Exercise sheet 7

Question 7.1

1.

loop1

0	1	2	3
0	0	0	0

loop2

0	1	2	3
2	3	1	1

loop3

0	1	2	3
2	5	6	7

2.

B

1	2	3	4	5	6	7
				1		

C

0	1	2	3
2	4	6	7

B

1	2	3	4	5	6	7
			1	1		

C

0	1	2	3
2	3	6	7

B

1	2	3	4	5	6	7
			1	1		3

C

0	1	2	3
2	3	6	6

B

1	2	3	4	5	6	7
		1	1	1		3

C

0	1	2	3
2	2	6	6

B

1	2	3	4	5	6	7
	0	1	1	1		3

C

0	1	2	3
1	2	6	6

B

1	2	3	4	5	6	7
	0	1	1	1	2	3

C

0	1	2	3
1	2	5	6

B

1	2	3	4	5	6	7
0	0	1	1	1	2	3

C

0	1	2	3
0	2	5	6

Question T.2

loop invarient: after ith iteration, C[j] (0≤j≤i) is the number of elements less than or equal to j.

Initialization: Before the first iteration, C[0] is the number of elements that are 0. Then after the first iteration, C[1] is the number of elements that are 0 or 1, and C[0] holds the same.

Maintanence: After the ith ireration, we assume the loop invarient holds. Then for the (i+1)th iteration, C[i+1] = C[i+1] + C[i] where C[i] is the number of elements less than or equal to i C[i+1] is the number of elements equal to (i+1) before iteration. Thus c[i+1] becomes to the number of elements less than or equal to (i+1).

Termination: After the last iteration, all c[j] (0≤j≤k) is the number of elements less than or equal to j.

The loop invarient holds.

Question T.3

will not be stable but will sort the numbers.

① justify not stable

from 1 to A.length, A[i] = A[j] and i<j; in the for loop, we first decide where A[i] should be placed, A[i] will be placed at B[C[A[i]]], after that C[A[i]] will decrease.

then in the array B[1 ... A.length], A[i] will be put behind A[j]

② justify sort

for every element A[i], it will be placed at B[C[A[i]]], where C[A[i]] is the number less than or equal to A[i]. It still ensures that in array B[1 ... A.length],

elements before B[C[A[i]]] is not larger than it, and elements behind B[C[A[i]]] is not smaller than it.

Question T.4

We can first count how many elements that are not larger than (a-1) and that are not larger than b, and calculate the number of elements in [a:b], which takes O(1).

Do count work takes O(n+k):

① let C[0~k] be a new array, with every element is 0

② count every number from 0~k occurs how many times and store it in C[i]

③ process C[0…k] make C[i] be the number of elements less than or equal to i.

Then calculate the number of elements in [a:b], it takes O(1)

④ calculate C[b] - C[a-1]

考虑 a=0

if a=0 calculate C[b]

Question T.5

① compare last character ② compare second character ③ compare first character

MOB	TAB	BAR
TAB	BAR	BIG
DOG	CAR	BOX
TUG	TAR	CAR
PIG	PIG	COW
BIG	BIG	DOG
BAR	MOB	MOB
CAR	DOG	PIG
TAR	COW	ROW
COW	ROW	TAB
ROW	WOW	TAR
WOW	BOX	TUG
BOX	TUG	WOW

Question T.6

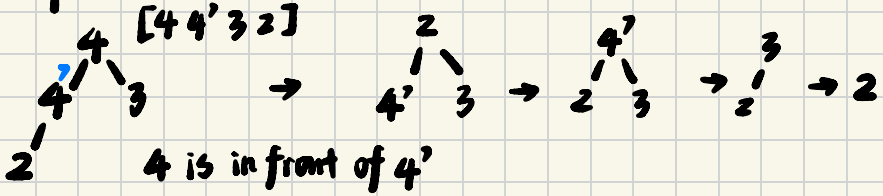
Insertion Sort : stable

Assume A[1 ... j-1] is sorted, and insert A[j] into the sequence. If there exist A[k] that is equal to A[j] k<j, and k is the largest number we can get for A[k]=A[j], then A[k+1 ... j-1] will be moved to A[k+2...j] and A[j] will be placed at A[k+1], which holds the stability.

Merge Sort : Stable

Assume $L[1 \cdots q]$, $R[q+1 \cdots r]$ is sorted on their own, and L is in left of R. Then do merge, if $L[i] = R[j]$ we will let $L[i]$ be placed in the new array first, then place $R[j]$, which holds the stability.

HeapSort : not stable



we get [2 3 4' 4], which shows heapsort is not stable.

QuickSort : not stable

[3 3' 2 1] 3 is in front of 3'

→ [1 3' 2 3] → [1 3' 2 3] → [1 2 3' 3]

which shows quicksort is not stable