Assignment 1

P Aashrith - EE18BTECH11035

Download all Codes from

https://github.com/Aashrith20/C−and−DS/tree/main/Assignment%201/codes

Download all latex-tikz codes from

https://github.com/Aashrith20/C−and−DS/blob/main/Assignment%201/ee18btech11035.tex

1 Problem

Consider a Sequence of 14 elements: A = [-5,-10,6,3,-1,-2,13,4,-9,-1,4,12,-3,0]. The subsequence sum $S(i, j) = \sum_{k=i}^{j} A[k]$. Determine the maximum of $S(i, j)$, where $0 \leq i \leq j < 14$. (Divide and Conquer approach may be used)

2 Solution

C Code can found in the following directory

https://github.com/Aashrith20/C−and−DS/blob/main/Assignment%201/codes/code.c

By Divide and Conquer :
Calculate mid index of array which is $(start+end)/2$ Then recursively call the function to solve the left and right subarray.

For left subarray start index is the same as previous start index but end index is mid.At each recursive call Store the returned value in a variable **LeftSubArrayMax** .

In the case of right subarray start index is $(mid + 1)$ index but end index is the same as previous end index.At each recursive call Store the returned value in a variable **RightSubArrayMax**.

Calculating the maximum subsequence sum of whole array:
Step 1:
Calculate the maximum of subsequence sum of all the subarrays in the left subarray with ending index as mid.

$$left_{sum} = max(S(i, mid))\forall\ (start \leq i \leq mid) \tag{2.0.1}$$

Step 2:
Calculate the maximum subsequence sum sum of all the subarrays in the right subarray with starting index as mid+1.

$$right_{sum} = max(S(mid + 1, i))\forall\ (mid + 1 \leq i \leq end) \tag{2.0.2}$$

Calculate the $left_{sum}$ and $right_{sum}$ at each recursion call.
To get the maximum subsequence sum return $max(LeftSubArrayMax, RightSubArrayMax, left_{sum}+ right_{sum})$

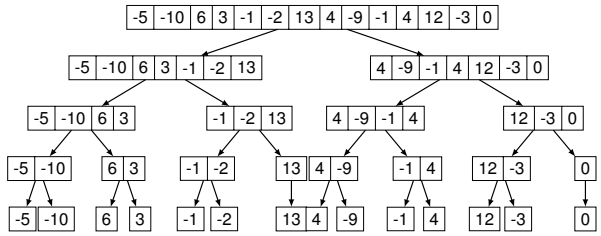

Fig. 0

Answer for this particular array is 29.

3 Vector Representation

$S(i, j)$ represents the sum of all elements of array from index i to j.

Let **x** be a vector defined as

$$\mathbf{x} = \left.\begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}\right\} \text{from index i to j}$$

$S(i, j)$ can be rewritten in the vector representation as follows

$$S(i, j) = \sum_{k=i}^{j} A[k] \tag{3.0.1}$$

Taking inner product of A and **x** gives $S\ (i, j)$ Where A is input vector

$$S(i, j) = A^T x \tag{3.0.2}$$

4 Complexities

Time Complexity$(T\ (N))$ **:**
T(N) is the order of the time taken for completion of the algorithm for an input array of length N

$$T\ (N) = 2T\ (N/2) + O\ (N) \tag{4.0.1}$$
$$T\ (N) = 2T\ (N/2) + cN \tag{4.0.2}$$
$$T\ (N/2) = 2T\ (N/4) + cN/2 \tag{4.0.3}$$

Substituting (4.0.3) in (4.0.2)

$$T\ (N) = 2\left[2T\ (N/4) + c\ (N/2)\right] + cN \tag{4.0.4}$$
$$T\ (N) = 4T\ (N/4) + 2cN \tag{4.0.5}$$
$$\text{Similarly, } T\ (N) = 8T\ (N/8) + 3cN \tag{4.0.6}$$
$$\text{In general } T\ (N) = 2^k T\left(N/2^k\right) + kcN \tag{4.0.7}$$

Upper limit of K occurs when :

$$N/2^k = 1 \tag{4.0.8}$$
$$2^k = N \tag{4.0.9}$$
$$k = \log N \tag{4.0.10}$$

Assuming T(1) = 1
Solving (4.0.7) by Substituting $k = \log N$

$$T\ (N) = 2^k T\left(N/2^k\right) + kcN \tag{4.0.11}$$
$$T\ (N) = NT\ (1) + cN \log N \tag{4.0.12}$$
$$T\ (N) = N + cN \log N \tag{4.0.13}$$

$$T\ (N) \approx O\,(\text{NlogN}) \tag{4.0.14}$$

Space Complexity$(S\ (N))$ **:**
S(N) is the order of the Space taken for completion of the algorithm for an input array of length N.
This algorithm takes only 6 variables

$$\text{So, } S\ (N) = O\,(1) \tag{4.0.15}$$

5 Optimization

C Code can found in the following directory

https://github.com/Aashrith20/C−and−DS/blob/main/Assignment%201/codes/optimized_code.c

Initialize 2 variables **GlobalMax** and **LocalMax** with $-\infty$
GlobalMax : It stores the overall maximum sum of a subarray.
LocalMax : It stores the temporary sum of a subarray which we are assuming to be GlobalMax.

While iterating the array If the LocalMax is positive continue the same subarray by adding element at that index to the LocalMax else begin a new subarray by updating the LocalMax to element at that index.

```
if(LocalMax<0)
{
    LocalMax = arr[i];
}
else
{
    LocalMax = LocalMax + arr[i];
}
```

At each iteration update the GlobalMax.

```
GlobalMax = max(GlobalMax,LocalMax);
```

Finally, return the GlobalMax.

6 Complexities

Time Complexity : T(N) is the order of the time taken for completion of the algorithm for an input array of length N.
Iteration is done N times

$$\text{So, } T\ (N) = O\ (N) \tag{6.0.1}$$

Space Complexity : S(N) is the order of the Space taken for completion of the algorithm for an input array of length N.
This algorithm takes only 2 variables

$$\text{So, } S\ (N) = O\ (1) \tag{6.0.2}$$