Questions on jump game

01 jump game

Eg 1: arr[] = [2 3 1 0 4] → Yes

Eg 2: arr[] = [1 2 3 1 1 0 2 5] → No not possible.

- array jut has the number then always possible, jumps possible.

Eg: arr[] = [3 2 1 0 4] → No

Eg. [1 2 4 1 1 0 2 5] → Yes

0 1 2 3 4 5 6 7

maxInd = ~~0~~ ~~1~~ ~~3~~ ~~3~~ 3

done ans

```
bool canJump(vector<int>& nums){
    int maxInd = 0;
    for(int i=0; i<nums.size(); i++){
        if(i > maxInd){
            return false;
        }
        maxInd = max(maxInd, i+nums[i]);
    }
    return true;
}
```

02 jump game II

arr[] = [2 3 1 1 4] (indexes 0 1 2 3 4)

min no. of more required to reach at last.

(index, no. of jumps)

F(0,0)

+1 → F(1,1)

+2 → F(2,2)

F(1,1): +1 → F(2,2); +2 → F(3,2); +3 → F(4,2)

F(4,2) → min no. jumps = 2

```
f(ind, jump){
    if(ind >= n-1)
        return jumps;
    mini = INT_MAX;
    for(i=1; i<=arr[ind]; ){
        mini = min(mini, f(ind+i, jump+1));
    }
    return mini;
}
```

- memoization will take

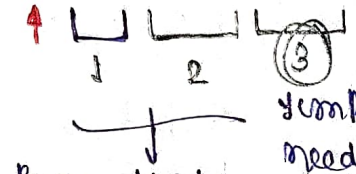
TC = O(N²)

SC = O(N²).

TC = N^N

SC = O(N).

Eg. arr = [2 3 1 4 1 1 1 2]



jumps = ~~0~~ ~~1~~ ~~2~~ (3) Ans

```
Func ( arr) {
  jumps = 0, l = 0, r = 0;
  while ( r < n-1) {
    Farthest = 0
    For ( ind = l → r) {
      Farthest = max (farthest,
                      arr [ind] + i);
    }
    l = r+1;
    r = Farthest;
    jumps = jumps + 1;
  }
  return jumps;
}
```

TC = O(N)

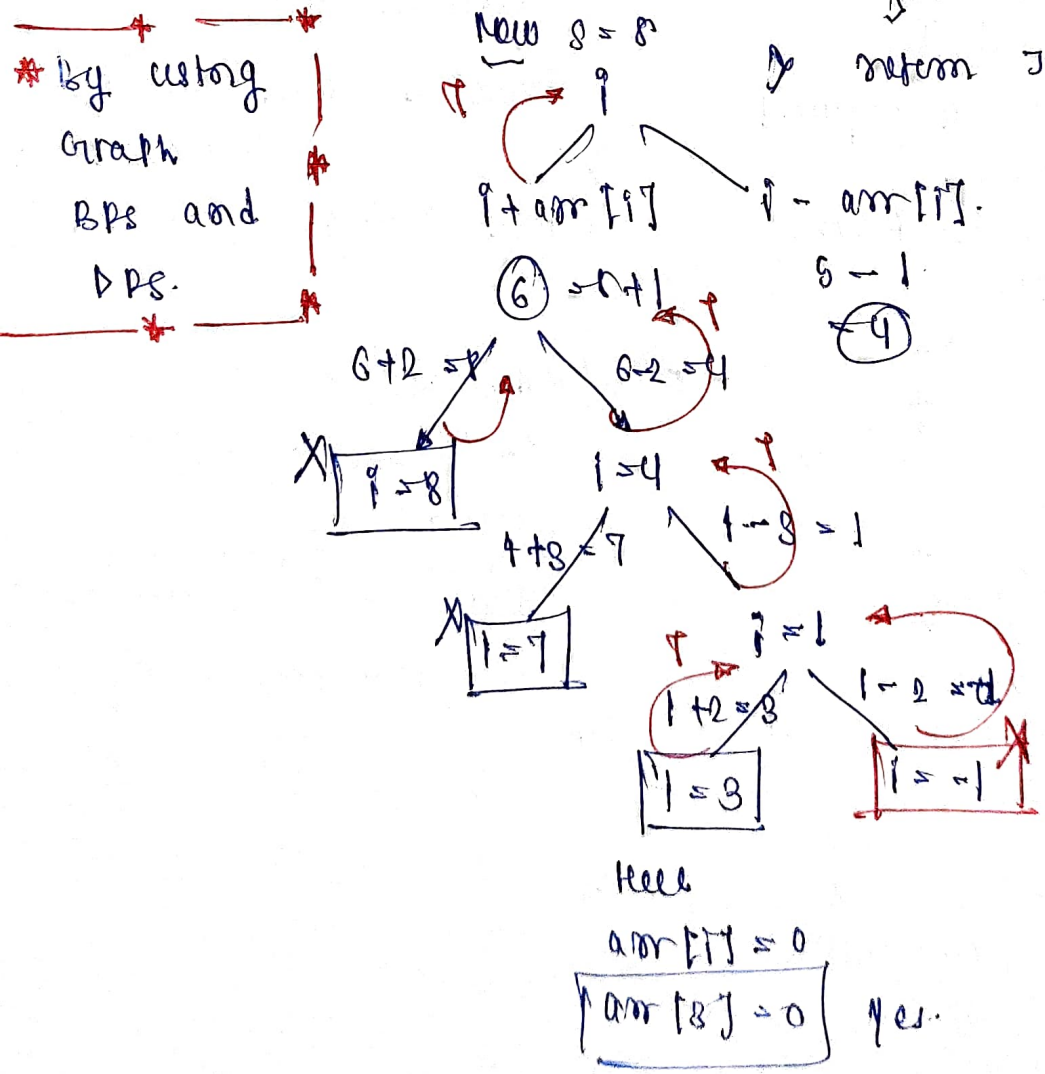
SC = O(1).

