DESIGN AND ANALYSIS OF ALGORITHM

EXRECISE: 4

PRATEEK MOHANTY

20BCE1482

MAXIUM SUB-ARRAY SUM

* BRUTE FORCE APPROACH-1

CODE

#include <iostream>

#include <vector>

using namespace std;

int maxSubarraySum(vector<int> A, int n)

{

    int i, j, k;

    int max\_sum = 0;

    for (i = 0; i < n; i++)

    {

        for (j = i; j < n; j++)

        {

            int sum = 0;

            for (k = i; k < j; k++)

                sum = sum + A[k];

            if (sum > max\_sum)

                max\_sum = sum;

        }

    }

    return max\_sum;

}

int main()

{

    int n;

    cin >> n;

    vector<int> vec1;

    for (int i = 0; i < n; i++)

    {

        int a;

        cin >> a;

        vec1.push\_back(a);

    }

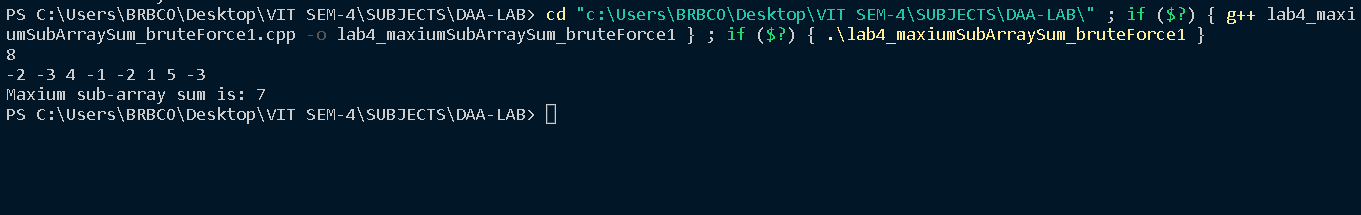
    cout << "Maxium sub-array sum is: ";

    cout << maxSubarraySum(vec1, n);

    return 0;

}

OUTPUT



* BRUTE FORCE APPROACH-2

CODE

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int main()

{

    int n;

    cin >> n;

    vector<int> vec1;

    cout << "Enter the elements of the vector" << endl;

    for (int i = 0; i < n; i++)

    {

        int a;

        cin >> a;

        vec1.push\_back(a);

    }

    // int sum = 0;

    int max = INT16\_MIN;

    int leftIndex = -1, rightIndex = -1;

    for (int i = 0; i < n - 1; i++)

    {

        int sum = vec1[i];

        for (int j = i + 1; j < n; j++)

        {

            sum += vec1[j];

            if (sum > max)

            {

                max = sum;

                leftIndex = i;

                rightIndex = j;

            }

        }

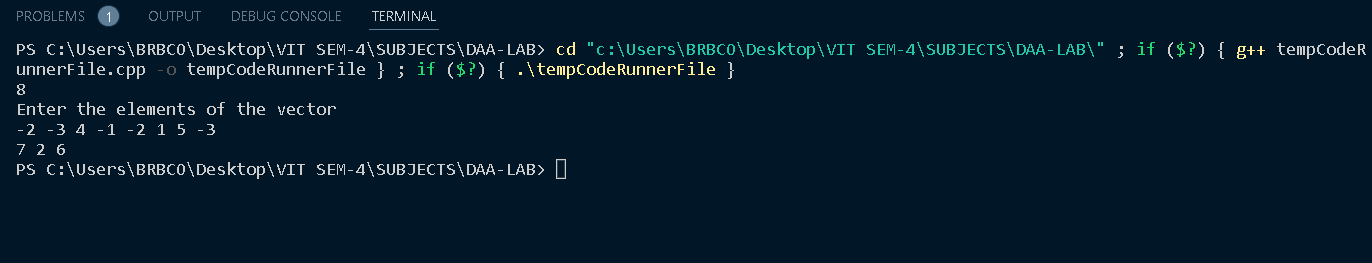
    }

    cout << max << " " << leftIndex << " " << rightIndex;

    return 0;

}

OUTPUT



* DIVIDE AND CONQUER METHOD

#include <iostream>

#include <vector>

using namespace std;

int max(int x, int y)

{

    return (x > y) ? x : y;

}

int maxSubArraySum(vector<int> vec1, int low, int high)

