Quiz 2.

1.

(a) An algorithm for a problem is optimal if its worst running time, which is in the big-O notation, matches a lower bound of the problem, and only difference between the upper bound and the lower bound of its running time is a constant coefficient. Therefore, the (1) is correct.

(b)

insertion sort	X	n^2.
merge sort	✓	$n \log n$
quick sort	X	
shell sort	X	

(1) The merge sort algorithm is the optimal, while other sort algorithms are not.

(2) The running time of the merge sort is $\Theta(n \log n)$ if there are n elements to be sorted.

2.

(a) John's solution is better.

Because John's solution ($2^{10} n \lg^2 n$) is more accurate than Mary's in terms of the estimated accuracy. ($n^{1.1} \lg n$).

(b) X: $O(n^{\frac{3}{2}}\lg n)$ y: $O(n\lg^5 n)$ Z: $(n^2 \lg\lg n)$

Clearly, Z student's ~~solution~~ algorithm is the best, as the estimated accuracy of Z's solution is more accurate than X's and y's. Hence, the order is:

Z's algorithm $(O(n^2\lg\lg n))$, X's $(o(n^{\frac{3}{2}}\lg n)$, y's $(O(n\lg^5 n))$.

3.

algorithm 1:

		Cost	times
1	$n \leftarrow V_1.\text{size}()$;	C_1	1
2	$m \leftarrow V_2.\text{size}()$;	C_2	1
3	if $m \leq n$ then return error	C_3	1
4	for $i \in \{1, \dots n\}$ do	C_4	$n+1$
5	for $j \in \{i+1, \dots m\}$ do	C_5	$\sum_{i=1}^{n}[(m+1)-(i+1)+1]$
6	$w[i] \leftarrow w[i] + V_1[i] * V_2[j]$	C_6	$\sum_{i=1}^{n}[m-(i+1)+1]$
7	return w.	C_7	1

C_5 and C_6 can be simplified as $\sum_{i=1}^{n}(m-i+1)$, $\sum_{i=1}^{n}(m-i)$

$$T(n) = C_1\cdot 1 + C_2\cdot 1 + C_3\cdot 1 + C_4(n+1) + C_5\cdot\sum_{i=1}^{n}(m-i+1) + C_6\cdot\sum_{i=1}^{n}(m-i) + C_7\cdot 1$$

$$= (C_1+C_2+C_3+C_4+C_7) + C_4\cdot n + C_5\cdot\frac{(m+m-n+1)\cdot n}{2} + C_6\cdot\frac{(m-1+m-n)\cdot n}{2}$$

$$\approx (C_5+C_6)\cdot(-\tfrac{1}{2}n^2) + (C_5+C_6)mn + (C_5-C_6)\tfrac{n}{2} + C_4\cdot n + \dots$$

$+(C_1 + C_2 + C_3 + C_4 + C_5)$

Because there is no best case scenario and worst case scenario. Thus, $T(n) = \Theta(n^2)$.

algorithm 2:

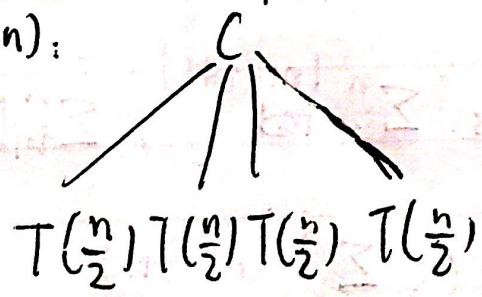
Based on the algorithm, we know that

~~$T(n) = 4T(\frac{n}{2}) + \Theta(1)$~~

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 4T(\frac{n}{2}) + cn & \text{else.} \end{cases}$$

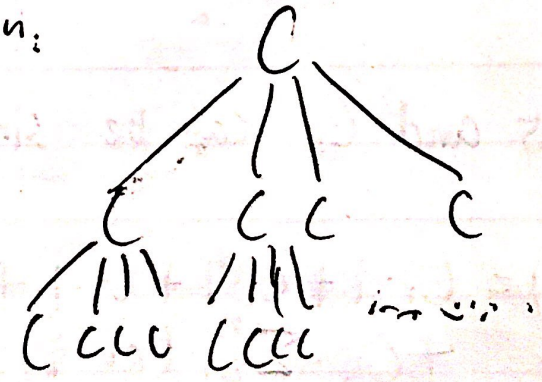
Divide and conquer:

$T(n)$:

