

Enrol in top rated Coding Courses and get assured Scholarship | Apply Now FREE



December 14, 2021   ■   Arrays / Data Structure

# Kadane’s Algorithm : Maximum Subarray Sum in an Array

**Problem Statement:** Given an integer array arr, find the contiguous subarray (containing at least one number) which has the largest sum and returns its sum and prints the subarray.

### Examples:

**Example 1:**

**Input:** arr = [-2,1,-3,4,-1,2,1,-5,4]

**Output:** 6

**Explanation:** [4,-1,2,1] has the largest sum = 6.

**Examples 2:**

**Input:** arr = [1]

**Output:** 1

**Explanation:** Array has only one element and which is giving positive sum of 1.

### Solution

**Disclaimer:** Don't jump directly to the solution, try it out yourself first.

### Solution 1: Naive Approach

**Approach:** Using three for loops, we will get all possible subarrays in two loops and their sum in another loop, and then return the maximum of them.

### Code:

### C++ Code

```
#include<bits/stdc++.h>

using namespace std;
int maxSubArray(vector < int > & nums, vector < int > & subarray) {
    int max_sum = INT_MIN;
    int n = nums.size();
    if (n == 1) {
        return nums[0];
    }
    int i, j;
    for (i = 0; i <= n - 1; i++) {
        for (j = i; j <= n - 1; j++) {
            int sum = 0;
            for (int k = i; k <= j; k++)
                sum = sum + nums[k];
            if (sum > max_sum) {
                subarray.clear();
                max_sum = sum;
                subarray.push_back(i);
                subarray.push_back(j);
            }
        }
    }
    return max_sum;
}

int main() {
    vector<int> arr{-2,1,-3,4,-1,2,1,-5,4};
    vector < int > subarray;
    int lon = maxSubArray(arr, subarray);
    cout << "The longest subarray with maximum sum is " << lon << endl;
    cout << "The subarray is " << endl;
    for (int i = subarray[0]; i <= subarray[1]; i++) {
        cout << arr[i] << " ";
    }
}
```

### Subscribe

I want to receive latest posts and interview tips

Name\*

Email\*

Join takeUforward

Search

### Recent Posts

- Find the highest/lowest frequency element
- A Guide on Online C Compiler
- Burst Balloons | Partition DP | DP 51
- Evaluate Boolean Expression to True | Partition DP: DP 52
- Palindrome Partitioning – II | Front Partition : DP 53

- Accolite Digital **Amazon** Arcesium Bank of America Barclays
- BFS Binary Search Binary Search Tree Commvault CPP DE
- Shaw DFS **DSA Self Paced** google
- HackerEarth infosys inorder Java Juspay Kreeti Technologies Morgan
- Stanley Newfold Digital Oracle post order queue recursion Samsung
- SDE Core Sheet **SDE Sheet** Searching set-bits sorting
- Strivers A2ZDSA Course sub-array subarray Swiggy takeuforward
- TCQ NINJA TCS TCS CODEVITA TCS DIGITA TCS Ninja **TCS**
- NQT** VMware XOR

```
}

```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity:  $O(N^3)$

Space Complexity:  $O(1)$

Java Code

```
import java.util.*;
class TUF {
    public static int maxSubArray(int[] nums, ArrayList < Integer > subarray) {
        int max_sum = Integer.MIN_VALUE;
        int n = nums.length;
        if (n == 1) {
            return nums[0];
        }
        for (int i = 0; i <= n - 1; i++) {
            for (int j = i; j <= n - 1; j++) {
                int sum = 0;
                for (int k = i; k <= j; k++)
                    sum = sum + nums[k];
                if (sum > max_sum) {
                    subarray.clear();
                    max_sum = sum;
                    subarray.add(i);
                    subarray.add(j);
                }
            }
        }
        return max_sum;
    }
    public static void main(String args[]) {
        int arr[]={-2,1,-3,4,-1,2,1,-5,4};
        ArrayList < Integer > subarray = new ArrayList < > ();
        int lon = maxSubArray(arr, subarray);
        System.out.println("The longest subarray with maximum sum is " + lon);
        System.out.println("The subarray is ");
        for (int i = subarray.get(0); i <= subarray.get(1); i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity:  $O(N^3)$ - TLE

Space Complexity:  $O(1)$

Python Code

