CSE 321 HW3 SOLUTIONS

171044098
Akif Kartal

1) To solve these recurrence relations we can use subsitution method step by step but for these dividing function there is a theorem called masters theorem which is much better than subsitition method.

- Master Theorem for Dividing Recurrence Relation Functions

$$T(n) = aT(n/b) + f(n)$$

$$a \geq 1 \;,\; b > 1 \text{ and } f(n) = \theta(n^k \log^p n)$$

Cases:

1) if $\log_b^a > k$ then $\theta(n^{\log_b^a})$

2) if $\log_b^a = k$ then

   if $p > -1$ $\theta(n^k \log^{p+1} n)$

   if $p = -1$ $\theta(n^k \log\log n)$

   if $p < -1$ $\theta(n^k)$

3) if $\log_b^a < k$ if $p \geq 0$ $\theta(n^k \log^p n)$

   if $p < 0$ $O(n^k)$

Let's start to solve by using this theorem.

a) $T(n) = 27T(n/3) + n^2$

$k = 2$ $\quad a = 27, \; b = 3$ $\quad \boxed{f(n) = \Theta(n^2) = \Theta(n^{2} \log^0 n)}$

$\log_b^a = \log_3^{27} = 3$

$\log_b^a > k \Rightarrow 3 > 2$ therefore,

$\Rightarrow \Theta(n^{\log_b^a}) = \Theta(n^3)$ (case 1)

---

b) $T(n) = 9T(n/4) + n$

$f(n) = \Theta(n) = \Theta(n^1 \log^0 n) \Rightarrow k = 1$

$a = 9, \; b = 4, \; \log_b^a = \boxed{\log_4^9 = 1,58}$

$\log_b^a > k \Rightarrow 1,58 > 1$ therefore,

$\Rightarrow \Theta(n^{\log_b^a}) = \Theta(n^{1,58})$ (Case 1)

---

c) $T(n) = 2T(n/4) + \sqrt{n}$

$f(n) = \Theta(\sqrt{n}) = \Theta(n^{1/2} \log^0 n) = \Theta(n^{1/2}) \Rightarrow k = 1/2 = 0,5$

$a = 2, \; b = 4 \Rightarrow \log_b^a = \log_4^2 = 0,5$

$\log_b^a = k \Rightarrow 0,5 = 0,5$ therefore,

$\Rightarrow \Theta(n^k \log^{p+1} n) \Rightarrow \Theta(n^{1/2} \log^1 n) \Rightarrow \Theta(n^{1/2} \log n)$ (case 2)

$\Rightarrow \boxed{\Theta(\sqrt{n} \log n)}$

d) $T(n) = 2T(\sqrt{n}) + 1$

To solve this recurrence, I will use substitution method.

$\Rightarrow T(n) = 2T(n^{1/2}) + 1$

$\Rightarrow$ Substitude $T(n^{1/2})$ as $T(n^{1/2}) = 2T((n^{1/2})^{1/2}) + 1$

$\Rightarrow T(n) = 2[2T((n^{1/2})^{1/2}) + 1] + 1$

$\Rightarrow T(n) = 4T(n^{1/4}) + 3$

$\Rightarrow$ Substitude $T(n^{1/4})$ as $T(n^{1/4}) = 2T((n^{1/4})^{1/2}) + 1$

$\Rightarrow T(n) = 4[2T(n^{1/8}) + 1] + 3$

$\Rightarrow T(n) = 8T(n^{1/8}) + 7$

if we contiune k times

$\Rightarrow \boxed{T(n) = 2^k T(n^{1/2^k}) + (2^k - 1)}$

$\Rightarrow$ Assume as in 2^{nd} question n is a power of 2.

