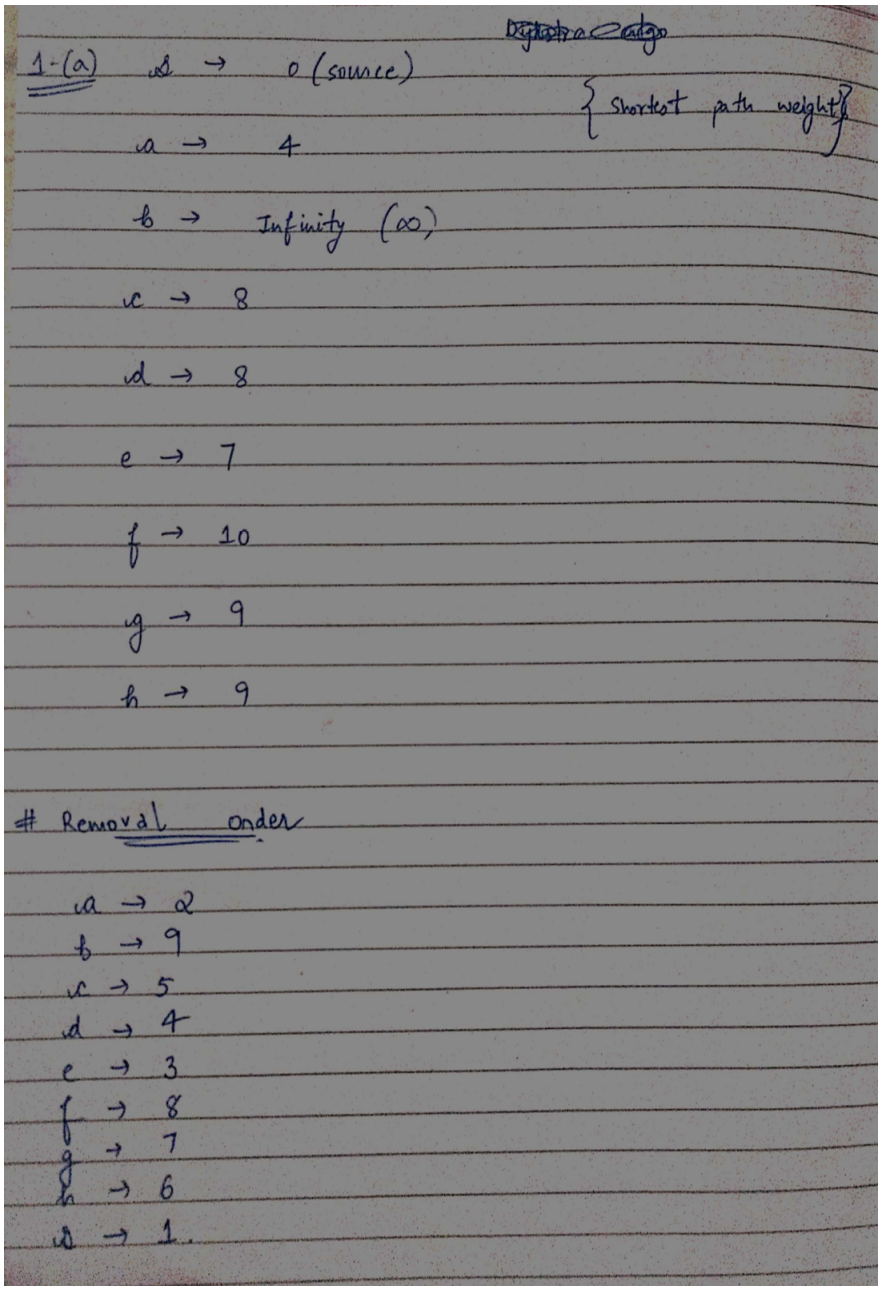
Problem Set 6

Name: Neeraj Pandey

Collaborators: None, None

Problem 6-1.

(a) Solution attached in the photo below:



(b) In the above solution, we noticed that s (source) is visited at the end and if we have to change the (f, d) value to -3, the graph will be $-Infinity$ at all sequences. Therefore, b is the only node that won't be effected. Rest all the nodes will be effected.

