Qn 8.1-2

$$\log(n!) = \cancel{\sum_{k=1}^{n} \log(k)}.\ \log(n \times (n-1) \times \cdots \times 2 \times 1)$$

$$= \log(n) + \log(n-1) + \cdots + \log(2) + \log(1)$$

$$= \sum_{k=1}^{n} \log(k)$$

$$= O(n * (\text{highest term in summation}))$$

$$= O(n \log n).$$

$$\log(n!) = \sum_{k=1}^{n} \log(k)$$

$$= \sum_{k=1}^{n/2} \log(k) + \sum_{k=n/2+1}^{n} \log(k)$$

$$\geq \sum_{k=n/2+1}^{n} \log(k).$$

$$\geq \sum_{k=(n/2+1)}^{n} \log(n/2).$$

$$\geq \sum_{k=(n/2+1)}^{n} (\log n - 1)$$

$$\geq \sum_{k=(n/2+1)}^{n} (\log n - 1)$$

$$\geq \Omega(n(\log n - 1))$$

$$\geq \Omega(n \log n - n)$$

$$\geq \Omega(n \log n).$$

$$\Rightarrow \log(n!) = \underline{\Theta(n \log n)}.$$

Qn 8.1-3 All hail HITLER! HAIL HIM! I ~~TELL~~ FORCE YOU TO HAIL HIM!

fraction = $\left(\frac{1}{2}\right)$

$$\Rightarrow \quad \frac{1}{2} \cdot n! \leq l \leq 2^h.$$

OR

$$h \geq \log\left(\frac{n!}{2}\right)$$

$$\geq \log(n!) - \log 2$$

$$\geq \log(n!)$$

$$= \Omega(n \log n)$$

No

fraction = $\left(\frac{1}{n}\right)$

$$\Rightarrow \quad \frac{1}{n} \cdot n! \leq l \leq 2^h$$

OR

$$h \geq \log((n-1)!)$$

$$= \Omega((n-1)\log(n-1)) \sim \Omega(n \log n) > \Omega(n)$$

No

fraction = $\left(\frac{1}{2^n}\right)$

$$\Rightarrow \quad \frac{1}{2^n} \cdot n! \leq l \leq 2^h$$

OR

$$h \geq \log\left(\frac{n!}{2^n}\right)$$

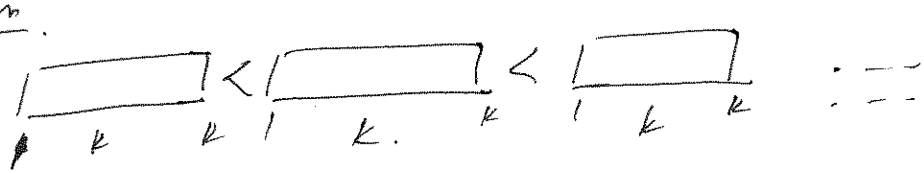
$$= n \log n - n$$

$$= \Omega(n \log n - n)$$

No

Q: 8.1-4

Given.



n/k boxes.

k elements in each subsequence.

n/k subsequences.

$$A[1 \dots k] < A[k+1 \dots 2k] < A[2k+1 \dots 3k] \dots < A\left[\left(\frac{n-1}{k}\right)k+1 \dots n\right]$$

$$\cancel{A\left[\left(\frac{n-k}{k}\right)k\right.}$$

$$\boxed{A\left[\left(\frac{n}{k}-i\right)k+1 \dots \left(\frac{n}{k}-i+1\right)k\right]} \longrightarrow \text{sorted. (each subsequence is sorted)}$$

Sort separately, since the subsequences are already sorted.

Then
$$T(n) \geq \sum_{i=1}^{n/k} k \log k$$
$$\geq \Omega\left(\frac{n}{k} \cdot k \log k\right) = \Omega(n \log k)$$

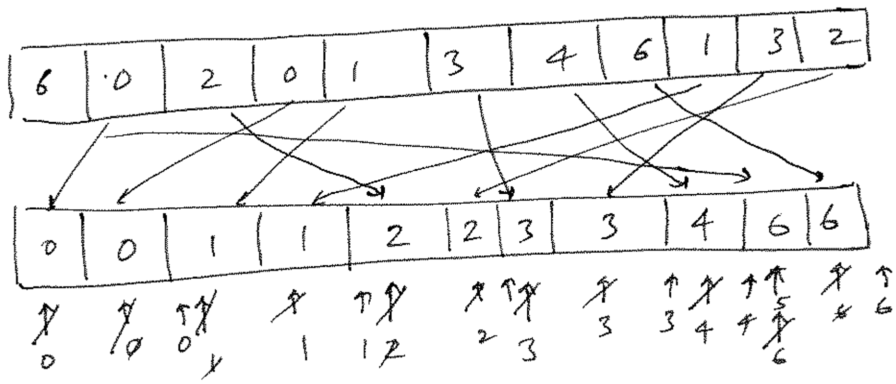
Qn 8.2-1

Array

6	0	2	0	1	3	4	6	1	3	2

$k = 6$, $n = 11$.

	0	1	2	3	4	5	6
~~Next~~ Count Values	2	2	2	2	1	0	2



Sorted

$\underline{Qn}$ $\underline{8.2\text{-}3}$

Counting Sort

let $C[0..k]$ be new array

for $i = 0$ to k.

$C[i] = 0$

for $j = 1$ to n

$C[A[j]]$ ++.