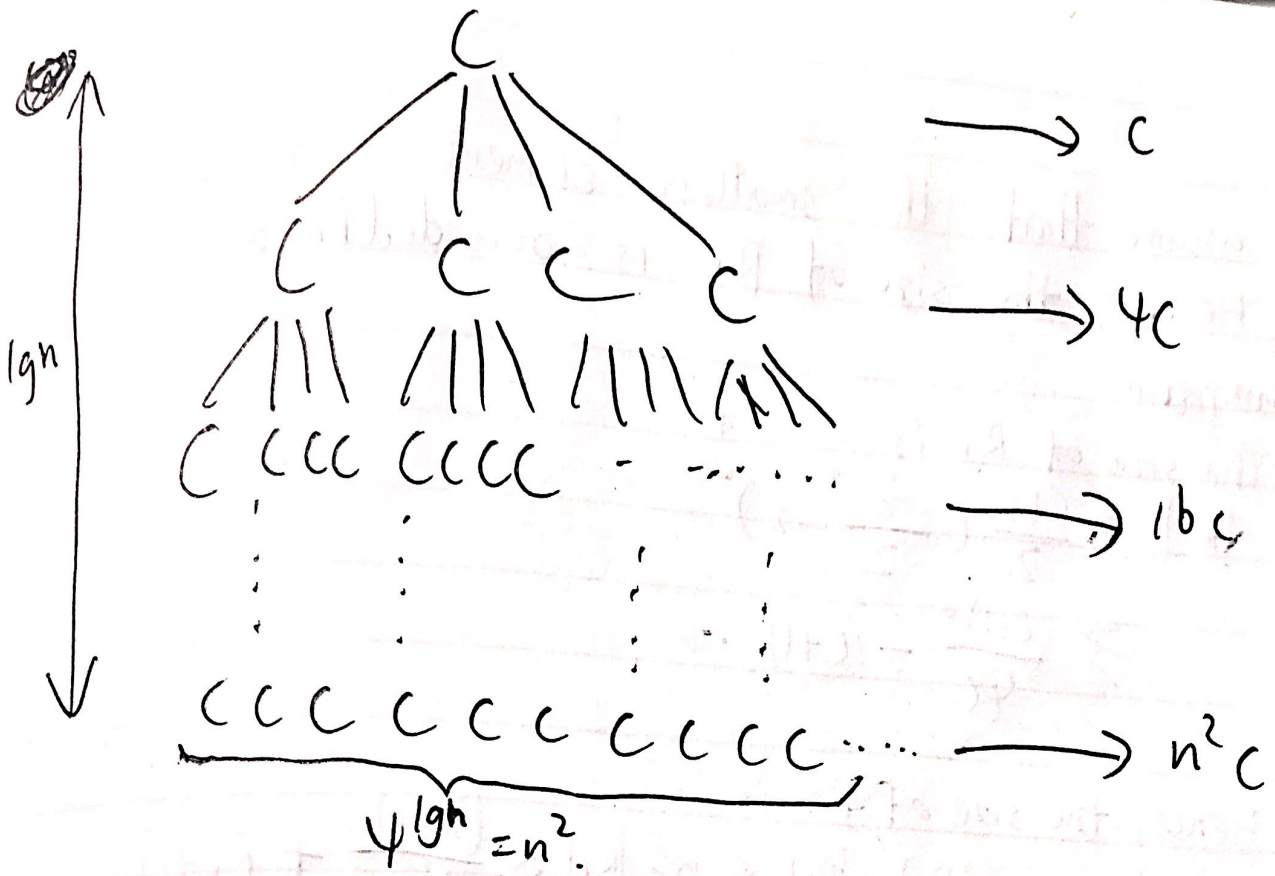


(a) (b)

cn:



(c)



Total is $c + 4c + 16c \cdots + n^2 c = c \cdot \frac{1-4^{lgn}}{1-4}$

$= \frac{n^2-1}{3} \cdot c$

$= \frac{c}{3} n^2 - \frac{1}{3} c$

Therefore, $T(n) = \Theta(n^2)$.

4.

We assume that the smallest element in the ith is in R_3. Hence, the size of R_1 is bounded based on our assumption.

The size of R_1 is:

$$|R_1| \geq \frac{c+1}{2}\left(\frac{\frac{n}{c}}{2} - 2\right)$$

$$\geq \frac{(c+1)n}{4c} - (c+1).$$

Hence, the size of R_3 is

$$|R_3| = n - |R_1| - |R_2| \leq n - |R_1| \leq \frac{(3c-1)n}{4c} + (c+1).$$

Thus, $T_n = \begin{cases} \Theta(1) & \text{if } n \leq 4 \cdot 39 = 156 \\ T\left(\frac{n}{c}\right) + T\left(\frac{(3c-1)n}{4c} + (c+1)\right) + O(n) & \text{if } n > 156 \end{cases}$

We assume that for any $n > 0$, $T(n) \leq bn$ for a existing constant $b > 0$ and for all $n \leq 156$, it exists that $T(n) \leq bn$.

Based on Big-O notation, for $n > n_0$,

$$T(n) \leq T\left(\frac{n}{c}\right) + T\left(\frac{(3c-1)n}{4c} + (c+1)\right) + c_1 n$$

$$\leq b \cdot \frac{n}{c} + b \cdot \frac{(3c-1) \cdot n}{4c} + b(c+1) + c_1 \cdot n$$

$$\leq \frac{bn}{c} + \frac{3nb}{4} - \frac{nb}{4c} + b(c+1) + c_1 \cdot n$$

$$= \frac{3bn}{4c} + \frac{3bn}{4} + bc + b + c_1 \cdot n$$

$$= bn + \left(\frac{3bn}{4c} - \frac{bn}{4} + bc + b + c_1 \cdot n\right)$$

The expression $\frac{3bn}{4c} - \frac{bn}{4} + bc + b + c_1 \cdot n$ is negative if $n > n_0$ and

$$b \geq \frac{4c \cdot c_1 \cdot n}{-3n + nc - 4c^2 - 4c}$$

Thus, for a constant b, $T(n) \leq bn$.