Maximum product of a triplet (subsequence of size 3) in array

Difficulty Level : Medium • Last Updated : 13 Jul, 2022

Given an integer array, find a maximum product of a triplet in array.

Examples:

```
Input:  [10, 3, 5, 6, 20]
Output: 1200
Multiplication of 10, 6 and 20

Input:  [-10, -3, -5, -6, -20]
Output: -90

Input:  [1, -4, 3, -6, 7, 0]
Output: 168
```


Approach 1 (Naive, O(n^3) time, O(1) Space)

A simple solution is to check for every triplet using three nested loops. Below is its implementation :

C++

```cpp
// A C++ program to find a maximum product of a
// triplet in array of integers
#include <bits/stdc++.h>
using namespace std;

/* Function to find a maximum product of a triplet
   in array of integers of size n */
int maxProduct(int arr[], int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // will contain max product
    int max_product = INT_MIN;

    for (int i = 0; i < n - 2; i++)
        for (int j = i + 1; j < n - 1; j++)
            for (int k = j + 1; k < n; k++)
                max_product = max(max_product,
                        arr[i] * arr[j] * arr[k]);

    return max_product;
}

// Driver program to test above functions
int main()
{
    int arr[] = { 10, 3, 5, 6, 20 };
    int n = sizeof(arr) / sizeof(arr[0]);

    int max = maxProduct(arr, n);

    if (max == -1)
        cout << "No Triplet Exists";
    else
        cout << "Maximum product is " << max;

    return 0;
}
```

Java

```java
// A Java program to find a
// maximum product of a
// triplet in array of integers

class GFG {

// Function to find a maximum
// product of a triplet in array
// of integers of size n
static int maxProduct(int []arr, int n)
{

    // if size is less than
    // 3, no triplet exists
    if (n < 3)
        return -1;

    // will contain max product
    int max_product = Integer.MIN_VALUE;

    for (int i = 0; i < n - 2; i++)
        for (int j = i + 1; j < n - 1; j++)
            for (int k = j + 1; k < n; k++)
                max_product = Math.max(max_product,
                          arr[i] * arr[j] * arr[k]);

    return max_product;
}

    // Driver Code
    public static void main (String [] args)
    {
        int []arr = { 10, 3, 5, 6, 20 };
        int n = arr.length;;

        int max = maxProduct(arr, n);

        if (max == -1)
            System.out.println("No Triplet Exists");
        else
            System.out.println("Maximum product is " + max);
    }
}
```

Python3

```python
# Python3 program to find a maximum
# product of a triplet in array
# of integers
import sys

# Function to find a maximum
# product of a triplet in array
# of integers of size n
def maxProduct(arr, n):

    # if size is less than 3,
    # no triplet exists
    if n < 3:
        return -1

    # will contain max product
    max_product = -(sys.maxsize - 1)

    for i in range(0, n - 2):
        for j in range(i + 1, n - 1):
            for k in range(j + 1, n):
                max_product = max(
                    max_product, arr[i]
                    * arr[j] * arr[k])

    return max_product

# Driver Program
arr = [10, 3, 5, 6, 20]
n = len(arr)

max = maxProduct(arr, n)

if max == -1:
    print("No Tripplet Exits")
else:
    print("Maximum product is", max)

# This code is contributed by Shrikant13
```

C#

```
// A C# program to find a
// maximum product of a
// triplet in array of integers
using System;
class GFG {

// Function to find a maximum
// product of a triplet in array
// of integers of size n
static int maxProduct(int []arr, int n)
{

    // if size is less than
    // 3, no triplet exists
    if (n < 3)
        return -1;

    // will contain max product
    int max_product = int.MinValue;

    for (int i = 0; i < n - 2; i++)
        for (int j = i + 1; j < n - 1; j++)
            for (int k = j + 1; k < n; k++)
                max_product = Math.Max(max_product,
                          arr[i] * arr[j] * arr[k]);

    return max_product;
}

    // Driver Code
    public static void Main ()
    {
        int []arr = { 10, 3, 5, 6, 20 };
        int n = arr.Length;;

        int max = maxProduct(arr, n);

        if (max == -1)
            Console.WriteLine("No Triplet Exists");
        else
            Console.WriteLine("Maximum product is " + max);
    }
```

```
}

// This code is contributed by anuj_67.
```

PHP

```
<?php
// A PHP program to find a
// maximum product of a
// triplet in array of integers

// Function to find a maximum
// product of a triplet
// in array of integers of
// size n
function maxProduct($arr, $n)
{
    $INT_MIN = 0;

    // if size is less than
    // 3, no triplet exists
    if ($n < 3)
        return -1;

    // will contain max product
    $max_product = $INT_MIN;

    for ($i = 0; $i < $n - 2; $i++)
        for ($j = $i + 1; $j < $n - 1; $j++)
            for ($k = $j + 1; $k < $n; $k++)
                $max_product = max($max_product,
                        $arr[$i] * $arr[$j] * $arr[$k]);

    return $max_product;
}

    // Driver Code
    $arr = array(10, 3, 5, 6, 20 );
    $n = sizeof($arr);
    $max = maxProduct($arr, $n);
    if ($max == -1)
        echo "No Triplet Exists";
    else
```

```
        echo "Maximum product is " ,$max;

// This code is contributed by nitin mittal.
?>
```

