---Recitation Class 3

Q1 – For each of the following recurrences find a close-form expression.

(A) $T(n) = 8T\left(\frac{n}{2}\right) + n^2$

(B) $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + n^2$

Answer

(A) Cost at level i is $8^i \left(\frac{n}{2^i}\right)^2 = 2^{3i}\frac{n^2}{2^{2i}} = 2^i n^2$

$T(n) = n^2 \sum_{i=0}^{\log n} 2^i = n^2\left(2^{\log n+1} - 1\right) = \theta(n^3)$

(B) Use induction to prove $T(n) \leq 2n^2 \; for \; n \geq 1.$

Q2 – In an eccentric party you have been blindfolded! Guests $\{g_1, \dots, g_n\}$ have formed a line. You are supposed to find the guest of a given height as quickly as you can. When you touch a person, they tell you exactly one of the following

1. I am the one!
2. I am shorter.
3. I am taller.

Moreover, they let you know if you're at either end of the line. The only thing that you know about the line is the way it was created: guests started by forming a line in increasing order of height; then, they cyclically rotated.

Would your solution work if two or more guests could be of the same height?

Answer

Think of $G[g_1, \dots, g_n]$ as an array of heights which is a rotated sorted array, and h is a given height. Here is an algorithm to find the location of h.

```
Find_height(G[1..n])

1) Find middle point mid = (l + n)/2

2) If G[mid] = h, return mid.

3) Else If G[1] <= G[mid]   // means that G[l..mid] is sorted
    a) If h lies in range from G[l] to G[mid], recur for G[l..mid].
    b) Else recur for G[mid+1..n]

4) Else // means that G[mid+1..n] must be sorted
    a) If h lies in range from G[mid+1] to G[n], recur for G[mid+1..n].
    b) Else recur for G[l..mid]
```

The algorithm is very similar to binary search algorithm, which on each call of the function, some constant time is spent to compare h with the middle element and if is not the same as the middle element, spends some constant time to decide on which half to recur, then recurs on one of the halves; Thus the running time is $T(n) = T\left(\frac{n}{2}\right) + c$, which implies that $T(n) = \theta(\log n)$.

Q3 – Suppose we have two binary integers $x = \langle x_n, x_{n-1}, \dots, x_1\rangle$ and $y = \langle y_n, y_{n-1}, \dots, y_1\rangle$. We want to compute their product $z = xy$. For simplicity, assume that $n = 2^k$ for some k.

(A) What is the naïve way of doing this? What is the time complexity?
(B) Now try to design an algorithm using divide and conquer.
Hint: $x = \langle x_n, x_{n-1}, \dots, x_1\rangle$ can be written as $x = x_L . 2^{n/2} + x_R$. For instance, $x = \langle 10101101\rangle$ can be written as $x_L . 2^4 + x_R$ with $x_L = \langle 1010\rangle$ and $x_R = \langle 1101\rangle$.
(C) Analyze the performance and find the close-form expression of its running time.
(D) How can we improve the performance?
Hint: can we reduce the number of multiplications?

<u>Answer</u>

(A) We can multiply x by each digit of y_i of y separately, which each takes $\theta(n)$, and in total is $\theta(n^2)$, and then add up all of them. Adding two binary numbers with $2n$ digits takes $\theta(n)$ time, which there are $n-1$ such additions. This naïve way takes $\theta(n^2)$ time.

(B) (C) Let $T(n)$ be the running time needed to multiply two binary numbers with n digits. From the hint, we can rewrite $x = x_L . 2^{n/2} + x_R$, $y = y_L . 2^{n/2} + y_R$, where x_L, x_R, y_L, y_R are binary numbers with $n/2$ digits. Thus we have $xy = x_L y_L . 2^n + (x_R y_L + x_L y_R) . 2^{\frac{n}{2}} + x_R y_R)$. As we can see in the new formula, we can calculate each of $x_L y_L$, $x_R y_L$, $x_L y_R$, $x_R y_R$ separately first, the running time of each of them is $T\left(\frac{n}{2}\right)$. Note that we have four of such multiplications. Finally, we have four additions which each cost $\theta(n)$. Thus we get that $T(n) = 4T\left(\frac{n}{2}\right) + cn$. The running time with this algorithm is $\mathrm{T(n)} = \theta(n^2)$.

(D) Let $A = x_L y_L, B = (x_R y_L + x_L y_R)\ , C = x_R y_R$. In the previous two sections, we needed four multiplications of two numbers of $\frac{n}{2}$ digits, in order to calculate A, B, C. There is another way to obtain A, B, C which only needs 3 multiplications. Note that $B = (x_L + x_R) * (y_L + y_R) - (A + C)$. In order to calculate B, we do 1 multiplication of $T\left(\frac{n}{2}\right)$, and 4 additions and subtraction together of cost only $\theta(n)$. Thus, in order to calculate A, B, C, we have 3 multiplications with total cost of $3T\left(\frac{n}{2}\right)$ and a constant number of addition and subtractions of cost $\theta(n)$. The total is $T(n) = 3T\left(\frac{n}{2}\right) + cn$, which implies $\mathrm{T(n)} = \theta(n^{\log_2 3})$.