Homework 5: Approximation & Randomized Algorithm, and basics of Game Theory
CS 6161: Design and Analysis of Algorithm (Fall'20)

Due: December 6'th, 6 pm

General Instructions The assignment is meant to be challenging. Feel free to discuss with fellow students, however please write up your solutions independently (e.g., start writing solutions after a few hours of any discussion) and acknowledge everyone you discussed the homework with on your writeup. The course materials are all on UVA Collab. You may refer to any materials covered in our class. However, any attempt to consult outside sources, on the Internet or otherwise, for solutions to any of these homework problems is *not* allowed.

Whenever a question asks you to design or construct an algorithm, please formally write the pseudo-code in an algorithm box with all the essential elements (the input, output, and operations); whenever a question asks you to "show" or "prove" a claim, please provide a formal mathematical proof. Unless specified otherwise, in every problem set, when we ask for an answer in asymptotic notation, we are asking either for a $\Theta(\cdot)$ result, or else the tightest $O(\cdot)$ result you can come up with.

Submission Instructions Please write your solutions in LaTeXusing the **provided template**, hand-written solutions will not be accepted. **You must submit your answer in PDF version via Gradescope.** You should have received a notice about signing up for Gradescope — if not, please let the TAs know as soon as possible. Hope you enjoy the homework!

Problem 1

In Lecture 18, we learned about the SAT problem. A 3-SAT formula is a clause with exactly 3 distinct literals, e.g., $C = x_1 \vee \overline{x_2} \vee x_3$. The Maximum 3-Satisfiability (`MAX-3SAT`) problem is: for any m 3-SAT formulas given by n boolean variables $X = \{x_1, \cdots, x_n\} \in \{0,1\}^n$, find a boolean variable assignment for X that maximizes the number of satisfied clauses. Since we know this problem is NP-hard, we can only hope to come up with some randomized approximation algorithms.

1. **[10 Points]** Please give a polynomial-time randomized algorithm that satisfies at least $\frac{7m}{8}$ fraction of clauses in expectation.

2. **[15 Points]** Given any $\delta \in (0,1)$, please design a polynomial-time randomized algorithm that satisfies at least $\frac{7m}{8}$ fraction of clauses with probability at least $1-\delta$. A running time analysis is required (the running time should depend on both m and δ).

   Hint: *To solve the second problem, consider running your first algorithm repeatedly.*

Problem 2

Recall that the NP-complete minimum set cover problem is the following: given a universe set $U = \{u_1, u_2, ..., u_n\}$, a collection $S = \{S_1, S_2, ..., S_m\}$, where each S_j is a subset of U for $1 \leq j \leq m$ and an integer $r \leq m$, you are asked to determine whether there exists r subsets from S that cover the entire universe U. The `Max-Coverage` is a closed related problem where you are given the same input — universe $U = \{u_1, u_2, ..., u_n\}$, a collection subsets $S = \{S_1, S_2, ..., S_m\}$ and an interger $r \leq m$ — but is asked to find r subsets $S_{i_1}, \cdots, S_{i_r} \in S$ to maximize the total elements they contain, i.e., maximizing $|S_{i_1} \cup S_{i_2} \cdots \cup S_{i_r}|$.

1. **[7 Points]** Prove that `Max-Coverage` is NP-hard.

2. **[13 Points]** Design a polynomial-time multiplicative $(1-1/e)$-approximate algorithm for `Max-Coverage`. **Hint:** *Think about where you see* $(1 - 1/e)$*-approximation in our lectures.*

Problem 3

In lecture 25, we learnt that finding the Nash equilibrium (NE) of a game is in general computationally hard. But in this problem, we will explore one possible way of finding NEs of a game, that is to iteratively eliminate the strictly dominated strategies — an action $a_i \in A_i$ is *strictly dominated* if $u_i(a_i, a_{-i}) < u_i(a'_i, a_{-i})$ for *some* $a'_i \in A_i$ and for all $a_{-i} \in A_{-i}$. That is, a_i can never be better than a'_i and thus is said to be "dominated" by a'_i.

We can write this method as an algorithm, which takes as input a set of n action sets $A_1, \ldots, A_n$ and a set of n utility functions $u_1, \ldots, u_n$, where each u_i is a function $u_i : A_1 \times \cdots \times A_n \to \mathbb{R}$.

