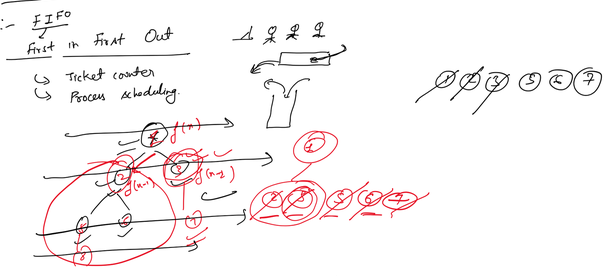
Queues and Deques

Wednesday, 31 March 2021 4:56 PM

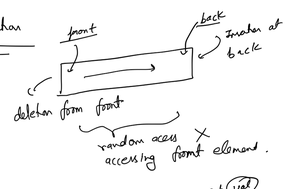
Arrays
LL
Stacks → LIFO

Queue :- FIFO
First In First Out

↳ Ticket counter
↳ Process scheduling

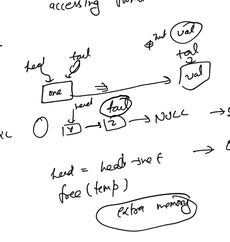


Implementation



random access ✗
accessing front element.

Linked List



Insertion → Enqueue
deletion front → dequeue

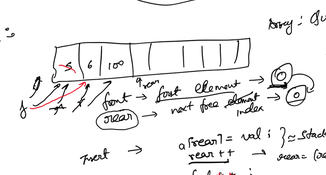
queue
front back

Pros: Dynamic → O(1)

head = head->next
free(temp)
Return memory → O(1)

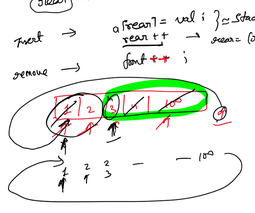
Array :-

Array : Queue

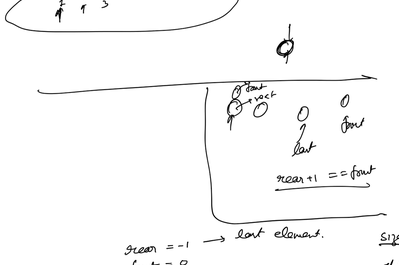


front → front element
rear → next free element index

Insert → a[rear] = val ; rear++ → rear = (rear+1)%N;
remove → front++ ;



rear − front

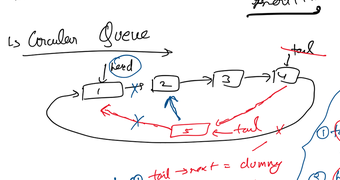


rear = -1
front = 0 → first element.

Insert → rear = (rear+1)%N
a[rear] = val.
size++ ;

size = 0
if (rear+1)%N == front && rear != -1) ↳ full.

delete → front = (front+1)%N
size--



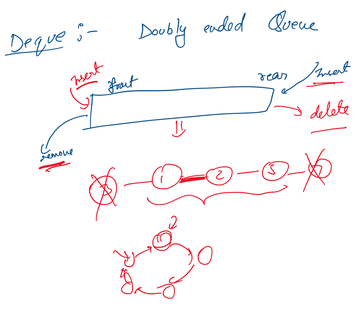
N = 4
Insert(7)
Insert(9)
remove()
Insert(10)
Insert(12)

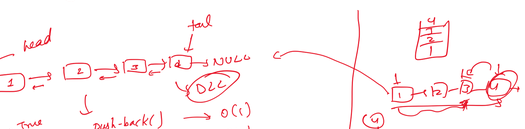
↳ **Circular Queue**

temp = head
① tail->next = head->next
② head = head->next
③ free(temp)

① tail->next = dummy
② dummy->next = temp
③ tail = dummy

Deque :- Doubly ended Queue



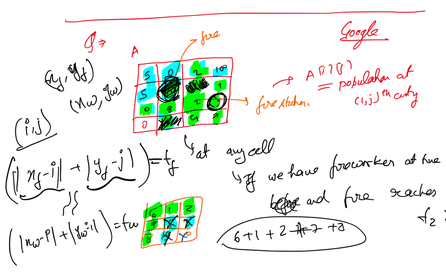


Linked List

O(1) → True
O(N) → False

push-back() → O(1)
pop-front() → O(1)
push-front() → O(1)
pop-back() → O(N)

