HW - 2

Can Duyar
171044075

I applied them for all solutions of Question-1.

Master Theorem

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

$f(n) = \Theta(n^k \log^p n)$

$a \geq 1$
$b > 1$
and
$f(n)$ is asymptotically +

Case 1: if $\log_b^a > k$ then $\Theta(n^{\log_b^a})$

Case 2: if $\log_b^a = k$

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

ii) if $p = -1 \to \Theta(n^k \log\log n)$

iii) if $p < -1 \to \Theta(n^k)$

Case 3:

if $\log_b^a < k$

if $p \geq 0 \to \Theta(n^k \log^p n)$

if $p < 0 \to \Theta(n^k)$

Q1)

a) $T(n) = 16T(\frac{n}{4}) + n!$ (a, b, $f(n)$)

→ we need to compare values of $\log_b^a$ and k to define the case!

$\log_b^a = \log_4^{16} = \log_4^{4^2} = 2$

$\log_b^a = 2$

$f(n) = n!$ → we can write it as n^n

$f(n) = n^n \hookrightarrow n^k \log^p n$ ($p \to 0$)

$p = 0$
$k = n$
in this case

$k = n$

$\log_b^a < k$ ($\log_b^a = 2$, $k = n$)

because k is equal to n then we can say that this is in 3. case. If we check the value of p then we see that $p \geq 0$ because $p = 0$ so we can express the complexity as $\Theta(n^k \log^p n)$ for $k = n$, $p = 0$

$T(n) = \Theta(n^n) = \boxed{\Theta(n!)}$ → Result

b) $T(n) = \sqrt{2}\, T(n/4) + \log n$ $\quad a = \sqrt{2},\ b = 4,\ f(n) = \log n$

$\log_b^a = \log_4^{\sqrt{2}} = \log_{2^2}^{2^{\frac{1}{2}}} = \frac{1}{4}$

$f(n) = \log n \rightarrow n^k \log^p n$ $\quad \boxed{\begin{matrix} k=0 \\ p=1 \end{matrix}}$

I need to compare values of k and $\log_b^a$ to define the case. I explained these cases with details in my first page.

$\rightarrow$ In this case, $\log_b^a = \frac{1}{4}$ and $k=0$ then $k < \log_b^a$ so it's the first case that I explained in my first page. In this case, complexity will be $\Theta(n^{\log_b^a})$ format.

$$\Longrightarrow \Theta(n^{\log_b^a}) \xrightarrow{\frac{1}{4}} \boxed{\Theta(\sqrt[4]{n})} \rightarrow \text{Result.}$$

---

c) $T(n) = 8T\left(\frac{n}{2}\right) + 4n^3$ $\quad a = 8,\ b = 2,\ f(n) = 4n^3$

$\log_b^a = \log_2^{2^3} = 3$

$f(n) = 4n^3 \rightarrow n^k \log^p n$

$\underline{k=3} \quad \underline{p=0}$

If we compare the values of k and $\log_b^a$ then we define the case.

$k=3$ and $\log_b^a = 3$ also.

if $k = \log_b^a$ then it's second case.

Now, we need to check value of p.

$p=0$ and $p > -1$ so it's the first case of case-2 (As I explained in the first page). The complexity will be $\Theta(n^k \log^{p+1} n)$ format. for $k=3$ and $p=0$

Therefore,

$$\Theta(n^k \log^{p+1} n) = \Theta(n^3 \log^{0+1} n)$$

$$= \boxed{\Theta(n^3 \log n)} \rightarrow \text{Result.}$$

d) $T(n) = 64T(n/8) - n^2\log n \rightarrow f(n)$

→ We can not solve it by using Master theorem because $f(n)$ is not positive. We need asymptotically positive $f(n)$ to solve it by using Master Theorem. That's why it cannot be solved.

---

e) $T(n) = 3T(n/3) + \sqrt{n} \rightarrow f(n) = n^k \log^p n$

$\log_b^a = \log_3^3 = 1$

$f(n) = \sqrt{n} = n^{\frac{1}{2}} \longrightarrow n^k \log^p n$

$k = \frac{1}{2}$ and $p = 0$

→ We need to compare values of $\log_b^a$ and k to define the case!

$\log_b^a = 1$, $k = \frac{1}{2}$ > so $k < \log_b^a$ and we need to apply first case (As I explained in my first page) The complexity will be $\Theta(n^{\log_b^a})$ in the first case.

