

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 // O(n^2) time | O(n) space
4 function diskStacking(disks) {
5   disks.sort((a, b) => a[2] - b[2]);
6   const heights = disks.map(disk => disk[2]);
7   const sequences = new Array(disks.length);
8   let maxHeightIdx = 0;
9   for (let i = 1; i < disks.length; i++) {
10     const currentDisk = disks[i];
11     for (let j = 0; j < i; j++) {
12       const otherDisk = disks[j];
13       if (areValidDimensions(otherDisk, currentDisk)) {
14         if (heights[i] <= currentDisk[2] + heights[j]) {
15           heights[i] = currentDisk[2] + heights[j];
16           sequences[i] = j;
17         }
18       }
19     }
20     if (heights[i] >= heights[maxHeightIdx]) maxHeightIdx = i;
21   }
22   return buildSequence(disks, sequences, maxHeightIdx);
23 }
24
25 function areValidDimensions(o, c) {
26   return o[0] < c[0] && o[1] < c[1] && o[2] < c[2];
27 }
28
29 function buildSequence(array, sequences, currentIdx) {
30   const sequence = [];
31   while (currentIdx !== undefined) {
32     sequence.unshift(array[currentIdx]);
33     currentIdx = sequences[currentIdx];
34   }
35   return sequence;
36 }
```

Our Tests

```
1 // Test Case 1: Expected: [1, 3, 5]
2 // Test Case 2: Expected: [1, 3, 5]
3
4 // Test Case 3: Expected: [1, 3, 5]
5 // Test Case 4: Expected: [1, 3, 5]
6
7 // Test Case 5: Expected: [1, 3, 5]
8 // Test Case 6: Expected: [1, 3, 5]
9
10 // Test Case 7: Expected: [1, 3, 5]
11 // Test Case 8: Expected: [1, 3, 5]
12
13 // Test Case 9: Expected: [1, 3, 5]
14 // Test Case 10: Expected: [1, 3, 5]
```

Solution 1

Solution 2

Solution 3

```
1 function diskStacking(disks) {
2   // Write your code here.
3 }
4
5 // Do not edit the line below.
6 exports.diskStacking = diskStacking;
7
```

Custom Output

Submit Code

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```

1 100 "Task 1: Basic MPI" -- "MPI Hello World" 10 1
2
3 #include <mpi.h>
4
5 int main()
6 {
7     MPI_Init(&argc, &argv);
8     MPI_Comm_rank(MPI_COMM_WORLD, &rank);
9     MPI_Comm_size(MPI_COMM_WORLD, &size);
10
11     if (rank == 0)
12     {
13         // Send message to all processes
14         MPI_Bcast(&rank, 1, MPI_INT, 0, MPI_COMM_WORLD);
15     }
16     else
17     {
18         // Receive message from process 0
19         MPI_Bcast(&rank, 1, MPI_INT, 0, MPI_COMM_WORLD);
20     }
21
22     // Print rank of each process
23     printf("Process %d\n", rank);
24
25     MPI_Finalize();
26 }

```

Run or submit code when you're ready.