O(n) ⇐

log(n) BS

★ arr[mid] < arr[hi]

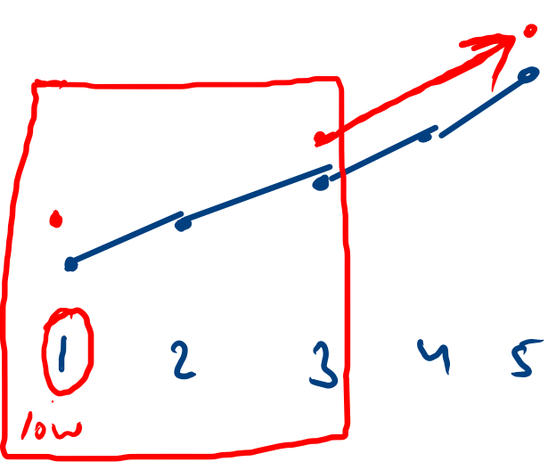
hi = mid

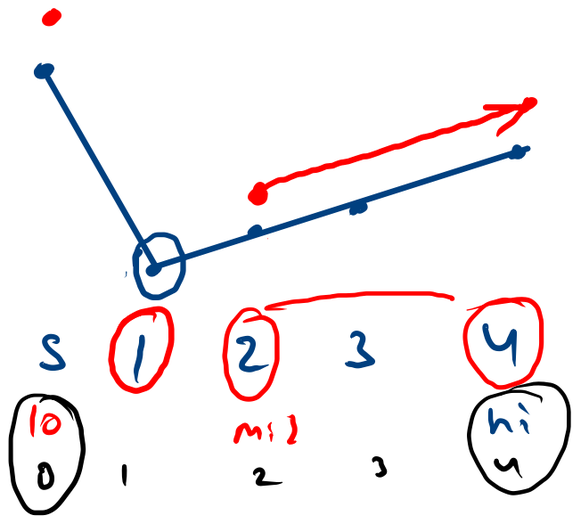
else

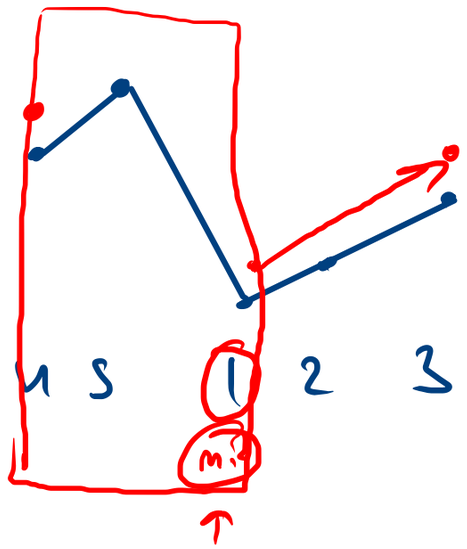
lo = mid+1

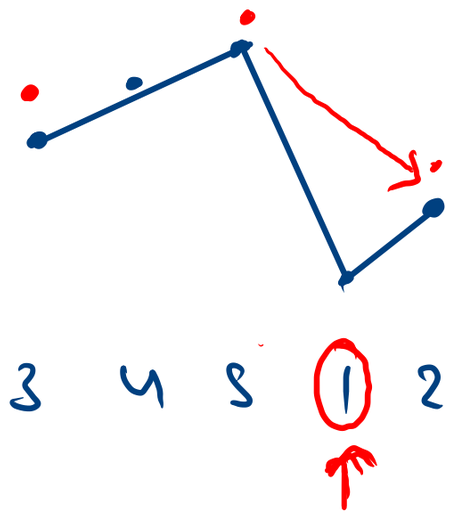
