COMP 6651: Final Exam

Fall 2020

Submission through Moodle is due by December 11th at 22:00

Instructions:

- Solve the following problems, write up your solutions, digitize them into a single PDF file, and upload it on Moodle by the deadline. No late submissions will be accepted. Double-check your submission to make sure that no pages are missing.
- For any algorithms we have done in class, you do not need to provide the pseudocode. However, if you are modifying the algorithm, you do need to provide the pseudocode.
- Do not spend too much time on any one problem. Read through all of them first and attack them in the order that allows you to make the most progress.
- Answers must be **brief and precise**.
- **The** 20% **rule:** You will receive 20% of the points for any (sub)problem for which you write "I do not know how to answer this question." You will receive 10% if you leave a question blank. If instead you submit irrelevant or erroneous answers you will receive 0 points. You may receive partial credit for the work that is clearly "on the right track."

1. **(Computational)** [10 pts]

   Consider the following LP in standard form. Rewrite it in slack form and solve it using Simplex. For each step of Simplex write down the entering variable, the leaving variable, and the basic feasible solution.

$$\begin{array}{lrl} \text{Maximize} & \frac{2}{3}x_1 + x_2 & \\ \text{Subject to} & x_1 + x_2 & \leq 8 \\ & \frac{1}{2}x_1 + x_2 & \leq \frac{13}{2} \\ & x_1 & \leq 5 \\ & x_2 & \leq 6 \\ & x_1, x_2 & \geq 0 \end{array}$$

2. **(Flow Networks)** [10 pts]

   You are given a list of courses $c_1, c_2, \ldots, c_m$ and a student's list of requirements $r_1, \ldots, r_n$. Requirement $r_i = (k_i, S_i)$ consists of a natural number k_i and a subset of courses $S_i \subseteq \{c_1, \ldots, c_m\}$. The requirement states that the student must take exactly k_i different courses from the set S_i (assume that $|S_i| \geq k_i$). Each course can be used to satisfy at most one requirement. The student doesn't even know whether all the requirements can be satisfied or not. Help the student determine if all requirements can be satisfied, and if so, help the student design a schedule of which courses to take so that all the requirements are met.

   (a) State this problem as a flow network $G = (V, E), c, s, t$. State precisely: what V is, what E is, what capacities are assigned to edges $e \in E$.

   (b) How can you use maximum flow value to decide if it is possible to satisfy all requirements or not? How can you use maximum flow to design an actual schedule provided that it is possible to satisfy all requirements?

   Example: $m = 4$ and $n = 2$. Requirement $r_1 = (2, \{c_1, c_2, c_3\})$ whereas requirement $r_2 = (1, \{c_3, c_4\})$. Then it is possible to satisfy all the requirements by taking course c_3 to satisfy requirement r_2 and taking courses c_1, c_2 to satisfy requirement r_1 (note that since c_3 is counted towards r_2, it can't be counted towards r_1, since each course can be used to satisfy at most one requirement).

   End of Example

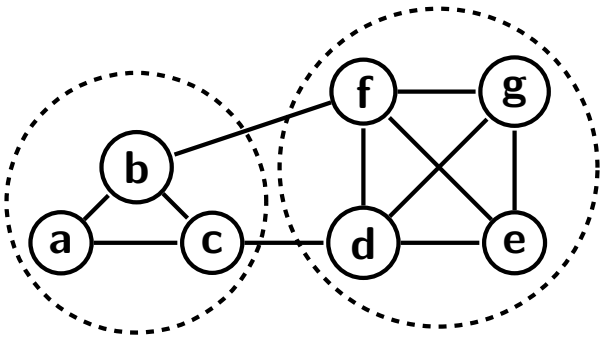
3. **(Complexity Theory)** [20 pts]

   (a) For each statement below state whether it is TRUE or FALSE (exclusive). For each answer provide a brief justification or a small counter-example. Guesses without justification or with incorrect justification receive 0 marks.

   - If $L \in \mathcal{NP}$ then $L^* \in \mathcal{NP}$.
   - If $L_1 \leq_p L_2$ and $L_2 \leq_p L_3$ then $L_1 \leq_p L_3$.

