Kadane's Algorithm

Print Screen

Maximum Subarray Sum

continuous

n

1	2	3	4	5

1, 2, 3, 4, 5

12, 23, 34, 45

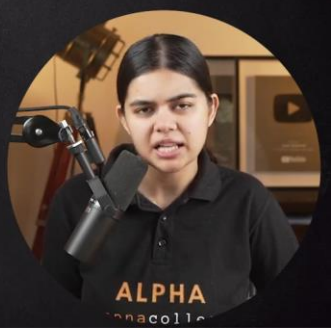
123, 234, 345

1234, 2345

12345

$\#subarr = \frac{n*(n+1)}{2}$

$n=5 \Rightarrow \frac{5*6}{2} = 15$



Print Screen

Maximum Subarray Sum

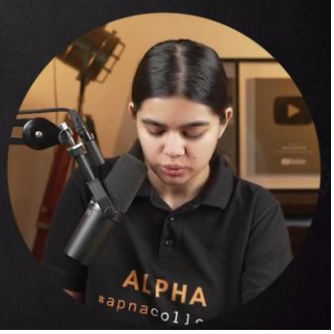
continuous part

1	2	3	4	5

start: 0, 1, 2

end (st to n-1): 0,1,2,3,4 ; 1,2,3,4 ; 2,3,4

```
for(st = 0; st < n; st++) {
    for(end = st; end < n; end++) {
        => (st to end)
    }
}
```



Print Screen

code.cpp

```cpp
#include <iostream>
#include <vector>
using namespace std;

int main() {
    int n = 5;
    int arr[5] = {1, 2, 3, 4, 5};

    for(int st=0; st<n; st++) {
        for(int end=st; end<n; end++) {
            for(int i=st; i<=end; i++) {
                cout << arr[i];
            }
            cout << " ";
        }
        cout << endl;
    }

    return 0;
}
```

PORTS PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL

```
apnacollege@Shradha DSAseries % g++ -std=c++11 code.cpp && ./a.out
1 12 123 1234 12345
2 23 234 2345
3 34 345
4 45
5
apnacollege@Shradha DSAseries %
```

Print Screen

Maximum Subarray Sum

Brute Force Approach

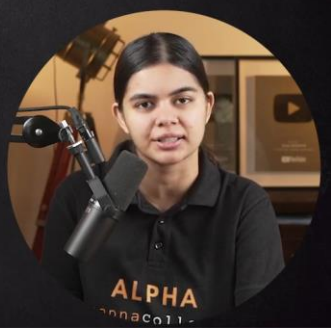
{ 3, -4, 5, 4, -1, 7, -8 }

st = 0

$O(n^3)$

```
for(st = 0; st < n; st++) {
    for(end = st; end < n; end++) {
        ⇒ (st to end)
    }
}
```

Print Screen



Maximum Subarray Sum

Brute Force Approach

{ 3, -4, 5, 4, -1, 7, -8 }

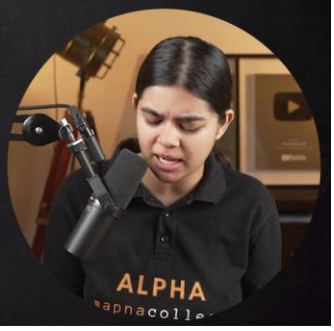
st = 0 end

CS = -1 + 5 = 4

$O(n^3)$

```
maxSum
for(st = 0; st < n; st++) {
    currSum = 0
    for(end = st; end < n; end++) {
        cs += arr[end]
        maxSum = max(CS, MS);
    }
}
return MS
```

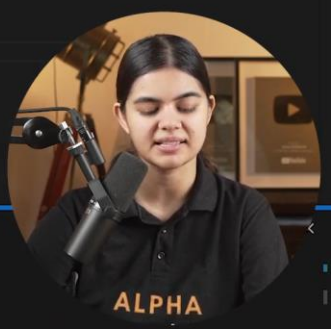
Print Screen



code.cpp

```cpp
int main() {
    int n = 5;
    int arr[5] = {1, 2, 3, 4, 5};

    int maxSum = INT_MIN;

    for(int st=0; st<n; st++) {
        int currSum = 0;
        for(int end=st; end<n; end++) {
            currSum += arr[end];
            maxSum = max(currSum, maxSum);
        }
    }

    cout << "max subarray sum = " << maxSum << endl;

    return 0;
}
```

```
PORTS  PROBLEMS  DEBUG CONSOLE  OUTPUT  TERMINAL
apnacollege@Shradha DSAseries % g++ -std=c++11 code.cpp && ./a.out
max subarray sum = 15
apnacollege@Shradha DSAseries %
```