lo .... mid

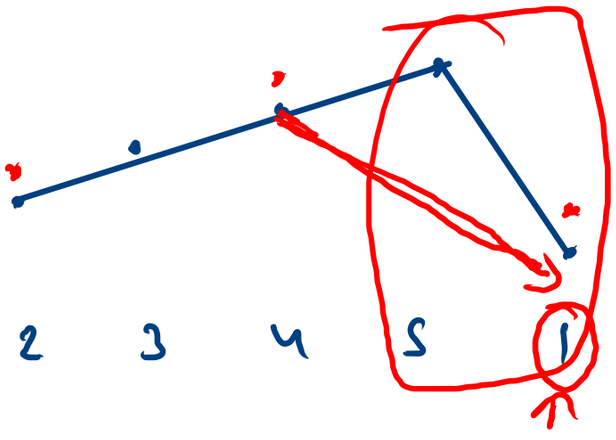
mid+1 .... hi

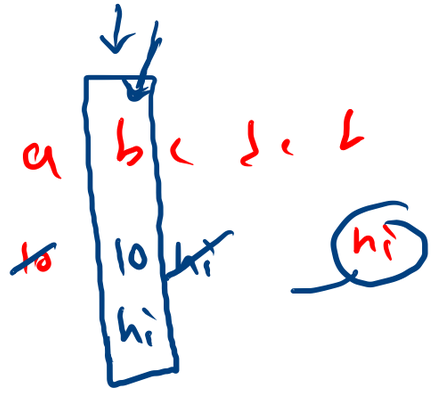




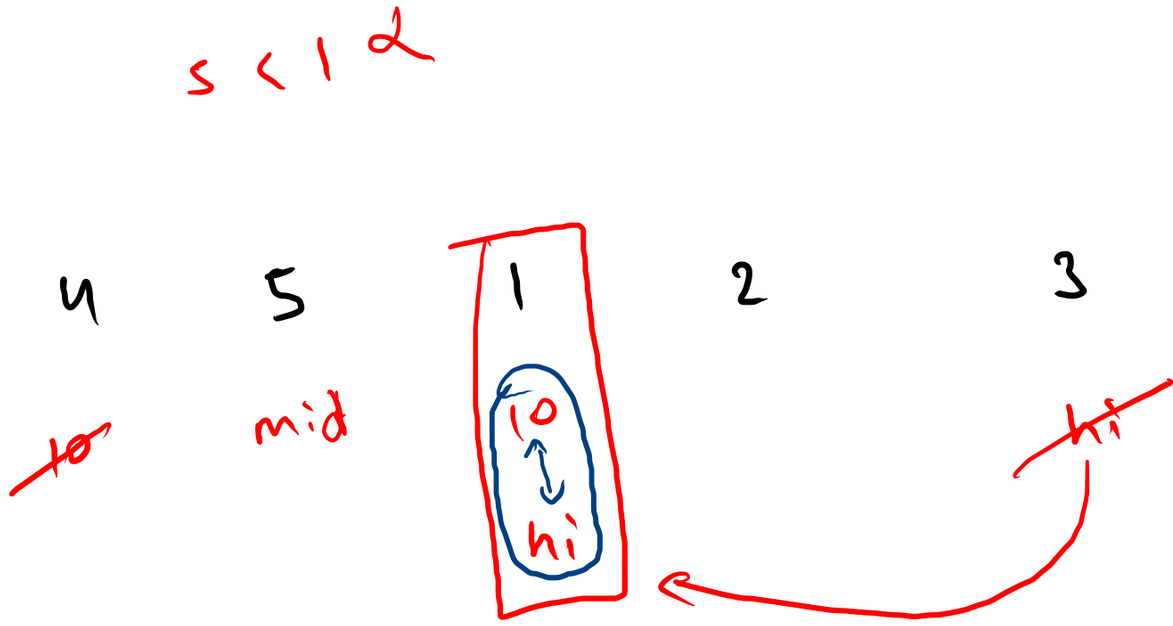


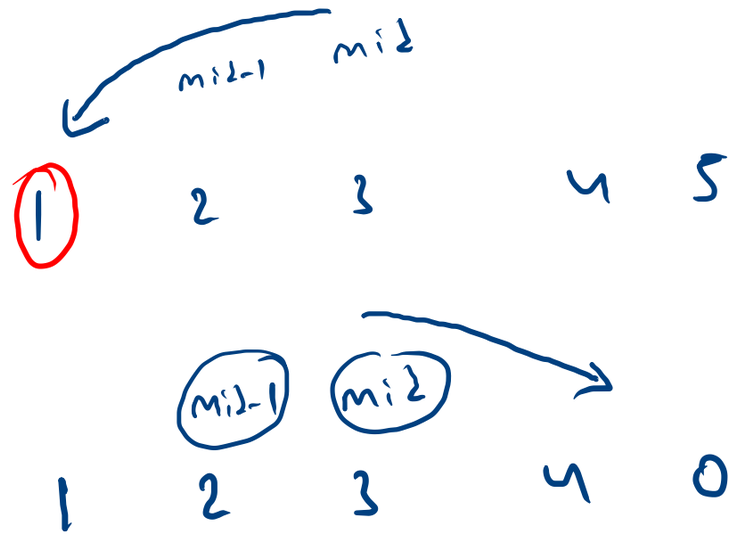






```java