$T(n) = \Theta(n^{\log_b^a}) \longrightarrow \boxed{\Theta(n)}$ → Result.

f) $T(n) = 2^n T(n/2) - n^n$ → $f(n)$

→ We cannot use Master Theorem to solve it! Because value of "a" is not constant and f(n) is not asymptotically positive. According to definition of Master Theorem,

$a \geq 1$

$b > 1$

and f(n) is asymptotically positive!

That's why it cannot be solved by using Master Theorem!

---

g) $T(n) = 3T\left(\frac{n}{3}\right) + \frac{n}{\log n}$ → $f(n)$

$\log_b^a = \log_3^3 = 1$

↳ f(n) is polynomially bigger or smaller than n, and is not equal to $\Theta(n^k \log^p n)$ for any $p \geq 0$. That's why we cannot apply Master theorem for solving this problem.

Q2)

a) $T(n) = 9T(\frac{n}{3}) + O(n^2)$

, dividing them into nine subproblems

one-third of the size

Combining the solutions in Quadratic time

↳ we can apply master theorem for solving it!

$T(n) = aT(\frac{n}{b}) + f(n)$ format will be like this and we know that $f(n)$ is in $O(n^k \log^p n)$. I explained it in my first page's master theorem part.

$T(n) = 9T(\frac{n}{3}) + O(n^2)$

↳ $O(n^k \log^p n)$, $k \to 2$, $p \to 0$

$a = 9$

$b = 3$

$k = 2$

$p = 0$

we need to compare values of k and $\log_b^a$ to define the case (As I explained in my first page)

$k = 2$

$\log_b^a = \log_3^{3^2} = 2$

→ $k = \log_b^a$ so it's in second case. Now, we need to check the value of p. $p = 0$ and $p > -1$. So we need to consider the first case of the second case (if $p > -1 \rightarrow O(n^k \log^{p+1} n)$).

$T(n) = O(n^k \log^{p+1} n)$ for $k = 2$ and $p = 0$.

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

(6)

b) $T(n) = 8T(\frac{n}{2}) + O(n^3)$

8 subproblems

half the size

combining the solutions in $O(n^3)$

→ We can apply master theorem for solving it!

$T(n) = aT(\frac{n}{b}) + f(n)$ format will be like this and we know that $f(n)$ is in $O(n^k \log^p n)$. I explained it in my first page's master theorem part.

$T(n) = \overset{a}{8}T(\frac{n}{\underset{b}{2}}) + \overset{f(n)}{O(n^3)} \rightarrow O(n^{\overset{3}{k}} \log^{\overset{0}{p}} n)$

$a = 8$
$b = 2$

$\log_b^a = \log_2 2^3 = 3$

$\boxed{\begin{matrix} k = 3 \\ p = 0 \end{matrix}}$

→ We need to compare values of k and $\log_b^a$ to define the case (As I explained in my first page)

$k = 3$

$\log_b^a = \log_2 2^3 = 3$

$k = \log_b^a$ so it's in second case. Now, we need to check the value of p. $p = 0$ and $p > -1$. So we need to consider the first case of the second case

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

$T(n) = O(n^{\overset{3}{k}} \log^{\overset{0}{p}+1} n)$ for $k = 3$ and $p = 0$

$\boxed{T(n) = O(n^3 \log n)}$

c) $T(n) = \underbrace{2}_{\text{two subproblems}}T(\underbrace{n/4}_{\text{Quarter of the size}}) + \underbrace{O(\sqrt{n})}_{\text{Combining the solutions in } O(\sqrt{n})}$

→ we can apply master theorem for solving it!

$T(n) = aT(n/b) + f(n)$ format will be like this and we know that $f(n)$ is in $O(n^k \log^p n)$. I explained it in my first page's master theorem part.

$T(n) = \underset{a}{2}T(n/\underset{b}{4}) + \underset{f(n)}{O(\sqrt{n})}$

↳ $O(n^k \log^p n)$

$a=2$
$b=4$

$\log_b^a = \log_{2^2}^2 = \boxed{\frac{1}{2}}$

$\log_b^a = \frac{1}{2}$

$\sqrt{n} = n^{\frac{1}{2}}$
so we can say that $\boxed{k = \frac{1}{2}}$ and $\boxed{p = 0}$

→ we need to compare values of k and $\log_b^a$ to define the case (As I explained in my first page)

$k = \frac{1}{2}$

$\log_b^a = \log_4^2 = \frac{1}{2}$

$k = \log_b^a$ so it's in second case. Now, we need to check the value of p. $p = 0$ and $p > -1$. so we need to consider the first case of the second case.