Kadane's Algorithm

Most Optimised

{ 3, -4, 5, 4, -1, 7, -8 }

Intuition

+ve + +ve → +

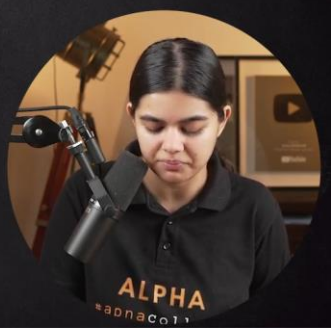
-ve + +ve → +

+ve + -ve → -ve

reset to 0

currSum = 0 maxSum = INT_MIN

```
for(i=0; i<n; i++){
    currSum += arr[i]
    maxSum = max(CS, MS)
    if(CS < 0)
        CS = 0
}
```



Kadane's Algorithm

Most Optimised

{ 3, -4, 5, 4, -1, 7, -8 }

CS = 0 3 -1 5 9 8 15 7

MS = -∞ 3 5 9 15

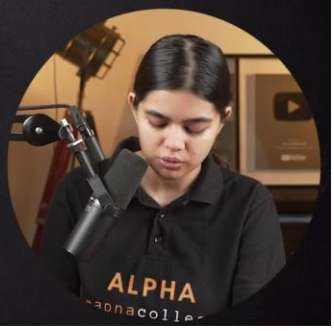
edge / corner

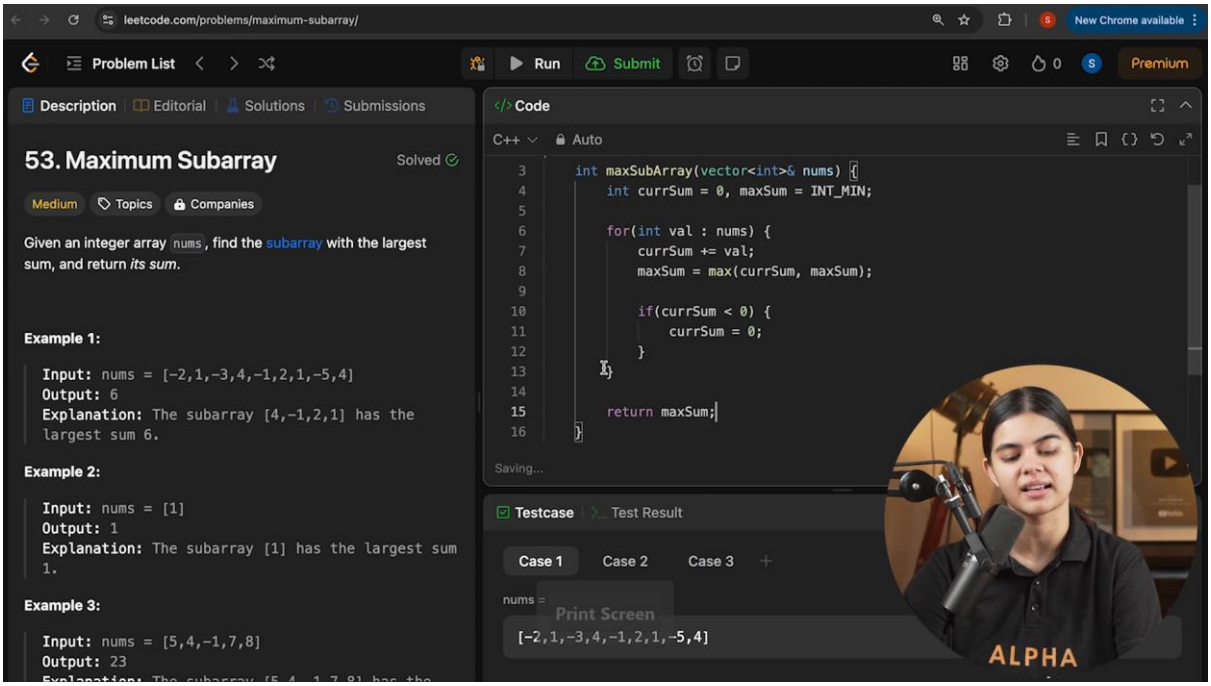
arr = [-1, -2, -3, -4, -5]

