Solution 1

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

using System;
using System.Collections.Generic;

public class Program {
  // O(n^2) time | O(n) space
  public static List<int[]> DiskStacking(List<int[]> disks) {
    disks.Sort((disk1, disk2) => disk1[2].CompareTo(disk2[2]));
    int[] heights = new int[disks.Count];
    for (int i = 0; i < disks.Count; i++) {
      heights[i] = disks[i][2];
    }
    int[] sequences = new int[disks.Count];
    for (int i = 0; i < disks.Count; i++) {
      sequences[i] = Int32.MinValue;
    }
    int maxHeightIdx = 0;
    for (int i = 1; i < disks.Count; i++) {
      int[] currentDisk = disks[i];
      for (int j = 0; j < i; j++) {
        int[] otherDisk = disks[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);
  }

  public static bool areValidDimensions(int[] o, int[] c) {
    return o[0] < c[0] && o[1] < c[1] && o[2] < c[2];
  }

  public static List<int[]> buildSequence(List<int[]> array, int[] sequences,
    int currentIdx) {
    List<int[]> sequence = new List<int[]>();
    while (currentIdx != Int32.MinValue) {
      sequence.Insert(0, array[currentIdx]);
      currentIdx = sequences[currentIdx];
    }
    return sequence;
  }
}

```