Heap One Shot

Assignment Solutions

PW SKILLS

Q1. Given an array of elements, sort the array in decreasing order using min heap.
Input : arr[ ] = {5, 3, 10, 1}
Output : arr[ ] = {10, 5, 3, 1}

Code link: https://pastebin.com/eqZ1p2bE

Explanation:

1. Build a min heap from the given array.
2. Extract elements from the min heap one by one in a loop and store them in a new array.
3. The extracted elements will be in increasing order because we are using a min heap. So, reverse the new array to get the elements in decreasing order.

Output:

```
Original array: 5 3 10 1
Array in decreasing order: 10 5 3 1


...Program finished with exit code 0
Press ENTER to exit console.
```

Q2. Given are N ropes of different lengths, the task is to connect these ropes into one rope with minimum cost, such that the cost to connect two ropes is equal to the sum of their lengths.
Input: arr[ ] = {4,3,2,6} , N = 4
Output: 29
Explanation:

1. **First, connect ropes of lengths 2 and 3. Now we have three ropes of lengths 4, 6, and 5.**
2. **Now connect ropes of lengths 4 and 5. Now we have two ropes of lengths 6 and 9.**
3. **Finally connect the two ropes and all ropes have connected.**

Code link: https://pastebin.com/rYPurupU

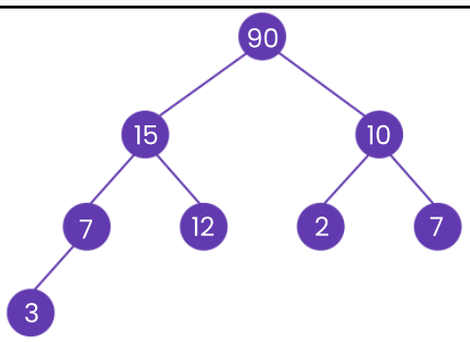
Explanation:

1. Create a priority queue (min heap) to store the lengths of the ropes.
2. Insert all the rope lengths into the priority queue.
3. Initialize a variable cost to keep track of the total cost of connecting the ropes.
4. While the priority queue has more than one rope length:
   - Extract the two minimum lengths from the priority queue.
   - Add the sum of these two lengths to the cost.
   - Insert the sum back into the priority queue.
5. Return the final cost as the minimum cost to connect all the ropes.

```
Minimum cost to connect ropes: 29


...Program finished with exit code 0
Press ENTER to exit console.
```

Q3.Given the level order traversal of a complete Binary Tree, determine whether the Binary Tree is a Max Heap or not.
A complete binary tree means that all levels are filled except possibly the last level, and all nodes in the last level are as far left as possible.
Input:
Binary Tree Level Order Traversal: 90, 15, 10, 7, 12, 2, 7, 3
Binary Tree:



Output:

```
The given binary tree is a max heap.
```

A1. Code: https://pastebin.com/UKSTNTZ6

Explanation:

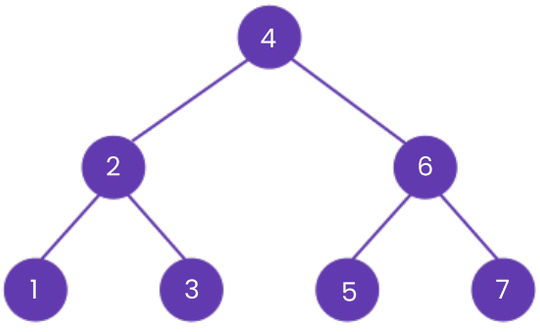
1. Given an array of size representing level order traversal of binary tree.
2. The function iterates through all internal nodes of the binary tree using a for loop, where i ranges from 0 to (n / 2) - 1.
3. For each internal node at index i, it calculates the indices of its left and right children as 2 * i + 1 and 2 * i + 2, respectively.
4. If the left child index is within the range of the input list and the left child value is greater than the parent value at index i, the function returns false, indicating that the binary tree is not a max heap.
5. Similarly, if the right child index is within the range of the input list and the right child value is greater than the parent value at index i, the function returns false.
6. If none of the above conditions are met, the function returns true, indicating that the binary tree is a max heap.

Time complexity: O(n), where n is the number of nodes in the binary tree. This is because the code iterates through all internal nodes of the binary tree, which is a linear operation.

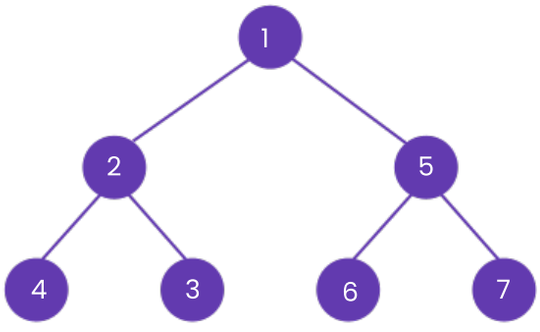
Space complexity: O(1), as it only uses a constant amount of extra space to store the indices of the left and right children of each internal node. The input list of integers is not considered extra space, as it is necessary to represent the binary tree in level order traversal.

Q4. Given a binary search tree which is also a complete binary tree. The problem is to convert the given BST into a Min Heap with the condition that all the values in the left subtree of a node should be less than all the values in the right subtree of the node. This condition is applied to all the nodes, in the resultant converted Min Heap.

Input:
Binary Search Tree



Output:
Min heap



A3. Solution Code: https://pastebin.com/n1M4aXJy

Output:

```
Binary Search Tree:
1 2 3 4 5 6 7
Min Heap:
4 3 4 1 6 5 7
```

Explanation:

1. Traverse the BST in any order and store the values of each node in an array.
2. Perform a bottom-up heap construction starting from the last element in the array, where each node is swapped with its smallest child until it satisfies the min heap property.
3. Once the heap construction is complete, replace the value of each node in the BST with the corresponding value from the array.

Time complexity: O(n), where n is the number of nodes in the tree. This is because we need to visit every node in the tree exactly once to extract its value and insert it into the min heap.

Space complexity: The space complexity of the conversion process depends on the implementation. If we are allowed to modify the original BST, we can perform the conversion in-place without using any additional space. However, if we need to create a new min heap while preserving the original BST, we need to allocate additional space for the new nodes. In this case, the space complexity would be O(n) to store the new nodes in the min heap.