Emirhan UZUN
171044019

# CSE 321 - INTRODUCTION TO ALGORITHMS
## Homework #3

1) I used Master's Theorem for some questions. This theorem is:

$$T(n) = a.T\left(\frac{n}{b}\right) + n^c$$

where $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
where $c = \log_b a$ then $T(n) = \Theta(n^c \log n)$
where $c > \log_b a$ then $T(n) = \Theta(f(n)) = \Theta(n^c)$

a) $T(n) = 27\,T(n/3) + n^2$
$a = 27$, $b = 3$, $c = 2$
$\log_b a = \log_3 27 = 3$ So $c < \log_b a$, then $T(n) = \Theta(n^3)$

b) $T(n) = 9T(n/4) + n$
$a = 9$, $b = 4$, $c = 1$
$\log_b a = \log_4 9 = 1{,}58$ So $c < \log_b a$, then $T(n) = \Theta(n^{\log_4 9}) = \Theta(n^{1.58})$

c) $T(n) = 2T(n/4) + n^{\frac{1}{2}}$
$a = 2$, $b = 4$, $c = 1/2$
$\log_b a = \log_4 2 = 1/2$ So $c = \log_b a$, then $T(n) = \Theta(\sqrt{n}.\log n)$

d) $T(n) = 2T(\sqrt{n}) + 1$
I will change the formula to simulate Master's Theorem formula.
Let's suppose $m = \log n$, $n = 2^m$
Now $T(2^m) = 2\,T(2^{m/2}) + 1$, let's say $S(m) = T(2^m)$ take log2
$S(m) = 2.S(m/2) + 1$ According to Master's Theorem;
$a = 2$, $b = 2$, $c = 0$ ; $\log_b a = \log_2 2 = 1$ So $c < \log_b a$ then $S(m) = \Theta(m)$
We said that $S(m) = T(2^m)$ so $T(2^m) = \Theta(m)$
$T(2^m) = \Theta(m)$ change with $2^m$ to $n$
$T(n) = \Theta(\log n)$

e) $T(n) = 2T(n-2) \quad T(0)=1 \ , T(1)=1$

$$T(n) = 2T(n-2)$$
$$= 2.(2T(n-4)) = 2^2.T(n-2*2)$$
$$= 2.2.(2.T(n-6)) = 2^3.T(n-2*3)$$
$$\vdots$$

$T(n) = 2^k T(n-2*k)$ , We know that $T(1)=1$

So $n-2*k=1 \Rightarrow k = \frac{n-1}{2}$

$$T(n) = 2^{\frac{n-1}{2}}.T(1) = 2^{\frac{n}{2}} \times \underbrace{2^{\frac{-1}{2}}}_{\text{constant}}$$ So $T(n) = \Theta(2^{\frac{n}{2}})$

f) $T(n) = 4T(n/2) + n \ , T(1)=1$

$I^{st}$ Way

Master's Theorem ; $a=4, b=2, c=1$

$\log_b a = \log_2 4 = 2$ So $c < \log_b a$ $\quad T(n) = \Theta(n^2)$

$II^{nd}$ Way

$$T(n) = 4.T(n/2) + n$$
$$= 4.(4.T(n/2^2) + n/2) + n$$
$$= 4.4.(4.T(n/2^3) + n/2^2) + n$$
$$\vdots$$

$T(n) = 4^k.T(n/2^k + n/2^{k-1}) + n$ , We know that $T(1)=1$

So $n/2^k + 2n/2^k = 1 \quad \frac{3n}{2^k} = 1 \quad 3n = 2^k \quad k = \log 3n$

$$T(n) = 4^{\log 3n}.T(1) + \log 3n$$
$$2^{\log 3n^2}.1 + \log 3n = \underbrace{9n^2}_{\substack{\text{big} \\ \text{growth rate}}} + \log 3n$$ So $\Theta(\max(n^2, \log 3n))$
$$= \Theta(n^2)$$

g) $T(n) = 2T(\sqrt[3]{n}) + 1$ , $T(3) = 1$

Change formula to simulate Master Theorem's formula

Let's suppose $m = \log_3 n$ , $n = 3^m$

$T(3^m) = 2T(3^{m/3}) + 1$ , let's say $S(m) = T(3^m)$ take log 3

$S(m) = T(3^m)$

$S(m) = 2 \cdot S(m/3) + 1$ According to Master's Theorem:

$a=2$ $b=3$ $c=0$

$\log_b a = \log_3 2 = 0{,}63$ So $c < \log_b a$ then $S(m) = \Theta(m^{0{,}63})$

