Problem Set 2

Name: Vivek Verma

Collaborators: None

Problem 2-1.

(a) $T(n) = 4(\frac{n}{2}) + O(n)$

Master theorem method:
$a = 4,\ b = 2,\ f(n) = O(n)$
$n^{log_b a} = n^{log_2 4} = n^2$
$f(n) = O(n) = O(n^{log_2 4 - \epsilon}) \quad , where\ \epsilon = 1$
Since, there exist some ϵ for which $f(n)$ grows slower than $(n^{log_2 4 - \epsilon})$, Case 1 can be applied here.
$\therefore\ T(n) = \theta(n^2)$

Recursion tree method:
Number of nodes at each level $l = 4^l$
Cost of each level $l = c.4^l \left(\frac{n}{2^l}\right)$
Number of levels = $log_2 n$

$$\begin{aligned} Total\ cost &= \sum_{l=0}^{log_2 n} c4^l \left(\frac{n}{2^l}\right) \\ &= n \sum_{l=0}^{log_2 n} c2^l \\ &= cn\left(2^{log_2 n + 1} - 1\right) \\ &= cn(2n-1) \\ &= O(n^2) \end{aligned}$$

The tree has $4^{log_2 n} = n^2$ number of total leaves and the cost of each leaf is $\theta(1)$.
$\therefore$ Total cost is lower bounded by n^2.

(b) $T(n) = 3T(\frac{n}{\sqrt{2}}) + O(n^4)$

Master theorem method:

$a = 3,\ b = \sqrt{2},\ f(n) = O(n^4)$

$n^{log_b a} = n^{log_{\sqrt{2}} 3}$

There cannot be any ϵ for which $f(n)$ will grow slower than $O(n^{log_{\sqrt{2}} 3-\epsilon})$

And there cannot be any k either for which $f(n)$ will behave asymptotically same as $(n^{log_{\sqrt{2}} 3} log^k n)$

Therefore, case 1 and case 2 cannot be applied.

Since, $f(n)$ grows faster than $n^{log_{\sqrt{2}} 3+\epsilon}$ for $\epsilon = 1$, and

$3f(\frac{n}{\sqrt{2}}) < cf(n)$ for $c = 0.9$

$\therefore T(n) = \theta(n^4)$

Recursion tree method:

Number of nodes at each level $l\ is\ 3^l$

Cost of each level $l\ is\ c\ 3^l \left(\frac{n}{\sqrt{2}^l}\right)^4$

Number of levels $= log_{\sqrt{2}} n$

$$
\begin{aligned}
Total\ cost &= \sum_{l=0}^{log_{\sqrt{2}} n} c3^l \left(\frac{n}{\sqrt{2}^l}\right)^4 \\
&= cn^4 \sum_{l=0}^{log_{\sqrt{2}} n} \left(\frac{3}{4}\right)^l \\
&= cn^4 \left(\frac{1 - (\frac{3}{4})^{log_{\sqrt{2}} n+1}}{1 - \frac{3}{4}}\right) \\
&= O(n^4)
\end{aligned}
$$

Since, the number of leaves in the tree are $3^{log_{\sqrt{2}}(n)} = 3^{2log(n)} = n^{2log3}$ and cost of each leaf is $\theta(1)$.

$\therefore$ Total cost of T(n) is lower bounded by n^{2log3}.

(c) $T(n) = 2T(\frac{n}{2}) + 5nlog(n)$

Master theorem method:

$a = 2,\ b = 2,\ f(n) = 5nlog(n) = \theta(nlog(n))$

$n^{log_b a} = n^{log_2 2} = n$

since $f(n) = \theta(n^{log_2 2} log^k n)$ for $k = 1$. Case 2 can be applied here.

$\therefore T(n) = \theta(nlog^2 n)$

Recursion tree method:

Number of nodes at each level $l\ is\ 2^l$.