$\Rightarrow n = 2^m$

$\Rightarrow T(2^m) = 2^k T(2^{m/2^k}) + (2^k - 1)$

$\Rightarrow$ Assume $T(2^{m/2^k}) = T(2)$

$\frac{m}{2^k} = 1 \Rightarrow m = 2^k \Rightarrow \boxed{k = \log_2 m}$

Since, $n = 2^m$

$m = \log_2 n$

$\boxed{k = \log_2(\log_2 n)}$

$\rightarrow 2^k - 1 = 2^{\log_2(\log_2 n)} - 1$

(By using logarithm property)

$= \log_2 n^{\log_2 2} = \log_2 n - 1$

$\Rightarrow \Theta(\log_2 n - 1) = \boxed{\Theta(\log_2 n)}$ 3

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

Again, let's use substitution method for this problem with fewer steps.

$\Rightarrow T(n) = 2T(n-2) \longrightarrow 1$

$\Rightarrow T(n-2) = 2T(n-4)$

$\Rightarrow T(n) = 2[2T(n-4)]$

$\Rightarrow T(n) = 4T(n-4) \longrightarrow 2$

$T(n-4) = 2T(n-6)$

$\Rightarrow T(n) = 4[2T(n-6)]$

$T(n) = 8T(n-6) \longrightarrow 3$

$\vdots$

$T(n) = 2^k T(n-2k)$

Assume $= n - 2k = 0$

$n = 2k$

$k = n/2$

$T(n) = 2^{n/2} T(0)$

$T(n) = 2^{n/2} \cdot 1 \quad (T(0) = 1)$

$T(n) = 2^{n/2}$

$\Rightarrow \boxed{\Theta(2^{n/2})}$

---

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

Since, we have a initial condition let's use again substition method with this initial condition.

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

$\Rightarrow T(n/2) = 4T((n/2)/2) + n/2$

$\Rightarrow T(n) = 4[4T(n/4) + n/2] + n$

$\Rightarrow T(n) = 16T(n/4) + 3n \rightarrow 2$

$\Rightarrow T(n/4) = 4T(n/8) + n/4$

$\Rightarrow T(n) = 16[4T(n/8) + n/4] + 3n$

$\Rightarrow T(n) = 64T(n/8) + 7n \rightarrow 3$

$\vdots$

$\Rightarrow T(n) = 4^k T(n/2^k) + (2^k - 1)$

Since, $T(1) = 1$

Assume, $\frac{n}{2^k} = 1$

$\Rightarrow n = 2^k \quad \Rightarrow k = \log n$

$T(n) = 4^{\log n} T(1) + (2^{\log n} - 1)$

$T(n) = 4^{\log n} + 2^{\log n} - 1$

$\Rightarrow \Theta(4^{\log n} + 2^{\log n} - 1)$

$\Rightarrow \Theta(4^{\log n})$ (ignore constant and lower degree functions)

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

Apply substition method:

$\Rightarrow T(n) = 2T(n^{1/3}) + 1 \rightarrow$ 1

$\Rightarrow T(n^{1/3}) = 2T(n^{1/9}) + 1$

$\Rightarrow T(n) = 2[2T(n^{1/9}) + 1] + 1$

$\Rightarrow T(n) = 4T(n^{1/9}) + 3 \rightarrow$ 2

$\Rightarrow T(n^{1/9}) = 2T(n^{1/27}) + 1$

$\Rightarrow T(n) = 4[2T(n^{1/27}) + 1] + 3$

$\Rightarrow T(n) = 8T(n^{1/27}) + 7 \longrightarrow$ 3

$\vdots$

$\Rightarrow T(n) = 2^k T(n^{1/3^k}) + (2^k - 1)$

Assume, $n = 3^m$

$T(3^m) = 2^k T(3^{m/3^k}) + (2^k - 1)$

Assume, $T(3^{m/3^k}) = T(3) = 1$

$\frac{m}{3^k} = 1 \Rightarrow m = 3^k \Rightarrow k = \log_3 m$

Since $n = 3^m$

$m = \log_3 n$

$\boxed{k = \log_3(\log_3 n)}$

$2^k - 1 \Rightarrow 2^{\log_3(\log_3 n)} - 1$

$\Rightarrow \log_3 n^{\log_3 2} = \log_3 n^{0,63} - 1$ $(\log_3 2 = 0,63)$

$\Rightarrow \Theta(\log_3 n^{0,63} - 1) = \boxed{\Theta(\log_3 n^{0,63})}$

6

2)

```
function f(n)
    if n<=1:
        print_line("**")
    else:
        for i=1 to n
            f(n/2)
        end for
```