{

    if (high <= low)

        return vec1[low];

    int mid = (low + high) / 2;

    int left\_max = INT16\_MIN;

    int sum = 0;

    for (int i = mid; i >= low; i--)

    {

        sum += vec1[i];

        if (sum > left\_max)

            left\_max = sum;

    }

    int right\_max = INT16\_MIN;

    sum = 0;

    for (int i = mid + 1; i <= high; i++)

    {

        sum += vec1[i];

        if (sum > right\_max)

            right\_max = sum;

    }

    int max\_left\_right = max(maxSubArraySum(vec1, low, mid), maxSubArraySum(vec1, mid + 1, high));

    return max(max\_left\_right, left\_max + right\_max);

}

int main()

{

    int n;

    cin >> n;

    vector<int> vec1;

    for (int i = 0; i < n; i++)

    {

        int a;

        cin >> a;

        vec1.push\_back(a);

    }

    cout << "Maximum sub array sum is: ";

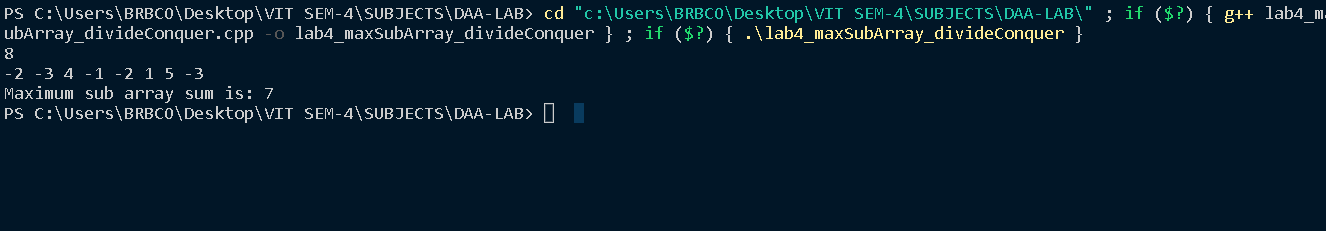
    int b = maxSubArraySum(vec1, 0, n - 1);

    cout << b;

    return 0;

}

OUTPUT



* USING AUXILIARY ARRAY

CODE

#include <iostream>

#include <vector>

using namespace std;

int main()

{

    int n;

    cin >> n;

    vector<int> vec1;

    for (int i = 0; i < n; i++)

    {

        int a;

        cin >> a;

        vec1.push\_back(a);

    }

    vector<int> res;

    res.push\_back(vec1[0]);

    for (int i = 1; i < n; i++)

    {

        if (vec1[i] + res[i - 1] > 0)

        {

            res[i] = vec1[i] + res[i - 1];

        }

        else

            res[i] = res[i - 1];

    }

    cout << "Maximum Sub Array sum is: ";

    int ans = 0;

    for (int i = 0; i < n; i++)

    {

        ans = max(ans, res[i]);

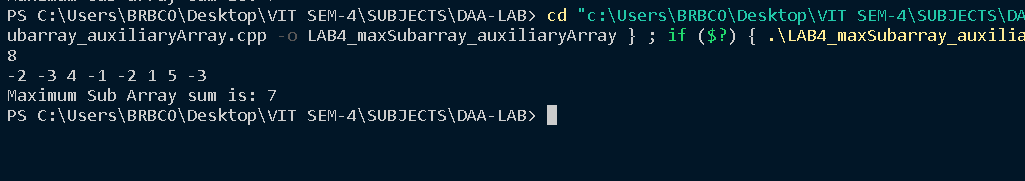
    }

    cout << ans;

    return 0;

}

OUTPUT



* KADANE’S ALGORITHM

CODE

#include <iostream>

#include <vector>

#include <climits>

using namespace std;

int main()

{

    int n;

    cin >> n;

    vector<int> vec1;

    cout << "Enter the elements of the vector" << endl;

    for (int i = 0; i < n; i++)

    {

        int a;

        cin >> a;

        vec1.push\_back(a);

    }

    int maxim = INT16\_MIN;

    int leftIndex = -1, rightIndex = -1;

    int sum = 0;

    for (int i = 0; i < n; i++)

    {

        sum += vec1[i];

        maxim = max(sum, maxim);

        if (sum < 0)

        {

            sum = 0;

        }

    }

    cout << maxim;

}

OUTPUT

