Our Solution(s) Run Code

Solution 1

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

package main

import "sort"

type Disk []int
type Disks []Disk

func (disks Disks) Len() int           { return len(disks) }
func (disks Disks) Swap(i, j int)      { disks[i], disks[j] = disks[j]
func (disks Disks) Less(i, j int) bool { return disks[i][2] < disks[j]

func DiskStacking(input [][]int) [][]int {
  disks := make(Disks, len(input))
  for i, disk := range input {
    disks[i] = disk
  }
  sort.Sort(disks)
  heights := make([]int, len(disks))
  sequences := make([]int, len(disks))
  for i := range disks {
    heights[i] = disks[i][2]
    sequences[i] = -1
  }
  for i := 1; i < len(disks); i++ {
    disk := disks[i]
    for j := 0; j < i; j++ {
      other := disks[j]
      // If the conditions of disk stacking are met
      if areValidDimensions(other, disk) {
        // If it's an increase in size
        if heights[i] <= disk[2]+heights[j] {
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func DiskStacking(disks [][]int) [][]int {
  // Write your code here.
  return nil
}

```

Our Tests

Custom Output Submit Code

```
}

func (s *TestSuite) TestInvert(t *testing.T) {
    expected := [][]int{{0, 1, 0}, {1, 0, 1}, {0, 1, 0}}
    input := [][]int{{0, 1, 0}, {1, 0, 1}, {0, 1, 0}}
    output := InvertImage(input)
    require.Equal(t, expected, output)
}

func (s *TestSuite) TestInvert(t *testing.T) {
    expected := [][]int{{1, 1, 0, 0}, {1, 0, 0, 1}, {0, 1, 1, 1}, {1, 0, 1, 0}}
    input := [][]int{{1, 1, 0, 0}, {1, 0, 0, 1}, {0, 1, 1, 1}, {1, 0, 1, 0}}
    output := InvertImage(input)
    require.Equal(t, expected, output)
}
```

Run or submit code when you're ready.