techiedelight.com

Maximum Product Subarray Problem | Techie Delight

3-4 minutes

Given an integer array, find the subarray that has the maximum product of its elements. The solution should return the maximum product of elements among all possible subarrays.

For example,

Input: { -6, 4, -5, 8, -10, 0, 8 }
Output: 1600
Explanation: The maximum product subarray is {4, -5, 8, -10} having product 1600

Input: { 40, 0, -20, -10 }
Output: 200
Explanation: The maximum product subarray is {-20, -10} having product 200

Practice this problem

The problem differs from the problem of finding the maximum product subsequence. Unlike subsequences, subarrays are required to occupy consecutive positions within the original array.

A naive solution would be to consider every subarray and find the

product of their elements. Finally, return the maximum product found among all subarrays. The implementation can be seen here. The time complexity of this solution is $O(n^2)$, where `n` is the size of the input.

A better solution will be to maintain two variables to store the maximum and minimum product ending in the current position. Then traverse the array once, and for every index `i` in the array, update the maximum and minimum product ending at `A[i]`. Update the result if the maximum product ending at any index is more than the maximum product found so far.

Following is the C, Java, and Python implementation based on the above idea:

- C
- Java
- Python

C

```
1  #include <stdio.h>
2  // Utility function to find a minimum of two numbers
3  int min(int x, int y) {
4      return (x < y) ? x : y;
5  }
6  // Utility function to find a maximum of two numbers
```

```
7  int max(int x, int y) {
8      return (x > y) ? x : y;
9  }
10 // Function to return the maximum product of a subarray of a
11 given array
12 int findMaxProduct(int arr[], int n)
13 {
14     // base case
15     if (n == 0) {
16         return 0;
17     }
18     // maintain two variables to store the maximum and
19 minimum product
20     // ending at the current index
21     int max_ending = arr[0], min_ending = arr[0];
22     // to store the maximum product subarray found so far
23     int max_so_far = arr[0];
24     // traverse the given array
25     for (int i = 1; i < n; i++)
26     {
27         int temp = max_ending;
28         // update the maximum product ending at the current
29 index
```

```
30          max_ending = max(arr[i], max(arr[i] * max_ending,
31  arr[i] * min_ending));
32          // update the minimum product ending at the current
    index
33          min_ending = min(arr[i], min(arr[i] * temp, arr[i] *
34  min_ending));
35          max_so_far = max(max_so_far, max_ending);
36      }
37      // return maximum product
38      return max_so_far;
39  }
40  int main(void)
41  {
42      int arr[] = { -6, 4, -5, 8, -10, 0, 8 };
43      int n = sizeof(arr) / sizeof(arr[0]);
44      printf("The maximum product of a subarray is %d",
45              findMaxProduct(arr, n));
46      return 0;
47  }
48
49
50
51
52
```

```
53
54
55
```

Download Run Code

Output:

The maximum product of a subarray is 1600

Java

Python

The time complexity of the above solution is O(n) and doesn't require any extra space.

Thanks for reading.

Please use our online compiler to post code in comments using C, C++, Java, Python, JavaScript, C#, PHP, and many more popular programming languages.

Like us? Refer us to your friends and help us grow. Happy coding 🙂