(b) Let $G = (V, E)$ be a simple undirected graph that represents a friendship network, i.e., vertices are people and there is an edge between two people if and only if they are friends (we assume friendship is reciprocal). A subset of people $S \subseteq V$ form a tightly-knit community, TKC for short, if all people in S are friends with each other. The TKC problem asks whether all people in a given friendship network G can be split into k disjoint TKCs (friendships across different TKCs are allowed). State the TKC problem as a language L_{TKC}. Prove that L_{TKC} is $\mathcal{NP}$-complete. Recall that $COL = \{\langle G, k\rangle : G \text{ has a valid coloring with at most } k \text{ colors}\}$ is $\mathcal{NP}$-complete.

Example: The following graph can be split into 2 TKCs (circled in the picture), but not into 1 TKC.



End of Example

4. **(Greedy)** [20 pts]

In the Interval Set Cover problem you are given a set of intervals $I_1, \ldots, I_n$ on the real line. Interval $I_j = [s_j, e_j)$ is a half-open interval. You are also given that together they cover the grand interval $[s, e) = \bigcup_{j=1}^{n} I_j$. However, some intervals might be redundant. We wish to find the smallest subset of intervals that cover the same grand interval $[s, e)$. In other words, we are looking for $S \subseteq \{1, \ldots, n\}$ such that $\bigcup_{j \in S} I_j = [s, e)$ and $|S|$ is as small as possible. Design an optimal and efficient greedy algorithm for this task.

(a) Explain your algorithm in plain English.

(b) Describe your algorithm in pseudocode.

(c) Formally prove correctness of your algorithm.

(d) State and justify the runtime of your algorithm.

Example: Let $I_1 = [4, 10)$, $I_2 = [1, 6)$, $I_3 = [1, 5)$ and $I_4 = [7, 9)$. Then $I_1 \cup I_2 \cup I_3 \cup I_4 = [1, 10)$. Thus, the grand interval is $[1, 10)$. This interval is covered by two input intervals $I_1 \cup I_3 = [1, 10)$, but not by any single input interval. Thus $S = \{1, 3\}$ is an optimal solution.

End of Example

5. **(Approximation Algorithms)** [20 pts]

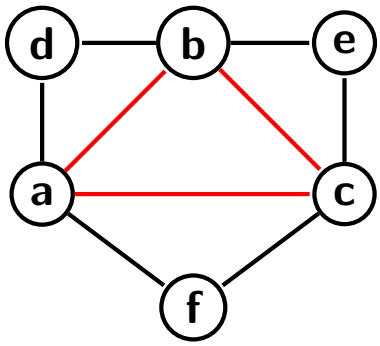
Let $G = (V, E)$ be a simple undirected graph. A cycle of length 3 in G is called a triangle. A subset of edges $C \subseteq E$ is an edge-cover of triangles if every triangle in G contains at least one edge from C. The Edge-Cover of Triangles problem asks to find an edge-cover of triangles of minimum size, i.e., minimize $|C|$. Design a simple and efficient 3-approximation algorithm for this problem. *Hint: borrow inspiration from the 2-approximation algorithm for Vertex Cover. In Vertex Cover you cover edges by vertices, in this problem you cover triangles by edges.*

(a) Describe your algorithm in plain English.

(b) Describe your algorithm in pseudocode.

(c) Prove that your algorithm achieves approximation ratio 3.

Example: In the following graph, $C = \{\{a, b\}, \{b, c\}, \{a, c\}\}$ (shown in red) is an edge-cover of all triangles, because each triangle contains at least one edge from C.



End of Example

6. **(Dynamic Programming)** [20 pts]

You are designing a roller coaster for an amusement park. You have access to an infinite supply of roller coaster segments of k different types. Each type is described by two parameters (e_i, m_i), where $e_i \in \mathbb{Z}_{\geq 0}$ is how much one segment of this type adds to excitement and $m_i \in \mathbb{Z}$ is how much one segment of this type adds to (or subtracts from, since we allow negative m_i) motion sickness. Your job is to decide which segments to place one after another to form a roller coaster consisting of n segments in total. You wish to maximize total excitement; however, you cannot just place one exciting segment after another, as this will make many riders motion sick. At the start of the roller coaster excitement is 0 and motion sickness is 0, and these two parameters accumulate additively. The goal is to maximize overall excitement while keeping the accumulated motion sickness between 0 and 100 (inclusive) during the entire ride. Design an efficient dynamic programming algorithm for this. *Hint: consider subproblems indexed by two parameters i and m, where $0 \leq i \leq n$ and $0 \leq m \leq 100$.*

(a) State this problem formally in the usual format **Input:** ..., **Output:** ....

