Advanced Algorithms

CS 5300

01/20/2023

HW #1

(1) Suppose computer A is running a sorting algorithm and it is supposed to sort an array of one million numbers. Suppose that computer A executes a billion instructions per second, & suppose computer A requires $100n \lg n$ instructions to sort n numbers. Find the time it takes computer A to sort one million numbers.

Sol) It is given that computer A runs an sorting Algorithms and,

It sorts one million numbers $= 10^6$ numbers

given that it executes a billion instructions per second $= 10^9$ instruction/sec

Now, we need to find the time that computer A takes to sort one million is

We know that

$$\text{Time} = \frac{\text{Distance}}{\text{Speed}}$$

$\therefore$ Distance $= 100 n \lg n$

Speed $= 10^9$ instruc/sec

$\Rightarrow \frac{100 \times 10^6 \times 19 10^6 \text{ instructions}}{10^9 \text{ instructions/sec}}$

$\Rightarrow 10^2 \times 10^6 \times 10 9 10^6 / 10^9 = \frac{10^8\ 19\ 10^6}{10^9}$ seconds

$\Rightarrow \frac{6 10 9 10}{10}$

$\Rightarrow 0.6 \times 19 10$ seconds

$\Rightarrow 0.6 \times \frac{\ln 10}{\ln 2}$ $\left(\because \log x^n = n \log x \quad \log_2 x = \frac{\ln x}{\ln 2}\right)$

$\Rightarrow 0.6 \times 3.3219$

Time $\simeq$ 1.99 seconds $\simeq$ 2 seconds

---

(2) Suppose we are comparing implementation of insertion sort and merge sort on the same machine. For inputs of size n, insertion sort runs in $2n^2$ steps, while merge sort runs in $50 n \lg n$ steps. For which values of n does insertion sort beat merge sort?

Sol) Given that insertion sort run $2n^2$ steps and Merge sort runs $50 n \lg n$ steps.

Let us assume that for the value of n insertion sort beat Merge sort.

$\therefore 2n^2 > 50 n \log n$

∴ The above condition satisfies the asked question such that insertion beats Merge sort

$2n^2 > 50n\lg n$

$2n > 50\lg n$

$n > 25\lg n$

$n > \lg n^{25}$

$\frac{n}{25} > \lg n$

$2^{n/25} > n$

Now, we have to find the smallest value of 'n' for which this inequality holds.

If $n = 25$

$2^{25/25} > 25$

$2 \not> 25$ (False)

If $n = 100$

$2^{100/25} \not> 100$

$2^4 > 100 \Rightarrow 16 \not> 100$ (false)

If $n = 200$

$2^{200/25} > 200$

$2^8 > 200$

$256 > 200$ (Holds true)

Hence, for $n = 200$; insertion beats Merge sort

Hence, proved #

Another way of solving:

$\Rightarrow 2n^2 < 50 n \lg n$

$n < 25 \lg n \Rightarrow \frac{n}{25} < \frac{\log n}{\log 2}$

$\Rightarrow \log_2 n^{n/25} < \log n$

$\Rightarrow 2^{n/25} < n$ (or) $n > 2^{n/25}$

$n = 25, \quad 25 > 2^{25/25} = 2$ (True)

$n = 50, \quad 50 > 2^{50/25} = 2^2 = 4$ (True)

$n = 75, \quad 75 > 2^{75/25} = 2^3 = 8$ (True)

$n = 175, \quad 175 > 2^{175/25} = 2^7 = 128$ (true)

$n = 200, \quad 200 > 2^{200/25} = 2^8 = 256$ (False)

for $n = 189 \Rightarrow 189 > 2^{189/25} = 188.70$ (True)

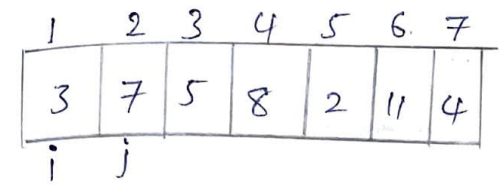
$n = 190 > 2^{190/25} = 2^{7.6} = 194.01$ (false)

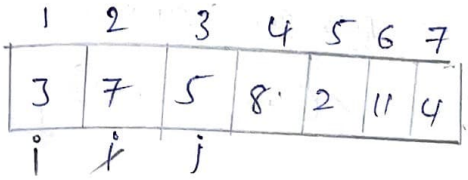
$\therefore$ for $n \geq 190$, insertion sort beats merge sort

