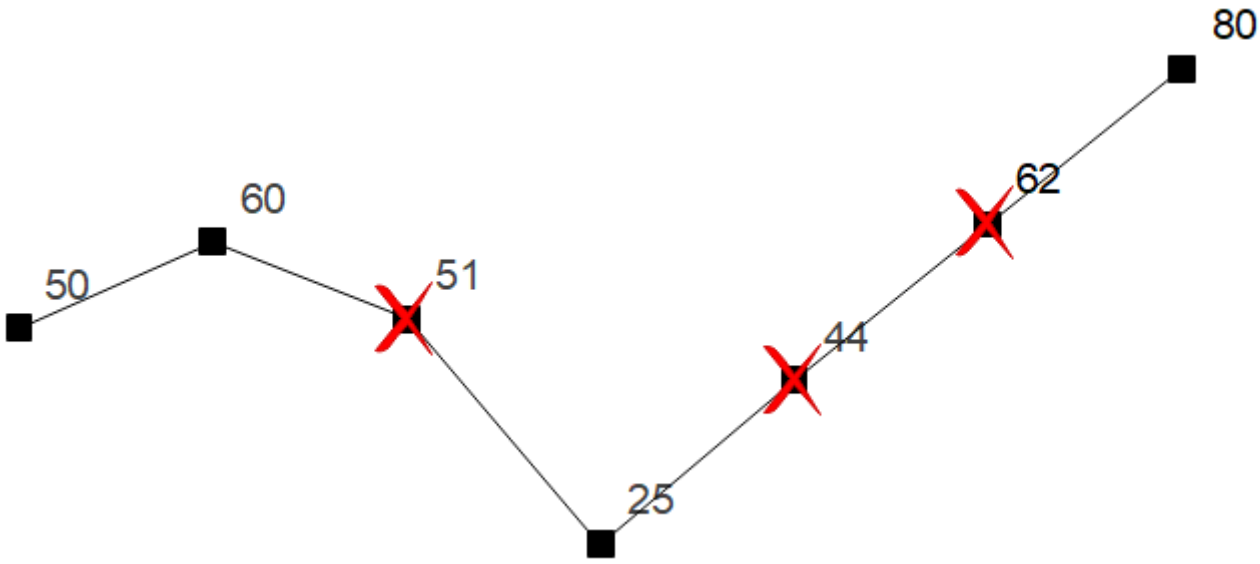
力扣

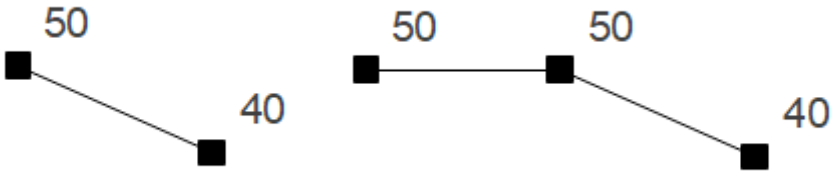
编号	分类	难度	我的题解	力扣题目链接
1	贪心、动态规划	中等	摆动序列	点击跳转

