Codility_

Candidate Report: trainingKPE89R-X42

Check out Codility training tasks

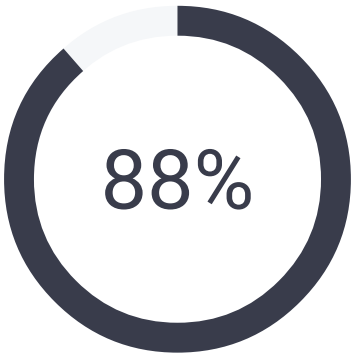
Test Name:

Summary Timeline

Tasks summary

Task	Time spent	Score
MaxProductOfThree C	10 min	88%

Total score



Tasks Details

Easy

1. MaxProductOfThree

Maximize A[P] * A[Q] * A[R] for any triplet (P, Q, R).

Task Score	Correctness	Performance
88%	75%	100%

Task description

A non-empty array A consisting of N integers is given. The *product* of triplet (P, Q, R) equates to A[P] * A[Q] * A[R] (0 ≤ P < Q < R < N).

For example, array A such that:

```
A[0] = -3
A[1] = 1
A[2] = 2
A[3] = -2
A[4] = 5
A[5] = 6
```

contains the following example triplets:

- (0, 1, 2), product is −3 * 1 * 2 = −6
- (1, 2, 4), product is 1 * 2 * 5 = 10
- (2, 4, 5), product is 2 * 5 * 6 = 60

Your goal is to find the maximal product of any triplet.

Write a function:

```
int solution(int A[], int N);
```

that, given a non-empty array A, returns the value of the maximal product of any triplet.

Solution

Programming language used:	C
Total time used:	10 minutes
Effective time used:	10 minutes
Notes:	*not defined yet*

Task timeline

11:13:29 11:22:33

Code: 11:22:33 UTC, c, final, score: **88** show code in pop-up

For example, given array A such that:

```
A[0] = -3
A[1] = 1
A[2] = 2
A[3] = -2
A[4] = 5
A[5] = 6
```

the function should return 60, as the product of triplet (2, 4, 5) is maximal.

Write an **efficient** algorithm for the following assumptions:

- N is an integer within the range [3..100,000];
- each element of array A is an integer within the range [−1,000..1,000].


```c
// you can write to stdout for debugging purposes, e.g.
// printf("this is a debug message\n");
#define get_min_three(x,y,z) ((x<=y && x<=z)?0:((y<=x && y<
#define abs(x) (x<0? (-x):x)

int solution(int A[], int N) {
    // write your code in C99 (gcc 6.2.0)
    int res=0, res_tmp=0;
    int i = 0;
    int M[3], Mindex[3];
    int tmp_index, tmp_val;
    int neg_cnt = 0;
    if (N==3) return A[0]*A[1]*A[2];

    M[0] = A[0];
    M[1] = A[1];
    M[2] = A[2];

    Mindex[0] = 0;
    Mindex[1] = 1;
    Mindex[2] = 2;

    for (i = 3; i<N; i++){
        tmp_index = get_min_three(abs(M[0]),abs(M[1]),abs(M
        if (abs(A[i]) > abs(M[tmp_index])){
            M[tmp_index] = A[i];
            Mindex[tmp_index] = i;
        }
    }
    res = M[0]*M[1]*M[2];

    neg_cnt = (M[0]<0)+(M[1]<0)+(M[2]<0);
    //printf("nc%d: %d*%d*%d ", neg_cnt,M[0],M[1],M[2]);
    if (neg_cnt==3) { //included all A[] are negative value
        tmp_val = -1000;
        for (i = 0; i<N; i++){
            if (A[i] > tmp_val) //pos
                tmp_val = A[i];
        }
        if (tmp_val==0)
            return 0;
        else if (tmp_val>0){
            tmp_index = get_min_three(abs(M[0]),abs(M[1]),a
            M[tmp_index] = tmp_val;
        }
        else{
            M[0] = A[0];
            M[1] = A[1];
            M[2] = A[2];

            Mindex[0] = 0;
            Mindex[1] = 1;
            Mindex[2] = 2;
            for (i = 3; i<N; i++){
            tmp_index = get_min_three(-M[0],-M[1],-M[2]);
            if (A[i] > M[tmp_index]){
                M[tmp_index] = A[i];
                Mindex[tmp_index] = i;
        }
    }
        }

        res = M[0]*M[1]*M[2];
    }
    else if (neg_cnt==1){ //considering pos and neg substit
        tmp_val = 0;

        //pos
        for (i = 0; i<N; i++){
            if ( A[i] > tmp_val && i^Mindex[0] && i^Mindex[
                tmp_val = A[i];
        }
        if (tmp_val>=0){
            tmp_index = M[0]<0?0:(M[1]<0?1:2);
            res_tmp = res*tmp_val/M[tmp_index];
    //printf("pos%d: %d*%d*%d ", tmp_val, M[0],M[1],M[2]);
```

```
77          }
78
79          //neg
80          for (i = 0; i<N; i++){
81              if ( A[i] < tmp_val && i^Mindex[0] && i^Mindex[
82                  tmp_val = A[i];
83          }
84          if (tmp_val<=0){
85              tmp_index = M[0]<0?(M[1]>=M[2]?2:1):(M[1]<0?(M[
86      //printf("neg%d: %d*%d*%d ", tmp_val, M[0],M[1],M[2]);
87              M[tmp_index] = tmp_val;
88              res = M[0]*M[1]*M[2];
89          }
90
91          if (res<res_tmp)
92              res = res_tmp;
93      }
94
95      return res;
96  }
```

Analysis summary

The following issues have been detected: wrong answers.

Analysis

Detected time complexity: **O(N * log(N))**

expand all	Example tests	
▶	example example test	✓ OK
expand all	**Correctness tests**	
▶	one_triple three elements	✓ OK
▶	simple1 simple tests	✓ OK
▶	simple2 simple tests	✓ OK
▼	small_random random small, length = 100	✗ WRONG ANSWER got 943825005 expected 964280454
1.	0.001 s	**WRONG ANSWER**, got 943825005 expected 964280454
expand all	**Performance tests**	
▶	medium_range -1000, -999, ... 1000, length = ~1,000	✓ OK
▶	medium_random random medium, length = ~10,000	✓ OK
▶	large_random random large, length = ~100,000	✓ OK
▶	large_range 2000 * (-10..10) + [-1000, 500, -1]	✓ OK
▶	extreme_large (-2, .., -2, 1, .., 1) and (MAX_INT)..(MAX_INT), length = ~100,000	✓ OK

The PDF version of this report that may be downloaded on top of this site may contain sensitive data including personal information. For security purposes, we recommend you remove it from your system once reviewed.