A Product Array Puzzle

Given an array arr[] of n integers, construct a Product Array prod[] (of same size) such that prod[i] is equal to the product of all the elements of arr[] except arr[i]. Solve it **without division operator and in O(n)**.

Example:
arr[] = {10, 3, 5, 6, 2}
prod[] = {180, 600, 360, 300, 900}

Algorithm:
1) Construct a temporary array left[] such that left[i] contains product of all elements on left of arr[i] excluding arr[i].
2) Construct another temporary array right[] such that right[i] contains product of all elements on on right of arr[i] excluding arr[i].
3) To get prod[], multiply left[] and right[].

Implementation:

```
#include<stdio.h>
#include<stdlib.h>

/* Function to print product array for a given array
 arr[] of size n */
void productArray(int arr[], int n)
{
  /* Allocate memory for temporary arrays left[] and right[] */
  int *left = (int *)malloc(sizeof(int)*n);
  int *right = (int *)malloc(sizeof(int)*n);

  /* Allocate memory for the product array */
  int *prod = (int *)malloc(sizeof(int)*n);

  int i, j;

  /* Left most element of left array is always 1 */
  left[0] = 1;

  /* Rightmost most element of right array is always 1 */
  right[n-1] = 1;

  /* Construct the left array */
  for(i = 1; i < n; i++)
    left[i] = arr[i-1]*left[i-1];
```

```
  /* Construct the right array */
  for(j = n-2; j >=0; j--)
    right[j] = arr[j+1]*right[j+1];

  /* Construct the product array using
    left[] and right[] */
  for (i=0; i<n; i++)
    prod[i] = left[i] * right[i];

  /* print the constructed prod array */
  for (i=0; i<n; i++)
    printf("%d ", prod[i]);

  return;
}

/* Driver program to test above functions */
int main()
{
  int arr[] = {10, 3, 5, 6, 2};
  int n = sizeof(arr)/sizeof(arr[0]);
  printf("The product array is: \n");
  productArray(arr, n);
  getchar();
}
```

Run on IDE

Time Complexity: O(n)

Space Complexity: O(n)

Auxiliary Space: O(n)

The above method can be optimized to work in space complexity O(1). Thanks to Dileep for suggesting the below solution.

```
void productArray(int arr[], int n)
{
  int i, temp = 1;

  /* Allocate memory for the product array */
  int *prod = (int *)malloc(sizeof(int)*n);

  /* Initialize the product array as 1 */
  memset(prod, 1, n);

  /* In this loop, temp variable contains product of
    elements on left side excluding arr[i] */
  for(i=0; i<n; i++)
  {
    prod[i] = temp;
    temp *= arr[i];
  }

  /* Initialize temp to 1 for product on right side */
  temp = 1;

  /* In this loop, temp variable contains product of
    elements on right side excluding arr[i] */
  for(i= n-1; i>=0; i--)
  {
    prod[i] *= temp;
```

```
    temp *= arr[i];
  }

  /* print the constructed prod array */
  for (i=0; i<n; i++)
    printf("%d ", prod[i]);

  return;
}
```

Run on IDE

Time Complexity: O(n)

Space Complexity: O(n)

Auxiliary Space: O(1)

Please write comments if you find the above code/algorithm incorrect, or find better ways to solve the same problem.

91 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

2.5 Average Difficulty : **2.5/5.0**
Based on **10** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.