Cost of each level $l\ is\ c\ 2^l\ \left(\frac{n}{2^l} log \frac{n}{2^l}\right) = cnlog\frac{n}{2^l}$

Number of levels $= log_2 n$

$$
\begin{aligned}
Total\ cost\ &= \sum_{l=0}^{log_2 n} cnlog\frac{n}{2^l} \\
&= cn \sum_{l=0}^{log_2 n} log\frac{n}{2^l} \\
&= cn \sum_{l=0}^{log_2 n} \left(log_2 n - log_2 2^l\right) \\
&= cn \sum_{l=0}^{log_2 n} log_2 n - cn \sum_{l=0}^{log_2 n} l \\
&= cnlog_2(n) * (log_2(n) + 1) - cn \left(\frac{log_2(n) * (log_2 n + 1)}{2}\right) \\
&= \frac{cn}{2} (log_2(n) * (log_2 n + 1)) \\
&= \theta(nlog^2 n)
\end{aligned}
$$

(d) $T(n) = T(n-2) + \theta(n)$
Substitution method:
Guess, $T(n) = cn^2$

Base case: $n = 1$,
since $T(1) = \theta(1) = c(1^2)$
therefore base case is true.

Inductive case: Assume $T(n-2) = c(n-2)^2$
Then,

$$\begin{aligned} T(n) &= T(n-2) + \theta(n) \\ &= c(n-2)^2 + \theta(n) \\ &= \theta(n^2) = c(n^2) \end{aligned}$$

$\therefore T(n) = \theta(n^2)$

Problem 2-2.

(a) Insertion sort and Selection sort are both **in-place** sorting algorithms.
Selection sort performs n swaps in the worst case. Cost of each swap is $nlog(n)$.
Hence the worst case time complexity of selection sort is $O(n^2log(n))$
Insertion sort performs n^2 swaps in worst case. And cost of each swap is $nlog(n)$.
Hence the worst case time complexity of insertion sort is $O(n^3log(n))$.

$\therefore$ **Selection sort** is best.

(b) Selection sort and Insertion sort both requires n^2 comparisons in worst case. Cost of each comparison is $O(log(n))$.
Hence the total cost of $O(n^2log(n))$.

Calculating total cost of merge sort:

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

Applying case 2 of *Master theorem*, the total cost of merge sort where each comparison takes $O(log(n))$ time,

$$T(n) = \theta(nlog^2(n)$$

$\therefore$ **Merge sort** is best.

(c) Selection sort and Merge sort both are independent of the array being already sorted.
Insertion sort requires $O(n + number\ of\ adjacent\ swaps)$ time. i.e
$O(n + log(log(n)))$
Hence, **insertion sort** is best in this case.

Problem 2-3. If Picard starts from northern end of the island, i.e $0_{th}\ kilometer$,

$$i = 0$$

Then in each step, teleport to 2^i and increment i by 1 in until $k <= 2^i$
Perform a binary search from 2^{i-1} to 2^i kilometers and find the location of Picard i.e k.

If Picard starts from southern end of the island, i.e $(n-1)_{th}\ kilometer$,

$$i = n - 1$$

Then in each step, teleport to 2^i and decrement i by 1 until $k >= 2^i$.
Perform a binary search from 2^i to 2^{i+1}.

From Start to finding i such that either $(2^{i-1} < k <= 2^i)$ or $(2^i <= k < 2^{i+1})$ is true,
takes $O(log(n))$ time. Then finding the location of Picard between the shorter range does not require more than $O(log(n))$ time.

$\therefore$ Total time complexity of finding Picard on the island is $O(log(n))$.

Problem 2-4. A sorted array and a Doubly linked list will be used to build the database.
Each element of Doubly linked list is a message.
Each element of Sorted array is an Object which contains two things:

- •Viewer ID
- •Dynamic array of pointers to messages by user in Doubly linked list

build(V):
Building a sorted array takes $O(nlog(n))$ time and creating an empty Doubly linked list takes constant time.

send(v, m):
To send message m to the chat from viewer with ID v,

- •Find the viewer v in sorted array in $O(log(n))$ time.
- •if the Dynamic array associated with viewer is empty, return.
- •Otherwise, Add the message at the end of Doubly linked list in $O(1)$ time,
- •and store the message pointer m_p pointing to the message in Doubly linked list to the dynamic array corresponding the viewer v in $O(1)$ time.

$\therefore$ this operation takes $O(log(n))$ time in worst case.

recent(k):
Return the last k entries of Doubly Linked List in $O(log(k))$ time.

ban(v):
To ban a viewer v, follow the steps:

- •Find the user in sorted array in $O(log(n))$ time.
- •Delete the messages sent by the user one by one, by getting the message pointers, delete the message pointed by message pointer, and then delete the message pointer from the dynamic array.

Problem 2-5. *This one is hard :(Will do it later*

(a)

(b)

(c)