CS 170 HW 4

Due on 2018-09-23, at 9:59 pm

1 (⭐) Study Group

List the names and SIDs of the members in your study group.

2 (⭐⭐) Maximum Subarray Sum

Given an array of \( n \) integers, the maximum subarray is the contiguous subarray (potentially empty) with the largest sum. Design a linear algorithm to find the sum of the maximum subarray. For example the maximum subarray of \([-2, 1, -3, 4, -1, 2, 1, -5, 4]\) is \([4, -1, 2, 1]\), whose sum is 6.

Please give a three-part solution of the following format:

(a) Clearly describe your algorithm. You can include the pseudocode optionally.

(b) Write a proof of correctness.

(c) Write a runtime analysis.

3 (⭐) MST Basics

For each of the following statements, either prove or supply a counterexample. Always assume \( G = (V, E) \) is undirected and connected. Do not assume the edge weights are distinct unless specifically stated.

(a) Let \( e \) be any edge of minimum weight in \( G \). Then \( e \) must be part of some MST.

(b) If \( e \) is part of some MST of \( G \), then it must be a lightest edge across some cut of \( G \).

(c) If \( G \) has a cycle with a unique lightest edge \( e \), then \( e \) must be part of every MST.

(d) For any \( r > 0 \), define an \( r \)-path to be a path whose edges all have weight less than \( r \). If \( G \) contains an \( r \)-path from \( s \) to \( t \), then every MST of \( G \) must also contain an \( r \)-path from \( s \) to \( t \).

4 (⭐) Prim’s Algorithm

A popular alternative to Kruskal’s algorithm is Prim’s algorithm, in which the intermediate set of edges \( X \) always forms a subtree, and \( S \) is chosen to be the set of this tree’s vertices. We can think of Prim’s algorithm as greedily processing one vertex at a time, adding it to \( S \). The pseudocode below gives the basic outline of Prim’s algorithm. See the book for a detailed example of a run of the algorithm.
$S = \{ v \}$
$X = \{ \}$

While $S \neq V$:

Choose $t \in V \setminus S$, $s \in S$ such that $\text{weight}(s, t)$ is minimized

$X = X \cup \{(s, t)\}$

$S = S \cup \{t\}$

Return $X$

(a) Run Prim’s algorithm on the following graph, starting from A, stating which node you processed and which edge you added at each step.

(b) Prim’s algorithm is very similar to Dijkstra’s in that a vertex is processed at each step which minimizes some cost function. These algorithms also produce similar outputs: the union of all shortest paths produced by a run of Dijkstra’s algorithm forms a tree. However, the trees they produce aren’t optimizing for the same thing. To see this, give an example of a graph for which different trees are produced by running Prim’s algorithm and Dijkstra’s algorithm. In other words, give a graph where there is a shortest path from a start vertex $A$ using at least one edge that doesn’t appear in any MST.

5 (⋆) Divide and Conquer for MST?

Is the following algorithm correct? If so, prove it. Otherwise, give a counterexample and explain why it doesn’t work.

procedure $\text{FindMST}(G$: graph on $n$ vertices$)$

If $n = 1$ return the empty set

$T_1 \leftarrow \text{FindMST}(G_1$: subgraph of $G$ induced on vertices $\{1, \ldots, n/2\})$

$T_2 \leftarrow \text{FindMST}(G_2$: subgraph of $G$ induced on vertices $\{n/2 + 1, \ldots, n\})$

$e \leftarrow$ cheapest edge across the cut $\{1, \ldots, \frac{n}{2}\}$ and $\{\frac{n}{2} + 1, \ldots, n\}$.

return $T_1 \cup T_2 \cup \{e\}$. 