We said that $S(m) = T(3^m)$ so $T(3^m) = \Theta(m^{0{,}63})$

$T(3^m) = \Theta(m^{0{,}63})$ Change $3^m$ with $n$

$T(n) = \Theta(\log_3 n)^{0{,}63} = \Theta(\log_3 n)^{\log_3 2}$

2) I wrote this relation

f(n)

if n<=1

print $\Big\}$ 1 constant

else

for i=1 to n $\to$ n times

f(n/2) $\to T(n/2)$

$$T(n) = \begin{cases} 1 & , n=1 \\ n \times T(n/2) & , n>1 \end{cases} \qquad T(1)=1$$

We assume n is a power of 2

So $n = 2^k$, $k = \log n$

$$T(2^k) = 2^k \times T(2^{k-1})$$

$$= 2^k \times (2^{k-1} \times T(2^{k-2}))$$

$$= 2^k \times 2^{k-1} \times (2^{k-2} \times T(2^{k-3}))$$

$$2^k \times 2^{k-1} \times \dots \times 2 = 2^{\frac{k.(k+1)}{2}}$$

So

$$T(2^k) = 2^{\frac{k(k+1)}{2}} \times T(1)$$

$$T(2^k) = (2^k)^{k/2} \times (2^k)^{1/2} \times T(1)$$

$$T(n) = n^{k/2} \times n^{1/2} \Rightarrow n^{\frac{k+1}{2}} \Rightarrow n^{\frac{\log n+1}{2}}$$

$$T(n) = \theta(n^{\frac{\log n+1}{2}}) = \theta(n^{\log n})$$

For line numbers, I wrote the function with Python format (in file)

Counter examples:

| $n$ | Line |
|---|---|
| $2 = 2^1$ | $2 = 2^1$ |
| $4 = 2^2$ | $8 = 2^2 . 2^1 = 2^3$ |
| $8 = 2^3$ | $64 = 2^3 . 2^2 . 2^1 = 2^6$ |
| $16 = 2^4$ | $1024 = 2^4 . 2^3 . 2^2 . 2^1 = 2^{10}$ |
| $\vdots$ | $\vdots$ |
| $n = 2^k$ | $= 2^k . 2^{k-1} . 2^{k-2} \dots 2^1 = 2^{\frac{k.(k+1)}{2}}$ |

For $n$, there is $2^{\frac{k.(k+1)}{2}}$ line.

$k = \log n$, so for $n$, $2^{\frac{\log n(\log n+1)}{2}}$ line printed.

3- That algorithm performs constant time with computation and then recursively calls itself three times. Each time on array whose size 2/3 of the original size

So $T(n) = 3T\left(\frac{2n}{3}\right) + 1$

$$T(n) = 1 + 3T\left(\frac{2n}{3}\right)$$
$$= 1 + 3 + 9T\left(\frac{4n}{9}\right)$$
$$\vdots$$
$$= 1 + 3 + 3^2 + \dots + 3^{\log_{3/2} n}$$

$$\sum_{k=0}^{n} a^k = \frac{a^{n+1} - 1}{a-1} \quad \text{for } a>1$$

So $\dfrac{3^{\log_{3/2} n+1} - 1}{3-1} \Rightarrow \Theta(3^{\log_{3/2} n})$

$$= \Theta\left(3^{(\log_3 n)/(\log_3 3/2)}\right) \quad \text{Change base}$$

$$= \Theta\left(n^{1/\log_3 3/2}\right) \qquad \log_3 3/2 = 0{,}36$$

$$= \Theta\left(n^{1/0{,}36}\right) = \Theta\left(n^{2{,}77}\right)$$

## 4- Average Case Analysis for Quick Sort:

- Assume that it is equally likely that the pivot element L[low] will be fixed in any position after rearrange

$T = T_1 + T_2 \rightarrow$ Random Variables

$T_1$: # of operations in rearrange

$T_2$: # of operations in recursive

$$A(n) = E[T] = \underbrace{E[T_1]}_{\text{high-low+2 operations (fixed)}} + \overbrace{E[T_2]}^{\text{depends on where the pivot has been placed}}$$

$$E[T_2] = \sum_{\underset{\text{position of pivot}}{x}} \underbrace{E[T_2 \mid X = x]}_{\text{cond. prob.}} \cdot \underbrace{P(X = x)}_{1/n}$$

$$A(n) = \underbrace{(n+1)}_{E[T_1]} + \sum_{i=1}^{n} E[T_2 \mid X = i] \cdot \underbrace{P(X=i)}_{=\frac{1}{n}}$$

$$= (n+1) + \sum_{i=1}^{n} \underbrace{[A(i-1) + A(n-i)]}\cdot \frac{1}{n}$$

$$\begin{array}{ll} & A(0) + A(n-1) \\ & A(1) + A(n-2) \\ & \vdots \\ + & A(n-1) + A(0) \\ \hline & 2(A(0) + A(n-1)) \end{array}$$

$$\begin{array}{rl} n \cdot A(n) = & n \cdot (n+1) + 2[A(0) + \dots + A(n-1)] \\ -\ (n-1) A(n-1) = & n \cdot (n-1) + 2 \cdot [A(0) + \dots + A(n-2)] \\ \hline n A(n) - (n-1) A(n-1) = & 2n + 2A(n-1) \end{array}$$

