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.

Solution:

(a) **Description of the algorithm**
The DQ approach will only give an $O(n \log n)$ algorithm. A linear algorithm can be: scan the array and record the sum. If sum becomes negative then set it to 0. Then the maximum subarray sum is the maximum value that sum has ever been during the scan.

Pseudocode

Set $maxSum := 0$, $sum := 0$
for i in 1:n **do**
  $sum = \max(0, sum + a[i])$
  $maxSum = \max(maxSum, sum)$
return $maxSum$

(b) **Proof of correctness.**
If all the numbers in the array are negative, then the algorithm just returns 0.

Otherwise, note that

- Sum is set to 0 right before the scan of the maximum subarray. (Otherwise there is a postive subarray right before the maximum subarray and they can combine together to become a subarray with alarger sum.)
- Sum will not be set to 0 during the scan of the maximum subarray. (Otherwise the maximum subarray starts with a negative subarray.)

Therefore, maximum subrray value is recorded by $maxSum$ and algorithm is correcrt.

(c) **Runtime analysis.**
Clearly the algorithm is $O(n)$.

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.

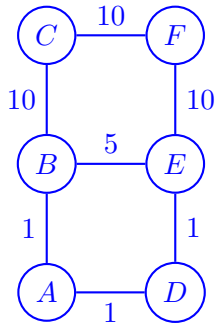
(d) For any $r > 0$, define an r-path to be a path whose edges all have weight less than r. If G contains an r-path from s to t, then every MST of G must also contain an r-path from s to t.

Solution:

1. True, e will belong to the MST produced by Kruskal.

2. True, suppose (u, v) is the edge. Let one side of the cut be everything reachable in the MST from u without using the edge (u, v). If this cut has an edge lighter than (u, v) then we could add this edge to the MST and remove (u, v). We know this edge is not already in the MST because otherwise both its endpoints would be reachable from u without using (u, v).

3. False. Let e be also the heaviest edge of a different cycle; then, we know that e can't be part of the MST. Concretely, in the following graph, edge (B, E) satisfies this condition, but will not be added to the graph.



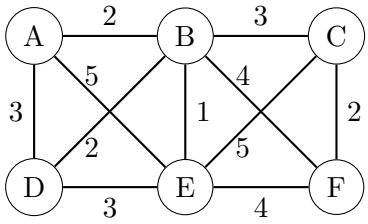
4. True. Let $v_1, v_2, \ldots, v_n$ denote the r path in G from $s = v_1$ to $t = v_n$. Consider the greatest i such that the MST has an r path from s to v_i and assume for contradiction that $i < n$. Then the edge (v_i, v_{i+1}) is not in the MST. Adding this edge to the MST forms a cycle, which must have edges of length less than r since otherwise we could replace one of them with (v_i, v_{i+1}). Thus there is an r path from v_i to v_{i+1} and hence from s to v_{i+1}, contradicting our initial assumption.

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 ≠ V:
    Choose t ∈ V\S, s ∈ S such that weight(s,t) is minimized
    X = X ∪ {(s,t)}
    S = S ∪ {t}
Return X
```

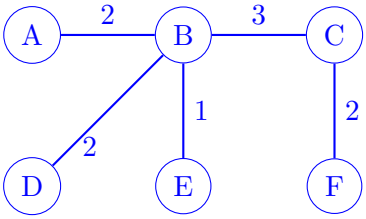
(a) Run Prim's algorithm on the following graph, starting from A, stating which node you processed and which edge you added at each step .



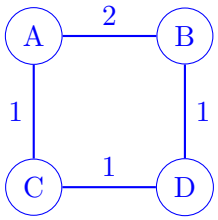
(b) Prim's algorithm is very similar to Dijkstra's in that a vertex is processed at each step which minimizes some cost function. These algorithms also produce similar outputs: the union of all shortest paths produced by a run of Dijkstra's algorithm forms a tree. However, the trees they produce aren't 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.

Solution:

(a) In the form *vertex* : *edge*, the processing was [A, B : (A,B), E : (B,E), D : (D,E), C : (B,C), F : (C, F)]
The resulting MST is:



(b)



The MST is $\{(A, C), (C, D), (B, D)\}$, but the shortest path from A to B is through the weight 2 edge.

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 FindMST(G: graph on n vertices)
If $n = 1$ return the empty set
$T_1 \leftarrow$ FindMST(G_1: subgraph of G induced on vertices $\{1, \ldots, n/2\}$)
$T_2 \leftarrow$ 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 $T_1 \cup T_2 \cup \{e\}$.

Solution: This algorithm does not work; multiple edges of the MST could cross this particular cut. Another way to see this is that the MSTs of the subgraph needn't also be part of the MST of the whole graph.
As a concrete counterexample, consider a wide rectangle and the horizontal cut between the top two vertices and the bottom two. Both edges on this cut should be in the MST.