Algorithms and data structures

Labwork 5: Searching and Sorting Algorithms

Dr. Doan Nhat Quang: doan-nhat.quang@usth.edu.vn

1. Sorting and Tree data structure

Exercise 1: This problem, we would like to study a sorting algorithm for a given array of n elements with the following steps: 1. Find the maximum element position (denoted by max) from 0 to n-1. 2. Flip all elements from the position 0 to max. (Flip means reverse, e.g: flipping (1, 0, 3) = (3, 0, 1)). 3. Flip the whole unsorted array. Now the maximum is found at the end of the unsorted array and the last element or the maximum is sorted. 4. Repeat this process until all elements are sorted.

- Write a program to complete the above algorithm.
- Calculate the complexity of your program (Best scenario, Worst scenario, Average).

Exercise 2: A binary tree can sort n elements of an array of data. First, create a complete binary tree with all leaves at one level, whose height h = (log_2 n) + 1, and store all array elements in the first n leaves. In each empty leaf, store an element E greater than any element in the array.

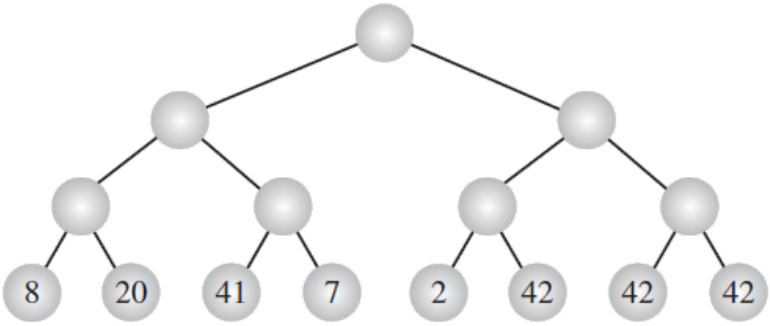
Figure (a) shows an example for data = {8, 20, 41, 7, 2}, h = ($log_2(5)$) + 1 = 3, and E = 42. Then, starting from the bottom of the tree, assign the minimum of its two children values to each node, as in Figure (b), so that the smallest element e_{min} in the tree is assigned to the root.

If a leaf node is to be removed, this node is replaced by a new node with the same value as its parent node.

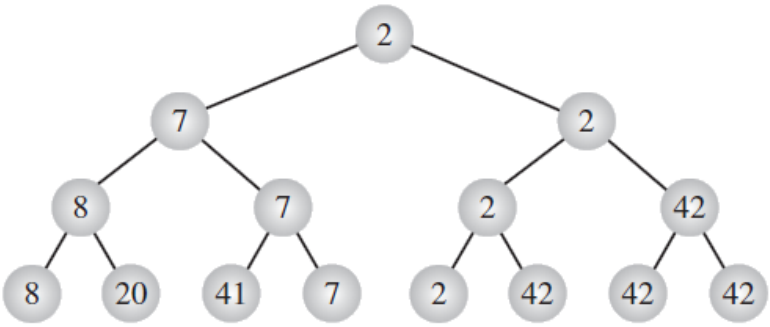
If a node is added to the tree, it will be a leaf node. Normally, a node with value E is replaced with a new value. It's necessary to verify recursively all values of its parent and make any possible modification, if necessary, so that the tree rules are respected.

Implement this tree structure in C/C++ with the necessary functions.

- write a function to initialize an array with n random values
- write a function to build this binary tree with the above definition with any data structure learned in lectures



(a)



(b)

- write a function to display the tree information
- write a function to search an input value using recursion. If found, display all the subtrees with the found node as the root of this subtree or return -1.
- write a function to insert new nodes into the tree and another one to remove nodes from the tree