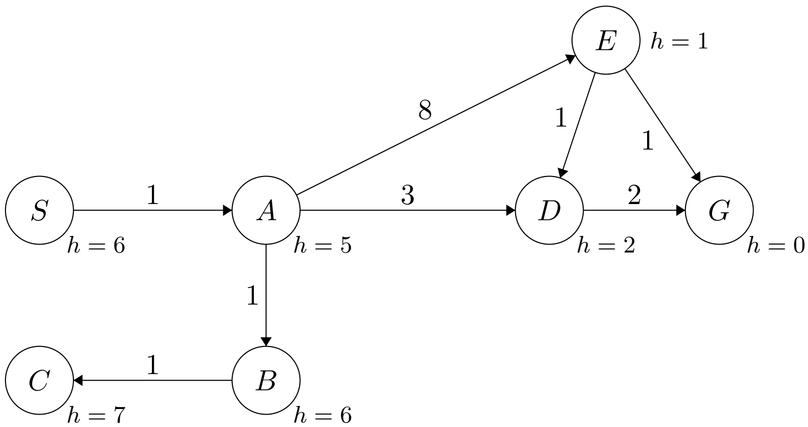
1. Sort all the edges by increasing order of their weight.
2. Pick the smallest edge and check if it forms a cycle with the spanning tree so far. If it doesn't form a cycle, add this edge to the spanning tree.
3. Repeat the previous step until there are $|V| - 1$ edges in the spanning tree, where $|V|$ is the number of vertices in the graph.

Run Kruskal's Algorithm to find a minimum spanning tree.

2 A* Search

2.1 Find the path from the start, S, to the goal, G, when running each of the following algorithms.

The **heuristic**, h, estimates the distance from each node to the goal.



(a) Which path does uniform cost search return?

Uniform cost search is the same as Dijkstra's, except that the search stops once we visit the goal state.

(b) Which path does greedy search return?

In **greedy search**, we ignore edge weights entirely and only use the heuristic to decide which node to visit next.

(c) Which path does A* search return?

A* search is an algorithm that combines the total distance from the start with the heuristic to optimize the search procedure.

3 Algorithms *Extra Practice*

Traversal Visit all the nodes in the graph.

- Depth-first traversal (preorder and postorder)
- Level-order traversal

Search Given s, find a goal v.

- Depth-first search
- Iterative-deepening depth-first search
- Breadth-first search

Single Pair Shortest Path Given s, find the shortest path to a goal v.

- *Uniform cost search*
- *Greedy search*
- A* search

Single Source Shortest Path Given s, find the shortest path to all nodes.

- Dijkstra's algorithm

Minimum Spanning Tree A *spanning tree*, or acyclic subgraph connecting all the nodes with the least total edge weight.

- Prim's algorithm
- Kruskal's algorithm

3.1 Is this algorithm for computing the *single pair shortest path* correct?

Given a starting vertex, s, and an ending vertex, v, compute the shortest path between s and v by running DFS, but at each node exploring the shortest outgoing edge first until v is reached. Return the s to v path in the DFS tree.

3.2 Briefly describe an efficient algorithm and the runtime for finding a minimum spanning tree in an undirected, connected graph $G = (V, E)$ when the edge weights satisfy:

(a) For all $e \in E$, $w_e = 1$. (All edge weights are 1.)

(b) For all $e \in E$, $w_e \in \{1, 2\}$. (All edge weights are either 1 or 2.)

3.3 Given a weighted, directed graph G where the weights of every edge in G are all integers between 1 and 10, and a starting vertex s in G, find the distance from s to every other vertex in the graph where the distance between two vertices is defined as the weight of the shortest path connecting them, or infinity if no such path exists.

(a) Design an algorithm for solving the problem better than Dijkstra's.

(b) Give the runtime of your algorithm.