Solution 1

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

#include <vector>
#include <algorithm>
#include <climits>
using namespace std;

bool areValidDimensions(vector<int> o, vector<int> c);
vector<vector<int>> buildSequence(vector<vector<int>> array,
                                  vector<int> sequences, int currentIdx);

// O(n^2) time | O(n) space
vector<vector<int>> diskStacking(vector<vector<int>> disks) {
  sort(disks.begin(), disks.end(),
       [](vector<int> &a, vector<int> &b) { return a[2] < b[2]; });
  vector<int> heights;
  for (int i = 0; i < disks.size(); i++) {
    heights.push_back(disks[i][2]);
  }
  vector<int> sequences;
  for (int i = 0; i < disks.size(); i++) {
    sequences.push_back(INT_MIN);
  }
  int maxHeightIdx = 0;
  for (int i = 1; i < disks.size(); i++) {
    vector<int> currentDisk = disks[i];
    for (int j = 0; j < i; j++) {
      vector<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);
}

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

vector<vector<int>> buildSequence(vector<vector<int>> array,
                                  vector<int> sequences, int currentIdx) {
  vector<vector<int>> sequence;
  while (currentIdx != INT_MIN) {
    sequence.insert(sequence.begin(), array[currentIdx]);
    currentIdx = sequences[currentIdx];
  }
  return sequence;
}

```