while(lo < hi){
    int mid = (lo+hi)/2;

    if(arr[mid] < arr[hi]){
        hi = mid;
    }else{
        lo = mid+1;
    }
}
return arr[lo];
```

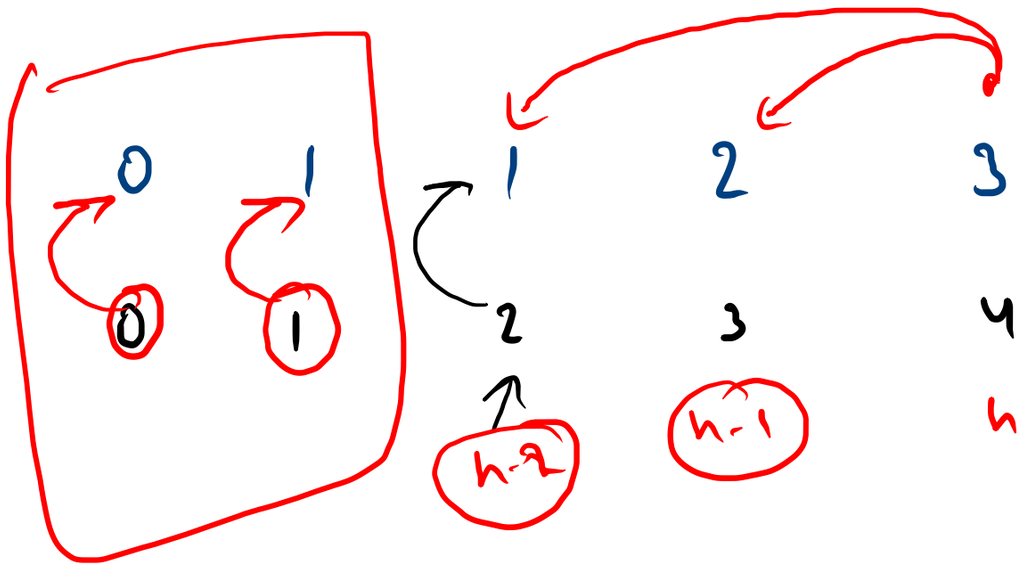




O(log(n))

* Solutions → O(n)

