CSL 3010: Design and Analysis of Algorithms July-November 2022

Lecture 1: Non Trivial Algorithms

Lecturer: Dr. Saket Saurabh *Scribe: Mukul, Sawan, Jaimin, Tanay*

1 Problem 1: Max element of array

Input : Given an Array A, consisting of n elements.
Output : Return maximum element from the array A

Approach : Iterate over the array and maintain a variable, say, *max_ele* which will keep track of the maximum element found in the array. If the incoming number is greater than the *max_ele* then update it's value. Finally, after coming out of the traversal return *max_ele* which will be the required output.
Analysis :
1.) Time complexity - O(n)
2.) Space complexity - O(1)
3.) Maximum Number of comparisons - n-1

2 Problem 2: Max and min element in array

Input : Given an Array A, consisting of n elements.
Output : Return maximum and minimum element from the array A

Approach : Iterate over the array and maintain a variable, say, *max_ele* which will keep track of the maximum element and min_ele to keep track of the minimum element found in the array. If the incoming number is greater than the *max_ele* then update it's value else if the value is lowe than min_ele the update it's value. Finally, after coming out of the traversal return both variables which will be the required output.
Analysis :
1.) Time complexity - **O(n)**
2.) Space complexity - **O(1)**
3.) Maximum Number of comparisons - 2n-2.

2.1 Improvement

Aim : we would like to build an algorithm which takes less number of comparisons than 2n-2.
Approach : If number of elements in the array are odd then initialize min and max as the first element.
If number of elements in the array are even then initialize min and max as minimum and maximum of the first two elements respectively.

For the rest of the elements, pick them in pairs and compare their maximum and minimum with max and min respectively and update the value accordingly. **Analysis** :
1.) Time complexity - **O(n)**
2.) Space complexity - **O(1)**
3.) Number of comparisons - If **n is odd**: **3*(n-1)/2**.
If **n is even: 3n/2 -2.**

3 Problem 3: Multiplication of two numbers

Consider two numbers *53* and *47*, to multiply these two, we require 4 digit multiplications namely (3x7, 5x7, 4x3, 4x5)

$$\begin{array}{r} 53 \\ \times\ 47 \\ \hline 371 \\ +2120 \\ \hline 2491 \\ \hline \end{array}$$

let's consider another set of numbers, *78* and *21*

$$\begin{array}{r} 78 \\ \times\ 21 \\ \hline 78 \\ +1560 \\ \hline 1638 \\ \hline \end{array}$$

Thus we can write: 78 x 21 = (7 x (10 + 8)) x ((2 x 10) + 1)
which is $= (7 \text{ x } 2).10^2 + (7.1 + 8.2) + 8.1$

Thus we can say, if we have two numbers X and Y and we consider it as two number of length n/2 each i.e.

$$\begin{aligned} \text{X} &= \| - - - - - A - - - - \| - - - - - B - - - - \| \\ \text{Y} &= \| - - - - - C - - - - \| - - - - - D - - - - \| \end{aligned}$$

$$\begin{aligned} \text{X} &= 10^{n/2}.A + B \\ \text{Y} &= 10^{n/2}.C + D \\ \text{XY} &= 10^n AC + 10^{n/2}BC + 10^{n/2}AD + BD \end{aligned}$$

Thus we can form a recurrence relation for the same, which is :-

$$\text{T(n)} = 4\text{T(n/2)} + \text{cn}$$

Now let's evaluate this recurrence for that we have two methods

- 1.) Expand and Verify
- 2.) Induction (Guess and verify)

3.1 Expand and verify

$$
\begin{aligned}
&\textit{Base case} : \mathrm{T}(1) = 1 \\
&\mathrm{T(n)} = 4\mathrm{T(n/2)} + \mathrm{cn} \\
&\qquad = 4[4\mathrm{T(n/4)} + \mathrm{c.(n/2)}] + \mathrm{cn} \\
&\qquad = 4^2 T(n/4) + 2cn + cn \\
&\qquad = 4^2[4T(n/8) + c.(n/4)] + 3cn \\
&\qquad = 4^3 T(n/8) + 4cn + 3cn
\end{aligned}
$$

after k steps this will evaluate to :

$$4^k T(n/k) + (1 + 2 + 4 + ...)cn$$

and finally to,

$$
\begin{aligned}
\mathrm{T(n)} &= \mathrm{n}^2 + cn^2 - n \\
\mathbf{T(n)} &= \mathbf{O(n^2)}
\end{aligned}
$$

3.2 Induction

Let us assume that this equation T(n) = 4T(n/2) + cn, T(1) = 1, has a solution of the form $\mathrm{an}^2 + \mathrm{bn}$, where a,b are constants. Then,

$$
\begin{aligned}
\mathrm{T(n)} &\leq 4\mathrm{T(n/2)} + \mathrm{cn} \leq an^2 - bn \\
\mathrm{T(n)} &\leq 4(an^2/4 - bn/2) + cn \leq an^2 - bn \\
\mathrm{an}^2 &- 2bn + cn \leq an^2 - bn
\end{aligned}
$$

This implies that our assumption was correct and indeed the solution for the equation in consideration is of the form $\mathrm{an}^2 - bn$, hence we can say from this too that $\mathrm{T(n)} = \mathbf{0(n^2)}$.

Continuing our approach for multiplying X and Y, we had,

$$
\begin{aligned}
\mathrm{X} &= 10^{n/2}.A + B \\
\mathrm{Y} &= 10^{n/2}.C + D \\
\mathrm{XY} &= 10^n AC + 10^{n/2}BC + 10^{n/2}AD + BD \\
\mathrm{XY} &= 10^n AC + 10^{n/2}BC + 10^{n/2}AD + BD - 10^{n/2}AC + 10^{n/2}AC - 10^{n/2}BD + 10^{n/2}BD \\
\mathrm{XY} &= (10^n - 10^{n/2)}AC + 10^{n/2}(A + B)(C + D) + (1 - 10^{n/2})BD
\end{aligned}
$$

We now only have 3 multiplications of n/2 each, so the recurrence will be

$$
\begin{aligned}
&\mathrm{T(n)} = 3\mathrm{T(n/2)} + \mathrm{cn} \\
&\textit{Base case} : \mathrm{T}(1) = 1 \\
&\qquad = 3[3\mathrm{T(n/4)} + \mathrm{c.(n/2)}] + \mathrm{cn} \\
&\qquad = 3^2 T(n/4) + 3cn/2 + cn \\
&\qquad = 3^3 T(n/8) + 3^2 c(n/4) + 3cn/2 + cn \\
&\qquad = 3^{\log_2 3} + cn[1 + (3/2) + (3^2/4) + ...]
\end{aligned}
$$

$$\mathbf{T(n) = O(n^{\log_2 3})}$$

Conclusion: Hence, we improved our time complexity from $\mathbf{O(n^2)} - -to - -\mathbf{O(n^{1.585})}$.
The computationally best algorithm for multiplication was given by **David Harvey** and **van der Hoeven** which uses the strategies of using number-theoretic transforms introduced with the Schonhage–Strassen algorithm to multiply integers using only **O(nlogn)**