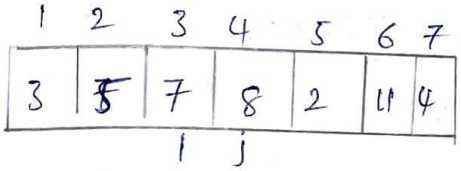
(3) Using the example we went over in the class as a model, illustrate the operations of insertion sort on the array $A = \langle 3, 7, 5, 8, 2, 11, 4 \rangle$

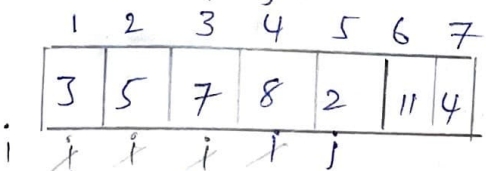
Sol)

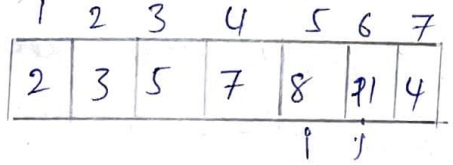
A

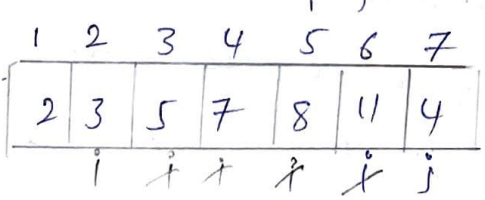


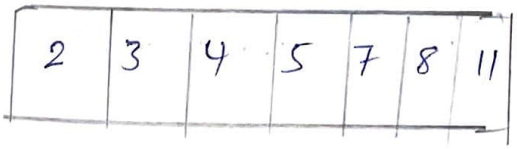












j	key	i
2	7	1
3	5	~~2~~ 1
4	8	3
5	2	4 ~~3~~ ~~2~~ ~~1~~ 0
6	11	5
7	4	~~6~~ ~~5~~ ~~4~~ ~~3~~ 2

Sorted Array =

$\langle 2, 3, 4, 5, 7, 8, 11 \rangle$

(4) Rewrite the insertion sort procedure to sort into non-increasing instead of non decreasing order

Sol) Insertion-Sort(A) for non-increasing:

```
for j=2 to A.length
    key = A[j]
    // insert A[j] into the sorted sequence
                                A[1, ....., j-1])
    i = j-1
    while i>0 and A[i] < key
        A[i+1] = A[i]
        i = i-1
    A[i+1] = key.
```

(5) Use the top down approach to illustrate the operations of merge sort on the array $A = \langle 3, 1, 7, 11, 2, 7, 15, 8, 13 \rangle$. Use the notes discussed in class as a guide.

