Summary

Time complexity : O(n log n) because heapify (called while extraction and insertion) takes O(log n) time and it will be called proportional to the number of elements n.
Space complexity is O(n) for the heap.

Problem Pattern

1. Take some example arrays, with elements as lengths of ropes and try to connect these ropes in different possible ways. Do you see a pattern for when you're obtaining the minimum answer?
2. Every time before connecting 2 ropes, you have to decide which 2 ropes to connect. What will this decision be based on? Think of a way to re-order your elements.
3. Will sorting help? If yes, is sorting just once enough? What will you do with the new rope you obtain after connecting 2 ropes?
4. If you need to sort again and again, would a data structure that always stores elements in a sorted order help? Consider using a heap.

Problem Approach

1. If you observe the problem closely, you'll notice that the lengths of the ropes which are picked first are included more than once in total cost. Therefore, the idea is to connect the smallest two ropes first and recur for remaining ropes.
2. So, we can use a heap to store the lengths of our ropes. As we form new ropes by connecting 2 other ropes, we keep adding the new lengths to our heap, so that we are easily able to pick the 2 smallest ropes.

Say your ropes : 2 5 6 1 4 → sorted → 1 2 4 5 6
First we'll pick 2 + 1 = 3
Now our array should become 3 4 5 6, i.e we will pick the merged rope first because its length is lesser than all the other ropes previously present in the array.

3. Let there be n ropes of lengths stored in an array len[0..n-1]

```
    1) Create a min heap and insert all lengths into the min
heap.

    2) Do the following while the number of elements in the min
heap is not 1.
```

        a) Extract the minimum and second minimum from min heap

        b) Add the above two extracted values and insert the added value to the min-heap.

        c) Maintain a variable for total cost and keep incrementing it by the sum of extracted values.

    3) Return the value of this total cost.

Problem Pseudocode

```
int minCost(int arr[], int n) {
        new PriorityQueue<Integer> pq

        for (i = 0 to n) :
            pq.add(arr[i])

        res = 0
        while (pq.size() > 1) {
            first = pq.poll()
            second = pq.poll()

            res += first + second
            pq.add(first + second)
        }
        return res
}
```

Alternate Approaches

You can solve this problem by using sorting algorithms, i.e. you can keep adding your new rope’s length to the array and call the sorting algorithm everytime. However, keep in mind that insertion in an array in an O(n) operation and you will also be sorting n times. So the time complexity will be higher.