Our Solution(s) Run Code

Solution 1

```java
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

import java.util.*;

class Program {
  // O(n^2) time | O(n) space
  public static List<Integer[]> diskStacking(List<Integer[]> disks) {
    disks.sort((disk1, disk2) -> disk1[2].compareTo(disk2[2]));
    int[] heights = new int[disks.size()];
    for (int i = 0; i < disks.size(); i++) {
      heights[i] = disks.get(i)[2];
    }
    int[] sequences = new int[disks.size()];
    for (int i = 0; i < disks.size(); i++) {
      sequences[i] = Integer.MIN_VALUE;
    }
    int maxHeightIdx = 0;
    for (int i = 1; i < disks.size(); i++) {
      Integer[] currentDisk = disks.get(i);
      for (int j = 0; j < i; j++) {
        Integer[] otherDisk = disks.get(j);
        if (areValidDimensions(otherDisk, currentDisk)) {
          if (heights[i] <= currentDisk[2] + heights[j]) {
            heights[i] = currentDisk[2] + heights[j];
            sequences[i] = j;
          }
        }
      }
      if (heights[i] >= heights[maxHeightIdx]) {
        maxHeightIdx = i;
      }
    }
    return buildSequence(disks, sequences, maxHeightIdx);
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```java
import java.util.*;

class Program {
  public static List<Integer[]> diskStacking(List<Integer[]> disks) {
    // Write your code here.
    return null;
  }
}

```

Our Tests

Custom Output Submit Code

```
    int[][] expected = new String[][];
    expected.add(new Integer[] {0, 1, 2});
    assertEquals(input, expected);
  }

  @Test
  public void Test3() {
    int[][] input = new String[][];
    input.add(new Integer[] {0, 1, 2});
    input.add(new Integer[] {0, 1, 2});
    int[][] expected = new String[][];
    expected.add(new Integer[] {0, 1, 2});
    expected.add(new Integer[] {0, 1, 2});
    assertEquals(input, expected);
  }
```

Run or submit code when you're ready.