Javascript

```
<script>

// JavaScript program to find a
// maximum product of a
// triplet in array of integers


// Function to find a maximum
// product of a triplet in array
// of integers of size n
function maxProduct(arr, n)
{

    // if size is less than
    // 3, no triplet exists
    if (n < 3)
        return -1;

    // will contain max product
    let max_product = Number.MIN_VALUE;

    for (let i = 0; i < n - 2; i++)
        for (let j = i + 1; j < n - 1; j++)
            for (let k = j + 1; k < n; k++)
                max_product = Math.max(max_product,
                          arr[i] * arr[j] * arr[k]);

    return max_product;
}


// Driver Code

        let arr = [ 10, 3, 5, 6, 20 ];
        let n = arr.length;;

        let max = maxProduct(arr, n);
```

```
        if (max == -1)
            document.write("No Triplet Exists");
        else
            document.write("Maximum product is " + max);

</script>
```

Output :

```
Maximum product is 1200
```

Approach 2: O(n) Time, O(n) Space

1. Construct four auxiliary arrays leftMax[], rightMax[], leftMin[] and rightMin[] of same size as input array.
2. Fill leftMax[], rightMax[], leftMin[] and rightMin[] in below manner.
   - leftMax[i] will contain maximum element on left of arr[i] excluding arr[i]. For index 0, left will contain -1.
   - leftMin[i] will contain minimum element on left of arr[i] excluding arr[i]. For index 0, left will contain -1.
   - rightMax[i] will contain maximum element on right of arr[i] excluding arr[i]. For index n-1, right will contain -1.
   - rightMin[i] will contain minimum element on right of arr[i] excluding arr[i]. For index n-1, right will contain -1.
3. For all array indexes i except first and last index, compute maximum of arr[i]*x*y where x can be leftMax[i] or leftMin[i] and y can be rightMax[i] or rightMin[i].
4. Return the maximum from step 3.

Below is its implementation :

C++

```
// A C++ program to find a maximum product of a triplet
// in array of integers
```

```
#include <bits/stdc++.h>
using namespace std;

/* Function to find a maximum product of a triplet
in array of integers of size n */
int maxProduct(int arr[], int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Construct four auxiliary vectors
    // of size n and initialize them by -1
    vector<int> leftMin(n, -1);
    vector<int> rightMin(n, -1);
    vector<int> leftMax(n, -1);
    vector<int> rightMax(n, -1);

    // will contain max product
    int max_product = INT_MIN;

    // to store maximum element on left of array
    int max_sum = arr[0];

    // to store minimum element on left of array
    int min_sum = arr[0];

    // leftMax[i] will contain max element
    // on left of arr[i] excluding arr[i].
    // leftMin[i] will contain min element
    // on left of arr[i] excluding arr[i].
    for (int i = 1; i < n; i++)
    {
        leftMax[i] = max_sum;
        if (arr[i] > max_sum)
            max_sum = arr[i];

        leftMin[i] = min_sum;
        if (arr[i] < min_sum)
            min_sum = arr[i];
    }

    // reset max_sum to store maximum element on
    // right of array
    max_sum = arr[n - 1];
```

```
    // reset min_sum to store minimum element on
    // right of array
    min_sum = arr[n - 1];

    // rightMax[i] will contain max element
    // on right of arr[i] excluding arr[i].
    // rightMin[i] will contain min element
    // on right of arr[i] excluding arr[i].
    for (int j = n - 2; j >= 0; j--)
    {
        rightMax[j] = max_sum;
        if (arr[j] > max_sum)
            max_sum = arr[j];

        rightMin[j] = min_sum;
        if (arr[j] < min_sum)
            min_sum = arr[j];
    }

    // For all array indexes i except first and
    // last, compute maximum of arr[i]*x*y where
    // x can be leftMax[i] or leftMin[i] and
    // y can be rightMax[i] or rightMin[i].
    for (int i = 1; i < n - 1; i++)
    {
        int max1 = max(arr[i] * leftMax[i] * rightMax[i],
                       arr[i] * leftMin[i] * rightMin[i]);

        int max2 = max(arr[i] * leftMax[i] * rightMin[i],
                       arr[i] * leftMin[i] * rightMax[i]);

        max_product = max(max_product, max(max1, max2));
    }

    return max_product;
}

// Driver program to test above functions
int main()
{
    int arr[] = { 1, 4, 3, -6, -7, 0 };
    int n = sizeof(arr) / sizeof(arr[0]);

    int max = maxProduct(arr, n);
```

```
    if (max == -1)
        cout << "No Triplet Exists";
    else
        cout << "Maximum product is " << max;

    return 0;
}
```

