COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 10

Announced on: Oct 10 (Thurs)

- This tutorial sheet contains problems in dynamic programming (DP). When presenting your solutions, please define the DP table clearly. Don't forget to write the base case. The correctness and running time arguments can be brief (1-2 sentences).
- Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. Recall that once the subproblem solutions stabilize in the Bellman-Ford algorithm (with $L_{k+1,v} = L_{k,v}$ for every destination v), they remain the same forevermore (with $L_{i,v} = L_{k,v}$ for all $i \geqslant k$ and $v \in V$). Is this also true on a per-vertex basis? That is, is it true that, whenever $L_{k+1,v} = L_{k,v}$ for some $k \geqslant 0$ and destination v, then $L_{i,v} = L_{k,v}$ for all $i \geqslant k$? Provide either a proof or a counterexample.

2. You are given a *strongly connected* directed graph $G = (V, E)$, which means that there is a directed path between every ordered pair of vertices. Each edge $e \in E$ of this graph is either colored red or blue. Design an algorithm to figure out whether there exists a cycle in G with *strictly more* red edges than blue edges. Clearly state the running time of your algorithm.

3. Consider the following parenthesization puzzles:

   a) *Parenthesize* $6 + 0 \cdot 6$ *to maximize the outcome.*

      Incorrect answer: $6 + (0 \cdot 6) = 6 + 0 = 6$.

      Correct answer: $(6 + 0) \cdot 6 = 6 \cdot 6 = 36$.

   b) *Parenthesize* $0.1 \cdot 0.1 + 0.1$ *to maximize the outcome.*

      Incorrect answer: $0.1 \cdot (0.1 + 0.1) = 0.1 \cdot 0.2 = 0.02$.

      Correct answer: $(0.1 \cdot 0.1) + 0.1 = 0.01 + 0.1 = 0.11$.

   The input to your algorithm is a sequence $x_0, o_0, x_1, o_1, \ldots, x_{n-1}, o_{n-1}, x_n$ of $n+1$ real numbers $x_0, x_1, \ldots, x_n$ and n operators $o_0, o_1, \ldots, o_{n-1}$. Each operator o_i is either addition ($+$) or multiplication ($\cdot$). Design a polynomial-time dynamic programming algorithm for finding the optimal (maximum-outcome) parenthesization of the given expression, and analyze the running time.

4. Consider the following *interval stabbing* problem: We are given a set S of n intervals, where

the i^{th} interval is denoted by $[\ell_i, r_i]$ and ℓ_i and r_i are integers. The goal is to determine if, for a given integer k, there exists a set of k integers $\{x_1, x_2, \ldots, x_k\}$ such that each interval in S contains at least one of the x_i's. Design a polynomial-time dynamic programming algorithm for this problem.

5. In this exercise, we will think about a voting problem involving the election of a committee. Let $C = \{c_1, c_2, \ldots, c_m\}$ denote the set of m candidates, and $V = \{v_1, v_2, \ldots, v_n\}$ denote the set of n voters. Each voter v_i has a strict and complete ranking R_i over the candidates in C.

   Consider the following voting rule f for selecting a committee of k candidates: Given any k-sized committee of candidates $W \subseteq C$, the score that voter v_i assigns to the committee W is equal to $m - j$ if voter v_i's *favorite* candidate in W is ranked at j^{th} position in its ranking R_i. The score of the committee W is the *minimum* of the scores it receives from all voters. The voting rule returns a committee with the highest score.

   For example, suppose there are three voters v_1, v_2, v_3 and five candidates $c_1, \ldots, c_5$. The rankings of the voters are:

$$v_1 : c_1 \succ c_2 \succ c_3 \succ c_4 \succ c_5,$$
$$v_2 : c_4 \succ c_2 \succ c_1 \succ c_5 \succ c_3,$$
$$v_3 : c_5 \succ c_4 \succ c_3 \succ c_2 \succ c_1.$$

   Then, the committee $\{c_2, c_5\}$ gets a score of 3 from voter v_1 because its favorite candidate in the committee, namely c_2, is ranked second. Likewise, the committee gets scores of 3 and 4 from voters v_2 and v_3, respectively, resulting in an overall score of $\min\{3, 3, 4\} = 3$.

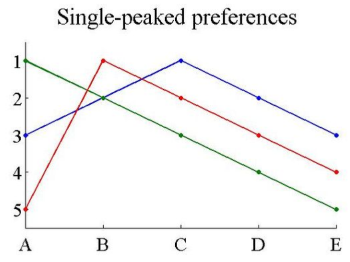

Figure 1: Example of single-peaked preferences. The axis is (A, B, C, D, E). The red voter corresponds to the ranking $B \succ C \succ D \succ E \succ A$. Thus, the "peak" of the red ranking is B. For any pair of candidates on the same side of the peak (say, C and D), the one closer to the peak is preferred over one that is further away. Image source: https://en.wikipedia.org/wiki/Single_peaked_preferences.

