COMP3711H

Homework 1

LAM, Pak Ho

A COMP3711H Written Assignment

February 21, 2026

Question 1

a. There are 6^i subproblems at the level i of recrusion tree.

b. The input size of each problem is $\frac{n}{2^i}$

c. $\left(\frac{n}{2^i}\right)^2$ work is done in one subproblem of level i.

d. Summing all subproblems in level i, the total work is $6^i \times \left(\frac{n}{2^i}\right)^2 = \left(\frac{3}{2}\right)^i \times n^2$

e. There are $\log_2 n + 1$ levels.

f. As the non recursive work is multiplied with $\frac{3}{2}$ which is larger than 1, the bound is dominated by the leaves. From d we know the total work at the bottom level with leaves is $\left(\frac{3}{2}\right)^{\log_2 n}$

The asymptotically tight upper bound on $T(n)$ is $\left(\frac{3}{2}\right)^{\log_2 n} \times n^2 = n^{log_2 3} \times n = n^{\log_2 6} \approx O(n^{2.585})$

Question 2

a. There are 3^i subproblems at the level i of recrusion tree.

b. The input size of each problem is $\frac{n}{2^i}$

c. $\left(\frac{n}{2^i}\right)^2$ work is done in one subproblem of level i.

d. Summing all subproblems in level i, the total work is $3^i \times \left(\frac{n}{2^i}\right)^2 = \left(\frac{3}{4}\right)^i \times n^2$

e. There are $\log_2 n + 1$ levels.

f. As the non recursive work is multiplied with $\frac{3}{4}$ which is smaller than 1, the bound is dominated by the root. Total non-recursive work $T(n) = \left(\frac{3}{4}\right)^i \times n^2 = n^2$ is dominated by root level with

$$\sum_{i=0}^{log_2 n} \left(\frac{3}{4}\right)^i \times n^2 \leq \frac{1}{1-\frac{3}{4}} n^2 = 4n^2$$

, we get $T(n) = O(n^2)$

Question 3

For Question 1: We assume that

$$T(n) \leq c \times n^{\log_2 6} \forall n = 2^p \text{ Large enough and } c \in \mathbb{R},$$

For $n = 1$,

$$T(1) = 1 \leq c \times 1^{\log_2 6} = c$$

It is true for $n = 1$.
Assume for $n = k$ its true,

$$T(k) \leq c \times k^{\log_2 6}$$

For $n = 2k$,

$$\begin{aligned} T(2k) &= 6T(k) + (2k)^2 \\ &\leq 6ck^{\log_2 6} + 4k^2 \\ &\leq (6c + 4)k^{\log_2 6} \\ &\leq \frac{(6c + 4)(2k)^{\log_2 6}}{6} \\ &\leq C(2k)^{\log_2 6} \end{aligned}$$

By mathematical induction, the upper bound holds for sufficient large n as the upper bound still holds. with a slightly larger constant C.

For Question 2: We assume that

$$T(n) \leq c \cdot n^2$$

For $n = 1$,

$$T(1) = 1 \leq c \times 1^2 = c$$

It is true for $n = 1$.
Assume for $n = k$ its true,

$$T(k) \leq c \times k^2$$

For $n = 2k$,

$$\begin{aligned} T(2k) &= 3T(k) + (2k)^2 \\ &\leq 3ck^2 + 4k^2 \\ &\leq (3c + 4)k^2 \\ &\leq \frac{(3c + 4)(2k)^2}{4} \\ &\leq c(2k)^2 \end{aligned}$$

By mathematical induction, the upper bound holds for sufficient large n as the upper bound still holds.