Java

```
// A Java program to find a maximum product of a triplet
// in array of integers
import java.util.*;

class GFG
{

/* Function to find a maximum product of a triplet
in array of integers of size n */
static int maxProduct(int []arr, int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Construct four auxiliary vectors
    // of size n and initialize them by -1
    int[] leftMin = new int[n];
    int[] rightMin = new int[n];
    int[] leftMax = new int[n];
    int[] rightMax = new int[n];
    Arrays.fill(leftMin, -1);
    Arrays.fill(leftMax, -1);
    Arrays.fill(rightMax, -1);
    Arrays.fill(rightMin, -1);

    // will contain max product
    int max_product = Integer.MIN_VALUE;

    // to store maximum element on left of array
    int max_sum = arr[0];

    // to store minimum element on left of array
```

```
int min_sum = arr[0];

// leftMax[i] will contain max element
// on left of arr[i] excluding arr[i].
// leftMin[i] will contain min element
// on left of arr[i] excluding arr[i].
for (int i = 1; i < n; i++)
{
    leftMax[i] = max_sum;
    if (arr[i] > max_sum)
        max_sum = arr[i];

    leftMin[i] = min_sum;
    if (arr[i] < min_sum)
        min_sum = arr[i];
}

// reset max_sum to store maximum element on
// right of array
max_sum = arr[n - 1];

// reset min_sum to store minimum element on
// right of array
min_sum = arr[n - 1];

// rightMax[i] will contain max element
// on right of arr[i] excluding arr[i].
// rightMin[i] will contain min element
// on right of arr[i] excluding arr[i].
for (int j = n - 2; j >= 0; j--)
{
    rightMax[j] = max_sum;
    if (arr[j] > max_sum)
        max_sum = arr[j];

    rightMin[j] = min_sum;
    if (arr[j] < min_sum)
        min_sum = arr[j];
}

// For all array indexes i except first and
// last, compute maximum of arr[i]*x*y where
// x can be leftMax[i] or leftMin[i] and
// y can be rightMax[i] or rightMin[i].
for (int i = 1; i < n - 1; i++)
```

```
    {
        int max1 = Math.max(arr[i] * leftMax[i] * rightMax[i],
                    arr[i] * leftMin[i] * rightMin[i]);

        int max2 = Math.max(arr[i] * leftMax[i] * rightMin[i],
                    arr[i] * leftMin[i] * rightMax[i]);

        max_product = Math.max(max_product, Math.max(max1, max2));
    }

    return max_product;
}

// Driver code
public static void main (String[] args)
{
    int []arr = { 1, 4, 3, -6, -7, 0 };
    int n = arr.length;

    int max = maxProduct(arr, n);

    if (max == -1)
        System.out.println("No Triplet Exists");
    else
        System.out.println("Maximum product is "+max);
}
}

// This code is contributed by mits
```

Python3

```
# A Python3 program to find a maximum product
# of a triplet in array of integers
import sys

# Function to find a maximum product of a
# triplet in array of integers of size n
def maxProduct(arr, n):

    # If size is less than 3, no triplet exists
    if (n < 3):
        return -1
```

```
# Construct four auxiliary vectors
# of size n and initialize them by -1
leftMin = [-1 for i in range(n)]
rightMin = [-1 for i in range(n)]
leftMax = [-1 for i in range(n)]
rightMax = [-1 for i in range(n)]

# Will contain max product
max_product = -sys.maxsize - 1

# To store maximum element on
# left of array
max_sum = arr[0]

# To store minimum element on
# left of array
min_sum = arr[0]

# leftMax[i] will contain max element
# on left of arr[i] excluding arr[i].
# leftMin[i] will contain min element
# on left of arr[i] excluding arr[i].
for i in range(1, n):
    leftMax[i] = max_sum

    if (arr[i] > max_sum):
        max_sum = arr[i]

    leftMin[i] = min_sum

    if (arr[i] < min_sum):
        min_sum = arr[i]

# Reset max_sum to store maximum
# element on right of array
max_sum = arr[n - 1]

# Reset min_sum to store minimum
# element on right of array
min_sum = arr[n - 1]

# rightMax[i] will contain max element
# on right of arr[i] excluding arr[i].
# rightMin[i] will contain min element
# on right of arr[i] excluding arr[i].
```

```
    for j in range(n - 2, -1, -1):
        rightMax[j] = max_sum

        if (arr[j] > max_sum):
            max_sum = arr[j]

        rightMin[j] = min_sum

        if (arr[j] < min_sum):
            min_sum = arr[j]

    # For all array indexes i except first and
    # last, compute maximum of arr[i]*x*y where
    # x can be leftMax[i] or leftMin[i] and
    # y can be rightMax[i] or rightMin[i].
    for i in range(1, n - 1):
        max1 = max(arr[i] * leftMax[i] * rightMax[i],
                   arr[i] * leftMin[i] * rightMin[i])
        max2 = max(arr[i] * leftMax[i] * rightMin[i],
                   arr[i] * leftMin[i] * rightMax[i])

        max_product = max(max_product, max(max1, max2))

    return max_product

# Driver code
arr = [ 1, 4, 3, -6, -7, 0 ]
n = len(arr)
Max = maxProduct(arr, n)

if (Max == -1):
    print("No Triplet Exists")
else:
    print("Maximum product is", Max)

# This code is contributed by rag2127
```