   We will assume that voter's preferences have the *single-peaked* property, defined as follows (see Figure 1): Consider an axis (or an ordering) σ of the candidates (not to be confused with the voters' preferences). Let R be any ranking of the candidates, and let $\texttt{top}(R)$ denote the top-ranked candidate in R. We say that R is single-peaked with respect to the axis σ if, for every pair of candidates c and c' that lie on the same side of $\texttt{top}(R)$ along the axis, if c is closer to $\texttt{top}(R)$, then c is preferred over c' in R. In other words, among the candidates on the same side of the "peak", the ones closer to the peak are preferred over those further away.

Show that when the rankings $R_1, \dots, R_n$ are single-peaked (with respect to a known axis σ), the output of the voting rule f can be computed in polynomial time.

Hint 1: You may use the following equivalent definition of single-peaked preferences: For any k, the set of top-k candidates in each ranking constitutes a connected segment with respect to the axis. For example, consider the red voter's ranking in Figure 1. Observe that the top-k prefixes are $\{B\}$, $\{B, C\}$, $\{B, C, D\}$, $\{B, C, D, E\}$ and $\{B, C, D, E, A\}$.

Hint 2: You may find it useful to refer to the interval stabbing problem.

6. ($\star$) How will your answer to the previous problem change if, instead of the minimum, the score of committee W is the *sum* of the scores it receives from all voters?

   Hint: Think about the "marginal" gain achieved by including a candidate in the committee relative to a voter's peak. The subproblems can be framed in terms of the size of the committee and the rightmost candidate in the committee.

7. ($\star$) Uh oh, another voting problem.

   This time, we will discuss a one-dimensional model of voting with a *continuum* of voters. Specifically, each point in the interval $[0, 1]$ denotes a *voter*. Additionally, there is a finite set of *candidates* $\mathcal{C} = \{c_1, c_2, \dots, c_m\}$ such that for every i, we have $c_i \in [0, 1]$. You may assume that c_i's are distinct. For any voter $v \in [0, 1]$, its *favorite* candidate, denoted by $\texttt{top}(v)$, is the candidate that is closest to it. That is, $\texttt{top}(v) := \arg\min_{c \in \mathcal{C}} |v - c|$.

   Unfortunately, in our problem, voters cannot vote directly for their favorite candidate; instead, they must delegate their vote to a *proxy*, who then votes for its own favorite candidate.[1] Specifically, a proxy is represented by a point in $[0, 1]$, and, in general, there can be multiple proxies. We will denote the set of proxies by $\mathcal{P} = \{p_1, p_2, \dots, p_k\}$. For any voter v, its *closest proxy* is given by $p_v := \arg\min_{p \in \mathcal{P}} |v - p|$. Each voter v delegates its vote to its closest proxy p_v, who then votes for its own favorite candidate given by $\texttt{top}(p_v) := \arg\min_{c \in \mathcal{C}} |p^v - c|$. Note that due to the delegation process, a voter's vote may get *distorted*, and the voter may end up voting for a non-favorite candidate, i.e., $\texttt{top}(p_v) \neq \texttt{top}(v)$ (see Figure 2). Our goal is to find an arrangement of the proxies so that this distortion is minimized.

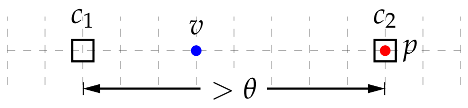

Figure 2: Failure of θ-representation. Voter v's favorite candidate is $\texttt{top}(v) = c_1$. However, its closest proxy is p, whose favorite candidate is $\texttt{top}(p_v) = c_2$. Since $|c_1 - c_2| > \theta$, the vote is significantly distorted.

   Formally, given a threshold $\theta \in [0, 1]$, an arrangement of proxies $\mathcal{P}$ is said to be *θ-representative* if, for every voter $v \in [0, 1]$, we have $|\texttt{top}(p_v) - \texttt{top}(v)| \leqslant \theta$. Your task is to design a

[1]See https://en.wikipedia.org/wiki/Proxy_voting and https://en.wikipedia.org/wiki/Liquid_democracy.

polynomial-time algorithm that, given as input the positions of the m candidates, a budget of k proxies (where $k \leqslant m$), and a distortion threshold θ, determines whether there exists a θ-representative proxy arrangement. If the answer is YES, your algorithm should also return the corresponding proxy arrangement.

For this problem, you can assume that the proxies are chosen from among the candidates, i.e., $\mathcal{P} \subseteq \mathcal{C}$. Thus, in particular, if proxy p lies at the same position as candidate c (as in Figure 2), then p obviously votes for candidate c.[2]

[2] If you can solve this problem *without* assuming that proxies are chosen from among the candidates, please get in touch with Rohit.