Our Solution(s) Run Code

Solution 1 Solution 2 Solution 3

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

package main

// O(n) time | O(n) space - where in is the length of the input array
func MinRewards(scores []int) int {
  rewards := make([]int, len(scores))
  fill(rewards, 1)
  localMinIdxs := getLocalMinIdxs(scores)
  for _, localMinIdx := range localMinIdxs {
    expandFromLocalMinIdx(localMinIdx, scores, rewards)
  }
  return sum(rewards)
}

func getLocalMinIdxs(arr []int) []int {
  localMinIdxs := []int{}
  if len(arr) == 1 {
    localMinIdxs = append(localMinIdxs, 0)
    return localMinIdxs
  }
  for i := 0; i < len(arr); i++ {
    if i == 0 && arr[i] < arr[i+1] {
      localMinIdxs = append(localMinIdxs, i)
    }
    if i == len(arr)-1 && arr[i] < arr[i-1] {
      localMinIdxs = append(localMinIdxs, i)
    }
    if i == 0 || i == len(arr)-1 {
      continue
    }
    if arr[i] < arr[i+1] && arr[i] < arr[i-1] {
      localMinIdxs = append(localMinIdxs, i)
```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

```go
package main

func MinRewards(scores []int) int {
  // Write your code here.
  return -1
}

```

Our Tests

Custom Output Submit Code

```
}

func (s *TestSuite) TestCase2() *TestCase {
    input := []int{3}
    output := MinMoves(input)
    expected := 0
    require.Equal(t, expected, output)
}

func (s *TestSuite) TestCase3() *TestCase {
    input := []int{1, 10}
    output := MinMoves(input)
    expected := 9
    require.Equal(t, expected, output)
}
```

Run or submit code when you're ready.