C#

```
// A C# program to find a maximum product of a triplet
// in array of integers
using System;

class GFG
```

```
{

/* Function to find a maximum product of a triplet
in array of integers of size n */
static int maxProduct(int []arr, int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Construct four auxiliary vectors
    // of size n and initialize them by -1
    int[] leftMin=new int[n];
    int[] rightMin=new int[n];
    int[] leftMax=new int[n];
    int[] rightMax=new int[n];
    Array.Fill(leftMin,-1);
    Array.Fill(leftMax,-1);
    Array.Fill(rightMax,-1);
    Array.Fill(rightMin,-1);

    // will contain max product
    int max_product = int.MinValue;

    // to store maximum element on left of array
    int max_sum = arr[0];

    // to store minimum element on left of array
    int min_sum = arr[0];

    // leftMax[i] will contain max element
    // on left of arr[i] excluding arr[i].
    // leftMin[i] will contain min element
    // on left of arr[i] excluding arr[i].
    for (int i = 1; i < n; i++)
    {
        leftMax[i] = max_sum;
        if (arr[i] > max_sum)
            max_sum = arr[i];

        leftMin[i] = min_sum;
        if (arr[i] < min_sum)
            min_sum = arr[i];
    }
```

```
        // reset max_sum to store maximum element on
        // right of array
        max_sum = arr[n - 1];

        // reset min_sum to store minimum element on
        // right of array
        min_sum = arr[n - 1];

        // rightMax[i] will contain max element
        // on right of arr[i] excluding arr[i].
        // rightMin[i] will contain min element
        // on right of arr[i] excluding arr[i].
        for (int j = n - 2; j >= 0; j--)
        {
            rightMax[j] = max_sum;
            if (arr[j] > max_sum)
                max_sum = arr[j];

            rightMin[j] = min_sum;
            if (arr[j] < min_sum)
                min_sum = arr[j];
        }

        // For all array indexes i except first and
        // last, compute maximum of arr[i]*x*y where
        // x can be leftMax[i] or leftMin[i] and
        // y can be rightMax[i] or rightMin[i].
        for (int i = 1; i < n - 1; i++)
        {
            int max1 = Math.Max(arr[i] * leftMax[i] * rightMax[i],
                        arr[i] * leftMin[i] * rightMin[i]);

            int max2 = Math.Max(arr[i] * leftMax[i] * rightMin[i],
                        arr[i] * leftMin[i] * rightMax[i]);

            max_product = Math.Max(max_product, Math.Max(max1, max2));
        }

        return max_product;
    }

    // Driver code
    static void Main()
    {
        int []arr = { 1, 4, 3, -6, -7, 0 };
```

```
    int n = arr.Length;

    int max = maxProduct(arr, n);

    if (max == -1)
        Console.WriteLine("No Triplet Exists");
    else
        Console.WriteLine("Maximum product is "+max);

}
}
```

