ADA-F23-A2 (Max Sub Array Time Complexity Analysis)

Muhamamd Bilal

November 27, 2023

Question

Write a C++ program to generate random numbers ranging from -1000 to 1000 to populate 20 files. Each file contains an array of elements. Each file is different in size. The sizes are n=10, 15, 20, 30, 40, 50, 60, 70, 90, 100, 150, 200, 250, 300, 350, 400, 450, 500, 750, 1000. Each file should contain both positive and negative numbers and at least 50 negative numbers. Your program should then read each file and find the maximum subarray using the following two different approaches: i. Brute Force Approach ii. Divide and Conquer Approach Analyze both approaches and estimate time complexity in terms of asymptotic notations. Find out the execution time taken for both approaches. Report this in a document comprising 2 to 10 pages. Compare your analysis in asymptotic notation and the execution time taken in the case of each file. The comparison should include text, tables, and graphs.

1 Time Complexity (Asymtotic Notations)

1.1 Brute Force

```
int maxSubArrayBF(int nums[], int n)
{

    int maxSum = 0; ........ 1

    for (int i = 0; i < n; i++) ........ n+1
    {

        int currSum = 0; ........ n

        for (int j = i; j < n; j++)  ........ n*n+1
        {
```

```
            currSum += nums[j];    ........ n*n

            if (currSum > maxSum)
                maxSum = currSum;    ........ n*n
        }
    }

    return maxSum;    ........ 1
}
```

We can express the time complexity of Brute Force Approach as follows:

$$T(n) = 1 + (n+1) + n + (n*n+1) + n*n + n*n + 1 = O(n^2) \quad (1)$$

1.2 Divide and Conquer

```
int maxSubArrayDC(int A[], int low, int high)
{
    if (high == low)
        return A[high];              ........ 1

    int mid = (high + low) / 2;

    int left = maxSubArrayDC(A, low, mid);  ........ T(n/2)

    int right = maxSubArrayDC(A, mid + 1, high);  ........ T(n/2)

    int crossings = maxCrossing(A, low, mid, high);  ........ O(n)

    return maximum(maximum(left, right), crossings);
}
```

Now the recurrence equation is written as :

$$T(n) = 2T(n/2) + (n+1)/2 = 2T(n/2) + n \quad (2)$$

.

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \quad \text{(i)} \\
T(n/2) &= 2T(n/4) + n/2 \\
&\text{Putting in (i):} \\
T(n) &= 2[2T(n/4) + n/2] + n \\
T(n) &= 4T(n/4) + 2n \\
T(n/4) &= 2T(n/8) + n/4 \\
&\text{Putting in (i):} \\
T(n) &= 4[2T(n/8) + n/4] + 2n \\
T(n) &= 8T(n/8) + 3n \\
&\text{Assuming } k \text{ steps:} \\
T(n) &= 2^k T(n/2^k) + kn \\
\text{Assume } n/2^k &= 1 \quad \text{(i.e., } n = 2^k\text{)} \\
k &= \log_2 n \\
T(n) &= 2^{\log n} * T(n/2^{\log n} + nlogn \\
T(n) &= n * T(n/4) + nlogn \\
T(n) &= n + nlogn
\end{aligned}
$$

$$
T(n) = O(nlogn) \tag{3}
$$

2 Comparison of Execution Times

Execution Time: Brute Force Execution Time increases notably with larger file sizes, while Divide and Conquer Execution Time remains comparatively lower and more stable.

Conclusion

- For smaller file sizes (up to 100), both algorithms perform similarly with negligible execution time differences.
- As file sizes grow larger, the Brute Force approach's execution time significantly increases compared to the Divide and Conquer approach.
- Divide and Conquer shows better scalability and efficiency, maintaining lower execution times even for larger file sizes compared to the Brute Force approach.

The Divide and Conquer approach demonstrates better performance in terms of execution time as the file size grows larger, indicating its efficiency in handling larger datasets compared to the Brute Force approach.

