Questions for self practice

IISc - DS221- Data Structures and Algorithms

Attempt these problems and write your solution on a paper before consulting internet or friends. You are not required to submit your solutions on Teams; these questions are for self-practice only.

1 Linear list

1. Design an efficient data structure for representing a subset S of integers from 1 to n. Operations we wish to perform on the set are:

   - Size reduction: Select an arbitrary integer (any element would do) from the set and delete it. The size reduction operation must happen in $O(1)$ time.
   - Add an integer i to the set (but do not permit if it is already present in your data structure). This should also happen in $O(1)$ time.

   Give a convincing explanation of how your data structure supports these operations in $O(1)$ time. *Hint: A single data structure will not suffice; try a combination of two data structures.*

2 Algorithmic analysis and complexity

1. Give using "big oh" notation, the worst case running time of the following program segment as a function of n. Write the recurrence relation and solve it. Assume n to be even.

```
int sample (int n) {
  if (n <= 1) return 1;
  else return (sample(n-2) + sample(n-2));
}
```

2. For what values of k is $n^k = O(n)$? Why? Consider $0 \leq k \leq 1000$ and k as real (not just integer).

3 Binary tree

1. One way to implement a binary tree is a linked representation where each element is represented by a node that has two link fields (`leftChild` and `rightChild`) plus a data field. Prove that in a binary tree with n nodes, $(n + 1)$ of the $2n$ link fields are NULL. *Hint: Try proof by induction.*

4 Priority queue

1. Suppose you have a Min-Heap H containing n keys stored as a binary tree. Suppose you wish to expand the features of your implementation and allow users to decrease the key at a given node. Assume that the user gives you an index i of a node in H and the new key (which is lower than the original key at that node). Design an efficient algorithm (pseudocode) to efficiently decrease the key at node i and remake H into a Min-Heap again. What is the worst case time complexity of your algorithm?

5 Dictionary

1. Suppose you have an array containing n distinct integers in ascending order. Design a 3-ary search algorithm similar to the binary search algorithm, which divides the range of search into 3 parts. (i) Write a pseudocode for your algorithm. (ii) Derive its worst case time complexity using recurrence relations.
2. Let S and T be two sets of numbers. Each of these set is represented as an unordered linked list. There are no duplicate elements within each list. All you are given are pointers to the heads of two lists. You do not know the list lengths. Describe an $O(min(|S|, |T|))$ expected-time algorithm to determine whether $S = T$. Describe your algorithm (step-by-step) in plain english. *Hint: Think about hashing.*

6 Graphs

1. Recall the *Compressed Sparse Row* (CSR) format for saving sparse matrices that was discussed in one of the lectures. Suppose you are working with a sparse undirected graph $G = (V, E)$ with $|E| \ll |V|^2$.
   (a) Inspired by the CSR storage format for sparse matrices, suggest a space-efficient data structure to store sparse undirected graph G.
   Hint: You should need one array of size $2|E|$ *and a second array of size* $|V| + 1$.
   (b) What is the main advantage, and the main disadvantage of the above representation over the adjacency list representation.
   (c) Write a pseudocode for breadth first search (BFS). Assume that your undirected graph is stored in the above CSR-like format.

7 Algorithms

1. Let A be an array of positive or negative integers of size n, where $A[1] < A[2] < \ldots < A[n]$. Write an algorithm (better than $O(n)$ time) to find an i such that $A[i] = i$ provided such an i exists. Give a precise description of the steps in your algorithm in plain english.
2. The diameter of a tree $T = (V, E)$ is defined by $\max_{\{u,v\} \in E} d(u, v)$, where $d(u, v)$ is the shortest path distance between u and v (measured in terms of #nodes between u and v, see Fig. 7 for an example). This path need not always pass through the root. This holds for all types of trees (not necessarily binary). Give an efficient algorithm (pseudocode) to compute the diameter of a tree. Use adjacency list representation for the tree.