PHP

```
<?php
// A PHP program to find a maximum product of a triplet
// in array of integers

/* Function to find a maximum product of a triplet
in array of integers of size n */
function maxProduct($arr, $n)
{
    // if size is less than 3, no triplet exists
    if ($n < 3)
        return -1;

    // Construct four auxiliary vectors
    // of size n and initialize them by -1
    $leftMin=array_fill(0,$n, -1);
    $rightMin=array_fill(0,$n, -1);
    $leftMax=array_fill(0,$n, -1);
    $rightMax=array_fill(0,$n, -1);

    // will contain max product
    $max_product = PHP_INT_MIN;

    // to store maximum element on left of array
    $max_sum = $arr[0];

    // to store minimum element on left of array
    $min_sum = $arr[0];

    // leftMax[i] will contain max element
```

```
// on left of arr[i] excluding arr[i].
// leftMin[i] will contain min element
// on left of arr[i] excluding arr[i].
for ($i = 1; $i < $n; $i++)
{
    $leftMax[$i] = $max_sum;
    if ($arr[$i] > $max_sum)
        $max_sum = $arr[$i];

    $leftMin[$i] = $min_sum;
    if ($arr[$i] < $min_sum)
        $min_sum = $arr[$i];
}

// reset max_sum to store maximum element on
// right of array
$max_sum = $arr[$n - 1];

// reset min_sum to store minimum element on
// right of array
$min_sum = $arr[$n - 1];

// rightMax[i] will contain max element
// on right of arr[i] excluding arr[i].
// rightMin[i] will contain min element
// on right of arr[i] excluding arr[i].
for ($j = $n - 2; $j >= 0; $j--)
{
    $rightMax[$j] = $max_sum;
    if ($arr[$j] > $max_sum)
        $max_sum = $arr[$j];

    $rightMin[$j] = $min_sum;
    if ($arr[$j] < $min_sum)
        $min_sum = $arr[$j];
}

// For all array indexes i except first and
// last, compute maximum of arr[i]*x*y where
// x can be leftMax[i] or leftMin[i] and
// y can be rightMax[i] or rightMin[i].
for ($i = 1; $i < $n - 1; $i++)
{
    $max1 = max($arr[$i] * $leftMax[$i] * $rightMax[$i],
                $arr[$i] * $leftMin[$i] * $rightMin[$i]);
```

```
        $max2 = max($arr[$i] * $leftMax[$i] * $rightMin[$i],
                    $arr[$i] * $leftMin[$i] * $rightMax[$i]);

        $max_product = max($max_product, max($max1, $max2));
    }

    return $max_product;
}

// Driver program to test above functions
    $arr = array( 1, 4, 3, -6, -7, 0 );
    $n = count($arr);

    $max = maxProduct($arr, $n);

    if ($max == -1)
        echo "No Triplet Exists";
    else
        echo "Maximum product is ".$max;


// This code is contributed by mits
?>
```

Javascript

```
<script>

// A javascript program to find a maximum product of a triplet
// in array of integers

/* Function to find a maximum product of a triplet
in array of integers of size n */
function maxProduct(arr , n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Construct four auxiliary vectors
    // of size n and initialize them by -1
    leftMin = Array.from({length: n}, (_, i) => -1);
    rightMin = Array.from({length: n}, (_, i) => -1);
```

```
leftMax = Array.from({length: n}, (_, i) => -1);
rightMax = Array.from({length: n}, (_, i) => -1);


// will contain max product
var max_product = Number.MIN_VALUE;

// to store maximum element on left of array
var max_sum = arr[0];

// to store minimum element on left of array
var min_sum = arr[0];

// leftMax[i] will contain max element
// on left of arr[i] excluding arr[i].
// leftMin[i] will contain min element
// on left of arr[i] excluding arr[i].
for (i = 1; i < n; i++)
{
    leftMax[i] = max_sum;
    if (arr[i] > max_sum)
        max_sum = arr[i];

    leftMin[i] = min_sum;
    if (arr[i] < min_sum)
        min_sum = arr[i];
}

// reset max_sum to store maximum element on
// right of array
max_sum = arr[n - 1];

// reset min_sum to store minimum element on
// right of array
min_sum = arr[n - 1];

// rightMax[i] will contain max element
// on right of arr[i] excluding arr[i].
// rightMin[i] will contain min element
// on right of arr[i] excluding arr[i].
for (j = n - 2; j >= 0; j--)
{
    rightMax[j] = max_sum;
    if (arr[j] > max_sum)
        max_sum = arr[j];
```

```
        rightMin[j] = min_sum;
        if (arr[j] < min_sum)
            min_sum = arr[j];
    }

    // For all array indexes i except first and
    // last, compute maximum of arr[i]*x*y where
    // x can be leftMax[i] or leftMin[i] and
    // y can be rightMax[i] or rightMin[i].
    for (i = 1; i < n - 1; i++)
    {
        var max1 = Math.max(arr[i] * leftMax[i] * rightMax[i],
                    arr[i] * leftMin[i] * rightMin[i]);

        var max2 = Math.max(arr[i] * leftMax[i] * rightMin[i],
                    arr[i] * leftMin[i] * rightMax[i]);

        max_product = Math.max(max_product, Math.max(max1, max2));
    }

    return max_product;
}

// Driver code
var arr = [ 1, 4, 3, -6, -7, 0 ];
var n = arr.length;

var max = maxProduct(arr, n);

if (max == -1)
    document.write("No Triplet Exists");
else
    document.write("Maximum product is "+max);

// This code is contributed by Amit Katiyar
</script>
```

Output :

```
Maximum product is 168
```

Approach 3: O(nlogn) Time, O(1) Space

1. Sort the array using some efficient in-place sorting algorithm in ascending order.
2. Return the maximum of product of last three elements of the array and product of first two elements and last element.

