Design and Analysis of Algorithms

Duration: 1 Hour
Maximum Marks: 15

1. Let $G = (V, E)$ be a weighted connected graph. Prove or disprove: A MCST can be found inductively as follows. Pick any vertex v and remove it from G to get G_v. Inductively find a MCST T_v for G_v. Add the minimum cost edge out of v in G to T_v to get T. This T is a MCST.

   [Hint: Observe that in this construction v will be a leaf in T]

   (3 marks)

2. Assume that you have an efficient implementation of a suffix trees (or suffix arrays). Describe an algorithm which takes a word w and two patterns (also words) u_1 and u_2 and outputs the left most and right most positions where either of these patterns occur. That is, the left most position that begins an occurrence of u_1 or u_2 and similarly the right most position that begins an occurrence of u_1 or u_2. Assume that the positions are numbered starting with 0.

   Here is an example:

   ```
   w = ababaabaaabaaaababab
   u1 = baa
   u2 = abaa
   ```

   The left most occurrence begins at position 2 (where `abaa` occurs). The right most occurrence is 10 (where `baa` occurs). What is the complexity of your solution? *(4 marks)*

3. Take the first four letters of your first name (pad it with b's if it is shorter than 4 letters), alternate it with 4 a's to get a word w. For example, if your name is *Arulmozhi* then the resulting word is *aarauala*.

   EITHER compute the Huffman code for this word and compute the number of bits saved in total using your encoding OR compute the suffix array for this word and use that to compute its Burrows Wheeler Transform. *(4 marks)*

4. Describe an algorithm that takes a directed graph $G = (V, E)$ as input and determines two sets of vertices Sr and Sn where Sr consists of all vertices v from which there is a path to every other vertex in V and Sn consists of all vertices v that can be reached from every vertex in V. (Of course Sr or Sn or both can be empty). Your algorithm should run in linear time in the size of the graph.

   [Hint: What has this got to do with Strongly Connected Components?] *(4 marks)*