Sol) Given that $A = \{3, 1, 7, 11, 2, 7, 15, 8, 13\}$

	1	2	3	4	5	6	7	8	9
A	3	1	7	11	2	7	15	8	13
	p								r

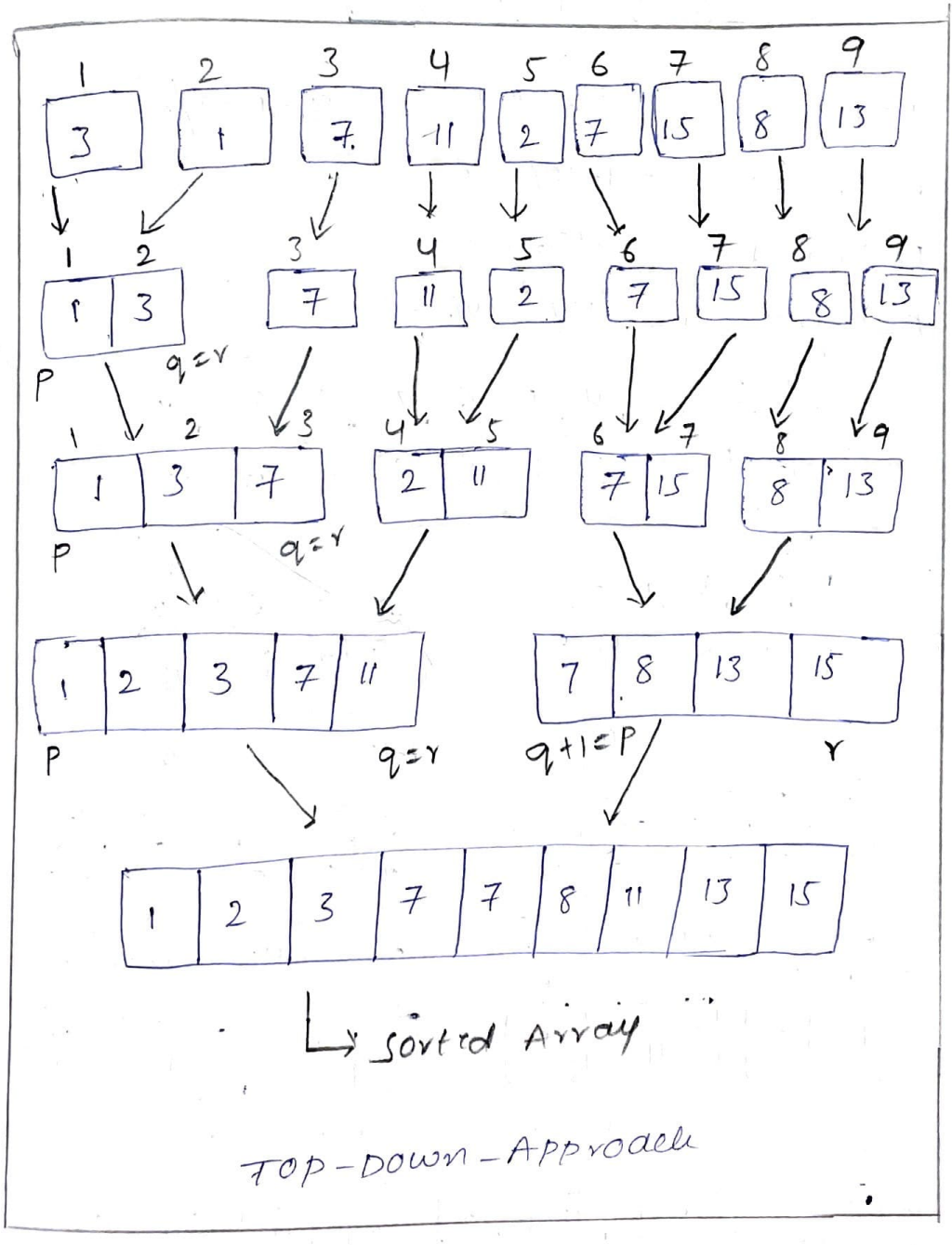
$p=1, r=9 \Rightarrow 1<9 = yes$

As we know that $q = \left(\frac{p+r}{2}\right)$ (∵ from Algorithm)

$p=1, r=9 \Rightarrow q = \frac{1+9}{2} = 10/2 = 5$

$p=1, r=5, q = 1+5/2 = 6/2 = 3$

$p=1, r=3, q = 1+3/2 = 4/2 = 2$



TOP-DOWN-Approach

⑥ Use the bottom-up approach to illustrate the operations of merge sort on the array $A = \langle 3, 5, 1, 11, 2, 7, 15, 8, 13, 17 \rangle$. Use the notes discussed in class as a guide.

