

## Grokking the Coding Interview: Patterns for Coding Questions

76% completed



Pattern: Merge Intervals



Pattern: Cyclic Sort



Pattern: In-place Reversal of a LinkedList



Pattern: Tree Breadth First Search



Pattern: Tree Depth First Search



Pattern: Two Heaps



Pattern: Subsets



Pattern: Modified Binary Search



Pattern: Bitwise XOR



Pattern: Top 'K' Elements



- Introduction
- Top 'K' Numbers (easy)
- Kth Smallest Number (easy)
- 'K' Closest Points to the Origin (easy)
- Connect Ropes (easy)**
- Top 'K' Frequent Numbers (medium)
- Frequency Sort (medium)
- Kth Largest Number in a Stream (medium)
- 'K' Closest Numbers (medium)
- Maximum Distinct Elements (medium)
- Sum of Elements (medium)
- Rearrange String (hard)
- Problem Challenge 1
- Solution Review: Problem Challenge 1
- Problem Challenge 2
- Solution Review: Problem Challenge 2
- Problem Challenge 3
- Solution Review: Problem Challenge 3

Pattern: K-way merge



- Introduction
- Merge K Sorted Lists (medium)
- Kth Smallest Number in M Sorted Lists (Medium)

## 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:

JavaPython3JS C++

```
1 const minimum_cost_to_connect_ropes = function(ropesLengths) {
2   result = [];
3   // TODO: Write your code here
4   return result;
5 };
6
7
8 console.log('Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([1, 3, 11, 5])}')
9 console.log('Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([3, 4, 5, 6])}')
10 console.log('Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([1, 3, 11, 5, 2])}')
11
```

RUNSAVERESET

Close

Output2.892s

Minimum cost to connect ropes:  
Minimum cost to connect ropes:  
Minimum cost to connect ropes:

### 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](#). 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:

JavaPython3C++JS

```
1 const Heap = require('./collections/heap'); //http://www.collectionsjs.com
2
3
```

Kth Smallest Number in a Sorted Matrix (Hard)

Smallest Number Range (Hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Pattern : 0/1 Knapsack (Dynamic Programming)

Introduction

0/1 Knapsack (medium)

Equal Subset Sum Partition (medium)

Subset Sum (medium)

Minimum Subset Sum Difference (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Pattern: Topological Sort (Graph)

Introduction

Topological Sort (medium)

Tasks Scheduling (medium)

Tasks Scheduling Order (medium)

All Tasks Scheduling Orders (hard)

Alien Dictionary (hard)

Problem Challenge 1

Solution Review: Problem Challenge 1

Problem Challenge 2

Solution Review: Problem Challenge 2

Miscellaneous

Kth Smallest Number (hard)

Conclusions

Where to Go from Here

Mark Course as Completed

Explore

Tracks

My Courses

Edpresso

Refer a Friend

Create

```
4 function minimum_cost_to_connect_ropes(ropesLengths) {
5   // add all ropes to the min heap
6   const minHeap = new Heap(ropesLengths, null, ((a, b) => b - a));
7
8   // go through the values of the heap, in each step take top (i.e., lowest) rope lengths from the min heap
9   // connect them and push the result back to the min heap.
10  // keep doing this until the heap is left with only one rope
11  let result = 0;
12  while (minHeap.length > 1) {
13    const temp = minHeap.pop() + minHeap.pop();
14    result += temp;
15    minHeap.push(temp);
16  }
17
18  return result;
19 }
20
21 console.log(`Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([1, 3, 11, 5])}`);
22 console.log(`Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([3, 4, 5, 6])}`);
23 console.log(`Minimum cost to connect ropes: ${minimum_cost_to_connect_ropes([1, 3, 11, 5, 2])}`);
```

RUN

SAVE

RESET

Close

Output3.930s

Minimum cost to connect ropes: 33  
Minimum cost to connect ropes: 36  
Minimum cost to connect ropes: 42

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.

Interviewing soon? We've partnered with [Hired](#) so that companies apply to you instead of you applying to them. [See how](#)

← Back

Next →

'K' Closest Points to the Origin (easy)

Top 'K' Frequent Numbers (medium)

✓

Mark as Completed

Report an Issue

Ask a Question