HW 10: Extra problems

Instructors: Har-Peled

CS/ECE 374A: Intro. Algorithms & Models of Computation, Fall 2024 Version: **1.0**

1 (100 PTS.) Easy peasy MST question. Or not.

Let $\mathsf{G} = (\llbracket n \rrbracket, \mathsf{E})$ be an undirected graph with n vertices and $m \geq n$ edges, and with distinct real weights on the edges. You are also given two vertices s and t in G. The *bottleneck* of a path π in G is $b(\pi) = \max_{e \in \pi} w(e)$. The ***bottleneck distance*** between s and t, denoted by $\mathsf{b}(s,t)$ is the minimum bottleneck of any path connecting s to t. The ***bridge*** β between s and t is the unique edge with the weight that realizes the bottleneck distance between s and t.

In addition, assume that for any vertex $v \in \llbracket n \rrbracket$, you are provided (in the input), the sorted list $L(v)$ of its adjacent edges in increasing order by weight (and you can access this list in constant time). Furthermore, let us assume that the bridge edge β has exactly T edges cheaper than it in G.

Describe an algorithm, as fast as possible, that computes the bottleneck shortest path from s to t, where the running time is only a function of T (and not of n or m). In particular, if T might be much smaller than n and m, and your algorithm should be faster than, say, $O(n)$.

(Hint: Verify that computing the bottleneck shortest path can be done by computing the MST, and then modify Prim's algorithm to get the desired algorithm. Do not worry about logarithmic factors in your running time. You can not use hashing in your solution.)

2 (100 PTS.) The best path (Fall 22).

Let $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ be a directed graph with n vertices and $m \geq n$ edges, and distinct positive real weights on the edges (here $w(e)$ denotes the weight of an edge $e \in \mathsf{E}$). For two sets X, Y let $X \oplus Y = (X \setminus Y) \cup (Y \setminus X)$ be their symmetric difference. For two paths π, σ in G, let e be the most expensive edge in $\mathsf{E}(\pi) \oplus \mathsf{E}(\sigma)$. If $e \in \pi$, then we write $\pi \succ \sigma$ (i.e., π is ***worse*** then σ). Clearly, this defines a natural ordering on the paths in G. The ***best*** path between s and t in G, is the unique path, such that all other paths (from s to t) are worse than it. Informally, the best path between s and t is the path minimizing the maximum weight edge on the path, and this property holds recursively on the two subpaths after we remove this edge.

Given vertices s and t, describe an algorithm, as fast as possible, that computes the best path from s to t.

Prove ***formally*** that the path your algorithm output is indeed the best path.

Partial credit would be given to efficient suboptimal algorithms.

(Hint: Think about the algorithm for the problem for the undirected case.)

3 (100 PTS.) Downwind from minus infinity (Fall 22).

Let G be a directed graph with n vertices and m edges, with distinct real weights (denoted by $w(\cdot)$) on the edges. A vertex $v \in \mathsf{V}(\mathsf{G})$ is ***sad*** if for any real number $\beta < 0$, there exists a walk π in G that ends in v, and $w(\pi) = \sum_{e \in \pi} w(e) < \beta$. Describe an algorithm (using or modifying algorithms seen in class), as fast as possible, that computes *all* the sad vertices of G.

4 Let $R_1, R_2, \ldots, R_n$ be a set of red intervals each of which is specified by its two end points. Let $B_1, B_2, \ldots, B_m$ be a set of blue intervals each of which is also specified by its two end points. You wish to find the smallest number of blue intervals that *cover* the red intervals. A blue interval B_j covers a red interval R_i if they contain the same point p on the line. All intervals are close in the sense that the end points are contained in the interval. Describe a greedy algorithm for this problem and prove its correctness.

5 We saw in lecture that Borouvka's algorithm for MST can be implemented in $O(m \log n)$ time where m is the number of edges and n is the number of nodes. We also saw that Prim's algorithm can be implemented in $O(m + n \log n)$ time. Obtain an algorithm for MST with running time $O(m \log \log n)$ by running Borouvka's algorithm for some number of steps and then switching to Prim's algorithm. This algorithm is better than either of the algorithms when $m = \Theta(n)$. Formalize the algorithm, specify the parameters and argue carefully about the implementation and running time details. Briefly justify the correctness of the algorithm assuming that the edge weights are unique.

6 Red street in the city Shampoo-Banana can be modeled as a straight line starting at 0. The street has n houses at locations $x_1, x_2, \ldots, x_n$ on the line. The local cable company wants to install some new fiber optic equipment at several locations such that every house is within distance r from one of the equipment locations. The city has granted permits to install the equipment, but only at some m locations on the street given y locations $y_1, y_2, \ldots, y_m$. For simplicity assume that all the x and y values are distinct. You can also assume that $x_1 < x_2 < \ldots < x_n$ and that $y_1 < y_2 < \ldots < y_m$.

- Describe a greedy algorithm that finds the minimum number of equipment locations that the cable company can build to satisfy the desired constraint that every house is within distance r from one of them. Your algorithm has to detect if a feasible solution does not exist. Prove the correctness of the algorithm. One way to do this by arguing that there is an optimum solution that agrees with the first choice of your greedy algorithm.
- **Not to submit:** The cable company has realized subsequently that not all locations are equal in terms of the cost of installing equipment. Asssume that c_j is the cost at location y_j. Describe a dynamic programming algorithm that minimizes the total cost of installing equipment under the same constraint as before. Do you see why a greedy algorithm may not work for this cost version?

