Same mistake of
Vineeth Varma

CSE 5311 Quiz 1 (Fall 2022) 50

1:10-1:50pm, 9/22 (Thursday)

Name: Sai Rohit Kalyan Gandham.

1. "The best worst-case running time that we've seen for a sorting algorithm is O(nlgn)". Is this statement correct? Why? [5 points]

Sol) for Heap, and merge Sort the Best, Worst-Case is $O_{Big O}$ $O(n\lg n)$, Hence it's Correct of Heap and merge But for Insertion and Quick Sort the Statement is false.

2. "The time for a randomized quicksort to sort n numbers is O(nlgn)." Is this statement correct? Why? [5 points]

5

2sol) No it Shouldn't always be $(n\lg n)$, if the random pivot is maximum Element then the Sorting algorithm gives $O(n^2)$

3. Use the master method to give the tight asymptotic bound for the recurrence (you are not required to verify it): $T(n) = 2T\left(\frac{n}{4}\right) + n$.[15 points] WKT master's method: $T(n) = aT(n/b) + n^d \log^k n$ ⇒ $a=2$, $b=4$, $d=1$

3sol) 15

compare $\log_b a \Rightarrow \log_{2^2} 2 \Rightarrow \left(\frac{1}{2}\right)$ and $d=1 \Rightarrow (d>1) \Rightarrow f(n) = n^d \Rightarrow n \Rightarrow O(n)$

4. Use the mater method to give the tight asymptotic bound for the recurrence $T(n) = 2T\left(\frac{n}{2}\right) + n\lg n$, and use the substitution method to verify its upper bound. [25 points]. **[Clue: the method's Case 2: If $f(n) = \Theta(n^{\log_b a} \lg^k n)$ for some constant k≥0, f(n) and $n^{\log_b a}$ grow at similar rates and the solution is $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$].**

10

↳ The answers in Back!

(13, 19, 9, 5, 14, 7, 4, 21, 6, 14)

p i j

19, 13.

5. The code implements the array partition for quicksort. Answer questions:

(1) The operation of PARTITION is applied on the array A = {13;19; 9; 5; 14; 7; 4; 21; 6; 14}. Draw the array right after the first execution of any **"exchange"** in the code. [10 points]

Sol) * After

19	13	9	5	14	7	4	21	6	14

* Because pivot 13 is less than 19 So pivot Exchange with 19

```
PARTITION(A, p, q) {    /* A[p. . q] */
    x←A[p]     /* pivot= A[p] */
    i←p
    for j←p+ 1 to q {
      do if A[j] ≤ x then {
            i ← i + 1
            exchange A[i] ↔A[j]
      }
    }
    exchange A[p] ↔ A[i]
    return i
```

(2) Suppose that the partitioning algorithm always produces a 99-to-1 proportional split. What is the recurrence (function)? [10 points]

$\boxed{T(n) = T(n-1) + n}$ Because Eq-17 decrementing By 1 and we are doing $i = i+1 \Rightarrow n$ $T(n) = T(n-1)$

(3) **What's the recurrence's upper bound**? (guess process or proof not required) [10 points]

* Recurrence's upper bound (means Worst-Case) Scenario ie $O(n^2)$ Because we have to traverse loop for other two or more times

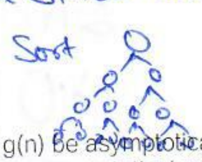
6. Show how to sort *n* integers in the range 0 to $n^3 - 1$ in O(*n*) time. (hint: radix sort). [20 points]

20

$\boxed{\frac{b}{r}\left(n+2^r\right)}$ → assume we have two Elements $(n=2)$ 0 to $(2^3-1) = (7)$

* Since we know the length and range of the Elements

So we will do the radix Sort



⇒ using $\frac{b}{r}\left(n+2^r\right)$ we can say Sorting them median and divide and Conquer

7. **[Bonus Question]** Let f(n) and g(n) be asymptotically positive functions. Use an example to disprove the **statement:** "*At least one of the two asymptotical relationships is true: (1) f(n) = O(g(n)); or (2) g(n) = O(f(n))*" [20 points] [Clue: functions such, as sin(x) + 1 and cos(x)+1, **don't serve the purpose**.]

$b = \log_2 n^3$ then $\frac{\log n^3}{\log n}(n+2^r) \Rightarrow \frac{3\log n}{\log n}(n+n) \Rightarrow 3(2n) \Rightarrow O(n)$

4Sol) $T(n) = 2T(n/2) + n\log(n)$

master method $\Rightarrow T(n) = aT(n/b) + f(n)$

$a = 2,\ b = 2,\ f(n) = n^d \log_2^{k+1} = n^1 \log_2^1$

Since $\log_b^a = 1$ and $k = 0,\ d = 1$

Case $O\left(n^{\log_b^a} \log^{k+1}_n\right)$

$= O(n\log^2 n)$

* Approving By Substitution method

Verify it's by Substitution method.

