Due at the end of class

Work in groups of three or four. This exercise will not be graded, but your involvement in the activities will affect your grade for participation. Please TURN IN ONE COPY per group, with names of all members at top.

1 Dijkstra’s Algorithm

Dijkstra(WeightedGraph g, Vertex s) {
    set d[v] = infinite for all nodes v
    pq = PriorityQueue containing pairs (v, cost), smallest to largest
    d[s] = 0
    add (v, d[v]) to pq for all nodes v
    while pq is not empty {
        remove lowest cost node v from pq
        for each nbr of v {
            if d[nbr] > d[v] + w(v, nbr) {
                d[nbr] = d[v] + w(v, nbr)
                update cost of nbr in pq
                parent[nbr] = v
            }
        }
    }
    // d encodes distances, parent encodes lowest cost paths
}

(a) Following a similar analysis to the runtime of Breadth-First-Search. What is the runtime of Dijkstra’s algorithm? For the updating the cost of the neighbor step, assume that this has running time $x$ units of work.
(b) If we use a special kind of heap, called a Fibonacci heap, the update step can be
done in $O(1)$ amortized time. If we use a binary heap, what is the cost? (Assume we
add a public method to our heap implementation that provides this functionality.)

(c) If we want to use a binary heap, we can simply add a new entry to the priority
queue rather than updating the existing entry. Why does the algorithm still work?
Should we make any other changes to the algorithm?

(d) In Dijkstra’s algorithm, the weights of edges must be non-negative. Give an exam-
ple graph that has negative weights where Dijkstra’s algorithm fails to produce the
correct answer.

(e) What does Dijkstra’s algorithm compute when all of the edges have equal weight?

(f) In the special case where all of the edges have equal weight, how can you simplify
the above algorithm? Try to justify each change that you make.

2 Prim’s Algorithm

Prim’s algorithm finds a minimum spanning tree of a graph. The algorithm is strikingly
similar to Dijkstra’s algorithm for finding shortest paths. In this exercise, your task is to
write pseudocode for Prim’s algorithm.
The input to Prim’s algorithm is a weighted undirected graph \( G = (V, E) \). We grow a minimum spanning tree starting from a single “root” vertex \( s \). The choice of \( s \) is arbitrary: choose any node. The algorithm maintains a set of nodes, called \( C \), and a set of edges \( T \) that form a minimum spanning tree over the nodes of \( C \).

Initially, \( C \) contains only the start node \( s \) and the set of edges \( T \) is empty. At each round of the algorithm, we identify an edge that connects a node in \( C \) to a node outside of \( C \) (i.e. a node in the set \( V - C \)). To ensure it remains a minimum spanning tree, among all of the edges that connect a node inside \( C \) to a node outside \( C \), we select the edge with minimum weight. This edge is added to \( T \) and the corresponding node that was previously outside of \( C \) is added to \( C \). When all the nodes are in \( C \), the algorithm terminates and the edges of \( T \) define a minimum spanning tree.

(a) Each time we add an edge to \( T \), how do we know it remains a tree? Recall the three properties of a tree: the number of edges is one less than the number of nodes, it is connected, and it has no cycles.

(b) Simulate the algorithm on the example shown in the figure. Write down the order in which edges are added to the tree. Start with node \( A \).

(c) Write pseudocode for Prim’s algorithm on the following page.
FindMST(WeightedGraph g, Vertex s) {
}

}