Below is the implementation of above approach :

C++

```
// A C++ program to find a maximum product of a
// triplet in array of integers
#include <bits/stdc++.h>
using namespace std;

/* Function to find a maximum product of a triplet
   in array of integers of size n */
int maxProduct(int arr[], int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Sort the array in ascending order
    sort(arr, arr + n);

    // Return the maximum of product of last three
    // elements and product of first two elements
    // and last element
    return max(arr[0] * arr[1] * arr[n - 1],
               arr[n - 1] * arr[n - 2] * arr[n - 3]);
}

// Driver program to test above functions
int main()
{
    int arr[] = { -10, -3, 5, 6, -20 };
    int n = sizeof(arr) / sizeof(arr[0]);

    int max = maxProduct(arr, n);

    if (max == -1)
        cout << "No Triplet Exists";
```

```
    else
        cout << "Maximum product is " << max;

    return 0;
}
```

Java

```
// Java program to find a maximum product of a
// triplet in array of integers

import java.util.Arrays;

class GFG {

    /* Function to find a maximum product of a triplet
   in array of integers of size n */
    static int maxProduct(int arr[], int n) {
        // if size is less than 3, no triplet exists
        if (n < 3) {
            return -1;
        }

        // Sort the array in ascending order
        Arrays.sort(arr);

        // Return the maximum of product of last three
        // elements and product of first two elements
        // and last element
        return Math.max(arr[0] * arr[1] * arr[n - 1],
                arr[n - 1] * arr[n - 2] * arr[n - 3]);
    }

// Driver program to test above functions
    public static void main(String[] args) {
        int arr[] = {-10, -3, 5, 6, -20};
        int n = arr.length;

        int max = maxProduct(arr, n);

        if (max == -1) {
            System.out.println("No Triplet Exists");
        } else {
            System.out.println("Maximum product is " + max);
```

```
        }

    }
}
/* This Java code is contributed by Rajput-Ji*/
```

Python3

```
# A Python3 program to find a maximum
# product of a triplet in an array of integers

# Function to find a maximum product of a
# triplet in array of integers of size n
def maxProduct(arr, n):

    # if size is less than 3, no triplet exists
    if n < 3:
        return -1

    # Sort the array in ascending order
    arr.sort()

    # Return the maximum of product of last
    # three elements and product of first
    # two elements and last element
    return max(arr[0] * arr[1] * arr[n - 1],
               arr[n - 1] * arr[n - 2] * arr[n - 3])

# Driver Code
if __name__ == "__main__":

    arr = [-10, -3, 5, 6, -20]
    n = len(arr)

    _max = maxProduct(arr, n)

    if _max == -1:
        print("No Triplet Exists")
    else:
        print("Maximum product is", _max)

# This code is contributed by Rituraj Jain
```

C#

```
// C# program to find a maximum product of a
// triplet in array of integers
using System;
public class GFG {

    /* Function to find a maximum product of a triplet
in array of integers of size n */
    static int maxProduct(int []arr, int n) {
        // if size is less than 3, no triplet exists
        if (n < 3) {
            return -1;
        }

        // Sort the array in ascending order
        Array.Sort(arr);

        // Return the maximum of product of last three
        // elements and product of first two elements
        // and last element
        return Math.Max(arr[0] * arr[1] * arr[n - 1],
                arr[n - 1] * arr[n - 2] * arr[n - 3]);
    }

// Driver program to test above functions
    public static void Main() {
        int []arr = {-10, -3, 5, 6, -20};
        int n = arr.Length;

        int max = maxProduct(arr, n);

        if (max == -1) {
            Console.WriteLine("No Triplet Exists");
        } else {
            Console.WriteLine("Maximum product is " + max);
        }

    }
}
// This code is contributed by 29AjayKumar
```

PHP

```
<?php
```

Javascript

```
<script>

// Javascript program to find a maximum
// product of a triplet in array of integers

// Function to find a maximum product of a
// triplet in array of integers of size n
function maxProduct(arr, n)
{

    // If size is less than 3, no
    // triplet exists
    if (n < 3)
    {
        return -1;
    }

    // Sort the array in ascending order
    arr.sort();

    // Return the maximum of product of last three
    // elements and product of first two elements
    // and last element
    return Math.max(arr[0] * arr[1] * arr[n - 1],
            arr[n - 1] * arr[n - 2] * arr[n - 3]);
}

// Driver code
var arr = [-10, -3, 5, 6, -20];
var n = arr.length;
var max = maxProduct(arr, n);

if (max == -1)
{
    document.write("No Triplet Exists");
}
else
{
    document.write("Maximum product is " + max);
}
```

```
// This code is contributed by Rajput-Ji

</script>
```

Output :

```
Maximum product is 1200
```

Approach 4: O(n) Time, O(1) Space

1. Scan the array and compute Maximum, second maximum and third maximum element present in the array.
2. Scan the array and compute Minimum and second minimum element present in the array.
3. Return the maximum of product of Maximum, second maximum and third maximum and product of Minimum, second minimum and Maximum element.

Note – Step 1 and Step 2 can be done in single traversal of the array.

