10 Algorithms (RKH/DJW)

(a) Let \texttt{dijkstra\_path}(g, a, b) be an implementation of Dijkstra’s shortest path algorithm that returns the shortest path from node \texttt{a} to node \texttt{b} in a graph \texttt{g}. Prove that the implementation can safely terminate when it first encounters node \texttt{b}. [5 marks]

(b) Consider all paths in a graph from \texttt{a} to \texttt{b}, ordered from shortest to longest. Assuming \texttt{p = dijkstra\_path}(g, a, b) is the first path in this collection, an algorithm to find the second path considers deviations from the vertices of \texttt{p}. An algorithm to do this is given below.

\begin{verbatim}
function second_path(Graph g, Vertex a, Vertex b):
p = dijkstra_path(g,a,b)
best_so_far = []
for i = 1 to len(p)-1:
    t = p[:i]  # First i elements of p
    c = g.get_edge_weight(p[i], p[i+1])
    g.set_edge_weight(p[i], p[i+1], infinity)
    t.append(dijkstra_path(g,p[i],b))
    if (len(best_so_far) == 0 or
        cost(t) < cost(best_so_far)):
        best_so_far = t
    g.set_edge_weight(p[i], p[i+1], c)
return best_so_far
\end{verbatim}

(i) Show the steps of this algorithm on the following graph, from \texttt{A} to \texttt{B}. [5 marks]

(ii) What is the asymptotic complexity of this algorithm in terms of the number of edges, \(E\), and the number of vertices, \(V\)? Assume the implementation of Dijkstra’s algorithm uses a priority queue based on a Fibonacci heap. [4 marks]

(iii) Show how to adapt this algorithm to find the top-\(k\) shortest paths in the collection. State the complexity of the adapted algorithm. [6 marks]