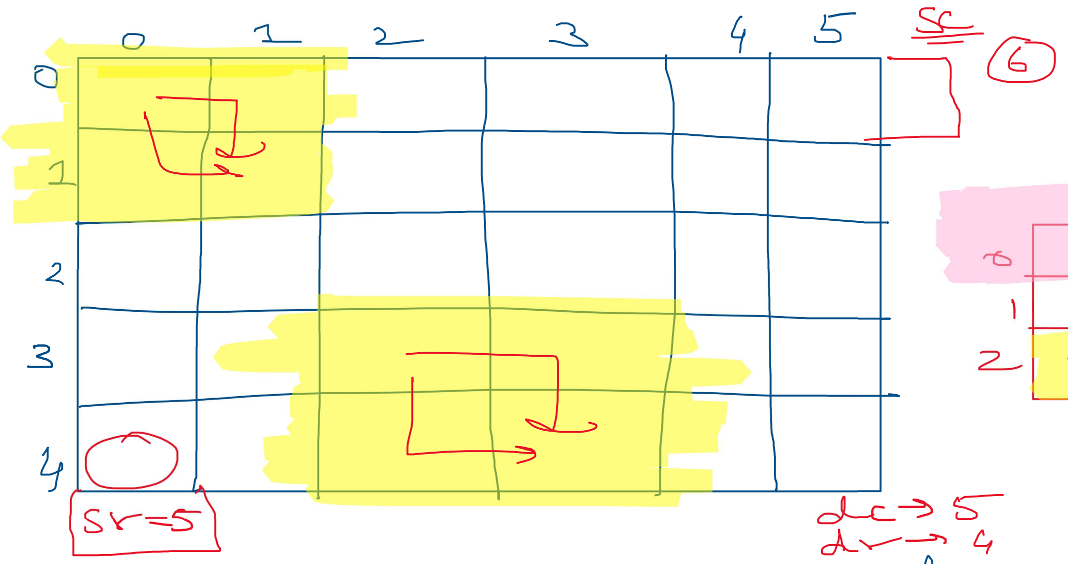
August 15, DP + Quick Sort

15 August 2020 08:21



(5 × 6)

	0	1	2	3
0	10	6	3	1
1	4	3	2	1
2	1	1	1	1

Find the no. of ways to reach end

Start from (0,0)

2 possible movements

$(i, j) \longrightarrow (i, j+1)$

$\downarrow$

$(i+1, j)$

findMazePath(sr, sc, dr, dc)

Base Conditions

1 → if (sr == dr && sc == dc)

{

return 1;

}

```
if (sc > dc) return 0;
if (sr > dr) return 0;
```

No. of ways → Right Jaane ke ways + Neeche Jane ke ways

```
return findMazePath(sr, sc+1, dr, dc) +
       findMazePath(sr+1, sc, dr, dc);
```

	0	1
0	2	1
1	1	

Recursion Tree

(sr, sc)

(sr, sc+1)   (sr+1, sc)

(sr, sc+1)

(sr, sc+2) (sr+1, sc+1)

(sr+2, sc+1) (sr+1, sc+2)

(sr+2, sc) (sr+1, sc+1)

(sr+2, sc+1) (sr+1, sc+2)

Overlapping Sub Problems ✓

Optimal Substructure ✓

Quick Sort

- → Recursive
- → In-Place Algorithm
- → $O(n \log n)$ → Best or Average Case
- → $O(n^2)$ → Worst Case

