Question 4 [25 points] Design an O(k|E|) time algorithm to find the shortest path between two vertices a and b in a directed graph G = (V, E) with weighted (can be negative) edges, where it is guaranteed that the shortest path between any two vertices has at most k edges.

For this question, we can take advantage of be Bellman's algorithm. In Bellman's algorithm, the longest possible path without a cycle is |V| - 1 edges. As a result, the algorithm should do |v| - 1 iterations. (This is proved in lecture notes and the overall running time for Bellman's algorithm is O(|V| * |E|) edges since there are |V| -1 iterations and |E| edges to scan for each iteration.

If we know that the shortest path between any two vertices has at most k edges, we can do k iterations instead of |V| -1. The reason is that the longest possible path without a cycle is k.

Below is the steps for my algorithm:

1. First, set all distance of vertices to infinity from a where a is initialized to 0.
2. Repeat the following steps **K** times
    1. For each edge (u,v) in the graph:
    2. Update the distance for each vertex if it can be reached in a shorter distance through that edge.
3. Check negative-weight cycles : for each edge(u,v) :

    if d[v] > d[u] + weight(u,v) → a negative-weight cycle exists.

if after K passes, d[v] fails to converge, there exist a negative cycle.

Therefore, the running time for this algorithm is O(K|E|) the reason is that the are K iterations and for each iteration, |E| edges will be scanned.