HW2

Keerthana Golla

September 2023

1 Problem - 1

In order to determine which algorithm is asymptotically faster between the three versions with division into groups of 3, 5 (standard Select), and 7, we can analyze their time complexity.

Standard Select with division into groups of 5: The algorithm first divides the array into $n/5$ groups of 5 elements each. Then it finds the median of each group (which takes constant time). Next, it recursively applies the Select algorithm to find the median of medians. Finally, it partitions the n/5 arrays medians based on the median of medians and then continues recursively on partitioned array .But it takes at-most one recursive call here since it either goes to if condition or else condition as discussed in the class. The recurrence relation for this algorithm can be expressed as
$T(n) = T(n/5) + T(7 * n/10 + 6) + O(n)$
where $T(n)$ is the time to find the median of an array of size n. which has a time complexity of $O(n)$ as discussed in the class with proof by Induction .
Algorithm with division into groups of 7: Similar to the standard Select, this algorithm divides the array into $n/7$ groups of 7 elements each and follows the same procedure. (4) $\left(\frac{1}{2}\left(\frac{n}{7}\right) - 2\right) \geq \frac{2n}{7} - 8$ .And the rest of the elements will be $n - \left(\frac{2n}{7} - 8\right) = \frac{5n}{7} + 8$
So, The recurrence relation for this algorithm is
$\mathrm{T(n)} = \mathrm{T}\left(\frac{n}{7}\right) + T\left(\frac{5n}{7} + 8\right) + \Theta(n)$
Since, the above cant be solved using master's theorem lets solve using proof of induction.similar way of how we got time complexity for select algorithm with 5 division by proof by Induction , we can also use same process here to get time complexity , $assume\mathrm{T(n)} \leq c * n$,
$\mathrm{T(n)} = \mathrm{T}\left(\frac{n}{7}\right) + T\left(\frac{5n}{7} + 8\right) + \Theta(n) = T\left(\frac{n}{7}\right) + T\left(\frac{5n}{7}\right) + \Theta(n)$
$\leq c \cdot \frac{n}{7} + c\left(\frac{5n}{7}\right) + (k * n)$,where k > 0
$\leq \frac{6cn}{7} + (k * n) \leq cn$. last inequality here is $c > 7k$, hence - time complexity is $O(n)$
Algorithm with division into groups of 3:

Similar to the standard Select, this algorithm divides the array into $n/3$ groups of 3 elements each and follows the same procedure. (2) $\left(\frac{1}{2}\left(\frac{n}{3}\right) - 2\right) \geq \frac{n}{3} - 4$. And the rest of the elements will be $n - \left(\frac{n}{3} - 4\right) = \frac{2n}{3} + 4$
so,the recurrence relation for this algorithm is
$T(n) = T(n/3) + T(2 * n/3) + O(n)$.
Since, the above cant be solved using master's theorem lets solve using proof by Induction(similar to what we did in class). assume $T(n) \leq c * n$,
$\mathrm{T(n)} = \mathrm{T}\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + \Theta(n)$
$\leq c \cdot \frac{n}{3} + c\left(\frac{2n}{3}\right) + (k * n)$,where k > 0
$\leq cn + (k * n) \leq cn$. For this , k should be zero which is contradicting , since k should be greater than 1 for the time complexity to be $\Theta(n)$.
So lets use recursion tree method, The shortest path to a leaf occurs when we take the heavy branch each time. The height k is given by $n\left(\frac{1}{3}\right)^k \leq 1$, meaning $n \leq 3^k$ or $k \geq \log_3 n$. The longest path to a leaf occurs when we take the light branch each time. The height k is given by $n\left(\frac{2}{3}\right)^k \leq 1$, meaning $n \leq \left(\frac{3}{2}\right)^k$ or $k \geq \log_{\frac{3}{2}} n$. On any full level, the additive terms sum to n. There are $\log_3 n$ full levels. Thus, $T(n) = n \log_3 n = O(n \log n)$.
Conclusion: If we compare the time complexities we can clearly say that groups of 3 takes asymptotically much time when compared to that of groups of 5,7 (since groups of 5,7 are taking only $O(n)$ and $n \log n$ asympotically larger than n)

2 Problem - 2

In order to determine which algorithm is asymptotically faster ,we need to compare the runtime of QuickSort when using Alice's pivot selection technique with the runtime when using Bob's pivot selection technique.

In General QuickSort, when the pivot provides separation of $a|b$, the recurrence relation for the expected runtime is given by:

$$T(n) = T(a) + T(b) + O(n)$$

Alice's Pivot Selection Technique:

It is mentioned that Alice's technique provides separation of $\frac{n}{4}|\frac{3n}{4}$, in constant time. We will pick that to be a pivot.So we got a pivot that divides array into $\frac{n}{4}|\frac{3n}{4}$. But it is not mentioned that the elements left to the pivot are less than the pivot. They just mentioned that on rank of pivot if x . so in order to partition and arrange the "x" elements that less than pivot to the left side and "n-x" elements that are greater than pivot on to the right side it takes O(n) .

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + O(n)$$

So, for Alice's technique: Since, the above cant be solved using master's theorem .So lets use recursion tree method, The shortest path to a leaf occurs when we take the heavy branch each time. The height k is given by $n\left(\frac{1}{4}\right)^k \leq 1$, meaning $n \leq 4^k$ or $k \geq \log_4 n$. The longest path to a leaf occurs when we take the light branch each time. The height k is given by $n\left(\frac{3}{4}\right)^k \leq 1$, meaning $n \leq \left(\frac{4}{3}\right)^k$ or $k \geq \log_{\frac{4}{3}} n$. On any full level, the additive terms sum to n. There are $\log_4 n$ full levels. Thus, $T(n) = O(n \log_4 n) = O(n \log n)$.

Bob's Pivot Selection Technique: similarly for Bob's technique provides separation of $\frac{n}{3}|\frac{2n}{3}$.
So, for Bob's technique:

$$T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + O(n)$$

Since, the above cant be solved using master's theorem . So lets use recursion tree method, The shortest path to a leaf occurs when we take the heavy branch each time. The height k is given by $n\left(\frac{1}{3}\right)^k \leq 1$, meaning $n \leq 3^k$ or $k \geq \log_3 n$. The longest path to a leaf occurs when we take the light branch each time. The height k is given by $n\left(\frac{2}{3}\right)^k \leq 1$, meaning $n \leq \left(\frac{3}{2}\right)^k$ or $k \geq \log_{\frac{3}{2}} n$. On any full level, the additive terms sum to n. There are $\log_3 n$ full levels. Thus, $T(n) = O(n \log_3 n) = O(n \log n)$.

Conclusion: If we compare the time complexities we can clearly say that both are asympotically equal.