

## Connect Ropes (easy)

We'll cover the following ^

- Problem Statement
- Try it yourself
- Solution
- Code
  - Time complexity
  - Space complexity

### Problem Statement #

Given 'N' ropes with different lengths, we need to connect these ropes into one big rope with minimum cost. The cost of connecting two ropes is equal to the sum of their lengths.

#### Example 1:

Input: [1, 3, 11, 5]

Output: 33

Explanation: First connect  $1+3(=4)$ , then  $4+5(=9)$ , and then  $9+11(=20)$ . So the total cost is 33 ( $4+9+20$ )

#### Example 2:

Input: [3, 4, 5, 6]

Output: 36

Explanation: First connect  $3+4(=7)$ , then  $5+6(=11)$ ,  $7+11(=18)$ . Total cost is 36 ( $7+11+18$ )

#### Example 3:

Input: [1, 3, 11, 5, 2]

Output: 42

Explanation: First connect  $1+2(=3)$ , then  $3+3(=6)$ ,  $6+5(=11)$ ,  $11+11(=22)$ . Total cost is 42 ( $3+6+11+22$ )

### Try it yourself #

Try solving this question here:

 Java

 Python3

 JS

 C++

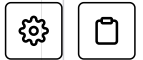
```
1 def minimum_cost_to_connect_ropes(ropeLengths):
2     result = []
3     # TODO: Write your code here
4     return result
```



```

5
6
7 def main():
8
9     print("Minimum cost to connect ropes: " +
10           str(minimum_cost_to_connect_ropes([1, 3, 11, 5])))
11     print("Minimum cost to connect ropes: " +
12           str(minimum_cost_to_connect_ropes([3, 4, 5, 6])))
13     print("Minimum cost to connect ropes: " +
14           str(minimum_cost_to_connect_ropes([1, 3, 11, 5, 2])))
15
16
17 main()
18
19

```



## Solution #

In this problem, following a greedy approach to connect the smallest ropes first will ensure the lowest cost. We can use a **Min Heap** to find the smallest ropes following a similar approach as discussed in Kth Smallest Number

(<https://www.educative.io/collection/page/5668639101419520/5671464854355968/5696381570252800/>). Once we connect two ropes, we need to insert the resultant rope back in the **Min Heap** so that we can connect it with the remaining ropes.

## Code #

Here is what our algorithm will look like:

Java

Python3

C++

JS

```

1 from heapq import *
2
3
4 def minimum_cost_to_connect_ropes(ropeLengths):
5     minHeap = []
6     # add all ropes to the min heap
7     for i in ropeLengths:
8         heappush(minHeap, i)
9
10    # go through the values of the heap, in each step take top (lowest) rope lengths from th
11    # connect them and push the result back to the min heap.
12    # keep doing this until the heap is left with only one rope
13    result, temp = 0, 0
14    while len(minHeap) > 1:
15        temp = heappop(minHeap) + heappop(minHeap)
16        result += temp
17        heappush(minHeap, temp)
18
19    return result
20
21
22 def main():
23
24     print("Minimum cost to connect ropes: " +
25           str(minimum_cost_to_connect_ropes([1, 3, 11, 5])))
26     print("Minimum cost to connect ropes: " +

```

```
27         str(minimum_cost_to_connect_ropes([3, 4, 5, 6])))
28     print("Minimum cost to connect ropes: " +
```



#### Time complexity #

Given 'N' ropes, we need  $O(N * \log N)$  to insert all the ropes in the heap. In each step, while processing the heap, we take out two elements from the heap and insert one. This means we will have a total of 'N' steps, having a total time complexity of  $O(N * \log N)$ .

#### Space complexity #

The space complexity will be  $O(N)$  because we need to store all the ropes in the heap.

[← Back](#)[Next →](#)

'K' Closest Points to the Origin (easy)

Top 'K' Frequent Numbers (medium)



**Mark as Completed**



Report an  
Issue



Ask a Question

([https://discuss.educative.io/tag/connect-ropes-easy\\_\\_pattern-top-k-elements\\_\\_grokking-the-coding-interview-patterns-for-coding-questions](https://discuss.educative.io/tag/connect-ropes-easy__pattern-top-k-elements__grokking-the-coding-interview-patterns-for-coding-questions))