1. (2 pt.) Consider the graph $G$ below.

\[ \begin{array}{cccc}
A & 2 & B & 9 \\
1 & 4 & 7 & 8 \\
D & 3 & E & 6 \\
\end{array} \]

(a) (1 pt.) In what order does Prim’s algorithm add edges to an MST when started from vertex $C$?

(b) (1 pt.) In what order does Kruskal’s algorithm add edges to an MST?

[We are expecting: For both, just a list of edges. You do not need to draw the MST, and no justification is required.]

2. (4 pt.) Consider the following graph:

\[ \begin{array}{cccc}
C & D \\
A & B & E \\
\end{array} \]

(a) (1 pt.) What is the global minimum cut of this graph?

[We are expecting: Just the global minimum cut. No explanation is required.]

(b) (1 pt.) What is the probability that Karger’s algorithm chooses an edge crossing the minimum cut with its first choice?

[We are expecting: Just the probability. No explanation is required.]

(c) (2 pt.) What is the exact probability that one run of Karger’s algorithm returns a minimum cut on this graph? How does it compare to the bound of $1/\binom{n}{2}$ that we saw in class?

[We are expecting: Your numerical answer (no justification required), as well as a statement about how this compares to $1/\binom{n}{2}$.]
Problems

You may talk with your fellow CS161-ers about the problems. However:

• Try the problems on your own before collaborating.
• Write up your answers yourself, in your own words. You should never share your typed-up solutions with your collaborators.
• If you collaborated, list the names of the students you collaborated with at the beginning of each problem.

3. (5 pt.) [Fish Stops.] Plucky the Pedantic Penguin is walking $t$ miles across Antarctica. They need to eat along the way, but they can only eat when there’s a fishing hole for them to catch fish. They can walk at most $m$ miles between meals, and they know how $n$ fishing holes are laid out along their route.

Plucky is given an array $F$ so that $F[i]$ gives the distance from the start of their journey to the $i$’th fishing hole. There are $n$ fishing holes along the way, including at the beginning and the end: $F[0] = 0, F[n - 1] = t$. For example, the array $F = [0, 3, 4, 6, 10, 12]$, with $t = 12$ corresponds to the setup below:

Plucky wants to stop as few times as possible, given that they can walk at most $m$ miles without eating. (It is okay if they walk exactly $m$ miles between meals). Plucky starts out hungry, so they will always fish at 0 miles; they will also always fish at their destination (at $t$ miles), whether or not Plucky is hungry.

In the example above, if $m = 4$, then Plucky should stop 5 times (including stops at the beginning and the end), for example at 0, 4, 6, 10, 12 miles.

Design a greedy algorithm for Plucky to use. The algorithm should have the following properties:

• Your algorithm should take as input the array $F$, as well as the parameters $m$ and $t$. You may assume that $F$ is sorted.
• Your algorithm should output a list fishStops which contains a shortest list of places that Plucky could stop for fish. In the example above, the algorithm could output $[0, 4, 6, 10, 12]$. If Plucky cannot make it to their destination $t$ miles away, then your algorithm should return Stay Home.
• Your algorithm should run in time $O(n)$.

Because we don’t want to make this problem set too long, you do not have to prove that your algorithm is correct. However, it is great practice to prove that it is correct for yourself, and also it will make you more confident that your algorithm works!

[We are expecting: Pseudocode AND an English description of what it is doing. You do not need to justify the running time or the correctness.]
4. (6 pt.) [Minimum-maximum spanning trees.] Let $G$ be a connected weighted undirected graph. In class, we defined a minimum spanning tree of $G$ as a spanning tree $T$ of $G$ which minimizes the quantity

$$X = \sum_{e \in T} w_e,$$

where the sum is over all the edges in $T$, and $w_e$ is the weight of edge $e$. Define a “minimum-maximum spanning tree” to be a spanning tree that minimizes the quantity

$$Y = \max_{e \in T} w_e.$$

That is, a minimum-maximum spanning tree has the smallest maximum edge weight out of all possible spanning trees.

(a) (2 pt.) Give an example of a graph $G$ which has a minimum-maximum spanning tree $T$ so that $T$ is not a minimum spanning tree.

[We are expecting: An example, with an informal explanation of why it is an example.]

(b) (4 pt.) Prove that a minimum spanning tree in a connected weighted undirected graph $G$ is always a minimum-maximum spanning tree for $G$.

[HINT: Suppose toward a contradiction that $T$ is an MST but not a minimum-maximum spanning tree, and say that $T'$ is a minimum-maximum spanning tree.

How can you use $T'$ to modify $T$, to come up with a cheaper MST than $T$ (and hence a contradiction)? (Sub-hint: consider the heaviest edge in $T$).]

[We are expecting: A formal proof.]

(c) (NOT REQUIRED, WORTH 1 BONUS pt.) Give a deterministic $O(m)$ algorithm to find a minimum-maximum spanning tree in a connected weighted undirected graph $G$ with $n$ vertices and $m$ edges.

[We are expecting: To get the bonus point, you should submit pseudocode, along with an English description of the idea of the algorithm, and an informal justification of correctness and running time.]

Feedback

Thanks everyone for a great quarter! Instead of a feedback poll on the HW, please fill out the course evaluation on Axess when it becomes available! Your feedback is extremely important.