03 Jump game III

	0	1	2	3	4	5	6
Eg	4	2	3	0	3	1	2

New S = 5

* By using Graph BFS and DFS.



Here

arr[1] = 0

arr[3] = 0 Yes.

04 Jump game IV

Eg. [100 −23 −23 404 100 23 23 23 3 404]
0 1 2 3 4 5 6 7 8 9

**

0 → 4 → 8 → 9
1 2 3
100 100 404 404.

min steps = 3.

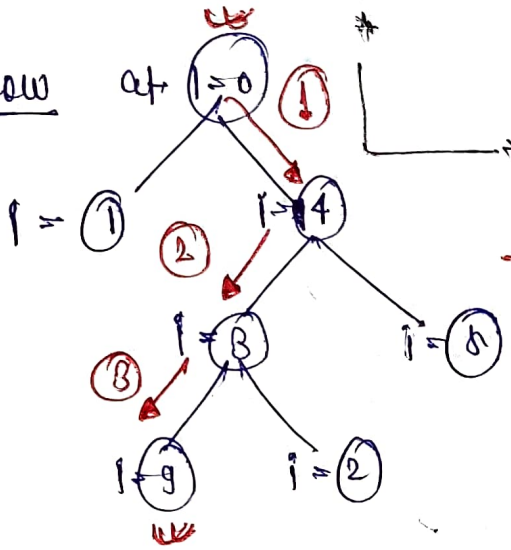
Each step (i)

(arr[i] == arr[j])

Hash map:

- 100 → 0, 4
- −23 → 1, 2
- 404 → 8, 9
- 23 → 5, 6, 7
- 3 → 3

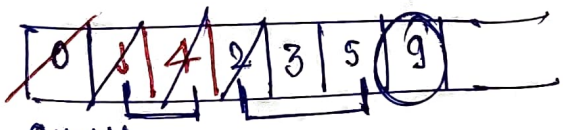
now at i=0 (1)



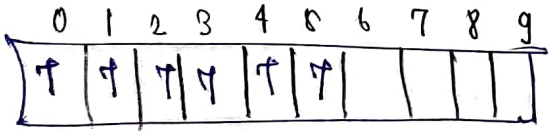
→ Here indexes act as nodes and steps will be edges

→ We have to find shortest path.

We BFS.