Google



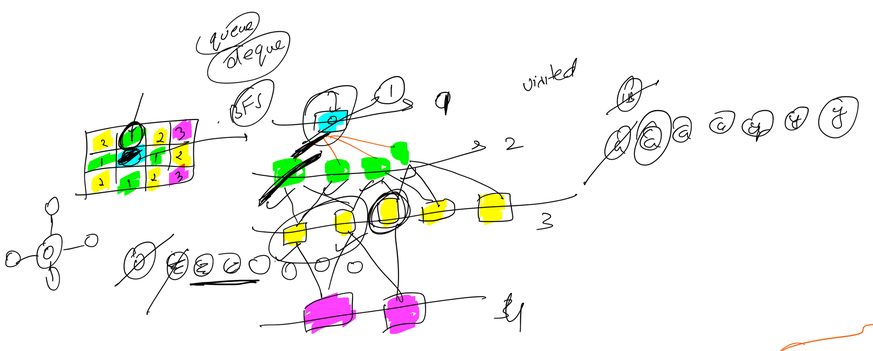
$\left(|x_f - i| + |y_f - j|\right) = f_f$ at any cell

if we have fireworker at fire f_1 and fire searches f_2

$f_2 \geq f_1$

$f_1 = \frac{\text{Distance}_{fire}}{\text{Speed}_{fire}} = \text{Distance}_{fire}$

$f_2 = \frac{\text{Distance}_{fire}}{\text{Speed}_{fire}}$



2018 → ICPC ONLINE ROUND
↳ Q3

visited[N][N] = { 0, 0, 0 ; 0 0 0 0 }

queue < pair < pair<int,int>, int> > q ;

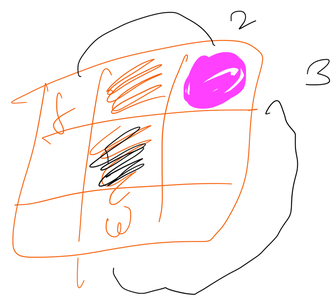
q.push({{x_f, y_f}, 1});
visited[x_f][y_f] = 1;
while(!q.empty()) {
t = q.front().second
x = q.front().first.first
y = q.front().first.second;
q.pop()
visited[x][y] = 1;
if(valid(x+1, y)) q.push({{x+1, y}, d+1});
}

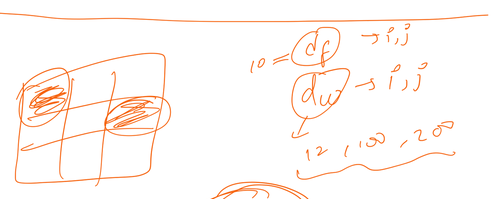
valid(x, y) {
x ≥ 0 x < N y ≥ 0 y < N
visited[x][y] == 0;
not A[x][y] == -1 // not blocked.
}

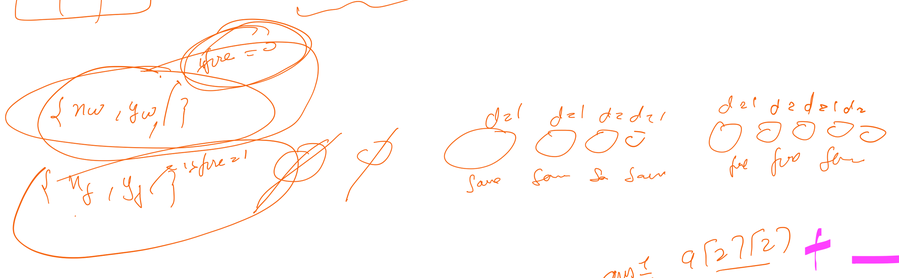
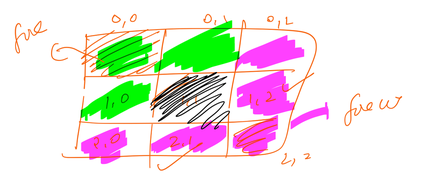
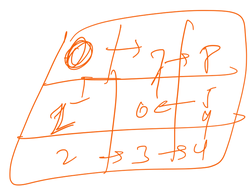
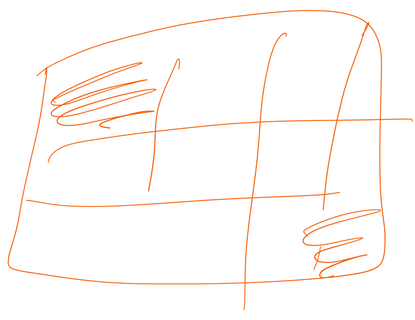
N×M
O(N×M)

{0, 2, d=1}
{0, 1, d=2}
{1, 1, d=2}
{1, 1, d=3}
{0, 0, d=3}

visited[ ][ ] ( )







ans = a[2][2] + —