H1.Peinan Weng

problem 1 *Karatsuba Example*

Carry out the Karatsuba algorithm for $1234 \cdot 4231$.

Note: Show each of the recurrence steps, and their calculations. You will not get any point if you do just a multiplication.

Answer

Step 1 Compute 12×42
$a = 1$ $b = 2$ $c = 4$ $d = 2$
Step 1-1 Compute $a \cdot c = 1 \times 4 = 4$
Step 1-2 Compute $b \cdot d = 2 \times 2 = 4$
Step 1-3 Compute $(a + b) \cdot (c + d) = 3 \times 6 = 18$
Step 1-4 Compute $Step\ 3 - Step2 - Step1 = 18 - 4 - 4 = 10$
Step 1-5 Compute $4(00) + 10(0) + 4 = 504$

Step 2 Compute 34×31
$a = 3$ $b = 4$ $c = 3$ $d = 1$
Step 2-1 Compute $a \cdot c = 3 \times 3 = 9$
Step 2-2 Compute $b \cdot d = 4 \times 1 = 4$
Step 2-3 Compute $(a + b) \cdot (c + d) = 7 \times 4 = 28$
Step 2-4 Compute $Step\ 3 - Step2 - Step1 = 28 - 4 - 9 = 15$
Step 2-5 Compute $9(00) + 15(0) + 4 = 1054$

Step 3 Compute 46×73
$a = 4$ $b = 6$ $c = 7$ $d = 3$
Step 3-1 Compute $a \cdot c = 4 \times 7 = 28$
Step 3-2 Compute $b \cdot d = 6 \times 3 = 18$
Step 3-3 Compute $(a + b) \cdot (c + d) = 10 \times 10 = 100$
Step 3-4 Compute $Step\ 3 - Step2 - Step1 = 100 - 18 - 28 = 54$
Step 3-5 Compute $28(00) + 54(0) + 18 = 3358$

Step 4 Compute 1234×4231
$a = 12$ $b = 34$ $c = 42$ $d = 31$
Step 4-1 Compute $a \cdot c = 12 \times 42 = 504$
Step 4-2 Compute $b \cdot d = 34 \times 31 = 1054$
Step 4-3 Compute $(a + b) \cdot (c + d) = 46 \times 73 = 3358$
Step 4-4 Compute $Step\ 3 - Step2 - Step1 = 3358 - 1054 - 504 = 1800$
Step 4-5 Compute $504(0000) + 1800(00) + 1054 = 5221054$

PROBLEM 2 *Proof by Induction*

Use the substitution method (thats the induction one) to prove the runtime of the following recurrence relation:

$T(n) = 2 \times T(\frac{n}{2}) + n^2$

How large does the constant c have to be?

Answer

Assume $\forall n_0 < n, T(n_0) \leq cn_0^2$

As for n,

$$\begin{aligned} T(n) &= 2 \times T(\frac{n}{2}) + n^2 \\ &\leq 2 \times (c \times (\frac{n}{2})^2) + n^2 \\ &= \frac{cn^2}{2} + n^2 \end{aligned}$$

In order to prove $T(n) = O(n^2)$, We only need to prove $T(n) \leq cn^2$.

Then we only need to prove $T(n) \leq \frac{cn^2}{2} + n^2 \leq cn^2$.

In that case, $\frac{c}{2} + 1 \leq c$, then $\frac{c}{2} \geq 1$, then $c \geq 2$.

Therefore, $T(n) = O(n^2)$, and the minimum c is 2.

PROBLEM 3 *Flip-flop*

Consider the recurrence $T(n) = 2 \times T(\frac{n}{2}) + f(n)$ in which:

$$f(n) = \begin{cases} n^3 & if \lfloor log(n) \rfloor \ is\ even \\ n^2 & otherwise \end{cases}$$

Show that $f(n) = \Omega(n^{\log_b(a)+\epsilon})$. Explain why the third case of the Master's theorem stated above does not apply. Prove a θ bound for the recurrence.

Answer

Question 1

$f(n) \geq n^2$ and $\forall\ 0 < \epsilon < 1\ \ n^{log_b(a)+\epsilon} = n^{1+\epsilon} < n^2 \Rightarrow f(n) = \Omega(n^{log_b(a)+\epsilon})$

Question 2

In order to use case 3 of the Master Theorem, following condition is required:

$$\exists k < 1\ \forall n > n_0\ \ af(\frac{n}{b}) \leq kf(n)$$

In this case, we need to prove $\frac{2f(\frac{n}{2})}{f(n)} \leq k < 1$, assume $n \to \infty$ and $\lfloor log(n) \rfloor$ odd

Then $\frac{2f(\frac{n}{2})}{f(n)} = \frac{2(\frac{n}{2})^3}{n^2} = \frac{n}{4} \to \infty\ \ (n \to \infty)$, so it doesn't satisfy this condition.

Question 3

$T(n) = 2 \times T(\frac{n}{2}) + f(n) \leq 2 \times T(\frac{n}{2}) + n^3$

According to the Master Theorem, $T(n) \leq T'(n) = 2 \times T(\frac{n}{2}) + n^3 = O(n^3)$

Then we need to prove the lower-bound.