Below is the implementation of the above approach :

C++

```
// A O(n) C++ program to find maximum product pair in
// an array.
#include <bits/stdc++.h>
using namespace std;

/* Function to find a maximum product of a triplet
   in array of integers of size n */
int maxProduct(int arr[], int n)
{
    // if size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Initialize Maximum, second maximum and third
```

```
    // maximum element
    int maxA = INT_MIN, maxB = INT_MIN, maxC = INT_MIN;

    // Initialize Minimum and second minimum element
    int minA = INT_MAX, minB = INT_MAX;

    for (int i = 0; i < n; i++)
    {
        // Update Maximum, second maximum and third
        // maximum element
        if (arr[i] > maxA)
        {
            maxC = maxB;
            maxB = maxA;
            maxA = arr[i];
        }

        // Update second maximum and third maximum element
        else if (arr[i] > maxB)
        {
            maxC = maxB;
            maxB = arr[i];
        }

        // Update third maximum element
        else if (arr[i] > maxC)
            maxC = arr[i];

        // Update Minimum and second minimum element
        if (arr[i] < minA)
        {
            minB = minA;
            minA = arr[i];
        }

        // Update second minimum element
        else if(arr[i] < minB)
            minB = arr[i];
    }

    return max(minA * minB * maxA,
               maxA * maxB * maxC);
}

// Driver program to test above function
```

```
int main()
{
    int arr[] = { 1, -4, 3, -6, 7, 0 };
    int n = sizeof(arr) / sizeof(arr[0]);

    int max = maxProduct(arr, n);

    if (max == -1)
        cout << "No Triplet Exists";
    else
        cout << "Maximum product is " << max;

    return 0;
}
```

Java

```
// A O(n) Java program to find maximum product
// pair in an array.
import java.util.*;

class GFG{

// Function to find a maximum product of
// a triplet in array of integers of size n
static int maxProduct(int []arr, int n)
{

    // If size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Initialize Maximum, second maximum
    // and third maximum element
    int maxA = Integer.MAX_VALUE,
        maxB = Integer.MAX_VALUE,
        maxC = Integer.MAX_VALUE;

    // Initialize Minimum and
    // second minimum element
    int minA = Integer.MIN_VALUE,
        minB = Integer.MIN_VALUE;

    for(int i = 0; i < n; i++)
```

```
        {

            // Update Maximum, second maximum
            // and third maximum element
            if (arr[i] > maxA)
            {
                maxC = maxB;
                maxB = maxA;
                maxA = arr[i];
            }

            // Update second maximum and
            // third maximum element
            else if (arr[i] > maxB)
            {
                maxC = maxB;
                maxB = arr[i];
            }

            // Update third maximum element
            else if (arr[i] > maxC)
                maxC = arr[i];

            // Update Minimum and second
            // minimum element
            if (arr[i] < minA)
            {
                minB = minA;
                minA = arr[i];
            }

            // Update second minimum element
            else if(arr[i] < minB)
                minB = arr[i];
        }

        return Math.max(minA * minB * maxA,
                        maxA * maxB * maxC);
    }

    // Driver code
    public static void main(String[] args)
    {
        int []arr = { 1, -4, 3, -6, 7, 0 };
        int n = arr.length;
```

```
    int max = maxProduct(arr, n);

    if (max == -1)
        System.out.print("No Triplet Exists");
    else
        System.out.print("Maximum product is " + max);
}
}

// This code is contributed by pratham76
```

Python3

```
# A O(n) Python3 program to find maximum
# product pair in an array.
import sys

# Function to find a maximum product
# of a triplet in array of integers
# of size n
def maxProduct(arr, n):

    # If size is less than 3, no
    # triplet exists
    if (n < 3):
        return -1

    # Initialize Maximum, second
    # maximum and third maximum
    # element
    maxA = -sys.maxsize - 1
    maxB = -sys.maxsize - 1
    maxC = -sys.maxsize - 1

    # Initialize Minimum and
    # second minimum element
    minA = sys.maxsize
    minB = sys.maxsize

    for i in range(n):

        # Update Maximum, second
        # maximum and third maximum
        # element
```

```
        if (arr[i] > maxA):
            maxC = maxB
            maxB = maxA
            maxA = arr[i]

        # Update second maximum and
        # third maximum element
        else if (arr[i] > maxB):
            maxC = maxB
            maxB = arr[i]

        # Update third maximum element
        else if (arr[i] > maxC):
            maxC = arr[i]

        # Update Minimum and second
        # minimum element
        if (arr[i] < minA):
            minB = minA
            minA = arr[i]

        # Update second minimum element
        else if (arr[i] < minB):
            minB = arr[i]

    return max(minA * minB * maxA,
               maxA * maxB * maxC)

# Driver Code
arr = [ 1, -4, 3, -6, 7, 0 ]
n = len(arr)

Max = maxProduct(arr, n)

if (Max == -1):
    print("No Triplet Exists")
else:
    print("Maximum product is", Max)

# This code is contributed by avanitrachhadiya2155
```