If we count how many lines print;

n	number of print
0	1
1	1
2	2 = T(1) * 2
3	3 = T(1) * 3
4	8 = T(2) * 4
5	10 = T(2) * 5
6	18 = T(3) * 6
7	21 = T(3) * 7
⋮	⋮
n	T(n/2) * n

Recurrence relation = $T(n) = n * T(n/2)$ apply backward subst.

$\Rightarrow T(n) = n * T(n/2) \rightarrow (1)$

$* \; T(n/2) = \frac{n}{2} T(n/4)$

$\Rightarrow T(n) = \frac{n^2}{2} T(n/4) \rightarrow (2)$

$* \; T(n/4) = \frac{n}{4} T(n/8)$

$\Rightarrow T(n) = \frac{n^3}{8} T(n/8) \rightarrow (3)$

$* \; T(n/8) = \frac{n}{8} T(n/16)$

$\Rightarrow T(n) = \frac{n^4}{64} T(n/16) \rightarrow (4)$

⋮

$\Rightarrow \boxed{T(n) = \frac{n^k}{2^{\log^2 k}} T(n/2^k)}$ → Recurrence relation

Assume $n = 2^k$ (Already given in question)
then $k = \log n$, Thus

$T(n) = n^{\log n} \cdot 2^{-(\log^2 \log n)} \cdot T(1)$ $\quad T(1) = 1$

$\boxed{T(n) = n^{\log n} \cdot 2^{-(\log^2(\log n))}}$

Result $= \Theta(n^{\log n} \cdot 2^{-(\log^2 \log n)})$

3)

Algorithm Function_f $(A[0 \dots n-1])$

$1 \leftarrow$ if $n = 2$ and $A[0] > A[1]$, then swap$(A[0], A[1])$

if $n > 2$ then {

$T(2n/3) \leftarrow$ Function_f$(A[0 \dots ceil(2n/3)]$

$T(2n/3) \leftarrow$ Function-f$(A[floor(n/3) \dots n])$

$T(2n/3) \leftarrow$ Function-f$(A[0 \dots ceil(2n/3)])$

}

$T(n) = 3T(2n/3) + 1$

if we use master theorem to solve this

$\Rightarrow f(n) = \Theta(1) \Rightarrow \Theta(n^0 \log^0 n) \Rightarrow k = 0$

$\Rightarrow a, 3, b = 3/2 \Rightarrow \log_b a \Rightarrow \boxed{\log_{1.5} 3 = 2,70}$

$\log_b a > k \Rightarrow 2,70 > 0$ therefore,

$\Rightarrow \Theta(n^{\log_b a}) \Rightarrow \boxed{\Theta(n^{2,70}) \quad (\text{case } 1)}$

4) To solve this question I will start with theoretical average case of algorithms. Analysis mostly taken from lecture.

Average case of Insertion Sort by Using number of swaps (comp.)

For each of the T_i's are random variables.

if we calculate $E[T_i]$ we will get;

$$E[T_i] = \sum_{j=1}^{i} j \cdot P(T_i = j)$$ → Probality that there are j comparisons in the i^{th} step

From this probility, There are 2 case i comparison

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

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

$$\Rightarrow \frac{i(i-1)}{2(i+1)} + \frac{2i}{i+1} = \frac{i^2 - i + 4i}{2(i+1)} = \cdots = \sum_{i=1}^{n-1} \frac{i}{2} + 1 - \frac{1}{i+1}$$

$$= \underbrace{\frac{n(n-1)}{4}} + n-1 - \underbrace{\sum_{i=1}^{n-1} \frac{1}{i+1}}_{\text{Harmonic series}}$$

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

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

Swaps in Insertion Sort (Theoritical)

1) A sorted list has no swaps.
2) A reverse sorted list of size n has $\frac{(n-1)n}{2}$ swap.
3) In the average, all lists of size n have $\frac{(n-1).n}{4}$ swap.

