CS180 Midterm Exam

1. (20 pt) Determine whether the following statements are **true or false**.

   (a) (5 pt) $2^n = \Theta(3^n)$

   (b) (5 pt) Suppose G is a connected, undirected graph. If removing an edge disconnects the graph, then the edge is a tree edge in the DFS tree.

   (c) (5 pt) Suppose G is a DAG and s is the first node in the topological ordering, t is the last node in the topological ordering, then there is always a path from s to t.

   (d) (5 pt) For any undirected connected graph G, there exists at least two nodes such that removing the nodes won't disconnect the graph.

2. (20 pt) As we learned in the class, a min-heap is useful when we are interested in the minimum value of set of numbers. Now, instead of the min, we are interesting in the K-th smallest value.

   (a) (10 pt) Please design a data structure to allow the following two operations: `push` (insert a new number) and `find_Kmin` (return the value of the K-th smallest number without removing it). Both operations should be done in $O(\log K)$ time.

   (b) (10 pt) In addition to `push` and `find_Kmin`, now we also want to support the `pop` operation to return and remove the K-th smallest element. Please design a data structure to support these three operations, and each operation should take $O(\log n)$ time with n being the size of the current set.

3. (20 pt) For a stable marriage problem with n men and n women, let m_1, m_2 be two of the men, w_1, w_2 be two of the women. Suppose m_1's preference list is $w_1 > w_2 > \ldots$; m_2's preference list is $w_2 > w_1 > \ldots$; w_1's preference list is $m_2 > m_1 > \ldots$; w_2's preference list is $m_1 > m_2 > \ldots$ (we only know the favorite and the second favorite of each of these four persons).

   (a) (10 pt) Show that in every stable matching, m_1, m_2 are matched to w_1, w_2 (which means we either have $(w_1, m_1), (w_2, m_2)$ or $(w_1, m_2), (w_2, m_1)$ in a stable matching).

   (b) (10 pt) For any even n, construct an instance of stable matching problem (i.e., provide a set of preferences) with at least $2^{n/2}$ stable matchings. Justify your claim.

4. (20 pt) Indiana Jones is stuck in the Emperor's Tomb and needs to find a way out. In front of him, there are n consecutive doors, each door is behind the previous, and he needs to open all the doors to get out. In the control room, there are n switches where each switch connects to a different door but he doesn't know which switch controls which door. Each switch can either be in an up or down position. but unfortunately, for some doors the "up" position will open the door while for others the "down" position will open the door. He needs to place all the n switches in the correct configuration to open all the doors. In order to find the correct configuration, each time Indiana Jones can walk to the control room, set the switches to any configuration, and walk back to check which is the first closed door (since the doors are consecutive, he cannot observe the status of doors behind the first closed one). Design an algorithm to find the correct configuration to open all the doors within $O(n \log n)$ trials.

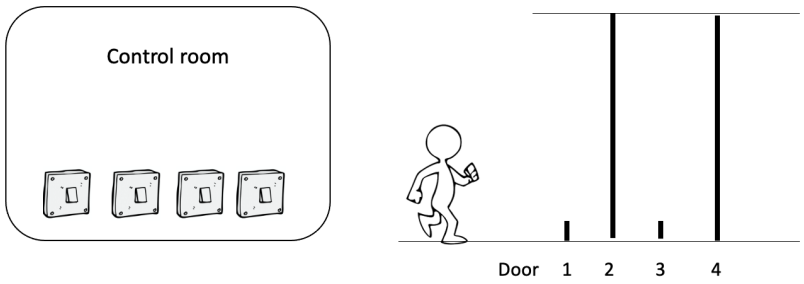

Figure 1: Illustration of Problem 3. In this case there are $n = 4$ doors, and 4 switches controlling those doors. He can only see the state of doors 1 and 2, not 3 or 4

5. (20 pt) There's a 1D segment and you want to travel from west-most point (s) to the east-most point (t). There are n teleporters on this 1D segment and each teleporter has two endpoints. Whenever you reach one endpoint, it will teleport you to another endpoint (it may transport you from east to west or west to east, depending on which endpoint you reach). All the endpoints are located between s and t and none of the endpoints are located at the same position. We assume you must always move **eastward** on the segment.

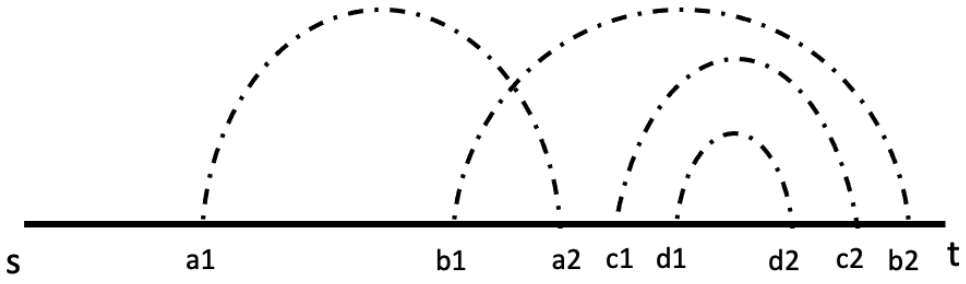

Figure 2: An example for Problem 5. We are given an input with 4 teleporters with endpoints a_1, a_2; b_1, b_2; c_1, c_2, d_1, d_2. Starting from s, you should keep moving to the right, so the path will be s (walk to) a_1 (teleport to) a_2 (walk to) c_1 (teleport to) c_2 (walk to) b_2 (teleport to) b_1 (walk to) a_2 (teleport to) a_1 (walk to) b_1 (teleport to) b_2 (walk to) t, and the path has reward t since you are teleported 5 times.

(a) Prove that no matter how those teleporters are placed, you will always reach the t. (Hint: you can view this as a graph problem, where each segment is a node, and each one has an edge pointing to the next segment)

(b) Now you are allowed to add another set of K teleporters (each with two endpoints). How to place those endpoints to *maximize* the number of times you get teleported in your trip from left-most point to right-most point? Given the input of the locations of existing teleporters and the number K, design an algorithm to solve this problem in $O(n)$