CS 473 – Algorithms I
Assignment 2

Muhammad Arham Khan
21701848
27/12/2020

a) This was a typical example of using the Djikstra shortest path algorithm since we needed to find the path with the least amount of weakness. Hence, I first implemented structures for the Vertices, Edges and then the Distance struct as well which was stored in the Heap array.
I stored the vertices in a dynamic array of Vertice struct and stored all the edges in a dynamic array of Edge struct. Furthermore, to maintain an adjacency list, each vertex struct also had a vertex pointer list that pointed to it's neighboring vertices.

   Instead of going with a separate heap implementation, I created the heap insert, extract and decreaseKey functions in the same file with operations on a 'heap' array. So, the algorithm works by first starting at the source vertex (vertex 0) and then adding it to the heap, and then iterating through the heap elements until it is empty, trying to find distances between the source node and the current retrieved node and replacing the distance value of the retrieved node only if it is smaller than the previous value. Hence, we also maintain a distances array which is first initialized to a MAX value and then replaced during comparative iterations. One each replacement, the distance array's value in incremented with the current node's weight.

   Hence, when the algorithm terminates, the distance array contains the minimum weakness of each path from the source to target vertex. This is then output to the target file. Since the program takes O(log n) time in inserting the values to the heap and then iterated through them O(n) times so the runtime complexity of this algorithm is O(nlogn). But Since it also iterates through all the edges of the vertices and tries to find the smallest path, the final runtime complexity of the algorithm is O(E + VlogV) where E is EdgeCount and V is VerticeCount.

b) Part b involved changing two minor things in the first part's code. In the first place, the min-heap had to be replaced with a max-heap so that the edges with the highest probabilities could be retrieved first. Secondly, the initial values in the probabilities array could not be 0, they had to be replaced with 1 to enable multiplication. Finally, instead of adding the weakness values to add to the probabilities array, I multiplied the values instead since that's how probabilities are calculated.

c) The script and the code in part a returns values that are the same as the answers in b because in the first place, the value provided is assumed to be the weakness and hence, it needs to be exponent inverted through -log so that the max values are returned instead of the min values from the heap. This way all the values populated as the results are actually the inverts of the max values. The – sign is necessary since the algorithm uses the addition operation on weights instead of the multiplication one and -log is basically the log of the 1/x. Finally in the response we take the exponent because although the values retrieved are correct considering the provided data, they need to converted back by using a negative exponent of 2 so that we have the correct values (same as the ones retrieved in b). Hence, this inversion of input values and the conversion back to normal works using code in part a.