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

$T(n) = O(n^k \log^{p+1} n)$ for $k = \frac{1}{2}$ and $p = 0$

$$\boxed{T(n) = O(\sqrt{n} \log n)}$$

→ As a Result, we found the the running times of these algorithms as following.

a) $T(n) = O(n^2 \log n)$ → for Algorithm X

b) $T(n) = O(n^3 \log n)$ → for Algorithm Y

c) $T(n) = O(\sqrt{n} \log n)$ → for Algorithm Z.

$O(n^3 \log n) > O(n^2 \log n) > O(\sqrt{n} \log n)$

$T(n) = O(\sqrt{n} \log n)$

* I would choose the Algorithm-Z. Because it has the lowest order exponent so it will be faster than others.

3) a)

i)

[6,2,8,4,7,3,9,5]

[6,2,8,4] [7,3,9,5]

[6,2] [8,4] [7,3] [9,5]

[2,6] [4,8] [3,7] [5,9]

Every pair of 2 compared at least once so we have maximum number of comparison in this case. It's the worst case of mergesort algorithm.

[2,4,6,8] [3,5,7,9]

[2,3,4,5,6,7,8,9] → Indexes are totally changed after these steps

---

ii) [1,2,3,4,5,6,7,8]

[1,2,3,4] [5,6,7,8]

[1,2] [3,4] [5,6] [7,8]

[1,2] [3,4] [5,6] [7,8]

They are already sorted so we have minimum number of comparison in this example. It's the best case of the merge sort algorithm.

[1,2,3,4] [5,6,7,8]

[1,2,3,4,5,6,7,8] → Indexes didn't change after these steps

II)

i)

pivot

1 3 5 7 9 11 13 15

If the array is already sorted then worst case occurs in QuickSort.

there is no element which is smaller than pivot at the right side of it.

3 5 7 9 11 13 15

still there is no element which is smaller than pivot at the right side of it!

There is no element smaller than pivot at the right side

5 7 9 11 13 15

7 9 11 13 15

9 11 13 15

$\Longrightarrow$ It becomes equal to linear search because the array is already sorted. In this exemple, we have <u>maximum number of swap operations</u>!

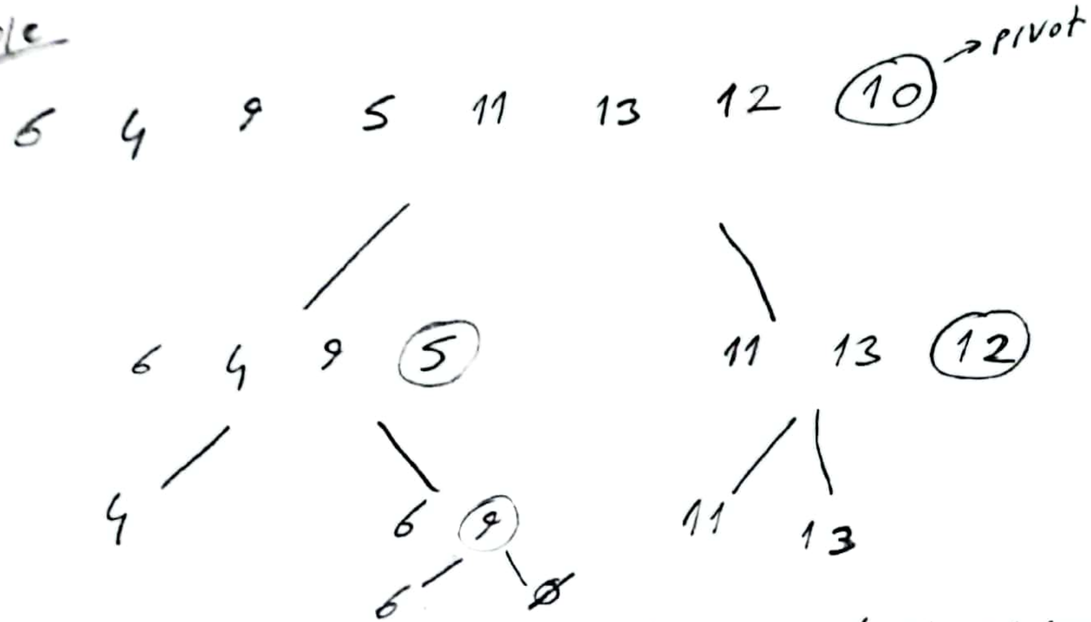
11 13 15

13 15

15

ii) Best case occurs when the pivot element divides the list into two equal halves by coming exactly in the middle point in this case time complexity will be $O(n\log n)$

Example



It's the best case of QuickSort algorithm and it requires the minimum number of swap operations. It would be better, if we have odd number of elements (to divide them as equal halves) but we have 8 elements in this case.

Q4) This is a binary search algorithm that searches only for "0". Target value was selected as "0" in this algorithm.

