

Our Solution(s)

Run Code

Your Solutions

Run Code

Solution 1

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <algorithm>
5 #include <climits>
6 using namespace std;
7
8 bool areValidDimensions(vector<int> a, vector<int> b) {
9     vector<vector<int>> buildSequence(vector<vector<int>> array,
10                                     vector<int> sequences, int currentIdx) {
11
12         // O(n^2) time | O(n) space
13         vector<vector<int>> diskStacking(vector<vector<int>> disks) {
14             sort(disks.begin(), disks.end(),
15                 [](vector<int> &a, vector<int> &b) { return a[2] < b[2]; });
16             vector<int> heights;
17             for (int i = 0; i < disks.size(); i++) {
18                 heights.push_back(disks[i][2]);
19             }
20             vector<int> sequences;
21             for (int i = 0; i < disks.size(); i++) {
22                 sequences.push_back(INT_MIN);
23             }
24             int maxHeightIdx = 0;
25             for (int i = 1; i < disks.size(); i++) {
26                 vector<int> currentDisk = disks[i];
27                 for (int j = 0; j < i; j++) {
28                     vector<int> otherDisk = disks[j];
29                     if (areValidDimensions(otherDisk, currentDisk)) {
30                         if (heights[i] <= currentDisk[2] + heights[j]) {
31                             heights[i] = currentDisk[2] + heights[j];
32                             sequences[i] = j;
33                         }
34                     }
35                 }
36             }
37             return heights;
38         }
39     }
40 }
```

Our Tests

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <algorithm>
5 #include <climits>
6 using namespace std;
7
8 bool areValidDimensions(vector<int> a, vector<int> b) {
9     vector<vector<int>> buildSequence(vector<vector<int>> array,
10                                     vector<int> sequences, int currentIdx) {
11
12         // O(n^2) time | O(n) space
13         vector<vector<int>> diskStacking(vector<vector<int>> disks) {
14             sort(disks.begin(), disks.end(),
15                 [](vector<int> &a, vector<int> &b) { return a[2] < b[2]; });
16             vector<int> heights;
17             for (int i = 0; i < disks.size(); i++) {
18                 heights.push_back(disks[i][2]);
19             }
20             vector<int> sequences;
21             for (int i = 0; i < disks.size(); i++) {
22                 sequences.push_back(INT_MIN);
23             }
24             int maxHeightIdx = 0;
25             for (int i = 1; i < disks.size(); i++) {
26                 vector<int> currentDisk = disks[i];
27                 for (int j = 0; j < i; j++) {
28                     vector<int> otherDisk = disks[j];
29                     if (areValidDimensions(otherDisk, currentDisk)) {
30                         if (heights[i] <= currentDisk[2] + heights[j]) {
31                             heights[i] = currentDisk[2] + heights[j];
32                             sequences[i] = j;
33                         }
34                     }
35                 }
36             }
37             return heights;
38         }
39     }
40 }
```

Solution 1 Solution 2 Solution 3

```
1 #include <vector>
2 using namespace std;
3
4 vector<vector<int>> diskStacking(vector<vector<int>> disks) {
5     // Write your code here.
6     return {};
7 }
8
```

Custom Output

Submit Code

```
1 // Copyright © 2020 AlgoExpert, LLC. All rights reserved.
2
3 #include <vector>
4 #include <algorithm>
5 #include <climits>
6 using namespace std;
7
8 bool areValidDimensions(vector<int> a, vector<int> b) {
9     vector<vector<int>> buildSequence(vector<vector<int>> array,
10                                     vector<int> sequences, int currentIdx) {
11
12         // O(n^2) time | O(n) space
13         vector<vector<int>> diskStacking(vector<vector<int>> disks) {
14             sort(disks.begin(), disks.end(),
15                 [](vector<int> &a, vector<int> &b) { return a[2] < b[2]; });
16             vector<int> heights;
17             for (int i = 0; i < disks.size(); i++) {
18                 heights.push_back(disks[i][2]);
19             }
20             vector<int> sequences;
21             for (int i = 0; i < disks.size(); i++) {
22                 sequences.push_back(INT_MIN);
23             }
24             int maxHeightIdx = 0;
25             for (int i = 1; i < disks.size(); i++) {
26                 vector<int> currentDisk = disks[i];
27                 for (int j = 0; j < i; j++) {
28                     vector<int> otherDisk = disks[j];
29                     if (areValidDimensions(otherDisk, currentDisk)) {
30                         if (heights[i] <= currentDisk[2] + heights[j]) {
31                             heights[i] = currentDisk[2] + heights[j];
32                             sequences[i] = j;
33                         }
34                     }
35                 }
36             }
37             return heights;
38         }
39     }
40 }
```

```
10     return vector<int>{vector{1, 2, 3, 4, 5},
11                       vector{6, 7, 8, 9, 10}};
12 }
13
14 // Returns "Next Step 2", 10000
15 vector<vector<int>> nextStep2(const vector<int>& v, int n, int m, int k) {
16     vector<vector<int>> result{vector{v, v}, vector{v, v}};
17     return result;
18 }
19
20 // Returns "Next Step 2", 10000
21 vector<vector<int>> nextStep2(const vector<int>& v, int n, int m, int k) {
22     vector<vector<int>> result{vector{v, v}, vector{v, v}};
23     return result;
24 }
```

Run or submit code when you're ready.