Problem 4 (sorting): Selection and Merge sort Analysis

Selection sort:

let's consider, O_b = number of operations before sorting routine
O_0 = number of operations in outer for loop
O_i = number of operations in inner for loop
PO_s = number of operations in if-condition

In here, we are working on the partial selection sort for P elements, the selection sort will have to iterate P-1 times to sort a set.

$$f(N) = O_b + \sum_{i=0}^{P-1} \left(O_0 + \sum_{J=i+1}^{n-1} \left(O_i + PO_s \right) \right)$$

let, $O_i + PO_s = O_{is}$

$$= O_b + \sum_{i=0}^{P-1} \left(O_0 + ((n-1)-(i+1)+1) O_{is} \right)$$

$$= O_b + \sum_{i=0}^{P-1} \left(O_0 + (n-1) O_{is} - i O_{is} \right), \text{ let } n-1 = n'$$

$$= O_b + \sum_{i=0}^{P-1} \left(O_0 + n' O_{is} \right) - O_{is} \sum_{i=0}^{P-1} i$$

$$= O_b + (P-1-0+1)(O_0 + n' O_{is}) - O_{is}(P-1-0+1)\, i$$

$$= O_b + P(O_0 + n' O_{is}) - i O_{is} P$$

$$= O_b + O_0 P + P n' O_{is} - i O_{is} P$$

$$= O_b + O_0 P + P(n-1) O_{is} - i O_{is} P$$

$$= O_b + O_0 P + O_{is} P n - P O_{is} - i O_{is} P$$

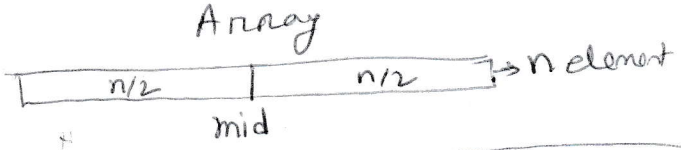
$$= P n O_{is} + (O_0 + i O_{is} - O_{is}) P + O_b'$$

$$= P n c'' + c' P + O_b$$

↑ it is ~~order of~~ $O(P \cdot n)$

Merge Sort - time Analysis:

For A merge sort recursive algorithm, first we can derive the recurrence relation of the merge sort. The recurrence relation of the merge sort:



$T(n) = T(n/2) + T(n/2) + n$

$T(n) = 2T(n/2) + n$ ..... (I)

Substitute $n/2$ in place of n in eq (I)

$T(n/2) = 2T(n/4) + n/2$ ..... (II)

$T(n)$ = time is taken for sorting n elements using merge sort

$T(n/2)$ = time is taken for sorting left $(n/2)$ elements and right.

n = Time is taken to merge n-elements

from equation (II), $T(n/2)$ substitutes in equation (I)

$T(n) = 2(2T(n/4) + n/2) + n$

$= 2^2 T(n/2^2) + n + n$

$= 2^2 T(n/2^2) + 2n$ ..... (III)

Again, substitute $n/4$ in place of n in equation (I)

$\therefore T(n/4) = 2T(n/8) + n/4$ ..... (IV)

from equation (IV), substitute $T(n/4)$ to equation (III)

We get $T(n) = 2^2(2T(n/8) + n/4) + 2n$

$= 2^3 T(n/2^3) + 3n$ ..... (V)

If we again substitute the $n/8$ in place of n in equation (I)

Then we will get a equation (V) as

$$T(n) = 2^4 T(n/2^4) + 4n \text{ ---- (VI)}$$

So, from equation (III), (V), (VI), in general term we can write as

$$T(n) = 2^k T(n/2^k) + kn \text{ ---- (VII)}$$

In Base condition,

When only one element $T(1)=1$. Every time number of element becomes half or $(n/2)$, so the term $n/2^k$ becomes one after some steps.

Let $\frac{n}{2^k} = 1$, then $n = 2^k$

$\Rightarrow \log_2 n = \log_2 2^k$ [Apply $\log_2$ in both sides]

$\Rightarrow \log_2 n = k$

Apply k into equation (VII)

$$T(n) = nT(1) + n\log_2 n$$

↑ it is a $\underline{O(n\log_2 n)}$

☐ Does the size of p change the analysis and when does 1 sort outperform the other?

⇒ The size of p changes the timing and operational analysis for selection sort, but it does not effect on or change merge sort.

⇒ When input length or array size is large, and P is small then selection ~~sho~~ sort may outperform the merge sort. Otherwise merge sort outperform the other.