CS 170 Homework 5 (Optional)

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, explicitly write "none".

2 Adding Many Edges At Once

Given an undirected, weighted graph $G(V, E)$, consider the following algorithm to find the minimum spanning tree. This algorithm is similar to Prim's, except rather than growing out a spanning tree from one vertex, it tries to grow out the spanning tree from every vertex at the same time.

procedure FindMST($G(V, E)$)
  $T \leftarrow \emptyset$
  while T is not a spanning tree **do**
    Let $S_1, S_2 \dots S_k$ be the connected components of the graph with vertices V and edges T
    For each $i \in \{1, \dots, k\}$, let e_i be the minimum-weight edge with exactly one endpoint in S_i
    $T \leftarrow T \cup \{e_1, e_2, \dots e_k\}$
  return T

For example, at the start of the first iteration, every vertex is its own S_i.

For simplicity, in the following parts you may assume that no two edges in G have the same weight.

(a) Show that this algorithm finds a minimum spanning tree.

(b) Give a tight upper bound on the worst-case number of iterations of the while loop in one run of the algorithm. Justify your answer.

(c) Using your answer to the previous part, give an upper bound on the runtime of this algorithm.

3 Minimum ∞-Norm Cut

In the Minimum Infinity-Norm Cut problem, you are given a connected undirected graph $G = (V, E)$ with positive edge weights w_e, and you are asked to find a cut in the graph where the largest edge in the cut is as small as possible (note that there is no notion of source or target; any cut with at least one node on each side is valid).

Solve this problem in $O(|E|\log|V| + |V| + |E|)$ time. **Give a 3-part solution.**

Hint: Minimum Spanning Tree does not require edge weights to be positive.

4 Penguin Fishing

PNPenguin has just had his 170th birthday, and to celebrate this great milestone, the penguins are planning a feast to celebrate PNPenguin's birthday for this year and future years, and they need to catch fish to serve at this feast.

The PNPenguins live in Antarctica, which can be represented as N ice floes and M two-way bridges each connecting two ice floes. Some of the ice floes contain fish. The penguins can walk much faster than they can swim, and their goal is to station a certain number of penguins in various places, so that they can catch all of the fish. Hiring penguin fishermen is expensive, so the penguins want to catch all the fish using as few penguin fishermen as possible. Each penguin fisherman can only travel between ice floes connected by a bridge; however, they will be able to catch all the fish in the ice floes they can travel between.

Unfortunately, due to global warming, the penguins' habitat is slowly melting. For each of the next N years, an ice floe m_i will melt. When an ice floe melts, if the ice floe has fish, the fish at that ice floe will remain, but all of the bridges connecting that ice floe to other floes will disappear. Knowing this, write an efficient algorithm asymptotically faster than $O(N^2)$ to find the number of penguins p_i needed to catch all the fish in year i for each of the next N years.

Please provide a 3-part solution.

5 Preventing Conflict

A group of n guests shows up to a house for a party, but the host knows that m pairs of these guests are enemies (a guest can be enemies with multiple other guests). There are two rooms in the house, and the host wants to distribute guests among the rooms, breaking up as many pairs of enemies as possible. The guests are all waiting outside the house and are impatient to get in, so the host needs to assign them to the two rooms quickly, even if this means that it's not the best possible solution. Come up with a $O(n+m)$-time algorithm that breaks up at least half of the pairs of enemies.

Give a 3-part solution.

6 Twenty Questions

Your friend challenges you to a variant of the guessing game 20 questions. First, they pick some word $(w_1, w_2, ..., w_n)$ according to a known probability distribution $(p_1, p_2, ..., p_n)$, i.e. word w_i is chosen with probability p_i. Then, you ask yes/no questions until you are certain which word has been chosen. You can ask any yes/no question, meaning you can eliminate any subset S of the possible words with the question "Is the word in S?".

Define the cost of a guessing strategy as the expected number of queries it requires to determine the chosen word, and let an optimal strategy be one which minimizes cost. Design an $O(n \log n)$ algorithm to determine the cost of the optimal strategy.

Give a 3-part solution.

Note: *We are only considering deterministic guessing strategies in this question. Including randomized strategies doesn't change the answer, but it makes the proof of correctness more difficult.*

7 Rigged Tournament

Peter is in charge of organizing a football tournament with n teams. The tournament is a single-elimination tournament: if teams i and j play, the team that loses is out of the tournament and cannot play any more games. There are no ties.

Peter's shady friend Jeff has given him the following inside information: if teams i and j play, then they will score a combined total of $f(i, j) \geq 0$ points in that game and furthermore Jeff can rig the match so that the team of his choice wins. Peter wishes to find a tournament schedule which (1) maximizes the number of points scored in the tournament, and (2) makes his favorite team, team i^*, win the tournament. Give an efficient algorithm to solve this problem and provide its runtime; proof of correctness is not required.

Note: teams need not play an equal number of games in the tournament. For example, if the teams are $\{1, 2, 3, 4\}$, then a valid tournament schedule (where (i, j) means i plays j and i wins) is $[(1, 2), (1, 3), (4, 1)]$. Here, team 4 wins the tournament.

8 Sum of Products

This question guides you through writing a proof of correctness for a greedy algorithm. You have n computing jobs to perform, with job i requiring t_i units of CPU time to complete. You also have access to n machines that you can assign these jobs to. Since the machines are in high demand, you can only assign one job to any machine. The jth machine costs c_j dollars for each unit of CPU time it spends running a job, so assigning job i to machine j will cost you $t_i \cdot c_j$ dollars (each job takes the same amount of CPU time to complete, regardless of which machine is used). Your goal is to find an assignment of jobs to machines that minimizes the total cost.

Assume the jobs and machines are sorted and have distinct runtimes/costs, i.e. $t_1 > t_2 > \ldots > t_n$ and $c_1 > c_2 > \ldots > c_n$.

(a) Describe the assignment of jobs to machines minimizes the total cost (no proof necessary).

Hint: What machine should we assign the longest job to? What machine should we assign the second longest job to? It might help to solve a small example by hand first.

(b) Given an assignment of jobs to machines, consider the following modification: If there is a pair of jobs i, j such that job i is assigned to machine i', job j is assigned to machine j', and $t_i > t_j$ and $c_{i'} > c_{j'}$, instead assign job i to machine j' and job j to machine i'. Show that this modification decreases the total cost of an assignment.

(c) Use part b to show that the assignment you chose in part a has the minimum total cost (Hint: Show that for any assignment other than the one you chose in part a, you can apply the modification in part b. Conclude that the assignment you chose in part a is the optimal assignment.)

9 Counting Shortest Paths

Given an undirected unweighted graph G and a vertex s, let $p(v)$ be the number of distinct shortest paths from s to v. We will use the convention that $p(s) = 1$ in this problem. Give an $O(|V| + |E|)$-time algorithm to compute $p(v) \bmod 1700$ for all vertices. Only the main idea and runtime analysis are needed.

Hint: For any vertex v, how can we express $p(v)$ as a function of other $p(u)$?

Note: As a secondary question, you should ask yourself whether the runtime would remain the same if we were computing $p(v)$ rather than $p(v) \bmod 1700$.