Given Array

7	2	1	6	8	5	3	4

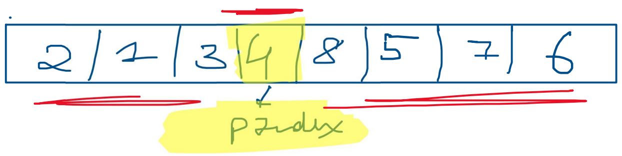
→

Sorted Array

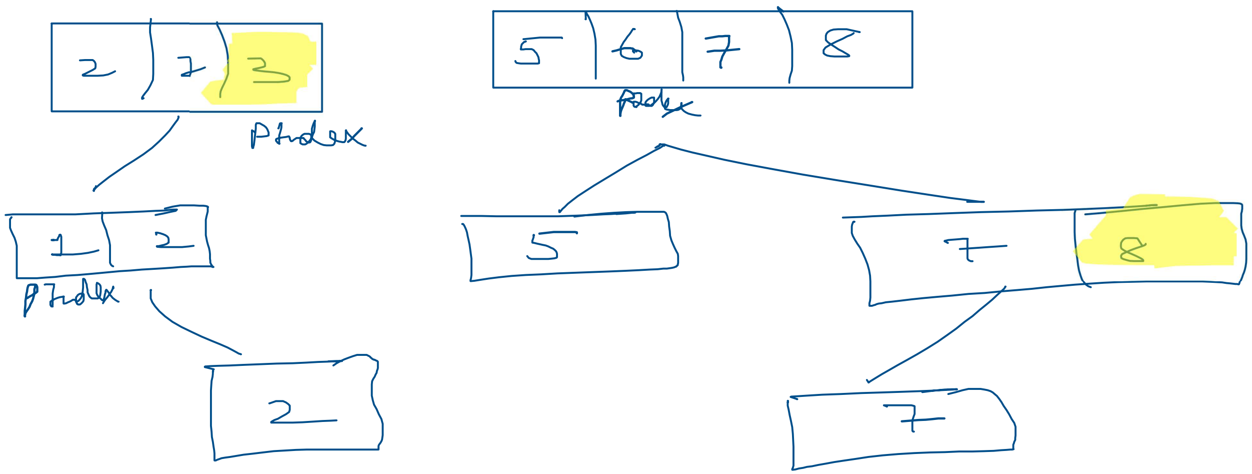
1	2	3	4	5	6	7	8

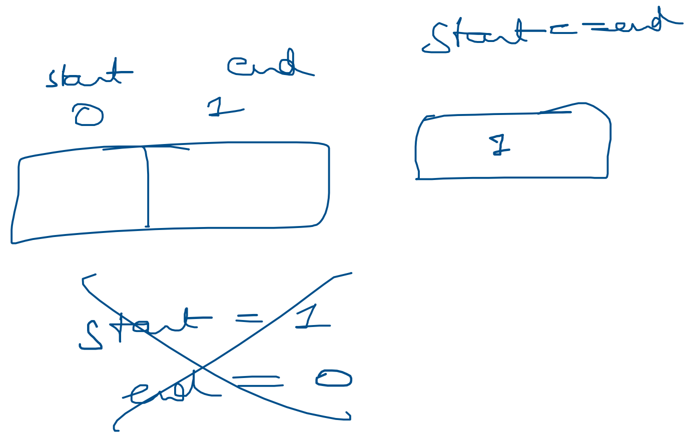
We will pick one element -> pivot
We will keep it on its place or we can say that
We will move elements smaller than it to the left and
elements greater than it to the right.

This process is called partitioning



(0, pIndex - 1) (pIndex + 1, end)






```
void quickSort(A,start,end)
{
    if(start>=end) return;
    pIndex = Partitioning(A,start,end);
    quickSort(A,start,pIndex-1);
    quickSort(A,pIndex+1,end);
}
```

or

```
void quickSort(A,start,end) → T(n)
{
    if(start<end) → c
    {
        pIndex =
        Partitioning(A,start,end);    an + b
        quickSort(A,start,pIndex-1); → T(n/2)
        quickSort(A,pIndex+1,end);   → T(n/2)
    }
}
```

Pivot

i

2	1	3	4	8	5	7	6

pIndex

```
int Partitioning(A,start,end)
{
    int pivot = A[end]; //end is last index not
    length
    pIndex = start;
    for(int i=start; i<=end-1;i++)
    {
        if(A[i]<pivot)
        {
            Swap(A,i,pIndex);
            pIndex++;
        }
    }
    Swap(A,end,pIndex);
    return pIndex;
}
```

→ Make Swap function

$$T(n) = 2\left[T\left(\frac{n}{2}\right)\right] + cn$$

$$\underline{T\left(\frac{n}{2}\right)} = 2T\left(\frac{n}{4}\right) + c\frac{n}{2}$$

$$T(n) = 4T\left(\frac{n}{4}\right) + 2cn$$

$$T(n) = 2^k \times T\left(\frac{n}{2^k}\right) + kcn$$

$$\frac{n}{2^k} = 1 \qquad k = \log n$$

$$T(n) = n \times c + nc \log n$$

$$\rightarrow O(n \log n)$$

If ~~sorted~~ → Already

1	2	3		5	6	7	

$n-1$ → 0

$$T(n) = T(n-1) + c$$

$$T(n-1) = T(n-2) + c$$

$T($