[CSN212] Assignment 4
Graph Algorithms
Maximum Marks 100

NOTE: Feel free to use any algorithm or proof covered in class as a black box.

1 Careful Deletion of vertices [10 Marks]

Given an undirected graph. Design and analyze an algorithm to find an order of deletion of vertices, such that any deletion does not disconnect the residual graph.

2 Even Cyclic Graph [10 Marks]

A graph is called an even cyclic graph if every cycle in the graph has even length. Describe and analyze an algorithm to report if a directed graph is even cyclic.

3 Critical Edges [20 Marks]

Critical edge in a graph is an edge whose removal causes the maximum increase in a computed entity.

1. Design and analyze an algorithm to compute the *critical* edge for computing an MST of the graph.
2. Design and analyze an algorithm to compute the *critical* edge for shortest $s - t$ path.

4 MST under vertex insertion [20 Marks]

Suppose that a graph G has a minimum spanning tree already computed. Design and analyze an algorithm to update the minimum spanning tree if we add a new vertex and incident edges to G.

5 Modified Shortest Paths [20 Marks]

1. **Monotonic shortest path.** Given a directed graph $G = (V, E)$ having positive edge weights, a monotonic path is one whose subsequent edge weights are either strictly increasing or strictly decreasing (strictly implies equal weights of adjacent edges are not permitted). Design and analyze an algorithm to compute the shortest monotonic path from a given source s.
2. **Bitonic shortest path.** Given a directed graph $G = (V, E)$ having positive edge weights, a bitonic path is one whose subsequent edge weights are first strictly increasing up to some vertex and thereafter strictly decreasing (strictly implies equal weights of adjacent edges are not permitted). Design and analyze an algorithm to compute the shortest bitonic path from a given source s.

6 Compression Algorithm [20 Marks]

Lempel-Ziv parsing is a method behind many popular compression software such as `zip`, `gzip`. It works by replacing substrings of a string by pointers to their earlier occurrences. For example, `mississippi` could be parsed to a list of *phrases* $< 0, \mathtt{m} >< 0, \mathtt{i} >< 0, s >< 1, 1 >< 3, 4 >< 0, \mathtt{p} >< 1, 1 >< 3, 1 >$, where $< 0, c >$ is used for new symbols and $< d, \ell >$ is used for the distance d, $d > 0$, from the previous occurrence of the substring of length ℓ.

Given string $S[1..n]$, a greedy Lempel-Ziv parsing works so that when it has outputed the parsing of $S[1..j-1]$, it finds the longest prefix of $S[j..n]$ that is also a prefix of $S[i..n]$, for some $i < j$, and outputs $< j - i, \mathtt{lcp}(i, j) >$, where $\mathtt{lcp}(i, j)$ denotes the length of the longest common prefix of $S[i..n]$ and $S[j..n]$. If no such i can be found with $\mathtt{lcp}(i, j) > 0$, it outputs $< 0, S[j] >$.

It can be shown that greedy parsing minimizes the number phrases outputed, over all possible parsings. However, this parsing may not be *bit-optimal*: Consider the case where the numbers in the phrases are encoded using a *variable-length encoder*, meaning that each number is assigned a code of varying length (in bits). Let $c1(d)$ and $c2(\ell)$ be functions returning the length of the code for each distance d and length ℓ, respectively. Give a polynomial time algorithm to find a *bit-optimal Lempel-Ziv parsing*, where the output size (in bits) of the parsing is minimum. To simplify the exposition, you may assume that the input contains all alphabet symbols in the beginning, so that all parses start deterministically and the encoding of new symbols can be ignored.

Hint. Create a DAG where shortest path corresponds to bit-optimal parsing.

7 Text Book Problems [Solve Dont Submit]

1. CLRS 22.5-6, 22.5-7,23-1,23-3,24-4,25-2
2. Consider the following algorithm to compute Strongly Connected components of a graph G.
   (a) Run the DFS of graph G_R (reverse all edges of G), and store its DFS finish times in reversed order.
   (b) Pick vertices in the above order, and perform a DFS to report the entire SCC containing the vertex.

   Prove the correctness of the above algorithm.

   Note: CLRS 22.5. This is the famous Kosaraju algorithm, from S. Rao Kosaraju who is an alumnus of IIT Kharagpur, who is currently a Professor at John Hopkin's University. Maybe someone among you would be responsible for such a fundamental algorithm in the future similar to Prof. Kosaraju :) .