Sol) Given Array: $A = \{3, 5, 1, 11, 2, 7, 15, 8, 13, 1\}$

3 5 1 11 2 7 15 8 13 1

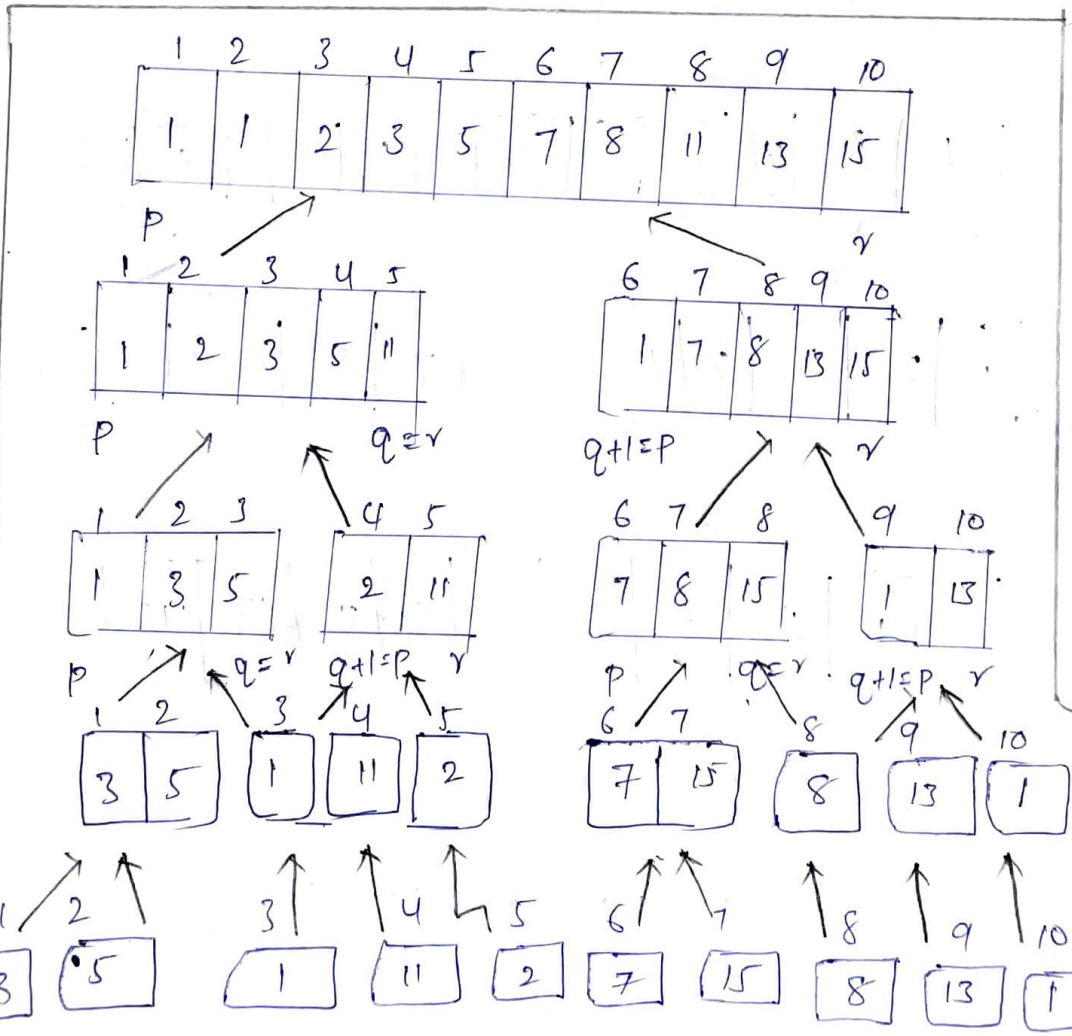
$q = p + r/2$

$p=1, r=10 \Rightarrow 1 < 10 \Rightarrow$ yes

$p=1, r=10, q = 1+10/2 = 5$

$p=1, r=5, q = 1+5/2 = 3$

$p=1, r=3, q = 1+3/2 = 4/2 = 2$



(7) Express the function $\frac{n^3}{10} - 2n^2 - 4n + 1$ in terms of θ - Notation.

Sol) To represent the any function in terms of θ, it can be represented as highest power of n

→ In the given function highest power is 3

∴ So, it can be represented as $\theta(n^3)$

As $T(n) = an + b = \theta(n)$ where a, b constants

$T(n) = an^2 + bn + c = \theta(n^2)$ where a, b, c are constants

Similarly,

$T(n) = n^3/10 - 2n^2 - 4n + 1 = \theta(n^3)$

where $1/10, 2, 4, 1$ are constants.

Another way of solving:

$$f(n) = \frac{n^3}{10} - 2n^2 - 4n + 1$$

let $f(n) = \theta(g(n))$

then there must exist c_1, c_2, n_0 such that

$c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$

we can observe that

$$c_1 \leq \frac{f(n)}{g(n)} \leq c_2 \quad \forall\ n > n_0$$

$\Rightarrow \lim_{n \to \infty} \frac{f(n)}{g(n)}$ must be bounded & positive

This is possible only for $g(n) = n^3$ because for other function $\lim_{n \to \infty} \frac{f(n)}{g(n)}$ will either be zero (or) infinity (∞)

Hence, $f(n) = \Theta(n^3)$

$$\frac{n^3}{100} \leq \frac{n^3}{10} - 2n^2 - 4n + 1 \leq n^3 \quad \therefore n_0 = 100$$

we can observe above inequality holds for all $n > n_0$

Hence, $\frac{n^3}{10} - 2n^2 - 4n + 1 = \Theta(n^3)$