Breaking the Records

Editorial by StefanK

Observation

To determine when Maria breaks her running records for best and worst scores, we must loop through her scores for each game in the season and track her current best and worst scores after each game. In addition, we must create two counter variables to track the number of times that the variables storing her best and worst scores are changed.

Solution

Because Maria's score from the first game of the season is the baseline for her records throughout the season, we initialize her best and worst score variables to her score for the first game (i.e., $score[0]$). We then loop through the remaining elements in the scores array and determine whether or not the current game's score breaks her best or worst record; as these are mutually exclusive (i.e., you'll never break the best and worst record during the same game), we can use an if - else if clause to reduce some redundant operations.

Notes

This approach is very straightforward and uses multiple variables to keep track of everything. It also makes up to two comparisons for each game because 'equal to' scenarios do not break a record. How can we reduce the number of comparisons using fewer variables while still writing readable code? Take some time to think about potential observations and review the code below.

Featured Solutions

Python 2

```
n = int(raw_input().strip())
score = map(int, raw_input().strip().split(' '))

max_score = score[0]
min_score = score[0]

ans1 = 0
ans2 = 0
for val in score:
    if val > max_score:
        max_score = val
        ans1 = ans1 + 1
    if val < min_score:
        min_score = val
        ans2 = ans2 + 1
print ans1, ans2
```

Java

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] score = new int[n];
        for(int score_i=0; score_i < n; score_i++){
            score[score_i] = in.nextInt();
        }
        int best = 0;
        int worst = 0;
        int bestCounter = 0;
```

STATISTICS

Difficulty: Easy

Time Complexity: $\mathcal{O(n)}

Required Knowledge: Arrays, Loops

Publish Date: Feb 10 2017

Originally featured in University CodeSprint 2

NEED HELP?

View discussions

View top submissions

```
        int worstCounter = 0;
        best = score[0];
        worst = score[0];
        for (int i = 1; i < score.length; i++) {
            if (score[i] < worst) {
                worst = score[i];
                worstCounter++;
            }
            if (score[i] > best) {
                best = score[i];
                bestCounter++;
            }
        }
        System.out.println(bestCounter + " " + worstCounter);
    }
}
```

Set by AllisonP

Problem Setter's code:

Java (Multiple Variables)

```
import java.util.*;

public class Solution {

    public static String solve(int[] scores) {
        // Set initial score
        int min = scores[0];
        int max = min;

        // Variables to track number of broken records
        int minCount = 0;
        int maxCount = 0;

        // Tally number of broken records
        for (int i = 1; i < scores.length; i++) {
            // Temp var to hold score for current record
            int current = scores[i];

            // Check if breaking a max record
            if (current > max) {
                max = current;
                maxCount++;
            }
            // If not breaking a max record, check if breaking a min record
            else if (current < min) {
                min = current;
                minCount++;
            }
        }

        // Return number of broken records in the correct order
        return maxCount + " " + minCount;
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] score = new int[n];
        for(int score_i=0; score_i < n; score_i++){
            score[score_i] = in.nextInt();
        }
        in.close();

        System.out.println(solve(score));
    }
}
```

Java (Tree)

```
import java.util.*;

class Solution {
    public class Record {
        private int score;
```

```
        public Record left;
        public Record right;

        /**
        *    A best or worst score record.
        *    @param The record's score.
        **/
        public Record(int score) {
            this.score = score;
            this.left = null;
            this.right = null;
        }

        /**
        *    Checks for and inserts new records.
        *    @param score The score for the (potential) new best or worst record.
        **/
        public void insert(int score) {
            if (score < this.score) {
                if (this.left == null) {
                    this.left = new Record(score);
                }
                else {
                    this.left.insert(score);
                }

            }
            else if (score > this.score) {
                if (this.right == null) {
                    this.right = new Record(score);
                }
                else {
                    this.right.insert(score);
                }
            }
        }

        /**
        *    @return The number of times the worst record was broken (edges in left branch of
        **/
        public int depthLeft() {
            System.err.println("Worst Record: " + this.score);
            return (this.left == null) ? 0 : (1 + this.left.depthLeft());
        }

        /**
        *    @return The number of times the best record was broken (edges in right branch of
        **/
        public int depthRight() {
            System.err.println("Best Record: " + this.score);
            return (this.right == null) ? 0 : (1 + this.right.depthRight());
        }
    }

    public String solve(int[] scores) {
        Record root = new Record(scores[0]);

        for (int i = 1; i < scores.length; i++) {
            root.insert(scores[i]);
        }

        // Return number of broken records in the correct order
        return root.depthRight() + " " + root.depthLeft();
    }

    public static void main(String[] args) {
        Scanner in = new Scanner(System.in);
        int n = in.nextInt();
        int[] score = new int[n];
        for(int score_i=0; score_i < n; score_i++){
            score[score_i] = in.nextInt();
        }
        in.close();
        Solution s = new Solution();
        System.out.println(s.solve(score));
    }
}
```

Feedback

Was this editorial helpful?

Yes No