$\frac{1}{n \cdot (n+1)}$ multiply by $\Rightarrow \frac{A(n)}{n+1} - \frac{A(n-1)}{n} = \frac{2}{n+1}$ $\left(t(n) = \frac{A(n)}{n+1}\right)$

$$t(n) = t(n-1) + \frac{2}{n+1}$$

$A(1) = 0$ // initial

$t(0) = \frac{A(0)}{1}$

$$t(n) = \sum_{i=2}^{n} \frac{2}{i+1} = 2\underbrace{H(n+1)}_{\text{Harmonic}} - 3$$

$$A(n) = t(n) \cdot (n+1) = 2 \cdot (n+1) \overbrace{H(n+1)}^{\ln(n+1)} - 3 \cdot (n+1) \in \Theta(n \log n)$$

- Average Case Analysis for Insertion Sort:

Let $T_i$ = # of basic operator at step $i$ ; $1 \le i \le n-1$

$$T = T_1 + T_2 + \dots + T_{n-1} = \sum_{i=1}^{n-1} T_i$$

$$A(n) = E[T] = E\left[\sum_{i=1}^{n-1} T_i\right] = E[T_1] + E[T_2] + \dots + E[T_{n-1}]$$

$$E[T_i] = \sum_{j=1}^{i} j \cdot P(T_i = j)$$

1 comparison will occur if $x = L[i] > L[i-1]$

2 " " " " $L[i-2] < x < L[i-1]$

⋮

$i$ " " " " $x < L[i]$

There are $(i+1)$ intervals that $x$ can fall in

$$P(T_i = j) = \begin{cases} \frac{1}{i+1} & \text{if } 1 \le j \le i-1 \\ \frac{2}{i+1} & \text{if } j = i \end{cases}$$

$$E[T_i] = \left[\sum_{j=1}^{i-1}\left(j \cdot \frac{1}{i+1}\right)\right] + i \cdot \frac{2}{i+1} = \frac{i(i-1)}{2(i+1)} + \frac{2}{(i+1)} = \frac{i^2 - i + 4i}{2(i+1)} = \frac{i(i+3)}{2(i+1)}$$

$$A(n) = E[T] = \sum_{i=1}^{n-1} E[T_i] = \sum_{i=1}^{n+1}\left(\frac{i}{2} + 1 - \frac{1}{i+1}\right) = \frac{n.(n-1)}{4} + n - 1 - \underbrace{\sum_{i=1}^{n-1} \frac{1}{i+1}}_{\text{Harmonic}}$$

$$= \frac{n.(n-1)}{4} + n - H(n)$$

$$A(n) \in \Theta(n^2)$$

Now, according to analyzes
QuickSort is $\Theta(n \log n)$
InsertionSort is $\Theta(n^2)$

There are 10 swap count comparisons in my report file.
As a result, I decide the quick sort is faster than insertion sort.

5 - a) $T(n) = 5T(n/3) + n^2$

$a=5$, $b=3$, $c=2$
$\log_b a = \log_3 5 = 1{,}46$
So $c > \log_b a$ then $T(n) = O(f(n)) = O(n^2)$

b) $T(n) = 2T(n/2) + n^2$
$a=2$, $b=2$, $c=2$
$\log_b a = \log_2 2 = 1$
So $c > \log_b a$ then $T(n) = O\, f(n)) = O(n^2)$

c) $T(n) = T(n-1) + n$
$= [T(n-2) + n - 1] + n$
$= T(n-3) + (n-2) + (n-1) + n$

$T(n) = T(n-k) + (n-(k-1)) + (n-(k-2)) + \dots + (n-1) + n$
$n-k=0 \quad k=n$

$T(n) = T(n-n) + (n-n+1) + (n-n+2) + \dots + (n-1) + n$
$T(n) = 0 + 1 + 2 + 3 + \dots + n$
$T(n) = \dfrac{n.(n+1)}{2} = \dfrac{n^2+n}{2}$ So $T(n) = O(n^2)$

→ Now, all algorithms have same growth order. But I would choose algorithm C. Because it needs less space and its component is linear time.