1. 摆动序列

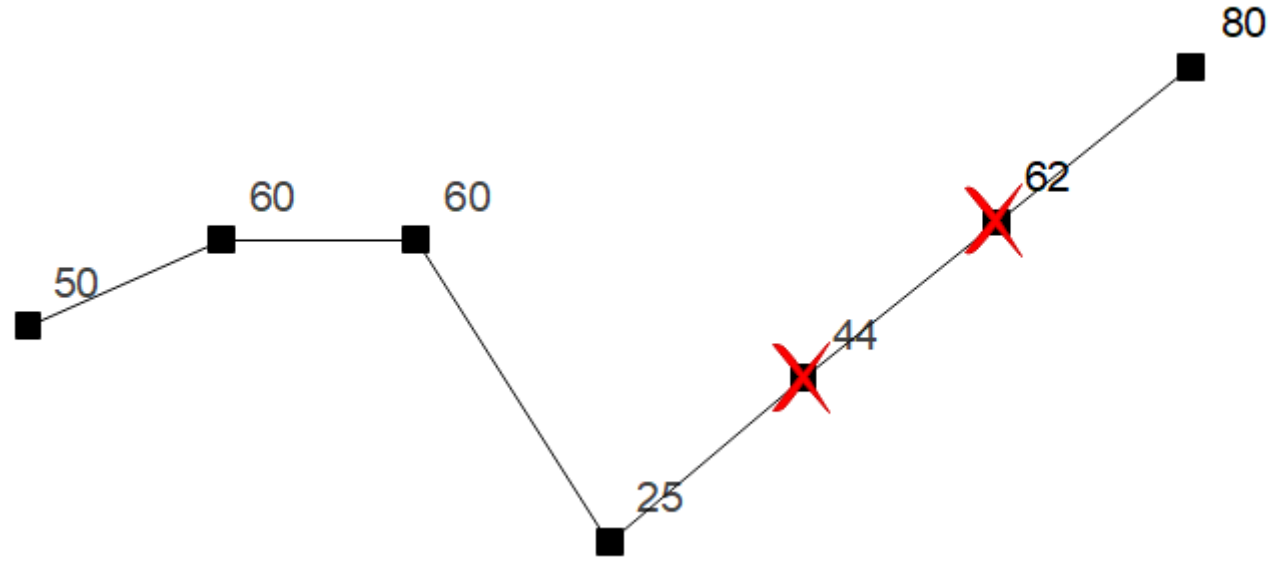
贪心解法

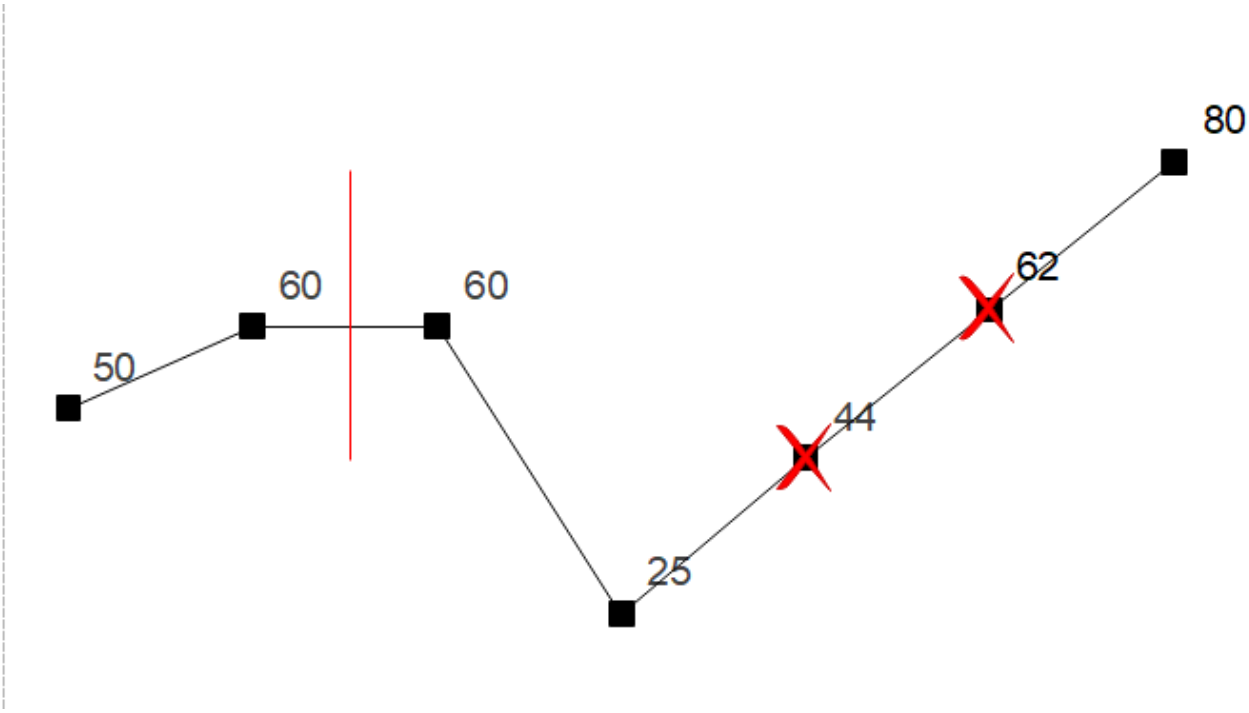
这里寻找贪心算法的局部最优，其实就是统计**局部**峰值点，统计峰值点关键在于如何解决边界峰值问题,如图2，3所示，[50,40]，其实可以看做[50,50,40]，那本质上这里出现了一个坡度，所以也是峰值点。说到这里我们也有了思路，用坡度去统计峰值点更合





值得注意的是，如图4所示，出现等值点60其实也算是一个峰值点，这会使我们的统计变得困难，但我们上面提到边界峰值点,如图5所示，我们可以分割成两部分，所以这里其实也等价于边界峰值点的统计问题。





领域点统计峰值
这里我们采用三个点去统计，会出现各种各样问题，特别边界点问题最不好统计，因此也没有再去深究代码问题（以下代码面对一些情况是有问题的）

```
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        int len = 0;
        int i = 0, j = 1;

        if (nums.size() == 1) return 1;

        while (nums[i] == nums[j]) {
            i++;
            j++;
```

```
            if (j == nums.size()) return 1;
        }

        len = 1;

        while (j < nums.size() - 1) {
            if ((nums[j] - nums[i]) * (nums[j] - nums[j + 1]) > 0) len++;

            j++;
            while (j < nums.size() - 1 && (nums[j] - nums[i]) * (nums[j] - nums[j
+ 1]) == 0) {
                j++;
                if (j < nums.size() - 1 && nums[j] - nums[j + 1] != 0) {
                    if ((nums[j] - nums[i]) * (nums[j] - nums[j + 1]) > 0) {
                        len++;
                    }
                    i = j - 1;
                    j++;
                    break;
                }
                else break;
            }
            i++;
        }
        return len + 1;
    }
};
```

坡度统计峰值法

```
class Solution {
public:
    int wiggleMaxLength(vector<int>& nums) {
        if (nums.size() <= 1) return nums.size();
        int curDiff = 0; // 当前一对差值
        int preDiff = 0; // 前一对差值
        int result = 1;  // 记录峰值个数，序列默认序列最右边有一个峰值
        for (int i = 0; i < nums.size() - 1; i++) {
            curDiff = nums[i + 1] - nums[i];
            // 出现峰值
            if ((curDiff > 0 && preDiff <= 0) || (preDiff >= 0 && curDiff < 0)) {
                result++;
                preDiff = curDiff;
            }
        }
        return result;
    }
};
```