Question 4

a. It is false. A counterexample

$$f(n) = 2n, g(n) = n$$

It fulfills the statement,Where c is a constant with

$$f(n) = 2n \leq 2n = 2g(n)$$

since it fulfills

$$f(n) \leq c \cdot g(n) \text{ and } f(n) = O(g(n))$$

However

$$\begin{aligned} 2^{f(n)} &= 2^{2n} = 4^n \\ 2^{g(n)} &= 2^n \\ 2^{f(n)} &\not\leq O\left(2^{g(n)}\right) \end{aligned}$$

b. It is true, according to the statement we assume

$$f(n) \leq c \cdot g(n) \text{ and } f(n) = O(g(n))$$

Thus,

$$\begin{aligned} (f(n))^3 &= (c \cdot g(n))^3 = c^3 \cdot (g(n))^3 \\ &= O(g(n))^3 \end{aligned}$$

Considering that c^3 is also a constant and can be written as C to obtain

$$f(n)^3 \leq C \cdot (g(n))^3 \text{ and } f(n)^3 = O(g(n)^3)$$

Question 5

a. $T(n) = O(n^{log_2 4}) = O(n^2)$

b. $T(n) = O(n^{log_2 7})$

c. $T(n) = O(n^2 \cdot n) = O(n^3)$

d. $T(n) = O(n^{logn})$

e. $T(n) = O(3^n)$

Question 6

$$2^{\sqrt{\log n}} < n \log n < n^{\frac{4}{3}} < n^{\log n} < 3^n < 2^{n^2} < 2^{2^n}$$

Question 7

a. Picking other numbers where $n < 3$ or $n > 8$ does not affect the probability of picking 3 or 8 for comparision, we will only not compare 3 and 8 when we pick 5 or 6 (in this set) since it breaks 3 and 8 into 2 groups, So, as we already break 3 and 8 into 2 groups in the first run with 5 as the pivot, it is impossible for us to compare 3 and 8 afterwards

$$\text{The probability } P = 0$$

b. Picking other numbers where $n < 8$ or $n > 15$ does not affect the probability of picking 8 or 15 for comparision, we will only not compare 8 and 15 when we pick 9 or 12 (in this set) since it breaks 3 and 8 into 2 groups, So, as we already put 9 and 15 into a same group in the first run with 5 as the pivot, that group contains 6,8,9,12 and 15 and only 9 and 12 is between the range 8 - 15. Noted that we only compare 8 and 15 if we select 8 or 15 as the pivot. The between set is [8,9,12,15] and we may not compare 8 and 15 if we select 9 or 12 in this set, so

$$\text{The probability } P = \frac{2}{4} = 0.5$$

c. Notice that 5 is compared with 8 and the numbers 1 and 3 will never be compared with 8 since they are break into another group, we use $P(n)$ to denote the probability of n compares with 8. $\mathbb{E}(1) = 0, \mathbb{E}(3) = 0$ and $\mathbb{E}(5) = 1$ Notice that 6 and 9 must be compared with 8 as they are cloest to 8 and we do not know in the pair (6 and 8) or (8 and 9) which number is larger, we compute the probability of 8 is compared to 12 with only 9 is between, which is the probability $P = \frac{2}{3}$. The total expected number is

$$\begin{aligned}\mathbb{E} &= P(1) + P(3) + P(5) + P(6) + P(9) + P(12) + P(15) \\ &= 0 + 0 + 1 + 1 + 1 + \frac{2}{3} + 0.5 = \frac{25}{6} = 4.167\end{aligned}$$

Question 8

a. The cost of merging all arrays with standard merge algorithm stated in the Question will be

$$(2 + 3 + 4 + ... + k)n = \frac{k(k+1)}{2} = O(k^2 n)$$

Noted that the first merge cost $2n$ and the second merge cost $3n$ since the previous array has the length $2n$ and so on. The cost is formed by $O(x + y)$ with $y = n$ as the newly added array but x is increasing from 1 to k as it carry more and more previous arrays.

b. The algorithm is proposed as follows: Pair the k sorted array 2 by 2 and merge each array using the standard merge procedure, which will result at $\frac{k}{2}$ sorted array with length $2n$, we repeat this process until 1 array remains.