Case 1 Assume $\lfloor log(n) \rfloor$ is even and $\lfloor log(\frac{n}{2}) \rfloor$ is odd.

$T(n) = 2 \times T(\frac{n}{2}) + n^3 = 4 \times T(\frac{n}{4}) + n^3 + 2(\frac{n}{2})^2 > 4 \times T(\frac{n}{4}) + n^3$

According to the Master Theorem, $T(n) > T''(n) = 4 \times T''(\frac{n}{4}) + n^3 = \Omega(n^3)$

Case 2 Assume $\lfloor log(n) \rfloor$ is odd and $\lfloor log(\frac{n}{2}) \rfloor$ is even.

$T(n) = 2 \times T(\frac{n}{2}) + n^2 = 4 \times T(\frac{n}{4}) + n^2 + \frac{n^3}{4} > 4 \times T(\frac{n}{4}) + \frac{n^3}{4}$

According to the Master Theorem, $T(n) > T'''(n) = 4 \times T'''(\frac{n}{4}) + \frac{n^3}{4} = \Omega(n^3)$

Then $T(n) = \Omega(n^3)$, so the θ bound of $T(n)$ is n^3

PROBLEM 4 *Skyline*

Design and analyze a divide and conquer algorithm that computes the right and left niftyness functions of a skyline s with n buildings. The input $A[1, ..., N]$ consists of the heights of each building; assume all buildings have unique heights. Your solution should have a running time of $\Theta(n \log n)$.

Note: Explain the correctness of your algorithm and run-time.

Answer (In this answer, I use numbers to denote the height of each building.)

The algorithm is like merge-sort. However, we need to count numbers here.

First of all, the sum of left and right niftyness of each element in an array only depends on its order in the array. For example, x is the nth smallest element then $n_l(x) + n_r(x) = n - 1$. Then for an array with n elements, the total niftyness of each element is $\frac{n(n-1)}{2} = \frac{n^2-n}{2}$. That means for an array with n elements, we only need to calculate n_l or n_r then another one could be get.

```
Pseudo Code
INPUT: Array $A[1, ..., N]$
OUTPUT: Left Niftyness $n_l(A[1, ..., N])$ and Ascending Sorted Array $A[1, ..., N]$
    IF $N = 1$
        $n_l(A) = 0$
    ELSE IF $N = 2$
        IF $A[1] \leq A[2]$
            $n_l(A) = 1$
        ELSE IF $A[1] > A[2]$
            swap $A[1]$ and $A[2]$
            $n_l(A) = 0$
        ENDIF
    ELSE IF $N > 2$
        compute $n_l(A[1, ..., \lfloor \frac{n}{2} \rfloor])$ and $n_l(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$
        for each element $A[x]$ in $n_l(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$, insert $A[x]$ into $A[1, ..., \lfloor \frac{n}{2} \rfloor]$
make sure the array after insertion is ascending sorted. After each insertion store
the index of $A[x]$ in the new array as $I[x]$.
        $n_l(A) = n_l(A[1, ..., \lfloor \frac{n}{2} \rfloor]) + \sum I[x] - n_r(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$
    ENDIF
```

Correctness Proof

For $N = 1$ and $N = 2$, obviously the answer is correct and A will be in ascending order after compute.

For $N > 2$, because those 2 original sub-array are both ascending sorted, after we insert an $A[x]$ into $A[1, ..., \lfloor \frac{n}{2} \rfloor]$, elements in the left of $A[x]$ must be smaller than $A[x]$ and then the number of $A[x]$'s left elements can be denoted as $I[x]$(Assume index starts from 0). The sum of $I[x]$ denotes the left niftyness of elements in the latter part $A[\lfloor \frac{n}{2} \rfloor + 1, ..., N]$. And by doing such insertion, we could make sure that the output array are totally ascending sorted.

Notice that we do not calculate the left niftyness of elements in the former part $A[1, ..., \lfloor \frac{n}{2} \rfloor]$, add $n_l(A[1, ..., \lfloor \frac{n}{2} \rfloor])$ to the result.

In addition, there is another case we need to consider about. When elements in the latter array are swapped after a certain sort, some descending pairs are counted into $\sum I[x]$, so we need to subtract $n_r(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$ from the result.

In summary, the result is $n_l(A) = n_l(A[1, ..., \lfloor \frac{n}{2} \rfloor]) + \sum I[x] - n_r(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$, and because for an array with n elements, the total niftyness of each element is $\frac{n^2-n}{2}$, we do not need to calculate $n_r(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N])$ but use $\frac{n^2-n}{2} - n_l(A[\lfloor \frac{n}{2} \rfloor + 1, ..., N](n = N - \lfloor \frac{n}{2} \rfloor)$ instead.

In this case, $T(n) = 2 \times T(\frac{n}{2}) + n$ (actually the algorithm is merge sort with number counting), according to the Master Theorem, this algorithm has a running time of $\Theta(n \log n)$.