```
from typing import List
def maxSubArray(nums: List[int], subarray: List[int]) -> int:
    max_sum = -float('inf')
    n = len(nums)
    if n == 1:
        return nums[0]
    i, j = 0, 0
    for i in range(n):
        for j in range(i, n):
            sum = 0
            for k in range(i, j + 1):
                sum += nums[k]
            if sum > max_sum:
                subarray.clear()
                max_sum = sum
                subarray.append(i)
                subarray.append(j)
    return max_sum

if __name__ == "__main__":
    arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
    subarray = []
    lon = maxSubArray(arr, subarray)
    print("The longest subarray with maximum sum is", lon)
```

```
print("The subarray is")
for i in range(subarray[0], subarray[1] + 1):
    print(arr[i], end=" ")
print()
```

Output:

The longest subarray with maximum sum is 6

The subarray is

4 -1 2 1

Time Complexity: O(N^3)- TLE

Space Complexity: O(1)

Solution 2: A better approach

Approach :

We can also do this problem using only two loops, we will avoid 3rd loop which was used in the above approach, instead of that we will create a new variable curr\_sum, which stores the array sum from the ith index to the jth index.

Code:

C++ Code

```
#include<bits/stdc++.h>

using namespace std;
int maxSubArray(vector < int > & nums, vector < int > & subarray) {
    int max_sum = INT_MIN;
    for (int i = 0; i < nums.size(); i++) {
        int curr_sum = 0;
        for (int j = i; j < nums.size(); j++) {
            curr_sum += nums[j];
            if (curr_sum > max_sum) {
                subarray.clear();
                max_sum = curr_sum;
                subarray.push_back(i);
                subarray.push_back(j);
            }
        }
    }
    return max_sum;
}

int main() {
    vector<int> arr{-2,1,-3,4,-1,2,1,-5,4};
    vector < int > subarray;
    int lon = maxSubArray(arr, subarray);
    cout << "The longest subarray with maximum sum is " << lon << endl;
    cout << "The subarray is " << endl;
    for (int i = subarray[0]; i <= subarray[1]; i++) {
        cout << arr[i] << " ";
    }

}
```

Output:

The longest subarray with maximum sum is 6

The subarray is

4 -1 2 1

Time Complexity: O(N^2)

Space Complexity: O(1)

Java Code

```
import java.util.*;
class TUF {
    public static int maxSubArray(int[] nums, ArrayList < Integer > subarray) {
        int max_sum = Integer.MIN_VALUE;
        for (int i = 0; i < nums.length; i++) {
            int curr_sum = 0;
            for (int j = i; j < nums.length; j++) {
                curr_sum += nums[j];
                if (curr_sum > max_sum) {
                    subarray.clear();
                    max_sum = curr_sum;
                    subarray.add(i);
                    subarray.add(j);
                }
            }
        }
        return max_sum;
    }
    public static void main(String args[]) {
        int arr[]={-2,1,-3,4,-1,2,1,-5,4};
```

```
        ArrayList < Integer > subarray = new ArrayList < > ();
        int lon = maxSubArray(arr, subarray);
        System.out.println("The longest subarray with maximum sum is " + lon);
        System.out.println("The subarray is ");
        for (int i = subarray.get(0); i <= subarray.get(1); i++) {
            System.out.print(arr[i] + " ");
        }

    }
}
```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity:  $O(N^2)$

Space Complexity:  $O(1)$

Python Code

```
from typing import List

def maxSubArray(nums: List[int], subarray: List[int]) -> int:
    max_sum = -float('inf')
    for i in range(len(nums)):
        curr_sum = 0
        for j in range(i, len(nums)):
            curr_sum += nums[j]
            if curr_sum > max_sum:
                subarray.clear()
                max_sum = curr_sum
                subarray.append(i)
                subarray.append(j)
        return max_sum

if __name__ == "__main__":
    arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]
    subarray = []
    lon = maxSubArray(arr, subarray)
    print("The longest subarray with maximum sum is", lon)
    print("The subarray is")
    for i in range(subarray[0], subarray[1] + 1):
        print(arr[i], end=" ")
    print()
```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity:  $O(N^2)$

Space Complexity:  $O(1)$

Solution : 3 Optimal Solution: Kadane’s Algorithm

**Intuition:** Basically this problem can be done in linear time complexity with Kadane’s algorithm along with that will also get the subarray that is giving the largest positive sum.

Approach:

-> We will have the following variables in the beginning :

**msf** – max so far, **meh** – max sum ending at ith index, **start** – to get the starting index of ans’s subarray, **end** – to get the ending index of ans’s subarray.

-> Traverse the array from starting and add the ith element to max\_end\_here(meh) , now we will check that adding the ith element gives greater value than max\_so\_far(msf) or not, if yes then we will update our msf and also update the starting and ending index(initially starting index is zero).

