CSC373 Fall'20
Final Assessment
Date: December 18, 2020

Q1 [10 Points] Find the Single Attendee

There are $2n+1$ attendees at a party, which includes n couples and a single person. At the end of the party, all the attendees form a line in which each person stands next to their partner, except for the single person, who stands somewhere in the line. As an example, for $n = 3$, the seven attendees could be standing in the order $(A_1, A_2, B_1, B_2, C, D_1, D_2)$, where (A_1, A_2), (B_1, B_2), and (D_1, D_2) are couples and C is single.

Your job is to find the position of the single person (this would be 5 in the above example). But you don't know which ones are partners. All you can do is ask questions of the form "Are the i-th and j-th people in the line partners?" Design a divide-and-conquer algorithm for this problem which finds the position of the single person by asking $O(\log n)$ questions. Justify your answer.

Q2 [15 Points] Event Planner

There are n events, each takes one unit of time. Each event i will provide a profit of g_i dollars if it is started at or before time t_i, but will provide zero profit if it is not started by time t_i (so there is no point in scheduling event i unless it can be scheduled to start by time t_i). Here, $g_i, t_i \geq 0$ and t_i may NOT be an integer. An event can start as early as time 0 and no two events can be running simultaneously. The goal is to feasibly schedule a subset of the events to maximize the total profit.

(a) [2.5 Points] Prove that there exists an optimal schedule OPT in which every event that is scheduled is scheduled to start at an integral time. Note that in such a solution, each event i is either scheduled to start by time $\lfloor t_i \rfloor$ or not scheduled at all.

(b) [5 Points] Design an efficient greedy algorithm which only schedules events at integral start times. [Hint: Let $T = \max_i \lfloor t_i \rfloor$. Think about which event you would schedule to start at time T.]

(c) [5 Points] Prove that your algorithm always returns an optimal solution.

(d) [2.5 Points] Analyze the worst-case running time of your algorithm. Explicitly state the data structures that your algorithm uses.

Q3 [15 Points] Protect the Paintings Again

Recall the question about protecting paintings from midterm 1. A corridor of a museum is represented by the interval $[a, b]$ (with $a < b$) and contains valuable paintings. There are n guards stationed along the corridor. Guard i can protect the interval $[s_i, f_i]$, where $a \leq s_i \leq f_i \leq b$. We say that a subset of guards $P \subseteq \{1, \ldots, n\}$ is *acceptable* if the guards in P already collectively protect the entire corridor, i.e., $\cup_{i \in P}[s_i, f_i] = [a, b]$. Assume that the set of all guards $\{1, \ldots, n\}$ is acceptable, so there is at least one acceptable set.

In the midterm, we designed a greedy algorithm for finding an acceptable subset P of *minimum cardinality* $|P|$. Instead, suppose that each guard i has an associated non-negative cost c_i. Design a dynamic programming solution for finding an acceptable subset P with the smallest total cost $\sum_{i \in P} c_i$. For full credit, your solution must run in $O(n^2)$ time and space.

[Hint: Consider the set of all the "breakpoints": $\{a, b, s_1, f_1, s_2, f_2, \ldots, s_n, f_n\}$. Suppose the *distinct* breakpoints in the ascending order are $a = p_1 < p_2 < \ldots < p_m = b$ for some m. It may be useful to think of a subproblem where you want to cover the sub-interval $[p_1, p_j]$ using only some of the guards. Do not forget to bound the maximum number of distinct breakpoints m in terms of n.]

(a) [5 Points] Define an array storing the necessary information from subproblems. Clearly define what each entry means and how you would compute the desired solution given this array.

(b) [5 Points] Write a Bellman equation and briefly justify its correctness.

(c) [2.5 Points] In what order would you compute the entries in a bottom-up implementation?

(d) [2.5 Points] Analyze the worst-case running time and space complexity of your algorithm.

Q4 [15 Points] Divide the Workload

You are the CEO of a company which employs n workers to perform m tasks. Each worker i is supposed to work a total of w_i hours and each task j requires a total of t_j hours of work. Assume that $\sum_{i=1}^{n} w_i = \sum_{j=1}^{m} t_j$. The floor supervisor has come up with an ideal work schedule represented as matrix A, where row i represents worker i, column j represents task j, and $A_{i,j}$ is the number of hours worker i will spend on task j. Matrix A has the property that the sum along each row i is exactly w_i and the sum along each column j is exactly t_j.