→ we know that $mid = (left + right)/2$ if $A[mid] > 0$ then it continues with elements between left and mid. If $A[mid] > 0$ then it continues with elements between mid and right as recursive. This algorithm divides the problem in half each time.

1-step ⟹ $T(n) = T(n/2) + 1$

2-step ⟹ $T(n/2) = T(n/4) + 1 + 1 \cdots$

3-step ⟹ $T(n/4) = T(n/8) + \underbrace{1+1+1}\ldots$ → equal to number of steps!

⋮

p-step ⟹ $T\left(\frac{n}{2^{p-1}}\right) = T\left(\frac{n}{2^p}\right) + p$ → last step

→ We can use Master Theorem to find the complexity of the recurrence relation. Format of the master theorem will be,

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

$T(n) = 1T(n/2) + 1$ → $f(n)$ → $\Theta(n^k \log^p n)$
↓ a, ↳ b

we need to compare values of k and $\log_b^a$ to define the case!
(As I explained in my first page)

$T(n) = 1T(n/2) + 1$ ↳ $\Theta(\underbrace{n^k \log^p n}_{1})$
↓ a ↓ b

$n^k \log^p n = 1$

$\boxed{\begin{matrix} k=0 \\ p=0 \end{matrix}}$

$\log_b^a = \boxed{\log_2^1}$

$\log_b^a > k$ in this case and it's in the second case! Now, we need to check the value of "p" to define the subcase of the second case (I explained these subcases in my first page)

$p=0$ and $p > -1$
so it's the first case of the second case! In this case, time complexity will be

$T(n) = \Theta(n^k \log^{p+1} n)$ for $k=0$ and $p=0$

$\boxed{T(n) = \Theta(\log n)}$ → Result.

Q5)

pseudocode

a)

```
box-gift-pairs(boxes, gifts, l, h)
   if l < h
     pivo = call divide_to_subarrays(boxes, l, h, gifts[h])
     call divide-to-subarrays(gifts, l, h, boxes[pivo])
     updated-low = pivo +1
     call box-gift-pairs(boxes, gifts, updated_low, h)
     updated-high = pivo-1
     call box-gift-pairs(boxes, gifts, l, updated_high)
   endif
end

divide-to-subarrays(box-or-gift, l, h, pivo)
  g ← l
  t ← l

  while t < h do
    if box-or-gifts[t] == pivo)
      box-or-gift[t], box-or-gift[h] ← box-or-gift[h], box-or-gift[t]
      t ← t-1
    else if box-or-gift[t] < pivo
      box-or-gift[g], box-or-gift[t] ← box-or-gift[t], box-or-gift[g]
      g ← g+1
    endif
    t ← t+1
  box-or-gift[g], box-or-gift[h] ← box-or-gift[h], box-or-gift[g]
  end while
  return g
end
```

b) → This problem is based on Quicksort algorithm. It divides the problem to 2-subarrays between updated values of low — high and pivot point. We use different pivot values from the array. Therefore we find the proper boxes for each of the gifts.

Time complexity of Quicksort problem

$$T(n) = \underbrace{2}_{a}T(n/\underbrace{2}_{b}) + n$$ → The algorithm solves the problem by dividing them into 2 subproblems that have half the size and then combining the solutions in linear time.

We can use master theorem to solve this reccurence.

$T(n) = aT(n/b) + f(n)$ → format of master theorem.

$T(n) = \underbrace{2}_{a}T(n/\underbrace{2}_{b}) + n \rightsquigarrow \Theta(n^k \log^p n)$

We need to compare values of $\log_b^a$ and k to define the case!
(As I explained in my first page)

$\boxed{\log_b^a = \log_2^2 = 1}$

$n \rightsquigarrow \Theta(n^{\overset{1}{k}} \log^{\overset{0}{p}} n)$

$\boxed{\begin{matrix} k=1 \\ p=0 \end{matrix}}$

We found that $\log_b^a = 1$ and $k = 1$
$\log_b^a = k$ so it's in the second case! so we need to check the value of "p" to define the proper subcase of the second case.

$p = 0$ and $p > -1$ so it's the first case of the second case (I explained them in my first page to use them for other Questions)

If $p > -1$ then $\Theta(n^k \log^{p+1} n)$

the complexity will be $\Theta(n^{\overset{1}{k}} \log^{\overset{0}{p}+1} n)$ for $k=1$ and $p=0$

$T(n) = \Theta(n^k \log^{p+1} n) = \Theta(n \log n)$

$\boxed{T(n) = \Theta(n \log n)}$ → Result.

Scanned with CamScanner