Algorithm 1 Iterated Elimination of Strictly Dominated Strategies

IteratedElim$(A_1, \ldots, A_n, u_1, \ldots, u_n)$.
  Initialize a counter $t = 0$
  For each i, **Let** $B_i^t = A_i$
  while TRUE **do**
    For each i let:

    $\text{Dom}_t^i = \{a_i \in B_i^t$ such that there exists $a'_i \in B_i^t$ such that for all $s \in B_1^t \times \ldots \times B_n^t$, $u_i(a'_i, s_{-i}) > u_i(a_i, s_{-i})\}$

    if There exists an i such that $\text{Dom}_t^i \neq \emptyset$ **then**
      Let $B_i^{t+1} = B_i^t - \text{Dom}_t^i$ for all i
      Update $t = t + 1$
    else
      Break
    end if
  end while
  Return $B_1^t, \ldots, B_n^t$.

1. **[10 Points]** Now consider the following 2 player game, specified by the following payoff table:

	a_2	b_2	c_2
a_1	2,0	1,1	4,2
b_1	3,4	1,2	2,3
c_1	1,3	0,2	3,0

Player 1 picks actions from $\{a_1, b_1, c_1\}$, player 2 picks action from $\{a_2, b_2, c_2\}$. For example if player 1 takes action c_1 and player 2 takes action a_2, then player 1 receives utility 1, player 2 receives 3.

Which strategy(s) survive iterated elimination of strictly dominated strategies? Check whether they are pure Nash equilibria of the game?

2. **[15 Points]** Prove that if only a single strategy profiles survives iterated elimination of strictly dominated strategies, i.e., if at the end, for all i, $|B_i^t| = 1$ and $s_i \in B_i^t$ is the surviving action of player i, then s must be the *unique* pure strategy Nash equilibrium of the game.

   Note: *You will receive partial credit if you are only able to show it must be an NE.*

Problem 4

In this problem, we will explore another possible way of finding NEs of a game by iteratively picking the best response. Recall that, an action $a_i \in A_i$ is the *best response* to some $a_{-i} \in A_{-i}$, if $u_i(a_i, a_{-i}) \geq u_i(a_i', a_{-i})$ for any $a_i' \in A_i$ where $A_{-i} = \Pi_{j \neq i} A_j$.

The game class you will consider is called *singleton congestion game*, which is a simpler variant of the congestion game we saw in lecture 24. Formally, a singleton congestion game can be defined as follows:

- A set of n players $[n] = \{1, 2, \cdots, n\}$ and a set of m *resources* A
- Each player $i \in [n]$ has a set of actions $A_i \subseteq A$, which is a subset of resources that player i can access. Any action $a_i \in A$ is a single resource that suffices to satisfy i'th need.
- Each resource $j \in A$ has a "congestion" function $l_j : \{0, \ldots, n\} \to \mathbb{R} \geq 0$, such that $l_j(k)$ is the congestion resource j suffers when k players are using it. Naturally, we will assume l_j is non-decreasing in k for any j — i.e., more occupations, more congestion.
- For any *action profile* $a = (a_1, \cdots, a_n)$, $n_j(a) = |\{i : a_i = j\}|$ is the number of players using resource j. Player i's *cost* is then defined as the congestion at the resource it use, i.e., $c_i(a) = l_{a_i}(n_{a_i}(a))$. Each player would like to minimize their costs.

Though NE is hard to compute in general, it turns out that for the above specially structured singleton congestion game, a pure Nash equilibrium is guaranteed to exist and can be computed in $poly(m, n)$ time.

1. **[15 Points]** The best response dynamic algorithm is defined below. Show that if this algorithm halts in a singleton congestion game, it returns a *pure* strategy Nash equilibrium.

Algorithm 2 Best Response Dynamics

Initialize $a = (a_1, \ldots, a_n)$ to be an arbitrary action profile.
while There exists i such that $a_i \notin \arg\min_{\mathbf{b} \in A_i} c_i(\mathbf{b}, a_{-i})$ **do**
  Set $a_i = \arg\min_{\mathbf{b} \in A_i} c_i(\mathbf{b}, a_{-i})$
end while
Halt and return a.

2. **[15 Points]** Notably, all singleton congestion games have at least one *pure* strategy Nash equilibrium. Prove this result by showing that the best response dynamics algorithm always halt in a congestion game.

   Hint: *Define a "potential" function $\phi : A \to \mathbb{R}$, s.t. $\phi(a) = \sum_{j=1}^{m} \sum_{k=1}^{n_j(a)} \ell_j(k)$ and then show how the best response from any player changes this potential function. What does this observation imply?*

Problem 5

[3 Bonus Points] To receive this bonus credit, you must submit your homework with the provided template (e.g., end your solution of each problem with `\newpage`) and properly assign all problems to their respective pages on Gradescope.