There is just one problem. The floor supervisor has taken the liberty of using fractional values for $A_{i,j}$-s, forgetting the recent company policy that a worker must spend an *integral* number of hours on a task. Luckily, all the w_i-s and t_j-s are integral. Your goal is to prove that it is always possible to "round" matrix A into some matrix B while preserving the row and column sums (i.e. set each $B_{i,j}$ to be either $\lfloor A_{i,j} \rfloor$ or $\lceil A_{i,j} \rceil$ such that each row i of B still sums to w_i and each column j of B still sums to t_j). The example below shows such a rounding of a 3×3 matrix.

$$A = \left[\begin{array}{ccc|c} 2.6 & 0 & 0.4 & 3 \\ 0.8 & 2.9 & 1.3 & 5 \\ 1.6 & 0.1 & 5.3 & 7 \\ \hline 5 & 3 & 7 & \end{array}\right] \longrightarrow B = \left[\begin{array}{ccc|c} 2 & 0 & 1 & 3 \\ 1 & 3 & 1 & 5 \\ 2 & 0 & 5 & 7 \\ \hline 5 & 3 & 7 & \end{array}\right]$$

(a) [2.5 Points] Consider the matrix A' obtained by replacing each entry of A with its fractional part (e.g. replacing 1.3 with 0.3, 2.6 with 0.6, 0.1 with 0.1, etc). First, argue that A' must also have *integral* row and column sums. Next, argue that if A' can be rounded while preserving the row and column sums, then A can be as well.

(b) [10 Points] Note that each $A'_{i,j} \in [0, 1]$; hence, rounding it means setting it to either 0 or 1 (except, if $A'_{i,j} \in \{0, 1\}$ then the rounding must not change its value). Using network flow techniques,

show that A' can be rounded while preserving row and column sums. Justify your answer.

[Hint: Construct a network with integral edge capacities, use A' to construct a max flow with fractional flow values on edges, and then use the integrality property of the Ford-Fulkerson algorithm (i.e. that it finds a max flow in which each edge carries an integral amount of flow).]

(c) [2.5 Points] What is the worst-case running time of the naïve Ford-Fulkerson on your network?

Q5 [15 Points] Linear Programming

(a) [5 Points] Convert the following linear program to the standard form. You only need to write the final answer; no justification is needed.

$$\begin{array}{ll} \max & 3x + 5y + 2z \\ \text{s.t.} & 5y + 10z \leq 3 - 2x \\ & 2x \leq 2 - 3y - z \\ & x, y \geq 0, z \in \mathbb{R} \end{array}$$

(b) [5 Points] Write the dual of the linear program from part (a). You do *not* need to write this in the standard form and no justification is needed.

(c) [5 Points] Consider the optimization problem from part (a), but change the objective function to maximizing $f(x, y, z)$, where

$$f(x, y, z) = \begin{cases} 3x + 5y, & \text{if } z \geq 0, \\ 3x + 5y + 2z, & \text{if } z < 0. \end{cases}$$

Note that $f(x, y, z)$ is *not* linear, and hence, the new optimization problem is not linear as well. Nonetheless, show that it can be converted into an equivalent *linear* program. Provide this equivalent linear program in its standard form and justify the equivalence.

Q6 [20 Points] SAT

Recall that a CNF formula $\varphi = C_1 \wedge \ldots \wedge C_m$ is a conjunction of clauses, where each clause is a disjunction of (any number of) literals. Recall the NP-complete problem SAT.

SAT:
Input: A CNF formula φ.
Question: Does φ have a satisfying assignment?

Now, consider the following two variants of it.

TripleSAT:
Input: A CNF formula φ.
Question: Does φ have at least *three* different satisfying assignments?

TwoThirdsSAT:
Input: A CNF formula φ.

Question: Is there an assignment satisfying at least two-thirds ($2/3$) of the clauses of φ?

(a) [3 Points] Prove that TripleSAT is in NP.

(b) [7 Points] Prove that TripleSAT is NP-hard through a reduction from SAT.

(c) [3 Points] Prove that TwoThirdsSAT is in NP.

(d) [7 Points] Prove that TwoThirdsSAT is NP-hard through a reduction from SAT.

Q7 [15 Points] Sabotage!

There is an undirected graph $G = (V, E)$, where nodes in V are servers and edges in E are cables running between pairs of servers. A set of $k > 2$ servers $S = \{v_1, \ldots, v_k\} \subseteq V$ is trying to collaboratively solve a problem and you want to sabotage this!

Specifically, you want to remove a subset of edges $T \subseteq E$ such that all nodes in S become disconnected from one another (i.e. there is no path left between any two of them). You want $|T|$ to be as small as possible. Consider the following greedy algorithm.

Algorithm 1: Greedy-Sabotage

```
for i = 1, ..., k do
    Let E_i be the smallest subset of edges we need to remove to disconnect v_i from every
      other node in S. (It turns out that E_i can be computed efficiently, but do not worry
      about this.)
end
Remove the union of k − 1 smallest E_i-s (i.e. the union of all but the largest E_i).
```

(a) [5 Points] Prove that Greedy-Sabotage returns a feasible solution T (i.e. removing the set of edges T it returns will indeed disconnect every pair of nodes in S).

(b) [10 Points] Prove that Greedy-Sabotage achieves a $2 - 1/k$ approximation ratio. For partial credit, prove a slightly weaker approximation ratio of 2.

[Hint: Let T^* denote the optimal solution. For each $i \in \{1, \ldots, k\}$, let $V_i \subseteq V$ denote the set of nodes in the connected component containing v_i (but no other node in S) that remains after removing T^*. Can you relate the number of edges in T^* with one endpoint in V_i with $|E_i|$?]