```
for(int i=0;i<nums.size();i++){
    meh+=nums[i];

    if(meh>msf){ msf=meh; start=s; end=i; }

    if(meh<0){meh=0; s=i+1;}
}
```

-> In this step, we will print the answer subarray using the start and end variables.

->Return the largest subarray sum that is:- msf.

Code:

C++ Code

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

int maxSubArray(vector < int > & nums, vector < int > & subarray) {
    int msf = INT_MIN, meh = 0;
    int s = 0;
    for (int i = 0; i < nums.size(); i++) {
        meh += nums[i];
        if (meh > msf) {
            subarray.clear();
            msf = meh;
            subarray.push_back(s);
            subarray.push_back(i);
        }
        if (meh < 0) {
            meh = 0;
            s = i + 1;
        }
    }

    return msf;
}

int main() {
    vector<int> arr{-2,1,-3,4,-1,2,1,-5,4};
    vector < int > subarray;
    int lon = maxSubArray(arr, subarray);
    cout << "The longest subarray with maximum sum is " << lon << endl;
    cout << "The subarray is " << endl;
    for (int i = subarray[0]; i <= subarray[1]; i++) {
        cout << arr[i] << " ";
    }

}
```

Output:

The longest subarray with maximum sum is 6

The subarray is

4 -1 2 1

Time Complexity: O(N)

Space Complexity:O(1)

Java Code

```
import java.util.*;

class TUF{
    public static int maxSubArray(int[] nums,ArrayList<Integer> subarray) {
        int msf=Integer.MIN_VALUE , meh=0 ;
        int s=0;
        for(int i=0;i<nums.length;i++){
            meh+=nums[i];

            if(meh>msf)
            {
                subarray.clear();
                msf=meh;
                subarray.add(s);
                subarray.add(i);
            }
            if(meh<0)
            {
                meh=0;
                s=i+1;
            }
        }

        return msf;
    }
    public static void main(String args[])
    {
        int arr[]={-2,1,-3,4,-1,2,1,-5,4};
        ArrayList<Integer> subarray=new ArrayList<>();
        int lon=maxSubArray(arr,subarray);
        System.out.println("The longest subarray with maximum sum is "+lon);
        System.out.println("The subarray is ");
        for(int i=subarray.get(0);i<=subarray.get(1);i++)
        {
            System.out.print(arr[i]+" ");
        }
    }
}
```

```
    }  
}
```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity: O(N)

Space Complexity:O(1)

Python Code

```
from typing import List  
  
def maxSubArray(nums: List[int], subarray: List[int]) -> int:  
    msf = -float('inf')  
    meh = 0  
    s = 0  
    for i in range(len(nums)):  
        meh += nums[i]  
        if meh > msf:  
            subarray.clear()  
            msf = meh  
            subarray.append(s)  
            subarray.append(i)  
        if meh < 0:  
            meh = 0  
            s = i + 1  
    return msf  
  
if __name__ == "__main__":  
    arr = [-2, 1, -3, 4, -1, 2, 1, -5, 4]  
    subarray = []  
    lon = maxSubArray(arr, subarray)  
    print("The longest subarray with maximum sum is", lon)  
    print("The subarray is")  
    for i in range(subarray[0], subarray[1] + 1):  
        print(arr[i], end=" ")  
    print()
```

Output:

The longest subarray with maximum sum is 6  
The subarray is  
4 -1 2 1

Time Complexity: O(N^2)

Space Complexity: O(1)

Special thanks to [Abhay Rai](#) and [Sudip Ghosh](#) for contributing to this article on takeUforward. If you also wish to share your knowledge with the takeUforward fam, [please check out this article](#)

Maximum Subarray Sum | Leetcode | Kadane's Algorithm | Brute-Better-Optimal | CPP/Java



Load Comments

Copyright © 2022 takeuforward | All rights reserved