Rotten oranges

class pair

{

int i,j;

pair(int i,int j)

{

this.i=i;

this.j=j;

}

}

class Solution {

public int orangesRotting(int[][] grid)

{

Queue<pair> q=new LinkedList<>();

int time=0,rotten=0,total=0;

for(int i=0 ;i<grid.length ;i++)

{

for(int j=0 ;j<grid[0].length ;j++)

{

if(grid[i][j]==1 || grid[i][j]==2)

{

total++;

}

if(grid[i][j]==2)

{

q.add(new pair(i,j));

}

}

}

if(total==0)

return 0;

while(!q.isEmpty())

{

int size=q.size();

rotten +=size;

if(rotten==total)

{

return time;

}

time++;

for(int i=0; i<size ;i++)

{

pair p=q.peek();

if(p.j+1 <grid[0].length && grid[p.i][p.j+1]==1)

{

grid[p.i][p.j+1]=2;

q.add(new pair(p.i,p.j+1));

}

if(p.j-1 >=0 && grid[p.i][p.j-1]==1)

{

grid[p.i][p.j-1]=2;

q.add(new pair(p.i,p.j-1));

}

if(p.i+1 <grid.length && grid[p.i+1][p.j]==1)

{

grid[p.i+1][p.j]=2;

q.add(new pair(p.i+1,p.j));

}

if(p.i-1 >=0 && grid[p.i-1][p.j]==1)

{

grid[p.i-1][p.j]=2;

q.add(new pair(p.i-1,p.j));

}

q.poll();

}

}

return -1;

}

}