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

$T(n/2) = 2T(n/4) + \frac{n}{2}\log(\frac{n}{2}) \rightarrow ②$

$T(n/4) = 2T(n/8) + n/4 \log(n/4) \rightarrow ③$

$② \times ① \Rightarrow \quad 2\left[2T(n/4) + \frac{n}{2}\log(n/2)\right] + n\log(n)$

$\Rightarrow 4T(n/4) + n\log(n/2) + n\log(n) \hookrightarrow ④$

$③ \times ④ \Rightarrow 4T\left[2T(n/8) + \frac{n}{4}\log(n/4)\right] + n\log(n/2) + n\log n$

$\Rightarrow 8T(n/8) + n\log(n/4) + n\log(n/2) + n\log n$

$\Rightarrow T(1)$

$\Rightarrow 2^3 T(n/2^3) + n\log(n/2^2) + n\log(n/2^1) + n\log(\frac{n}{2^0})$

$T(1) \Rightarrow$

$2^k T(n/2^k) + n\log(\frac{n}{2^{k-1}}) + n\log(\frac{n}{2^{k-2}}) \ldots \rightarrow ⑤$

$\boxed{\frac{n}{2^k} = 1} \Rightarrow n = 2^k \Rightarrow k = \log n$

keep in ⑤ $\quad 2^{\log_2 n} + n\log(\frac{n}{2^{\log n - 1}}) + n\log(\frac{n}{\log n - 1})$

$+ \ldots + n\log(\frac{n}{2^0})$

Sol) $O(n\log^2(n))$

7) $f(n) = 2n^2 + n$

$g(n) = n^2$

Case let $f(n) \le c\,g(n)$

$n = 5$

$2(5)^2 + 5 \le c(5)^2 \Rightarrow c = 2$

$55 \le 2(5)(25)$

$55 \le 250$

Hence proved that $f(n) \le c\,g(n)$

$\Rightarrow f(n) = c\,g(n)$

$2n^2 + n = c(n^2)$

let $n = 0$

$0 + 0 = c(0)$

$0 = 0$

only for $n = 0$ this is possible and

again

$g(n) = n^2 + 1$

$f(n) = 2n^2$

let $n = 1$

$2(1)^2 = (1)^2 + 1 \Rightarrow 2 = 2$

$\Rightarrow O(1)$

So $g(n) = O(f(n))$

CSE 5311 Quiz 1 (Fall 2022)

1:10-1:50pm, 9/22 (Thursday)

Name:

1. **"The best worst-case running time that we've seen for a sorting algorithm is O(nlgn)".** Is this statement correct? Why? [5 points]

   False. Counting sort $\Rightarrow O(n+k)$

2. "The time for a randomized quicksort to sort n numbers is O(nlgn)." Is this statement correct? Why? [5 points]

   False. Worst case $O(n^2)$

3. Use the master method to give the tight asymptotic bound for the recurrence (you are not required to verify it): $T(n) = 2T\left(\frac{n}{4}\right) + n$.[15 points]

   $\log_4 2 = \frac{1}{2} \Rightarrow \Theta(n)$

4. Use the mater method to give the tight asymptotic bound for the recurrence $T(n) = 2T\left(\frac{n}{2}\right) + nlgn$, and use the substitution method to verify its upper bound. [25 points]

   $\log_2 2 = 1 \Rightarrow \Theta(n\lg^2 n)$ | $T(n) = 2T(n/2) + n\lg n = \cancel{2}c\frac{n}{\cancel{2}}\lg^2\frac{n}{2} + n\lg n$

   $= nc(\lg^2\frac{n}{2} + \lg n) = nc((\lg n - 1)^2 + \lg n)$

   $= \cancel{nc(\lg^2 n - 2\lg n + 1) = n\lg^2 n - n(\lg n - 1)}$ prove

   $\cancel{\text{+ve for } n > 2}$

   $= cn\lg^2 n - n\lg n(2c-1) + n$

   $= cn\lg^2 n - n(\lg n(2c-1) - 1)$

   +ve for $n > 2$ $c > 2$ prove

5. The code implements the array partition for quicksort. Answer questions:

   (1) The operation of PARTITION is applied on the array A = {13;19; 9; 5; 14; 7; 4; 21; 6; 14}. Draw the array right after the first execution of any **"exchange"** in the code. [10 points].

   {13; 9; 19; 5; 14; 7; 4; 21; 6; 14}

   (2) Suppose that the partitioning algorithm always produces a 99-to-1 proportional split. What is the recurrence (function)? [10 points]

   $T(n) = T(0.99n) + T(0.01n) + O(n)$

   (3) **What's the recurrence's upper bound**? (guess process or proof not required) [10 points]

   $O(n\lg n)$

```
PARTITION(A, p, q) {   /* A[p. . q] */
        x←A[p]    /* pivot= A[p] */
        i←p
        for j←p+ 1 to q {
         do if A[j] ≤ x then {
                i ← i + 1
              exchange A[i] ↔A[j]
         }
        exchange A[p] ↔ A[i]
        return i
    }
```

6. Show how to sort n integers in the range 0 to $n^3 - 1$ in O(n) time. (hint: radix sort). [20 points]

   $\Theta\left(\frac{b}{r}(n+2^r)\right)$ where $b = \lg n^3 = 3\lg n$, $r = \lg n$

   Then $\Theta\left(\frac{b}{r}(n+2^r)\right) = \left(\frac{3\cancel{\lg n}}{\cancel{\lg n}}(n+2^{\lg n})\right)$

   $= 3(n+n) = 6n$

   $= \Theta(n)$

7. [**Bonus Question**] Let f(n) and g(n) be asymptotically positive functions. Use an example to disprove the **statement:** "*At least one of the two asymptotical relationships is true: (1) f(n) = O(g(n)); or (2) g(n) = O(f(n))*" [20 points] [Clue: functions such, as *sin(x) + 1 and cos(x)+1*, don'**t serve the purpose.**]

   $f(n) = \begin{cases} 2^n & \text{for } n = \text{even} \\ 3^n & \text{for } n = \text{odd} \end{cases}$

   $g(n) = \begin{cases} 3^n & \text{for } n = \text{even} \\ 2^n & \text{for } n = \text{odd} \end{cases}$

   $\Rightarrow f(n) \neq O(g(n))$ and $g(n) \neq O(f(n))$