C#

```
// A O(n) C# program to find maximum product
// pair in an array.
using System;
using System.Collections;

class GFG{

// Function to find a maximum product of
// a triplet in array of integers of size n
static int maxProduct(int []arr, int n)
{

    // If size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Initialize Maximum, second maximum
    // and third maximum element
    int maxA = Int32.MinValue,
        maxB = Int32.MinValue,
        maxC = Int32.MinValue;

    // Initialize Minimum and
    // second minimum element
    int minA = Int32.MaxValue,
        minB = Int32.MaxValue;

    for(int i = 0; i < n; i++)
    {

        // Update Maximum, second maximum
        // and third maximum element
        if (arr[i] > maxA)
        {
            maxC = maxB;
            maxB = maxA;
            maxA = arr[i];
        }

        // Update second maximum and
        // third maximum element
        else if (arr[i] > maxB)
        {
            maxC = maxB;
            maxB = arr[i];
```

```
        }

        // Update third maximum element
        else if (arr[i] > maxC)
            maxC = arr[i];

        // Update Minimum and second
        // minimum element
        if (arr[i] < minA)
        {
            minB = minA;
            minA = arr[i];
        }

        // Update second minimum element
        else if(arr[i] < minB)
            minB = arr[i];
    }

    return Math.Max(minA * minB * maxA,
                    maxA * maxB * maxC);
}

// Driver code
public static void Main(string[] args)
{
    int []arr = { 1, -4, 3, -6, 7, 0 };
    int n = arr.Length;

    int max = maxProduct(arr, n);

    if (max == -1)
        Console.Write("No Triplet Exists");
    else
        Console.Write("Maximum product is " + max);
}
}

// This code is contributed by rutvik_56
```

Javascript

```
<script>
```

```javascript
// A O(n) javascript program to find maximum product
// pair in an array.


// Function to find a maximum product of
// a triplet in array of integers of size n
function maxProduct(arr , n)
{

    // If size is less than 3, no triplet exists
    if (n < 3)
        return -1;

    // Initialize Maximum, second maximum
    // and third maximum element
    var maxA = Number.MIN_VALUE,
        maxB = Number.MIN_VALUE,
        maxC = Number.MIN_VALUE;

    // Initialize Minimum and
    // second minimum element
    var minA = Number.MAX_VALUE,
        minB = Number.MAX_VALUE;

    for(i = 0; i < n; i++)
    {

        // Update Maximum, second maximum
        // and third maximum element
        if (arr[i] > maxA)
        {
            maxC = maxB;
            maxB = maxA;
            maxA = arr[i];
        }

        // Update second maximum and
        // third maximum element
        else if (arr[i] > maxB)
        {
            maxC = maxB;
            maxB = arr[i];
        }

        // Update third maximum element
```

```
        else if (arr[i] > maxC)
            maxC = arr[i];

        // Update Minimum and second
        // minimum element
        if (arr[i] < minA)
        {
            minB = minA;
            minA = arr[i];
        }

        // Update second minimum element
        else if(arr[i] < minB)
            minB = arr[i];
    }

    return Math.max(minA * minB * maxA,
                    maxA * maxB * maxC);
}

// Driver code
var arr = [ 1, -4, 3, -6, 7, 0 ];
var n = arr.length;

var max = maxProduct(arr, n);

if (max == -1)
    document.write("No Triplet Exists");
else
    document.write("Maximum product is " + max);

// This code is contributed by 29AjayKumar

</script>
```

Output :

```
Maximum product is 168
```

Exercise:

1. Print the triplet that has maximum product.
2. Find a minimum product of a triplet in array.

This article is contributed by **Aditya Goel**. Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above

Next

Maximum Product Subarray

RECOMMENDED ARTICLES

Page : 1 2 3

01 Maximum product of bitonic subsequence of size 3
31, Mar 20

02 Maximum product of an increasing subsequence of size 3
17, Dec 16

03 Maximum product of subsequence of size k
04, Jul 17

05 Largest triplet product in a stream
27, Jun 17

06 Maximum sum of absolute differences between distinct pairs of a triplet from an array
16, Feb 21

07 Maximum triplet sum in array
16, Dec 17

08 Maximum length subsequence

04 **Find distinct integers for a triplet with given product**
28, Jan 20

such that adjacent elements in the subsequence have a common factor
11, Feb 19

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Medium

Easy | Normal | Medium | Hard | Expert

Improved By : shrikanth13, nitin mittal, vt_m, Rasheed60, ukasp, Rajput-Ji, 29AjayKumar, rituraj_jain, Code_Mech, Mithun Kumar, gouravgarg48, rutvik_56, avanitrachhadiya2155, pratham76, rag2127, chinmoy1997pal, amit143katiyar, ruhelaa48, simranarora5sos, surinderdawra388, sayanmandal79

Article Tags : Amazon, Jugnoo, Snapdeal, Arrays, Sorting

Practice Tags : Amazon, Snapdeal, Arrays, Sorting

Improve Article | Report Issue

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

Load Comments

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

NodeJS