MAX PRODUCT OF THREE

Team members

1-يوسف محمد حسن حنفى 20211094

2-يوسف عمر حسن محمد عرفه 20211088

3-حماده توفيق عبدالحميد توفيق 202000246

4-أحمد محمد فتحى لطفى 20210107

5-أدم محمد زغلول 20210132

6-عبدالرحمن حازم توفيق محمد 20220256

7-صفى الدين محمد عبدالفتاح 20220246

1ST algorithm

Description:-

1-sort array

2-first product=arr[N-1]\*arr[0]\*arr[1]

3-second product=arr[N-1]\*arr[n-2]\*arr[n-3]

4-return MAX product

Pseudocode:-

int maxProduct(int[] arr) {

n = arr.size

for i from 0 to n-1

for j from i+1 to n-1

if nums[i] > nums[j]

swap(nums[i], nums[j])

first\_product= arr[N-1]\*arr[0]\*arr[1]

second\_product= arr[N-1]\*arr[n-2]\*arr[n-3]

if(first\_product>second\_product){

return first\_product

}

Else{

Return second\_product

}

}

Analysis:-

Time complexity:-

* Two nested loops to sort the array.
* The outer loop iterates n times which is the array size.
* The inner loop iterates n-i-1 times average.
* Swapping operations in inner loop is constant time of O(1).
* The time complexity of nested loops is O(n2).
* After swapping and sorting there is two possible products variables that calculates the two products and then returns the max product, all this operations takes constant time of O(1).
* Since there is the loops that takes time only so the time complexity of this algorithm is O(n2).

Space complexity:-

* The only space used is for the array which user input it’s elements.
* Therefore, the space complexity is O(n).

Implementation:-

def maxProduct(arr):

n = len(arr)

for i in range(n):

for j in range(i + 1, n):

if arr[i] > arr[j]:

arr[i], arr[j] = arr[j], arr[i]

first\_product = arr[-1] \* arr[0] \* arr[1]

second\_product = arr[-1] \* arr[-2] \* arr[-3]

if first\_product > second\_product:

return first\_product

else:

return second\_product

2nd algorithm

Description:-

This Python code defines a function, max**\_**product**\_**of\_three that calculates the maximum product of any three integers within a given array of integers.

It employs a recursive approach to explore all possible combinations of three numbers within the list and keeps track of the maximum product found.

If the input list has fewer than three elements the function returns 0.

Pseudocode:-

max\_product\_of\_three(nums):

if length of nums < 3:

return 0

max\_product = 1

function find\_max\_product(start, count, product):

if count == 3:

update max\_product if product is greater

return

for i from start to length of nums:

find\_max\_product(i + 1, count + 1, product \* nums[i])

find\_max\_product(0, 0, 1)

return max\_product

Analysis:-

Time complexity:-

* Checks if the length is less than 3 or not that takes constant time complexity of O(1).
* If it’s 3 or more the function goes to begin the recursive of all possible three numbers in the array.
* Function “find\_max\_product” called for each element in the array and inside the function there is a loop that iterates on every element.
* Since we are looking for all the possible combinations of three and iterating on every element then the time complexity is O(n3).

Space complexity:-

* Each call for the function, three parameters are passed that occupy constant space.
* The number of recursions depends on the size of the array.
* Therefore, the most number of recursive calls is “n”.
* Therefore, the space complexity is O(n).

Implementation:-

def max\_product\_of\_three(nums):

if len(nums) < 3:

return 0

max\_product = 1

def find\_max\_product(start, count, product):

nonlocal max\_product

if count == 3:

max\_product = max(max\_product, product)

return

for i in range(start, len(nums)):

find\_max\_product(i + 1, count + 1, product \* nums[i])

find\_max\_product(0, 0, 1)

return max\_product

Comparison:-

|  |  |  |
| --- | --- | --- |
|  | 1st algorithm(non-recursive) | 2nd algorithm(recursive) |
| Time comlexity | O(n2) | O(n3) |
| Space complexity | O(n) | O(n) |

* Since the two algorithms have the same space complexity and different time complexity.
* Therefore, the 1st algorithm(which is non-recursive) is better than the second algorithm(which is recursive) because it takes less time.