~~for~~ $i = 1$ to k

$C[i] = C[i] + C[i-1]$

for $j = 1$ to n.

$B[C[A[j]]] = A[j]$

$C[A[j]] = C[A[j]] - 1$

Stable → $\underline{\text{No}}$

Still works properly → $\underline{\text{Yes}}$

Qn 8.2-4

FallInRange (a, b, A, n, k)

```
for i = 1 to k                        ] O(k)
    C[i] = 0

for i = 1 to n                        ] O(n)
    C[A[i]]++.

for i = 1 to k                        ] O(k)
    C[i] = C[i] + C[i-1]
    if C[i] > a.
        count ++

return count.
```

O(n+k).

Qn 8.3-1

~~COW~~	SEA	~~T~~AB	BAR	BAR
~~DOG~~	TEA	~~BAR~~	BIG	BIG
~~SEA~~	MOB	EAR	BOX	BOX
~~RUG~~	~~TAB~~	TAR	~~EAR~~	COW
~~ROW~~	DOG	SEA	DIG	DIG
~~MOB~~	RUG	TEA	DOG	DOG
~~BOX~~	DIG	DIG		EAR
~~TAB~~	BIG	~~BIG~~		FOX
~~BAR~~	~~BAR~~	MOB		MOB
~~EAR~~	~~EAR~~	DOG		NOW
~~TAR~~	~~TAR~~	~~COW~~		ROW
~~DIG~~	COW	ROW		RUG
~~BIG~~	ROW	NOW		SEA
~~TEA~~	NOW	~~BOX~~		TAB
~~NOW~~	BOX	FOX		TAR
~~FOX~~	FOX	RUG		TEA.

Qn 8.3-2

Insertion Sort :

Merge Sort : Yes, if '$\leq$' used.

Heap Sort : No.

Quick sort : No.

Simple scheme : use '$\leq$' in merging!

Qn Correctness of Radix Sort using loop invariants

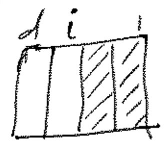
RadixSort(A, d)

for i = 1 to d.

stable sort(A,i)

Post : A is sorted

Loop Invariant :



for $k < i$: column 1 ... $i-1$ is sorted.

Initialisation: $i = 1$: for $k < i \Rightarrow k = 0 \Rightarrow \phi$ is sorted, True!

Maintainance: i : for $k < i \Rightarrow 1 < k < i \Rightarrow$

column 1 is sorted, using a stable sort
$\Rightarrow$ the order is kept the same during the sort.

column 2 is sorted with stable sort
$\Rightarrow$ the order of sort for column 1 remains the same and the new 2 columns get sorted with a stable sort on column 2.

⋮

column k is sorted + stable sort keeps the order same for all the parsed columns

Termination: $i \geq d$ or $i = d+1$

$\dot{0}$ ~~for $k < i$~~

columns $1 \ldots i-1$ sorted

$\Rightarrow$ $1 \ldots d$ sorted!

$\Rightarrow$ A sorted!

Qn 8.3-4

Sort n integers in the range $0 \ldots n^3-1$ in $O(n)$ time.

Sortn (A, n)

~~d~~= 3 * n.length.

RadixSort (A, d)

Qn 8.3-5

$d(n+k)$

Qn 8.4-1

.79
.13
.16
.64
.39
.20
.89
.53
.17
.42

0	1	2	3	4	5	6	7	8
	↓	↓	↓	↓	↓	↓	↓	↓
	0.13	.20	.39	0.42	.53	.64	.79	.89
	.16							
	.17							

.13	.16	.17	.20	.39	.42	.53	.64	.79	.89

Qn 8.4-2

Worst case : $\Theta(n^2)$ why?

↓

when all the numbers fall into a single bucket! OR Only one bucket holds all elements

then insertion sort takes $O(n^2)$ time

Simple change: Set a threshold for insertion sort. So, for a value of numbers in bucket $= k$.

if $k <$ ~~threshold~~ threshold value then insertion sort

else Merge Sort.

Qn 33.3-1

Vertices of CH(Q) = All vertices possible by every pair of nodes in graph Q. to make a convex hull.

→ One possible vertex is between p_1 and p_m.

Hence, in CH(Q).

Qn 33.3-2

Model of computation supports addition, comparison and multiplication.

~~Prove~~ Given lower bound for sorting = ~~O~~ $\Omega(n \log n)$.

lower bound for sorting = $\Omega(n \log n)$.

Qn 8-2

$k=1$ $[0 .. k]$

(a) CountingSort(A, n, k)

- stable sort
- $O(n+k)$ $k=1$
$= O(n)$.

$C[\,] \to$ length $= 2$ $=$ const

$B[\,] \to$ length $=$ A.length.

```
for i = 0 to 1
    C[i] = 0
for i = 0 to n
    C[A[i]]++

for i = 0 to k
    C[i] = C[i-1] + C[i] + 1

for i = 0 to n
    B[C[A[i]]] = A[i]
    C[A[i]]++.

return B.
```

(b)

Qn 8-4

(a)

redjugs → ϕ to n
blue jugs → 1 to n

for i = 1 to n
    for j = 1 to n
        if redjug[i] == ~~red~~ bluejug[j]
            return i, j.

(b) comparison sort for every one ~~max~~ min comparisons are log n.

⇒ for n → $\Omega(n \log n)$.

(c) QuickSort → randomized algorithm.