

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     // O(n^2) time | O(n) space
7     public static List<Integer[]> diskStacking(List<Integer[]> disks) {
8         disks.sort((disk1, disk2) -> disk1[2].compareTo(disk2[2]));
9         int[] heights = new int[disks.size()];
10        for (int i = 0; i < disks.size(); i++) {
11            heights[i] = disks.get(i)[2];
12        }
13        int[] sequences = new int[disks.size()];
14        for (int i = 0; i < disks.size(); i++) {
15            sequences[i] = Integer.MIN_VALUE;
16        }
17        int maxHeightIdx = 0;
18        for (int i = 1; i < disks.size(); i++) {
19            Integer[] currentDisk = disks.get(i);
20            for (int j = 0; j < i; j++) {
21                Integer[] otherDisk = disks.get(j);
22                if (areValidDimensions(otherDisk, currentDisk)) {
23                    if (heights[i] <= currentDisk[2] + heights[j]) {
24                        heights[i] = currentDisk[2] + heights[j];
25                        sequences[i] = j;
26                    }
27                }
28            }
29            if (heights[i] >= heights[maxHeightIdx]) {
30                maxHeightIdx = i;
31            }
32        }
33        return buildSequence(disks, sequences, maxHeightIdx);
34    }
35
36    private static boolean areValidDimensions(Integer[] otherDisk, Integer[] currentDisk) {
37        return (otherDisk[0] < currentDisk[0] && otherDisk[1] < currentDisk[1]);
38    }
39
40    private static List<Integer[]> buildSequence(List<Integer[]> disks, int[] sequences, int maxHeightIdx) {
41        List<Integer[]> sequence = new ArrayList<>();
42        int i = maxHeightIdx;
43        while (i >= 0) {
44            sequence.add(disks.get(i));
45            i = sequences[i];
46        }
47        sequence.reverse();
48        return sequence;
49    }
50}
```

Our Tests

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     // O(n^2) time | O(n) space
7     public static List<Integer[]> diskStacking(List<Integer[]> disks) {
8         disks.sort((disk1, disk2) -> disk1[2].compareTo(disk2[2]));
9         int[] heights = new int[disks.size()];
10        for (int i = 0; i < disks.size(); i++) {
11            heights[i] = disks.get(i)[2];
12        }
13        int[] sequences = new int[disks.size()];
14        for (int i = 0; i < disks.size(); i++) {
15            sequences[i] = Integer.MIN_VALUE;
16        }
17        int maxHeightIdx = 0;
18        for (int i = 1; i < disks.size(); i++) {
19            Integer[] currentDisk = disks.get(i);
20            for (int j = 0; j < i; j++) {
21                Integer[] otherDisk = disks.get(j);
22                if (areValidDimensions(otherDisk, currentDisk)) {
23                    if (heights[i] <= currentDisk[2] + heights[j]) {
24                        heights[i] = currentDisk[2] + heights[j];
25                        sequences[i] = j;
26                    }
27                }
28            }
29            if (heights[i] >= heights[maxHeightIdx]) {
30                maxHeightIdx = i;
31            }
32        }
33        return buildSequence(disks, sequences, maxHeightIdx);
34    }
35
36    private static boolean areValidDimensions(Integer[] otherDisk, Integer[] currentDisk) {
37        return (otherDisk[0] < currentDisk[0] && otherDisk[1] < currentDisk[1]);
38    }
39
40    private static List<Integer[]> buildSequence(List<Integer[]> disks, int[] sequences, int maxHeightIdx) {
41        List<Integer[]> sequence = new ArrayList<>();
42        int i = maxHeightIdx;
43        while (i >= 0) {
44            sequence.add(disks.get(i));
45            i = sequences[i];
46        }
47        sequence.reverse();
48        return sequence;
49    }
50}
```

Solution 1 Solution 2 Solution 3

```
1 import java.util.*;
2
3 class Program {
4     public static List<Integer[]> diskStacking(List<Integer[]> disks) {
5         // Write your code here.
6         return null;
7     }
8 }
9
```

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 import java.util.*;
4
5 class Program {
6     // O(n^2) time | O(n) space
7     public static List<Integer[]> diskStacking(List<Integer[]> disks) {
8         disks.sort((disk1, disk2) -> disk1[2].compareTo(disk2[2]));
9         int[] heights = new int[disks.size()];
10        for (int i = 0; i < disks.size(); i++) {
11            heights[i] = disks.get(i)[2];
12        }
13        int[] sequences = new int[disks.size()];
14        for (int i = 0; i < disks.size(); i++) {
15            sequences[i] = Integer.MIN_VALUE;
16        }
17        int maxHeightIdx = 0;
18        for (int i = 1; i < disks.size(); i++) {
19            Integer[] currentDisk = disks.get(i);
20            for (int j = 0; j < i; j++) {
21                Integer[] otherDisk = disks.get(j);
22                if (areValidDimensions(otherDisk, currentDisk)) {
23                    if (heights[i] <= currentDisk[2] + heights[j]) {
24                        heights[i] = currentDisk[2] + heights[j];
25                        sequences[i] = j;
26                    }
27                }
28            }
29            if (heights[i] >= heights[maxHeightIdx]) {
30                maxHeightIdx = i;
31            }
32        }
33        return buildSequence(disks, sequences, maxHeightIdx);
34    }
35
36    private static boolean areValidDimensions(Integer[] otherDisk, Integer[] currentDisk) {
37        return (otherDisk[0] < currentDisk[0] && otherDisk[1] < currentDisk[1]);
38    }
39
40    private static List<Integer[]> buildSequence(List<Integer[]> disks, int[] sequences, int maxHeightIdx) {
41        List<Integer[]> sequence = new ArrayList<>();
42        int i = maxHeightIdx;
43        while (i >= 0) {
44            sequence.add(disks.get(i));
45            i = sequences[i];
46        }
47        sequence.reverse();
48        return sequence;
49    }
50}
```

```

10  count = Integer(0); expected = 0; @range = Integer(0);
11  expected.add(Integer(0)); @, @, @;
12  while count < expected || @range < @ || @range < @;
13  }
14
15  @Post
16  @Test void test() {
17      count = Integer(0); @ = 0; @range = Integer(0);
18      @.add(Integer(0)); @, @, @;
19      @.add(Integer(0)); @, @, @;
20      count = Integer(0); expected = 0; @range = Integer(0);
21      expected.add(Integer(0)); @, @, @;
22      expected.add(Integer(0)); @, @, @;
23      while count < expected || @range < @ || @range < @;
24  }

```

Run or submit code when you're ready.