(b) State the semantic array.

(c) State the computational array.

(d) Briefly justify why the above two arrays are equal.

(e) What running time will an algorithm based on the computational array have? Why?

Example 1: $k = 2$, $n = 10$. The first segment type has $e_1 = 100, m_1 = 50$; the second segment type has $e_2 = 0, m_2 = -10$.

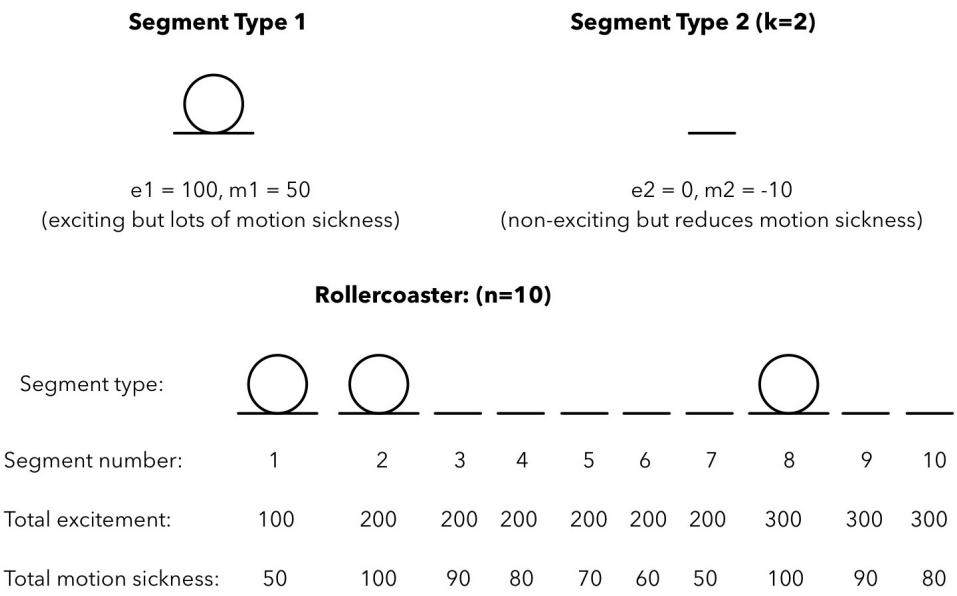


Figure 1: This figure illustrates Example 1 of Question 6.

We create a roller coaster consisting of the first two segments of type 1. The excitement after these two segments is 200, but motion sickness is 100. To reduce the motion sickness, the next 5 segments are of type 2 each. Thus, the excitement after the first 7 segments is 200, but motion sickness is 50 (reduced due to segments of type 2). This allows us to place another segment of type 1 as the 8^{th} segment, and to complete the roller coaster, we use segments of type 2 in the last two places. Thus, the overall excitement is 300 and motion sickness is maintained between 0 and 100 during the entire ride.

End of Example

Example 2: $k = 2$, $n = 10$. The first segment type has $e_1 = 100, m_1 = 11$; the second segment type has $e_2 = 10, m_2 = 1$.

Then we create a roller coaster consisting of the first 9 segments of type 1 and the last 10^{th} segment of type 2. Total excitement is $9e_1 + e_2 = 910$. Motion sickness after the first segment is $m_1 = 11$, after the second segment is $2m_1 = 22$, and so on. After the 9^{th} segment accumulated motion sickness is $9m_1 = 99$, so we cannot use type 1 for the

last segment, as it would push motion sickness beyond 100. Since we use segment of type 2 in the last place, the motion sickness after the last segment will be 100, which is within the tolerance level.

End of Example