File Size	Max Sum	BF ET (microseconds)	DC ET (microseconds)
10	1928	2	4
15	3328	2	6
20	4955	3	8
30	2501	5	12
40	7471	7	17
50	10087	10	20
60	2966	13	41
70	9732	16	29
80	2608	21	32
100	5472	33	36
100	5472	32	41
200	10647	110	80
250	11318	178	106
300	4430	259	117
350	10036	343	147
400	14847	450	152
450	16156	2201	182
500	9789	684	192
750	17290	1565	300
1000	11546	2773	397

Table 1: Comparison of Execution Time between Brute Force and Divide and Conquer Approaches

2.1 Graphs

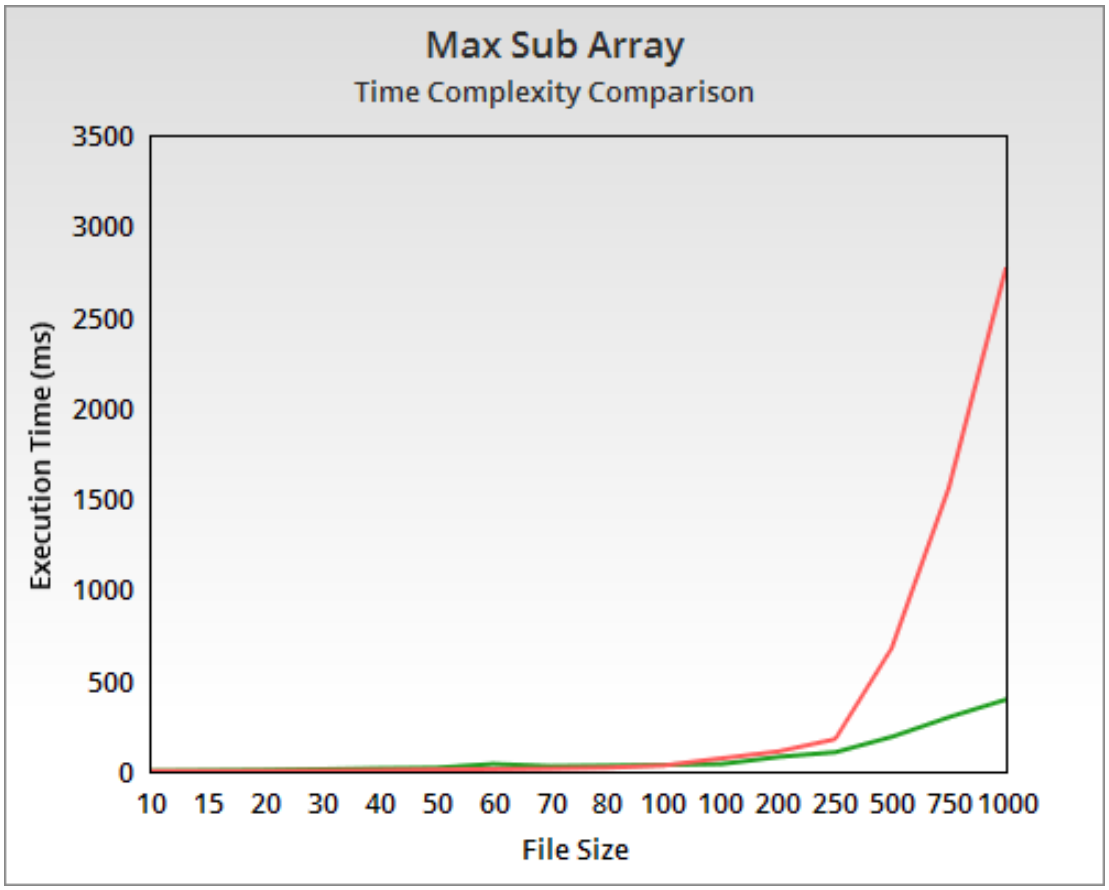

Figure 1: Comparison Grph (Red for BF and Green for DC)