Problem 6-2.

(a) The basic idea here is to use Dijkstra algorithm for shortest path. If k is small, we can have a tuple $< V, i >$ as to the cheapest path from s to V, taking exactly i red vertices and maximum k red vertices, and the states would be updated accordingly.
Detailed Algorithm: We start from s (source node), while keeping track of all the red vertices. While performing the Dijkstra, each time it visits the red vertex, it adds the counter element of that path and checks whether $counter < k$. If the condition is satisfied, the program proceeds to the next path. Recursively doing this, will give us the shortest (cheapest) path between two nodes. The program will run in a time complexity of $O(ElogV)$ for E edges and V vertices.

Problem 6-3.

(a) The basic idea here is to apply Johnson's algorithm to find the shortest distance among all shortest paths, as this is a sparse cyclic graph here with few edges which will use Bellman Ford algorithm to relax all the weights including negative ones. To perform this, we have to create a graph with V vertices (game levels) and E edges (as warp pipes) with each edge having a weight t_{ab}. Then we relax the weights using Bellman Ford that will eliminate all the negative weights. After relaxing the weights, as there is no negative weight left, we can perform Dijkstra algorithm to find the shortest path (cheapest) for every elevel that Dario plays and using fibonacci heaps to finish Dijkstra faster at each node in $O(VlogV + E)$ time. The condition says to strictly move back in time, as there is no negative weight then Dijkstra will prioritise the lower weight to find the levels where Dari can go back in time. The overall time complexity to run this algorithm will be $O(V^2logV + VE)$ time as Bellman Ford algorithm is called once ($O(VE)$) to relax the weights and Dijkstra is performed V times and using a Fibonacci heap, each of these iterations takes $O(VlogV + E)$

Problem 6-4.

(a) The basic idea here is to use Dijkstra Algorithm.
Detailed Algorithm: Given is a 2-way road where every road can be considered as an edge (total E edges) and every intersection as the vertex (total V vertices) of a directed cyclic graph. First step is to setup a graph with with E edges and V vertices; G = (V,E). Now the condition is to travel from the vertex X to vertex Y with at least 12 eggs. So, this a weighted graph with only positive weights (number of eggs), as number of eggs cannot be negative. We assume that Wester starts with k eggs in his bag initially, so he has to find the shortest (cheapest) path from the vertex X to vertex Y while taking into account that he has to throw eggs at every refused house he crossed on his way. Here, the number of edges n the graph will represent the number of eggs Wester has to throw at houses. To overcome this problem, applying Dijkstra algorithm to find the shortest (cheapest) path will help Wester to minimize the the number of eggs he has to throw at houses. While moving on the path from source, at the vertex X, Wester makes an egg counter to calculte the number of eggs that are left in the bag. To satisfy the condition, Wester should have number of eggs atleast greater than or equal to 12. In order to do this, the counter element should be always more than 12. At each vertex(intersection of roads), we will use the Dijkstra algorithm to look at the weights (number of eggs that Wester has to throw) and if the difference of the egg counter element and the weight is less than 12 and we proceedd to the next vertex. If it's not true, we perform graph traversal on it and reaching a grocery store will make the counter element to 12 again. Thus recursively performing the steps will provide us the shortest path to Y vertex from X and the final number of eggs available with Wester is equal to the counter element. As this algorithm performs a dijkstra approach, the time complexity will be equal to $O(ElogV)$ for E edges and V vertices.

Problem 6-5.

(a) Here, the basic idea is to perform Dijkstra algorithm so that Prakhar can find the shortest path to his favourite clearing from his current position while taking account two criteria points that he should not visit where groundhogs hide and the second that he should not visit visit the edges or vertices the groundhog fall under his safety distance. While performing dijkstra, Prakhar will visit each node an check if the given conditions are satisfied, if they are satisfied, he will procedd further and eventually find out the shortest distance. If the condition is not satisfied and a groundhog is present at a vertex, he traverse the graph and moves to the next path. Recursively performing this will help Prakhar find the shortest path. Performing this algorithm will take $O(ElogV)$ for E edges and V vertices.