Avarage case of Quick Sort by using number of swaps (comparison)

Let say T_i's are random variables then,

$T = T_1 + T_2$

T_1 = Partition

T_2 = recursive calls therefore,

$A(n) = E[T] = E[T_1] + E[T_2]$

If we solve this we will get;

$A(n) = t(n).(n+1) = 2(n+1).H(n) - 3(n+1)$

$\Rightarrow \boxed{\in \Theta(n\log n)}$

Number of swaps in implementation (Python)

I counted the number of swap operations in implementation of algorithms with size of 20 same 1 array and results are following;

Quick sort number of swap = 61

Insertion Sort number of swap = 119

Analysis and comment

if we compare the theoritical average cases and swap counts of course quick sort algorithm is much better. But Also by looking at this results, we also observe that insertion sort swap count much better than it's theoretical performance.

5) To solve these questions first, I will write simple pseucodes.

a)

A simple psuecode can be like this.

Algorithm test(n)
    if(n>0)
        s1 = test(n/3) → T(n/3)
        s2 = test(n/3) → T(n/3)
        s3 = test(n/3) → T(n/3)
        s4 = test(n/4) → T(n/3)
        s5 = test(n/5) → T(n/3)
        combine(s1, s2, s3, s4, s5) → n^2 (Quadratic time)

$T(n) = 5T(n/3) + n^2$ by using masters theorem,

$f(n) = \Theta(n^2) \Rightarrow \Theta(n^2 \log^0 n) \Rightarrow \boxed{k=2}$

$a = 5$, $b = 3 \Rightarrow \log_b^a \Rightarrow \log_3^5 = 1,46$

$\log_b^a < k \Rightarrow 1,46 < 2$ therefore,

$\Rightarrow \Theta(n^2) \Rightarrow \boxed{O(n^2)}$ (Case 3)

b) Algorithm test(n)
    if(n>0)
        s1 = test(n/2) → T(n/2)
        s2 = test(n/2) → T(n/2)
        combine(s1, s2) → n^2

$T(n) = 2T(n/2) + n^2$ by using Masters theorem,

$f(n) = \Theta(n^2) \Rightarrow \Theta(n^2 \log^0 n) \Rightarrow k = 2$

$a = 2,\ b = 2 \Rightarrow \log_b^a \Rightarrow \log_2^2 = 1$

$\log_b^a < k \Rightarrow 1 < 2$ therefore,

$\Theta(n^2) \Rightarrow \boxed{O(n^2)}$ (case 3)

---

c)

Algorithm test(n)

if(n>0)

S1 = test(n-1) $\rightarrow T(n-1)$

combine(S1) $\rightarrow n$

1

$T(n) = T(n-1) + n$ by using Subsitition method,

$\Rightarrow T(n-1) = T(n-2) + n - 1$

$\Rightarrow T(n) = T(n-2) + 2n - 1 \rightarrow$ 2

$\Rightarrow T(n-2) = T(n-3) + n - 2$

$\Rightarrow T(n) = T(n-3) + 3n - 3 \rightarrow$ 3

⋮

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

Assume $n - k = 0$

$n = k$

$T(n) = T(0) + 1 + 2 + 3 + \dots + (n-1) + n$

$T(n) = 1 + \frac{n.(n+1)}{2}$ (Assume $T(0)$ is 1)

$\rightarrow T(n) = \frac{n^2 + n}{2} + 1$ Thus,

$\boxed{= O(n^2)}$

Scanned with CamScanner

Lastly, if we compare these three algorithms,

$$a = b = c \Rightarrow O(n^2) = O(n^2) = O(n^2)$$

Since they are in same asymptotical order, I would choose any of them.