CSM 61B
Spring 2019

More MSTs & Sorting
Mentoring 10: April 8, 2019

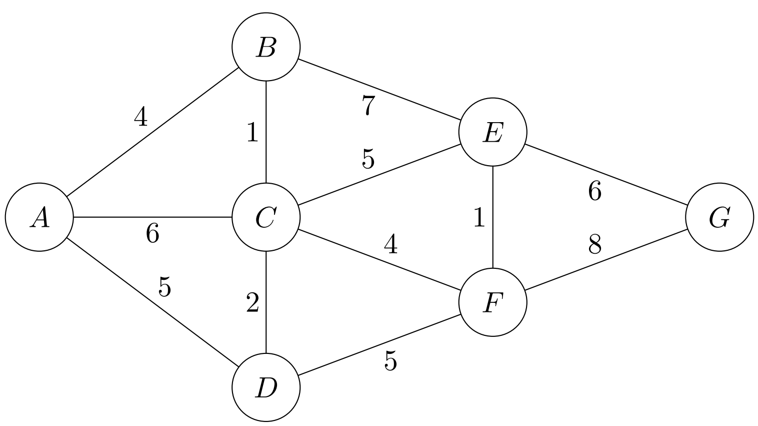
1 Prim's Algorithm

The cut property states that given any cut, the minimum weight crossing edge is in the MST. The converse is also true, that if an edge is in the MST, then it must be the minimum weight crossing edge across some cut. This property is a key idea behind Prim's algorithm.

1.1 Describe Prim's algorithm.

1.2 We can use a binary heap priority queue to implement Prim's. What would be the runtime of Prim's using this implementation?

1.3 Consider the telephone network from last week. Construct a minimum spanning tree by running Prim's Algorithm from node A.



2 Maximum Spanning Trees

2.1 We have two algorithms, Kruskal's and Prim's, that allow us to find a Minimum Spanning Tree. Consider the problem of finding a Maximum Spanning Tree

(a) Describe a modification to Kruskal's algorithm that would allow us to find a Maximum Spanning Tree of a graph

(b) Can we use a similar approach to modify Djikstra's algorithm to find the Maximum Path between two nodes?

3 Feeling Out of Sorts?

So far, we've learned a few different types of basic sorting algorithms. While sorting might seem like a simple idea, there are many real-world applications of sorting, and several different algorithms that we can use depending on the situation.

In the table below, fill out the best and worst-case runtimes for each of the sorting algorithms provided.

Algorithm	Best-case	Worst-case
Selection Sort		
Insertion Sort		
Merge Sort		
Heapsort		

3.1 Give a best and worst case input for insertion sort.

3.2 Do you expect selection or insertion sort to run more quickly on a reverse list?

3.3 In Heapsort do we use a min-heap or max-heap? Why?

3.4 Sort the following array using Heap Sort. [3, 2, 1, 5, 6, 8, 7]

4 Vertigo

4.1 We have a list of N elements that should be sorted, but to our surprise we recently discovered that there are at most k pairs out of order, or k **inversions**, in the list. The list `{ 0, 1, 2, 6, 4, 5, 3 }`, for example, contains 5 inversions: $(6,4),(6,5),(6,3),(4,3),(5,3)$.

For each value of k below, state the most efficient sorting algorithm and give a tight asymptotic runtime bound.

(a) $k \in O(\log N)$

(b) $k \in O(N)$

(c) $k \in O(N^2)$