Homework 5*

Data Structures
Fall 2019 CS203@IITG

---

(1) Analyze the time complexities of insert and delete operations for (i) compressed binary trie, and (ii) compressed multiway trie.

A radix tree is a compressed multiway trie with the number of children of any internal node is at most a given radix r. The following problem on radix tree: [CLRS] exercise 12-2 (page 304).

(2) Analyze the preprocessing time complexity of computing the suffix trie (compressed multiway trie comprising all the suffixes of a text string T given for preprocessing) with substring labels along the edges of the trie (i) when branching at each internal node is implemented with a static array (hence, it is a radix trie), and (ii) when branching at each internal node is implemented with a balanced BST.

Let T' be the suffix tree of text T concatenated with \$. Prove the following: Printing the suffix corresponding to every leaf node v of T' immediately after v is encountered during the inorder traversal of T' lists all the suffixes of $T\$$ in lexicographic order.

(3) To answer queries efficiently, each of the four applications mentioned for suffix tries in pages 598-600 of [HSA] require augmenting the standard suffix tree of the text T that gets preprocessed. Analyze the preprocessing time complexity in augmenting the standard suffix tree in each of these applications (excluding the time to build the standard suffix tree).

(4) Given the LCP array and the suffix array of text T, devise an efficient algorithm to compute LCP of any two suffixes that may possibly be needed in the query algorithm for determining whether the given pattern P occurs in T. (You may fix bugs in the algorithm presented in class, or devise a new algorithm.)

(5) In the definition of universal hash family $\mathcal{H}_{pm}$, if p is not a prime number, then determine whether $\mathcal{H}_{pm}$ continues to be a universal hash family.

As part of the FKS algorithm, we had shown that each key is hashed on an average 3 times before all the keys are placed without any pairwise collisions. Does it affect the time complexity or the expected space complexity (while considering the space of primary hash table and the secondary hash table) when the universal hash family $\mathcal{H}_{pm}$ is used?

(6) In the analysis of closed hashing, identify the specific places at which we used the assumption of uniform hashing.

Argue in favor of or against the following: the simple uniform hashing is a special case of the uniform hashing.

(7) Given a simple undirected unweighted graph in adjacency list data structure, devise an efficient algorithm to do the following: (i) every vertex v stored in any of the linked lists must need to have a pointer to an entry in the array part of the adjacency list that corresponds to vertex v, and (ii) every node that represents an edge $v'v''$ must have a pointer to the other node that also represents the same edge $v'v''$.

---

*Prepared by R. Inkulu, Department of Computer Science, IIT Guwahati, India. http://www.iitg.ac.in/rinkulu/

(8) For every vertex v that is reachable from s, prove that concatanating a shortest path from s to $v.pred$ with the arc from $v.pred$ to v yields a shortest path from s to v. Here, $v.pred$ is determined by traversing the given simple unweighted directed graph G in breadth-first manner, starting from a node $s \in G$.

Prove that the predecessor graph G' output by breadth-first traversal of a simple unweighted directed graph G from a node $v \in G$ is a shortest-path tree rooted at s. That is, show that for every vertex v that is reachable from s, there exists a unique path from s to v in G', and that unique path is a shortest path from s to v in G.

(Note that informal proofs for both of these were given in lectures.)