ESO207 Programming Assignment-2.1

Due on: 23:59 hrs, Oct 25, 2021
Maximum Marks 50

Instructions

- Please insert suitable comments in your pseudo-code and actual code so that anyone grading it may understand it easily.
- Present your arguments clearly and stay to the point.
- Runtime complexity of your program may be assessed from complexity analysis of your pseudo-code and by checking the fact that you have implemented that pseudo-code.
- Your code is allowed to be destructive. That is, input arguments may not denote the same object after the computation.
- No marks shall be awarded for an algorithm/program not working in the prescribed theoretical time bounds.
- **Notation:** Height of a tree T is denoted by $h(T)$. For a set S, $|S|$ stands for number of elements in S.

Assignment-2 is about implementing some operations of Union-Find-Split ADT used in algorithm to find largest common subsequence. Each set is represented as a `2-3` tree. For examples of these operations, you may refer to Lecture 23.

Q1 (Marks 25 + 25) You are given `2-3` trees T_1 and T_2, representing respectively finite sets S_1, S_2 of natural numbers. Further, it is given that for all $x \in S_1$ and for all $y \in S_2$, $x < y$.

(a) Write pseudo-code for the algorithm $Merge(T_1, T_2)$. $Merge(T_1, T_2)$ should return a 2-3 tree representation of set $S_1 \cup S_2$. Your algorithm should take $O(h(T_1) + h(T_2))$ time. Justify time complexity of your algorithm.

(b) Implement your algorithm of part (a) as an executable function $Merge(T_1, T_2)$. You are not allowed to use any library function in this implementation.

To help you write this code, you may divide it into several smaller functions. A modular and hierarchical design is encouraged.

For allowing us to test your program easily, you need to design following two functions. You need to submit just the working programs for these functions. No pseudocode for these is required. You may also use library functions (for example queue data structure) for these programs.

- **fun** Extract(T).
  Input: T is a 2-3 tree.
  action of program: Extract(T) prints elements of the set represented by T is ascending order.
- **fun** MakeSingleton(x)
  Input: x is a number.
  Output: MakeSingleton(x) returns a 2-3 tree representing set $\{x\}$

Please test your program using following function Test.

```
fun Test()
  T = MakeSingleton(1)
  for i = 2 to 500
   T= Merge(T,MakeSingleton(i))

  U = MakeSingleton(777)
  for i = 778 to 1000
    U= Merge(U,MakeSingleton(i))

  V=Merge(U,T)
  Extract(V)
```

Executing `Test()` should output $1, 2, 3, ..., 500,\ 777, 778, ..., 1000$.

We will test your program using different test functions.

Submission: One zipped tar file per team, containing pseudo-code file, source code files 'Merge', 'Extract' and 'MakeSingleton' (and other auxiliary files) should be uploaded on mookit.

——— End of Assignment ———