CSE 101 Spring 2024 Homework 3 Due: Wednesday 4/24 at 11:59pm

1. You are given a schedule for n buses in the city. The schedule is given to you as a 2-dimensional array B of dimensions $n \times k$. Each bus has k stops and each entry of the array is an ordered pair $B[i,j] = (S_{i,j}, t_{i,j})$ which means that the j^{th} stop that bus i makes is at station $S_{i,j}$ at time $t_{i,j}$ (measured in minutes counted from the beginning of the day.) Each row is ordered by times (i.e., $t_{i,1} < t_{i,2} < \ldots < t_{i,k}$.

   You can transfer from bus a to bus b at stop X if there exists entries $B[a,j] = (X, t_{a,j})$ and $B[b,j'] = (X, t_{b,j'})$ and $t_{a,j} < t_{b,j'}$.

   (a) Given bus stations X and Y and a starting time T, design an algorithm that returns the earliest time you can reach station Y from station X starting any time after time T using a sequence of transfers.

   (b) Given bus stations X and Y and a starting time T, design an algorithm that returns the fewest number of transfers necessary to reach station Y from station X starting any time after time T using a sequence of transfers.

2. For some non-negative integer d, we say that a d-regular graph is an undirected graph such that each vertex has a degree of d.

   Suppose you are given access to a *connected* d-regular graph $G = (V, E)$ and two vertices $s \in V$, $t \in V$. You wish to find the shortest path from s to t. (We can assume that the graph is very large and that we want to avoid having to look at the entire graph. So, for any vertex, you can look at its list of neighbors without having to look at the entire graph.)

   We can alter BFS so that it takes both s and t as inputs and when we reach t, we can stop so that we don't have to continue to explore unnecessary vertices.

```
BFS2(G, s, t):
(1)  for all u ∈ V:
(2)    dist(u) = ∞
(3)  dist(s) = 0
(4)  Q = [s]
(5)  while Q is not empty:
(6)    u = eject(Q)
(7)    for all edges (u, v) ∈ E:
(8)      if dist(v) = ∞:
(9)        inject(Q, v)
(10)       dist(v) = dist(u) + 1
(11)       if v == t:
(12)         break loop
```

   (Notice that `BFS2` is almost identical to `BFS` except for lines 11 and 12.)

   (a) i. Give an argument about why `BFS2` will correctly assign $dist(t)$ to the length of the shortest path from s to t.
   (3 points)

   ii. Assuming that ℓ is the length of the shortest path from s to t, what is the worst-case runtime of `BFS2` in terms of d and ℓ? (your answer should be in big-O notation in terms of d and ℓ.)
   (4 points)
   (Note: You can use the fact that for every possible distance $i = 1...L$, at some point in BFS, the queue will contain exactly all of the vertices that are distance i away from s.)

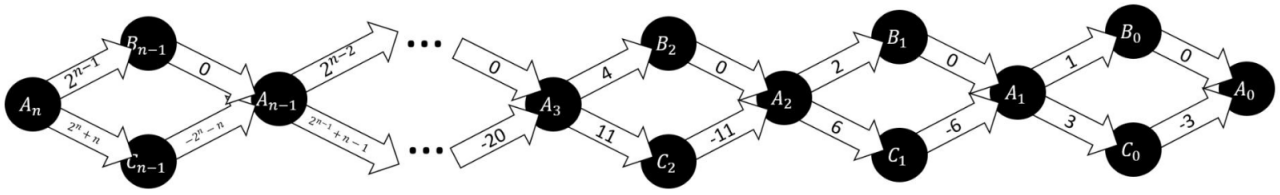
   (b) Design an algorithm that finds the shortest path from s to t in G that runs in time $O(d^{\ell/2})$.

i. High-level algorithm description (No correctness proof necessary.) (4 points)

ii. Runtime analysis. (4 points)

3. *[The purpose of this problem is to show that Dijkstra's algorithm could be exponential time if run on a graph with negative edge weights.]*

   For all integers $n \geq 0$, consider the graph $H(n)$ pictured below:



   That is, A_0 has no outgoing neighbors and the adjacency list for each other vertex is:

   - $A_k : (B_{k-1}, 2^{k-1}), (C_{k-1}, 2^k + k)$
   - $B_k : (A_k, 0)$
   - $C_k : (A_k, -2^k - k)$

   (a) Let $maxdist(v)$ be the maximum dist value that $dist(v)$ is set to (besides ∞) after running Dijkstra's on $H(n)$ starting at A_n.
   Prove by induction that $maxdist(v) \leq 2^n + n$ for all $v \in H(n)$.

   (b) Prove by induction that after running Dijkstra's on $H(n)$, starting at A_n, the vertex A_0 has been ejected from the priority queue 2^n times. (You can use the previous part.)

   (c) Give a lower bound for the runtime of Dijkstra's on this graph in Ω notation in terms of the number of vertices N.

4. Suppose Dijkstra's algorithm is run on the following graph, starting at node A.

   (a) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm. (5 points)

   (b) Draw the shortest path tree. (5 points)