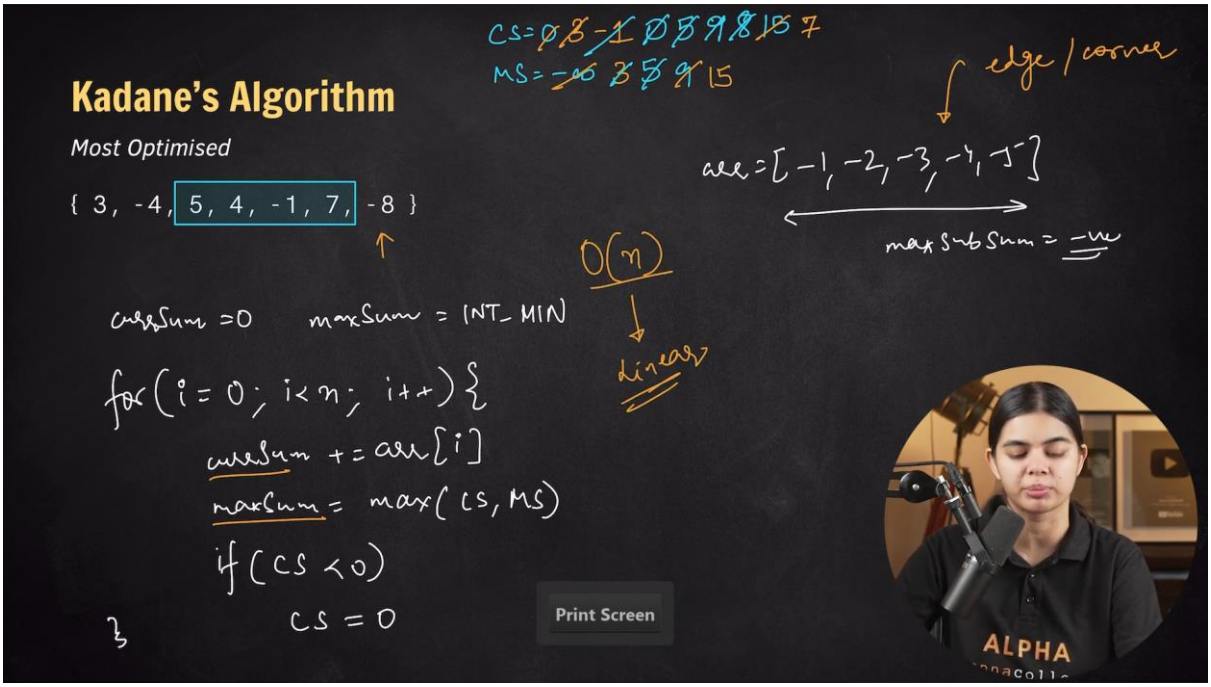
maxSubSum = -ve

currSum = 0 maxSum = INT_MIN

```
for(i=0; i<n; i++){
    currSum += arr[i]
    maxSum = max(CS, MS)
    if(CS < 0)
        CS = 0
}
```




leetcode.com/problems/maximum-subarray/
New Chrome available
Problem List
Run
Submit
Premium
Description
Editorial
Solutions
Submissions
53. Maximum Subarray
Solved
Medium
Topics
Companies
Given an integer array nums, find the subarray with the largest sum, and return its sum.
Example 1:
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
Explanation: The subarray [4,-1,2,1] has the largest sum 6.
Example 2:
Input: nums = [1]
Output: 1
Explanation: The subarray [1] has the largest sum 1.
Example 3:
Input: nums = [5,4,-1,7,8]
Output: 23
Code
C++
Auto
int maxSubArray(vector<int>& nums) {
int currSum = 0, maxSum = INT_MIN;
for(int val : nums) {
currSum += val;
maxSum = max(currSum, maxSum);
if(currSum < 0) {
currSum = 0;
}
}
return maxSum;
}
Saving...
Testcase
Test Result
Case 1
Case 2
Case 3
nums =
[-2,1,-3,4,-1,2,1,-5,4]
Print Screen
ALPHA


Kadane's Algorithm
Most Optimised
{ 3, -4, 5, 4, -1, 7, -8 }
currSum =0
maxSum = INT_MIN
for(i=0; i<n; i++){
currSum += arr[i]
maxSum = max(cs, MS)
if(cs<0)
cs = 0
}
O(n)
linear
edge / corner
Print Screen
ALPHA