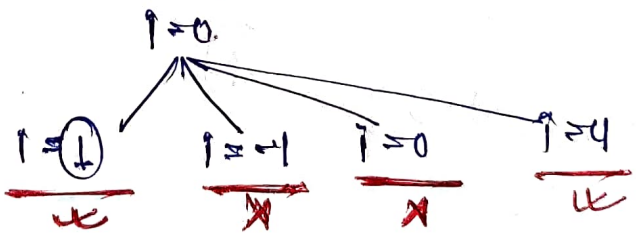
queue



visited

now

node i=0.

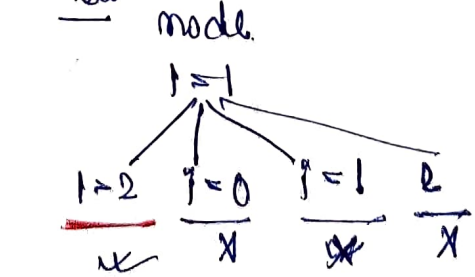


steps = ~~0~~ ~~1~~ ~~2~~ 3

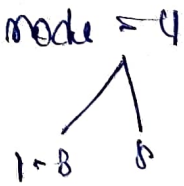
And will be given by BFS.

now

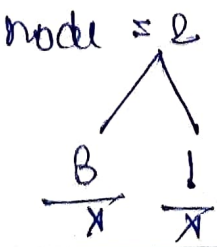
node i=1



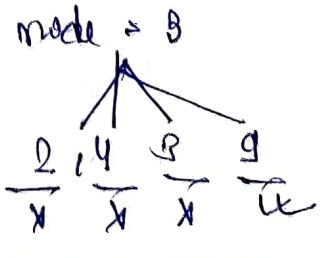
node = 4



node = 2

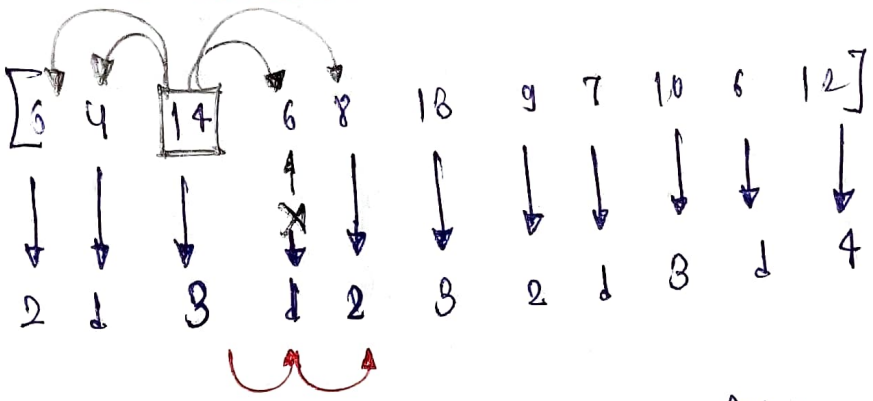


node = 8



TC = O(n)
SC = O(n).

05 jump game - 6



So ans will be 4

jumps

- 14 → 6 → 4 (calculated)
  total = 3.
- 14 → 4
  total = 2
- 14 → 6
  total = 2
- 14 → 8
  total = 3

