Queues

Queue - FIFO : First to first out
: Scheduling in the OS
: CPU Scheduling

Data: 3 12 6 14 7 8 3 12 1
2 4

X	X	X	14	7	8	3	12

Functions on Queue

1) Enqueue(x)
2) Dequeue()
3) size()
4) front(), rear() → Element we are about to delete : 3
5) Empty() → Element which we just inserted : 12

Implementation of Queue

	0	1	2	3	4	5	6	7	8	9
int arr[10] =	X	X	2	15	40					
	f				r					

int f=0, r=-1

Enqueue(int x) {
++r;
arr[r] = x
}

int rear() {
return arr[r]
}

→ Index recently added element

dequeue() {
++f;
}

int front() {
return arr[f]
}

arr[10]	Front = 0, Rear = -1	
3 Insertions	Front: 0	Rear: 2
4 Insertions	0	6
2 Deletions	2	6

Front: Index of element you are about to delete

Rear: Index of element which is recently inserted

int size() {
return r-f+1
}

bool Empty() {
if (size() == 0)
return True
return False
}

Note: In all these functions take care of overflow & underflow conditions

At start
r=-1, f=0
size = -1-0+1 = 0

Doubt:
f=0, r=-1

3 Insertions	f: 0	r: 2
2 Deletions	2	2

size = 1

Pros/Cons: int arr[10]

f = 0, r=-1

	f	r
70 Enqueue	0	69
50 Dequeue	50	69
40 Enqueue	50	109

→ 70 Inserts
- 50 Delete
+ 40 Delete
100 Slots

Size & memory of f

Space

Circular Queue

int arr[10]

0	1	2	3	4	5	6	7	8	9
X	X	X	X	14	11	4	-1	3	20
f								r	

f = 0, r=-1
Insert: 10, 9, 4, 14
f = 0, r = 3
Delete, 2 times
f = 2, r = 3
Insert: 5, 11, 14, -1, 3, 20
f = 2, r = 7
Delete 3 times
Insert: 25, 39
Index: 0 1
10 10
r=10 r=11

void enque(x) {
r++;
arr[r%N] = x
}

void deque() {
f++;
}

int rear() {
return arr[r%N];
}

int front() {
return arr[f%N]
}

int size() {
return r-f+1
}

r=0, f=-1, arr[100]

	f	r
Enque: 70	0	69
Deque: 50	50	69
Enque: 40	50	109

size = 109 - 50 + 1
= 80 s

→ Queue with Array LIFO [Size is not fixed]
Insert at back: O(1)
Delete from start: O(n)

→ Queues with linked list:
All operations: O(1)
Size is not fixed as well

→ Stack & Queue in Java:
C++
→ queue <int> q;
rear
front
enque
deque
size
O(1) time
C++ / Java / Python
[Priority Queue]

2Q) Given a Queue with data, reverse the data with stack.

Approach
1) 8 - 5
2) 5 - 8

5 6 11 -1 12

After reversing

12 -1 11 6 5

12
-1
11
6
5

12 -1 11 6 5

3Q) Given N elements to que, reverse the first k elements to Queue, Using Stack.

N = 7
K = 3

5 10 9 8 14 19 21

9 10 5 8 14 19 21

Approach 1:

5 10 9 8 14 19 21

Step1: All Queue to Stack1

21
19
14
8
9
10
5

Step2: N-k from Stack1 → Stack2

9
10
5

8
14
19
21

Step3: K from Stack1 → que
N-k from Stack2 → que

9 10 5 8 14 19 21

SC: O(N)
TC: O(N)

Approach 2: Using Single Stack

5 10 9 8 14 19 21

→ Using Stack
Reverse Whole q to Step1

21 19 14 8 9 10 5

→ Deque n-k from Stack 1 Stack 2

8
14
19
21

9 10 5

→ Stack to queue all elements Stack 3

9 10 5 8 14 19 21

Approach 2:

5 10 9 8 14 19 21

Step1:
Insert k Elements to Stack

8 14 19 21

9
10
5

Step2:
Insert k Elements in Queue

9 14 19 21 9 10 5

reversed

Note: Just they should come at start

Dequeue & Enqueue together N-k times

9 10 5 8 14 19 21

SC: O(k)
TC: O(N)

4Q) Generate k th number in Binary digits only 1 & 2

k = 10 | 1 2 11 12 21 22 111 112 121 122 211

Approach 1:
Using backtracking
1) Get all 1 digit Number
2) Get all 2 digit Number
3) Get all 3 digit Number
ans.size() ≥ k stop

Approach:
1 digit: 1 2
2 digit: 11 12 21 22
Step: 111 112 121 122 211 212 221 222

1 2 11 12 21 22 111 112 121 122

Delete front element: 1
Insert: 11 & 12 at rear
If que.size = k we stop
All Nos: 1 Size: 2 k=10

If No of elements in que = k : stop
Out = k

At Every Iteration:
→ we delete an Element
we Insert 2 Element
Overall we Insert 1 new Element

For a given Iteration we Insert 1 new Element.

Whatever they we repeat process K times

TC: O(k)
SC: O(k)

Note: We store data before & strings concatenate because may take

5th Q: Generate k th palindromic number using digits 1,2
Note: We only need to generate even length palindromes

k = 10 :

length: 11 22
length: 1111 1221 2112 2222
length: 111111 112211

11 22
1111 1221 2112 2222
111111 112211

1111 2112 1221 2222

1111 2112 1221 2222

And 1 or both ends 2 in between
we want get cancel over it

1111 1221 2112 2222

If we found k element we stop

6th Q: Generate k th palindromic Binary string [Consider base twice slices]

k = 10 :

1	2	3	4	5	6	7	8	9	10
0	1	11	101	111	1001	1111	10001	10101	11011

length: 11
length: 101 111
length: 1001 1111
length: 10001 10101 11011 11111
length: 100001 100111 111001 111111
length: 1001001 1001001

→ Functions
Time - Cost: Adding in middle
→ Odd sizes: 011
→ Odd - Freq: Add the mid character once again at center