DAA Lab

Practical 4

Amber Shukla

A1-B3-42

**Aim:** Implement maximum sum of subarray for the given scenario of resource allocation using the divide and conquer approach.

**Code:**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

vector<ll> arr;

void Subarray(ll ans)

{

    ll maxSum = INT\_MIN;

    ll start = 0, end = 0;

    for (ll i = 0; i < arr.size(); i++)

    {

        for (ll j = i; j < arr.size(); j++)

        {

            ll currentSum = 0;

            for (ll k = i; k <= j && currentSum <= ans; k++)

                currentSum += arr[k];

            if (currentSum == ans)

            {

                start = i;

                end = j;

                goto found;

            }

        }

    }

found:

    cout << "Maximum Sum Subarray is: [ ";

    for (ll i = start; i <= end; i++)

        cout << arr[i] << " ";

    cout << "]" << endl;

}

ll DAC(ll left, ll right, ll constraint)

{

    if (left > right)

        return INT\_MIN;

    if (left == right)

        if (arr[left] <= constraint)

            return arr[left];

        else

            return INT\_MIN;

    ll mid = left + (right - left) / 2;

    ll leftMax = DAC(left, mid, constraint);

    ll rightMax = DAC(mid + 1, right, constraint);

    ll sum = 0, leftSum = INT\_MIN, rightSum = INT\_MIN;

    for (ll i = mid - 1; i >= left; i--)

    {

        sum += arr[i];

        if (sum <= constraint)

            leftSum = max(leftSum, sum);

    }

    sum = 0;

    for (ll i = mid; i <= right; i++)

    {

        sum += arr[i];

        if (sum <= constraint)

            rightSum = max(rightSum, sum);

        if (leftSum != INT\_MIN && leftSum + sum <= constraint)

            rightSum = max(rightSum, leftSum + sum);

    }

    ll cross = max(leftSum, rightSum);

    return max({leftMax, rightMax, cross});

}

int main()

{

    ll size, constraint;

    cout << "Enter Size of Array: ";

    cin >> size;

    arr.resize(size);

    if (!arr.size())

    {

        cout << "ERROR: Empty Array" << endl;

        return 0;

    }

    cout << "Enter Array: ";

    for (ll i = 0; i < size; i++)

        cin >> arr[i];

    cout << "Enter Resource Constraint: ";

    cin >> constraint;

    ll ans = DAC(0, size - 1, constraint);

    if (ans == INT\_MIN)

    {

        cout << "No Feasible Subarray\n";

        return 0;

    }

    Subarray(ans);

    cout << "Maximum Sum = " << ans << endl;

    return 0;

}

**Output:**

A computer screen shot of a program

AI-generated content may be incorrect. **A screen shot of a computer

AI-generated content may be incorrect.**

**LeetCode:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Code:**

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

class Solution

{

public:

    int maxSubArray(vector<int> &nums)

    {

        int sum = -10001, max\_sum = sum;

        if (nums.size() == 1)

            return nums[0];

        for (int i : nums)

        {

            if (sum < 0)

                sum = 0;

            sum += i;

            if (sum > max\_sum)

                max\_sum = sum;

            ;

        }

        return max\_sum;

    }

};