Instead of using the way mentioned in part a, we merge the arrays 2 by 2, we have k sorted arrays so we need to perform $\frac{k}{2}$ actions of merging 2 arrays with length n. In this layer the time takes is $\frac{k}{2} \times (n + n) = nk$. After this action the number of arrays is reduced by half but each of them have a double size notice that in every layers until reaching the result, the time takes is same since the size is always halved and the length is always doubled. So, in general every layers has the same time cost of nk It takes $\log_2 k$ time to merge every arrays since the number of arrays is halved in every operation, which results at $\log_2 k$ layers for the aforementioned algorithm. So the total time is

$$nk \cdot \log_2 k = O(nk \log k)$$

Question 9

a. The algorithm is proposed as follows: We start dropping rung from $h = 0$ and increase h by $\sqrt{n}$ before next experiment everytime, if if reach a certain $h = c\sqrt{n}$ where the 1st jar breaks, we stop the loop process of increase h by $\sqrt{n}$ Now, we perform a linear experiment from $h = (c-1)\sqrt{n}$ until $h = c\sqrt{n} - 1$ and record h when the 2nd jar breaks during the linear experiment, since the total cost is $\frac{n}{m} + m$ if we increase the height of first jar experiment by m. We are minimizing the cost so the optimal value of m is $\sqrt{n}\, f(n) = \frac{n}{\sqrt{n}} + \sqrt{n} = 2\sqrt{n} = O(\sqrt{n})$

b. The algorithm is proposed as follows: We start dropping rung from $h = 0$ and increase h by $(\sqrt[3]{n})^2$ before next experiment everytime, if if reach a certain $h = c(\sqrt[3]{n})^2$ where the 1st jar breaks, we stop the loop process of increase h by $(\sqrt[3]{n})^2$. Then we perform the experiment as, We start dropping rung from $h = (c-1)(\sqrt[3]{n})^2$ and increase h by $\sqrt[3]{n}$ before next experiment everytime, if if reach a certain $h = d\sqrt[3]{n}$ where the 2nd jar breaks, we stop the loop process of increase h by $\sqrt[3]{n}$. Now, we perform a linear experiment from $h = (d-1)\sqrt[3]{n}$ until $h = \sqrt[3]{n}$ and record h when the 3rd jar breaks during the linear experiment. For the first jar, at most $n \div (\sqrt[3]{n})^2 = \sqrt[3]{n}$ is performed. and for second jar, at most $(\sqrt[3]{n})^2 \div \sqrt[3]{n} = \sqrt[3]{n}$ is performed. Lastly, in $\sqrt[3]{n}$ rungs at most $\sqrt[3]{n}$ experiments is needed

$$f(n) = \sqrt[3]{n} + \sqrt[3]{n} + \sqrt[3]{n} = 3\sqrt[3]{n} = O(\sqrt[3]{n})$$

Question 10

a. we denote the 8 numbers as $n_1, n_2, n_3..., n_8$. We first break them into 4 groups with 2 members, a_1, a_2, a_3, a_4 and so on. we compare 2 numbers in each group (which take 4 comparisions) and the "winners" of these 4 groups (the larger numbers in their respective groups) will be grouped as "max", the remaing numbers of the 4 groups will be grouped as "min". In the max group, we break them into 2 groups and do another 2 comparisions, the large numbers in these 2 comparisions will be compared for 1 time and we select the larger number as the max. in the max group, 3 more comparisions are done. Lastly, we seperate the numbers in min into 2 groups and select the 2 smaller numbers in each group and perform the final comparision with these 2 numbers, the smaller one will be the min, which makes min group also do 3 comparision. In total, we have done $4 + 3 + 3 = 10$ comparisions.

b. Since Divide and Conquer is a must, this algorithm use a slightly different as part a. For divide, We seperate the numbers into 2 groups, denote as L and R. And then we conquer them recursively such that we can obtain the max and min values of L and R respectively, $max_L, max_R, min_L, min_R$. Here in combine, 2 comparisions is needed to determine max value by comparing max_L with max_R and find min value by min_L with min_R, Hence the recurrence is

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

From this approach, we do 2 comparisions in the top level, and do 4 comparisions in the next level until reaching the base case. Notice that in other layers where $n \neq 2$ we need two comparisions to determine the max and min along 2 groups by comparing max_L with max_R and by comparing min_L with min_R. But in $n = 2$ case, only 1 comparision is needed as the larger one will be max and the smaller one will be min trivially. Hence the sum of need comparisions will be

$$2 + 4 + 8 + 16 + ... + \frac{n}{4} + \frac{n}{2} + n \div 2 = \frac{2 \times \left(2^{log_2 \frac{n}{2}} - 1\right)}{2 - 1} + \frac{n}{2} = \frac{3n}{2} - 2$$