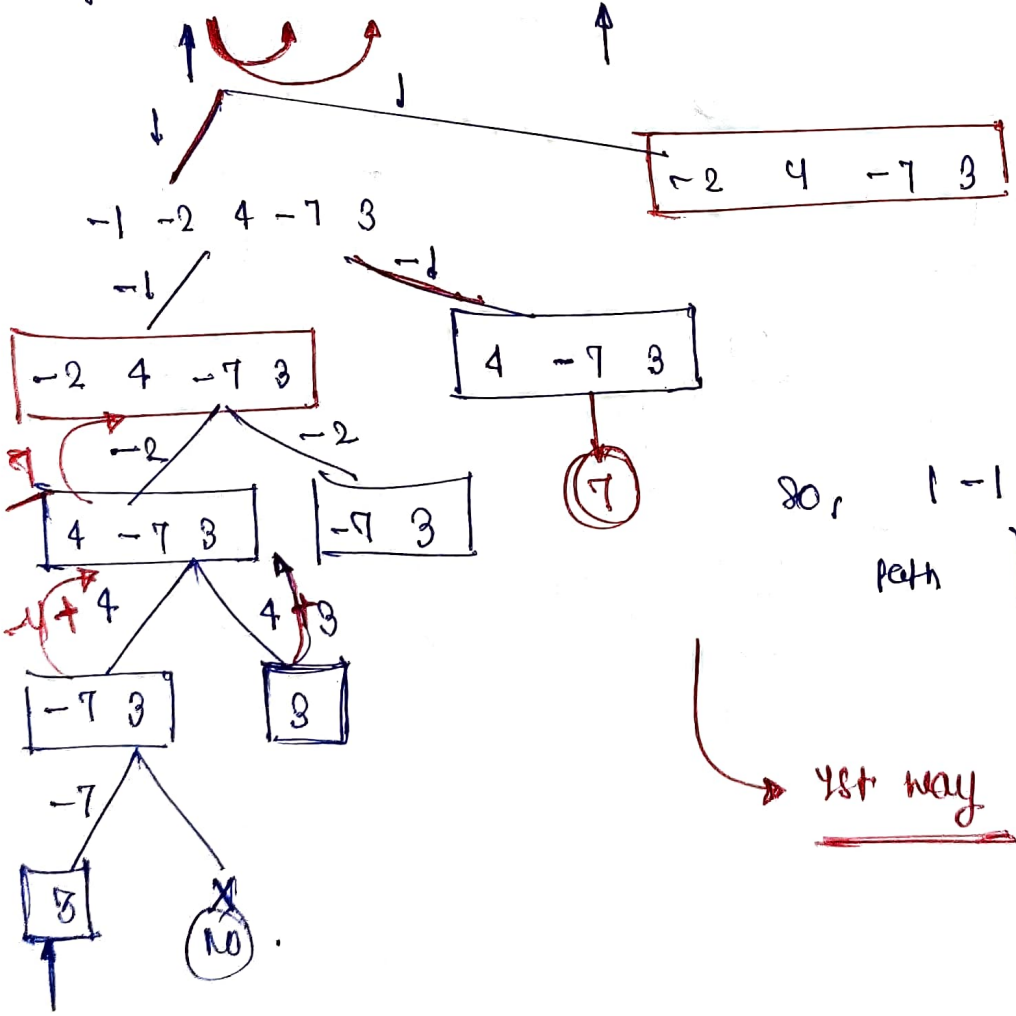
max = 3

new recursion

TC = O(n*k)

06 jump game - 6.

Eg. 1 -1 -2 4 -7 3.



memoization

dp

So, 1 -1 +7 = 0 +7 = 7

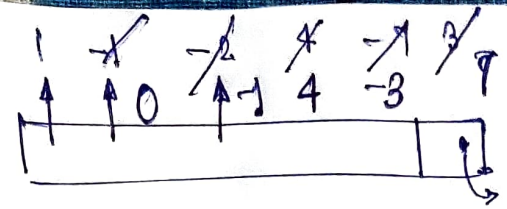
path 1 -1 4 3

1st way

1 -1 +(-2) +4 +(-7) +3

= -2

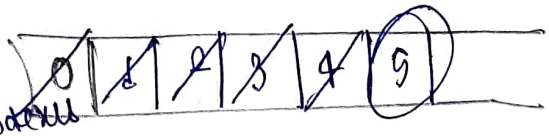
2nd way



last index pe aane ka max score leya hoga.

use deque type

[sliding window maximum].



index deque

$-1+1=0$

$-2+1=-1$

$4+0=4$

$-7+4=-3$

$8+4=1$

- agar arr change hua or usska value zada hai pehle queue wale index se to usko remove kr denge queue se.

07 Jump game 7

Eg. $s = 0\,|\,|\,0\,|\,|\,|\,0$

minJ = 2

maxJ = 3

true.

Eg $s = 0\,|\,|\,0\,|\,|\,|\,0$

minJ = 2

maxJ = 3

false

```
bool canReach(string s, int minJump, int maxJump) {
    int n = s.size();
    vector<bool> dp(n, false);
    dp[0] = true;
    int prev = 0;
    for (int i = 1; i < n; i++) {
        if (i >= minJump) prev = prev + dp[i - minJump];
        if (i > maxJump) prev = prev - dp[i - maxJump - 1];
        dp[i] = (s[i] == '0') && (prev > 0);
    }
    return dp[n-1];
}
```