CS 473: Algorithms, Spring 2018
HW 3 (due Wednesday, February 14th at 8pm)

This homework contains three problems. **Read the instructions for submitting homework on the course webpage**.

Collaboration Policy: For this home work, each student can work in a group with up to three members. Only one solution for each group needs to be submitted. Follow the submission instructions carefully.

1. You are part of a team that competes in a challenge to design a robot with the purpose of traversing a rectangular grid of M rows and N columns. Your team positions the robot at $(1, 1)$, which is the top-left cell. The goal is for the robot to reach (M, N), i.e. the bottom-right cell. In a single step, the robot can move only to the cells to its immediate right and bottom directions. More specifically, if the robot is at position (i, j), then it can move either to position $(i + 1, j)$ or to $(i, j + 1)$, provided that both positions are inside the grid.

   The challenge is that there are P obstacles in random positions on the grid, through which the robot cannot pass. Given the positions of the blocked cells, your task is to count all the number of paths that the robot can take to move from $(1, 1)$ to (M, N).

2. Your team lost in the above challenge, thus you resort to stop doing algorithms and find a real job. You are hired by a lottery company with the purpose of navigating n cities and collecting all the lottery tickets from the citizens of each city. However, being a true fan of algorithms, you want to traverse the cities in the most efficient manner.

   It turns out that some of the cities have roads between them and some do not. Also, some cities have lottery tickets for you to collect, while some other cities do not. Therefore, the input to your problem can be represented as an undirected weighted graph $G = (V, E)$, where each node is a city and each edge $\{u, v\}$ represents the road between cities u and v. The weight of this edge is equal to the distance between u and v. Furthermore, exactly k cities have lotteries to be collected, and these are given by set $S \subset V$ where $|S| = k$. Consider k being a **constant**. Since you live in city $s \notin S$, starting at s, your goal is to design an algorithm that computes the minimum total distance you have to travel to collect all the lottery tickets and return at s.

3. Consider the complete graph K_n on a set V of n vertices. A tournament $T = (V, E)$ is an orientation of the edges of K_n. Thus, for every two distinct nodes $u, v \in V$, either $(u, v) \in E$, or $(v, u) \in E$, but not both. The name "tournament" is natural as one can think of the set V as the set of players playing a round-robin tournament, that is every player plays against every other player exactly once, and there is a directed edge (u, v) if and only if u beats v in the single match.

   We say that T has the property **Beats-k** if for every set of k players, there exists another player that beats them all. That is, for every subset $S \subset V$ with $|S| = k$, there exists a node $u \in V \setminus S$ such that there is an edge from u to v for each $v \in S$. Note that node u may be different for different S. For example, a directed triangle T_3 with $V = \{1, 2, 3\}$ and $E = \{(1, 2), (2, 3), (3, 1)\}$ has **Beats-1**.

(a) Starting from K_n suppose we construct T by orienting each edge $\{u, v\}$ either (u, v) with probability $\frac{1}{2}$ or (v, u) with probability $\frac{1}{2}$.

Give a non-trivial lower for the probability that T satisfies the **Beats-k** property for a fixed k?

(b) Given k, show that there are tournaments with $O(2^k k^2)$ many players that satisfy **Beats-k** property? [**Hint:** Use solution of the first part.]

Optional Just to see the power of randomness try proving the second part without using a random graph.

The remaining problems are for self study. Do *NOT* submit for grading.

- You are playing a game with your friend. You both enter a dungeon with the purpose of collecting gold. The dungeon is represented as a rectangular grid of N rows and M columns, and each cell (i, j) except for the cell $(1, 1)$ (the top-left one) has a specific number of gold coins $c(i, j)$ on it. You and your friend both enter the dungeon through cell $(1, 1)$, and your goal is to reach the cell (N, M).

  You start playing a bizarre game. You and your friend always move together, to the same cells, but you take turns deciding at which cell to move, given that you can only move to the right and/or bottom. More specifically, there exists a fixed K such that, if you are at cell (i, j) and it is your turn to select the next cell to move, you can move to any cell that is at most K cells to the right and K cells to the bottom, i.e. to any cell in the rectangular grid created by (i, j) and $(i + K, j + K)$, with the restriction that you have to move to a different cell each time. When you move to a new cell, whoever's turn it was to decide the move takes all the gold of that cell.

  Suppose you have a map of the dungeon with the amount of gold on each cell, and you start the game by being the first to decide where you will move. Assuming that your best friend always plays optimally, find a path through the dungeon that will get you the maximum number of gold possible. Furthermore, if there are more than one paths that get you the maximum number of gold, pick the one that minimizes the gold that your friend will obtain.

  Hint: You might have to devise a way to combine two recurrence relations in order to solve the problem.

- Show that quick select takes $O(n)$ time in expectation. See if you can prove that the number of comparisons is at most $4n$ for an array of size n.

- Solve the first problem on shortest paths via dynamic programming in the file below. `https://courses.engr.illinois.edu/cs374/fa2015/labs/lab16-shortest-path-dynamic-prog.pdf`.

- Solve the first problem on shortest walks in the file below. `https://courses.engr.illinois.edu/cs374/fa2015/homework/hw8.pdf`.

- Read Jeff's notes on randomized quick sort but in particular the nuts and bolts problem. Solve some of the basic problems on probability and randomized algorithms at the back of the notes. `http://jeffe.cs.illinois.edu/teaching/algorithms/notes/09-nutsbolts.pdf`.