Maximum Product Subarray

Find the contiguous subarray within an array (containing at least one number) which has the largest product.

For example, given the array `[2,3,-2,4]`,
the contiguous subarray `[2,3]` has the largest product = `6`.

Solution 1

```java
public int maxProduct(int[] A) {
    if (A.length == 0) {
        return 0;
    }

    int maxherepre = A[0];
    int minherepre = A[0];
    int maxsofar = A[0];
    int maxhere, minhere;

    for (int i = 1; i < A.length; i++) {
        maxhere = Math.max(Math.max(maxherepre * A[i], minherepre * A[i]), A[i]);
        minhere = Math.min(Math.min(maxherepre * A[i], minherepre * A[i]), A[i]);
        maxsofar = Math.max(maxhere, maxsofar);
        maxherepre = maxhere;
        minherepre = minhere;
    }
    return maxsofar;
}
```

Note: There's no need to use O(n) space, as all that you need is a minhere and maxhere. (local max and local min), then you can get maxsofar (which is global max) from them.

There's a chapter in Programming Pearls 2 that discussed the MaxSubArray problem, the idea is similar.

written by rliu054 original link here

Solution 2

```
int maxProduct(int A[], int n) {
    // store the result that is the max we have found so far
    int r = A[0];

    // imax/imin stores the max/min product of
    // subarray that ends with the current number A[i]
    for (int i = 1, imax = r, imin = r; i < n; i++) {
        // multiplied by a negative makes big number smaller, small number bigger
        // so we redefine the extremums by swapping them
        if (A[i] < 0)
            swap(imax, imin);

        // max/min product for the current number is either the current number it
self
        // or the max/min by the previous number times the current one
        imax = max(A[i], imax * A[i]);
        imin = min(A[i], imin * A[i]);

        // the newly computed max value is a candidate for our global result
        r = max(r, imax);
    }
    return r;
}
```

written by mzchen original link here

Solution 3

```
class Solution {
 // author : s2003zy
 // weibo : http://weibo.com/574433433
 // blog : http://s2003zy.com
 // Time : O(n)
 // Space : O(1)
 public:
    int maxProduct(int A[], int n) {
        int frontProduct = 1;
        int backProduct = 1;
        int ans = INT_MIN;
        for (int i = 0; i < n; ++i) {
            frontProduct *= A[i];
            backProduct *= A[n - i - 1];
            ans = max(ans,max(frontProduct,backProduct));
            frontProduct = frontProduct == 0 ? 1 : frontProduct;
            backProduct = backProduct == 0 ? 1 : backProduct;
        }
        return ans;
    }
};
```

written by songzy982 original link here