7 Let $G = (V, E)$ be an edge-weighted undirected graph. We are interested in computing a minimum spanning tree T of G to find a cheapest subgraph that ensures connectivity. However, some of the nodes in G are unrealiable and may fail. If a node fails it can disconnect the tree T unless it is a leaf. Thus, you want to find a cheapest spanning tree in G in which all the unreliable nodes (which is a given subset $U \subset V$) are leaves. Describe an efficient for this problem. Note that your algorithm should also check wither a feasible spanning tree satisfying the given constraint exists in G.

8 Consider the language $L_{\text{OH}} = \{\langle M \rangle \mid M \text{ halts on at least one input string}\}$. Note that for $\langle M \rangle \in L_{\text{OH}}$, it is not necessary for M to *accept* any string; it is sufficient for it to *halt* on (and possibly rejects) some string. Prove that L_{OH} is undecidable.

9 Consider the following problem, called BoxDepth: Given a set of n axis-aligned rectangles in the plane, how big is the largest subset of these rectangles that contain a common point?

1. Describe a polynomial-time reduction from BOXDEPTH to MAXCLIQUE, and prove that your reduction is correct.
2. Describe and analyze a polynomial-time algorithm for BOXDEPTH. (**Hint:** Don't try to optimize the running time; $O(n^3)$ is good enough.)
3. Why don't these two results imply that P=NP?

10 This problem asks you to describe polynomial-time reductions between two closely related problems:

- SUBSETSUM: Given a set S of positive integers and a target integer T, is there a subset of S whose sum is T?
- PARTITION: Given a set S of positive integers, is there a way to partition S into two subsets S_1 and S_2 that have the same sum?

1. Describe a polynomial-time reduction from SUBSETSUM to PARTITION.
2. Describe a polynomial-time reduction from PARTITION to SUBSETSUM.

Don't forget to to prove that your reductions are correct.

11 Suppose you are given a graph $G = (V, E)$ where V represents a collection of people and an edge between two people indicates that they are friends. You wish to partition V into at most k non-overlapping groups $V_1, V_2, \ldots, V_k$ such that each group is very cohesive. One way to model cohesiveness is to insist that each pair of people in the same group should be friends; in other words, they should form a clique.

Prove that the following problem is NP-hard: Given an undirected graph G and an integer k, decide whether the vertices of G can be partitioned into k cliques.

12 Consider the following solitaire game. The puzzle consists of an $n \times m$ grid of squares, where each square may be empty, occupied by a red stone, or occupied by a blue stone. The goal of the puzzle is to remove some of the given stones so that the remaining stones satisfy two conditions: (1) every row contains at least one stone, and (2) no column contains stones of both colors. For some initial configurations of stones, reaching this goal is impossible.

Prove that it is NP-hard to determine, given an initial configuration of red and blue stones, whether this puzzle can be solved.

Solution: We show that this puzzle is NP-hard by describing a reduction from 3SAT. Let Φ be a 3CNF boolean formula with m variables and n clauses. We transform this formula into a puzzle configuration in polynomial time as follows. The size of the board is $n \times m$. The stones are placed as follows, for all indices i and j:

- If the variable x_j appears in the ith clause of Φ, we place a blue stone at (i, j).
- If the negated variable $\overline{x_j}$ appears in the ith clause of Φ, we place a red stone at (i, j).
- Otherwise, we leave cell (i, j) blank.

We claim that this puzzle has a solution if and only if Φ is satisfiable. This claim immediately implies that solving the puzzle is NP-hard. We prove our claim as follows:

$\Longrightarrow$ First, suppose Φ is satisfiable; consider an arbitrary satisfying assignment. For each index j, remove stones from column j according to the value assigned to x_j:

- If $x_j = \textsc{True}$, remove all red stones from column j.
- If $x_j = \textsc{False}$, remove all blue stones from column j.

In other words, remove precisely the stones that correspond to False literals. Because every variable appears in at least one clause, each column now contains stones of only one color (if any). On the other hand, each clause of Φ must contain at least one True literal, and thus each row still contains at least one stone. We conclude that the puzzle is satisfiable.

$\Longleftarrow$ On the other hand, suppose the puzzle is solvable; consider an arbitrary solution. For each index j, assign a value to x_j depending on the colors of stones left in column j:

- If column j contains blue stones, set $x_j = \textsc{True}$.
- If column j contains red stones, set $x_j = \textsc{False}$.
- If column j is empty, set x_j arbitrarily.

In other words, assign values to the variables so that the literals corresponding to the remaining stones are all True. Each row still has at least one stone, so each clause of Φ contains at least one True literal, so this assignment makes $\Phi = \textsc{True}$. We conclude that Φ is satisfiable.

This reduction clearly requires only polynomial time.

Rubric:[for all polynomial-time reductions] 10 points =

+ 3 points for the reduction itself
  - For an NP-hardness proof, the reduction must be from a known NP-hard problem. You can use any of the NP-hard problems listed in the lecture notes (except the one you are trying to prove NP-hard, of course).
+ 3 points for the "if" proof of correctness
+ 3 points for the "only if" proof of correctness
+ 1 point for writing "polynomial time"

- An incorrect polynomial-time reduction that still satisfies half of the correctness proof is worth at most 4/10.
- A reduction in the wrong direction is worth 0/10.