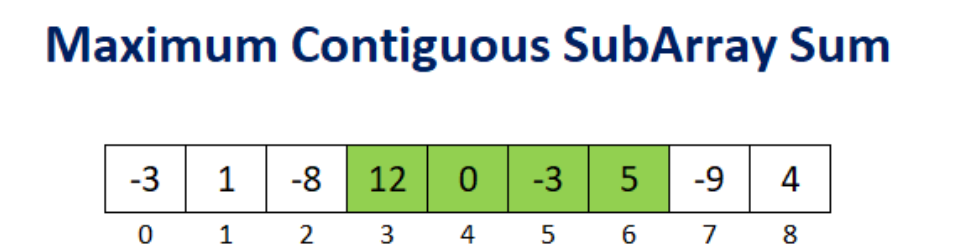
C++ Program - Maximum Subarray Problem

Kadane's algorithm is used to find the maximum sum of a contiguous subarray. Kadane's algorithm is based on the idea of looking for all positive contiguous subarray and find the maximum sum of a contiguous subarray.

In this algorithm, a variable called *max_sum* is created to store maximum sum of the positive contiguous subarray till current iterated element and a variable called *current_sum* is created to store sum of the positive subarray which ends at current iterated element. In each iteration, *current_sum* is compared with *max_sum*, to update *max_sum* if it is greater than *max_sum*.

Example:

To understand the kadane's algorithm, lets consider an array *Array = [-3, 1, -8, 12, 0, -3, 5, -9, 4]* and discuss each step taken to find the maximum sum of all positive contiguous subarray.



```
max_sum = current_sum = 0

Step 1: i = 0, Array[0] =  -3
current_sum = current_sum + (-3) = -3
Set current_sum = 0 because current_sum < 0

Step 2: i = 1, Array[0] =  1
current_sum = current_sum + 1 = 1
update max_sum = 1 because current_sum > max_sum

Step 3: i = 2, Array[0] =  -8
current_sum = current_sum + (-8) = -7
Set current_sum = 0 because current_sum < 0

Step 4: i = 3, Array[0] =  12
current_sum = current_sum + 12 = 12
update max_sum = 12 because current_sum > max_sum

Step 5: i = 4, Array[0] =  0
current_sum = current_sum + 0 = 12

Step 6: i = 5, Array[0] =  -3
current_sum = current_sum + (-3) = 9

Step 7: i = 6, Array[0] =  5
current_sum = current_sum + 5 = 14
update max_sum = 14 because current_sum > max_sum

Step 8: i = 7, Array[0] =  -9
current_sum = current_sum + (-9) = 5

Step 9: i = 8, Array[0] =  4
current_sum = current_sum + 4 = 9
```

Hence, after all iterations, the value of *max_sum* is 14. The stating index point and end index point of this subarray are 3 and 6 respectively.

Run this code

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

// function for kadane's algorithm
static int kadane(int Array[], int n) {
  int max_sum = 0;
  int current_sum = 0;

  for(int i=0; i<n; i++) {
    current_sum = current_sum + Array[i];

    if (current_sum < 0)
      current_sum = 0;

    if(max_sum < current_sum)
      max_sum = current_sum;
  }
  return max_sum;
}

// test the code
int main() {
  int MyArray[] = {-3, 1, -8, 12, 0, -3, 5, -9, 4};
  int n = sizeof(MyArray) / sizeof(MyArray[0]);
  cout<<"Maximum SubArray is: "<<kadane(MyArray, n);
  return 0;
}
```

The above code will give the following output:

```
Maximum SubArray is: 14
```

To get the location of maximum subarray, variables *max_start* and *max_end* are maintained with the help of variables *current_start* and *current_end*.

Run this code

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

// function for kadane's algorithm
static void kadane(int Array[], int n) {
  int max_sum = 0;
  int current_sum = 0;

  int max_start = 0;
  int max_end = 0;
  int current_start = 0;
  int current_end = 0;

  for(int i=0; i<n; i++) {
    current_sum = current_sum + Array[i];
    current_end = i;

    if (current_sum < 0) {
      current_sum = 0;
      //Start a new sequence from next element
      current_start = current_end + 1;
    }

    if(max_sum < current_sum) {
      max_sum = current_sum;
      max_start = current_start;
      max_end = current_end;
    }
  }

  cout<<"Maximum SubArray is: "<<max_sum<<"\n";
  cout<<"Start index of max_Sum: "<<max_start<<"\n";
  cout<<"End index of max_Sum: "<<max_end<<"\n";
}

// test the code
int main() {
  int MyArray[] = {-3, 1, -8, 12, 0, -3, 5, -9, 4};
  int n = sizeof(MyArray) / sizeof(MyArray[0]);
  kadane(MyArray, n);
  return 0;
}
```

The above code will give the following output:

```
Maximum SubArray is: 14
Start index of max_Sum: 3
End index of max_Sum: 6
```

Time Complexity:

The time complexity of Kadane's algorithm is *Θ(N)*.

Recommended Pages

C++ - Swap two numbers