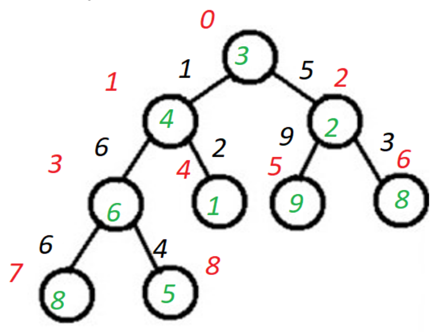
ALGORITHMS HW 2

We are given a **COMPLETE BİNARY TREE** where nodes and edges have **positive** weights. Node weights are stored in a *1*-dimensional array ***WN***. Edge weights are stored in a *2*-dimensional array ***WE*** where *0* denotes no edge.

Starting at the root of the tree and moving to either one of the children from the current node, **the goal** is to find the minimum total weight (i.e. sum of node and edge weights) path from the root to any one of the leaves.

Example:



Node weights array: ***WN*** = [3 4 2 6 1 9 8 8 5]

Edge weights array: ***WE*** =
$$\left[\begin{array}{ccccccccc}
0 & 1 & 5 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 6 & 2 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 9 & 3 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{array}\right]$$

Nodeid
Nodeweights
Edgeweights

Output: Min total weight path includes nodes 1-2-5 with total weight 9.

1. Implement the algortihm that generates complete binary tree with given size as input:
   -Generate the node and edge weights random between 1 and 20 inclusive.

2. Implement the greedy algorithm (i.e., write a function) of choosing the child with smallest sum of edge and node weights each time.

3. Implement a recursive algorithm (i.e., write a function) to find the minimum total weight. You must determine the input parameters. Also, give the time complexity of a recursive algorithm that implements this formulation? Show your work.

4. Implement a dynamic programming algorithm to solve the problem. You must determine the input parameters. Also, give the time complexity of your dynamic programming solution? Show your work.

5. In your main function:

   a. Show that the greedy algorithm does not solve this problem optimally.

   b. Run each of the recursive and dynamic functions with three different input sizes and compute the actual running times (in milliseconds or seconds) of these three algorithms. You will need to calculate the time passed before and after making a call to each function. Provide a 2x3 table involving the actual running times.