better implementation

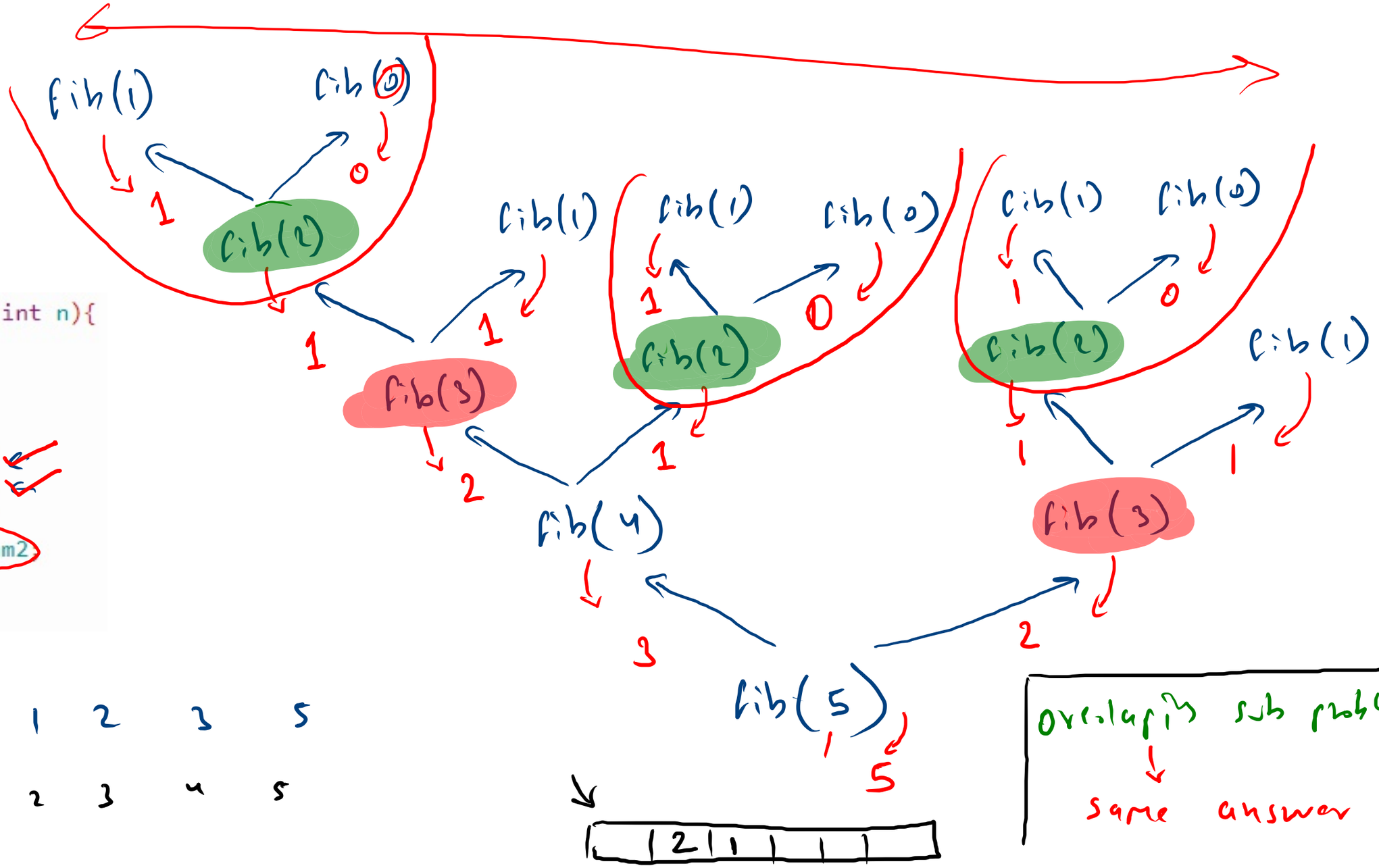


index	1	2	3	5	8	13	21
	2	3	4	5	6	7	8

problem

main

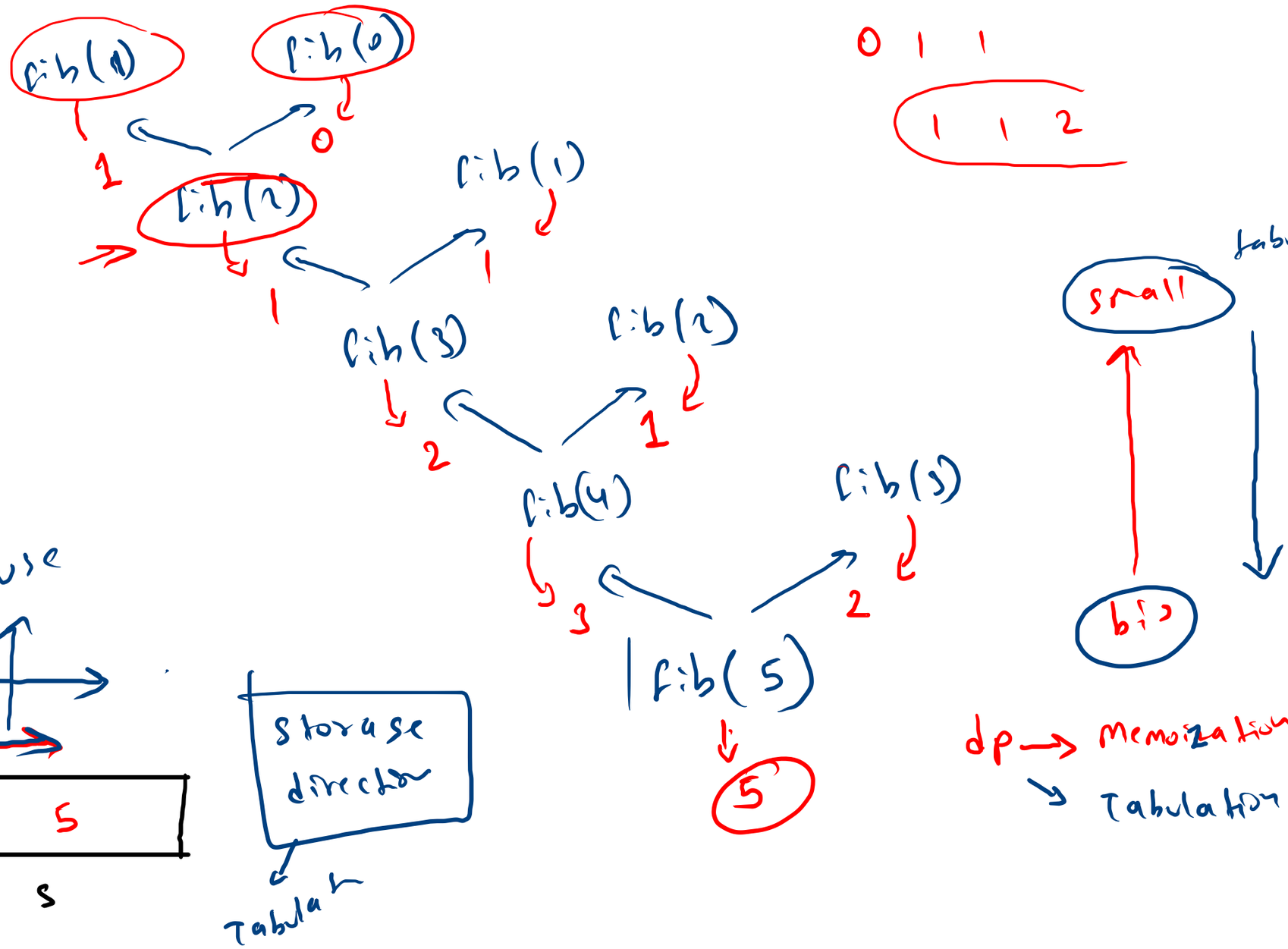
```java
public static int fibR(int n){
    if(n==0 || n==1){
        return n;
    }

    int nm1 = fibR(n-1);
    int nm2 = fibR(n-2);

    int nthfib = nm1 + nm2;
    return nthfib;
}
```



fib	0	1	1	2	3	5
index	0	1	2	3	4	5

n = 5

```java
public static int fibM(int n, int qb[]) {
  if (n == 0 || n == 1) {
    return n;
  }
  if(qb[n] != 0){
     return qb[n];
  }
  int nm1 = fibM(n - 1, qb);
  int nm2 = fibM(n - 2, qb);

  int nthfib = nm1 + nm2;
  qb[n] = nthfib;
  return nthfib;
}
```



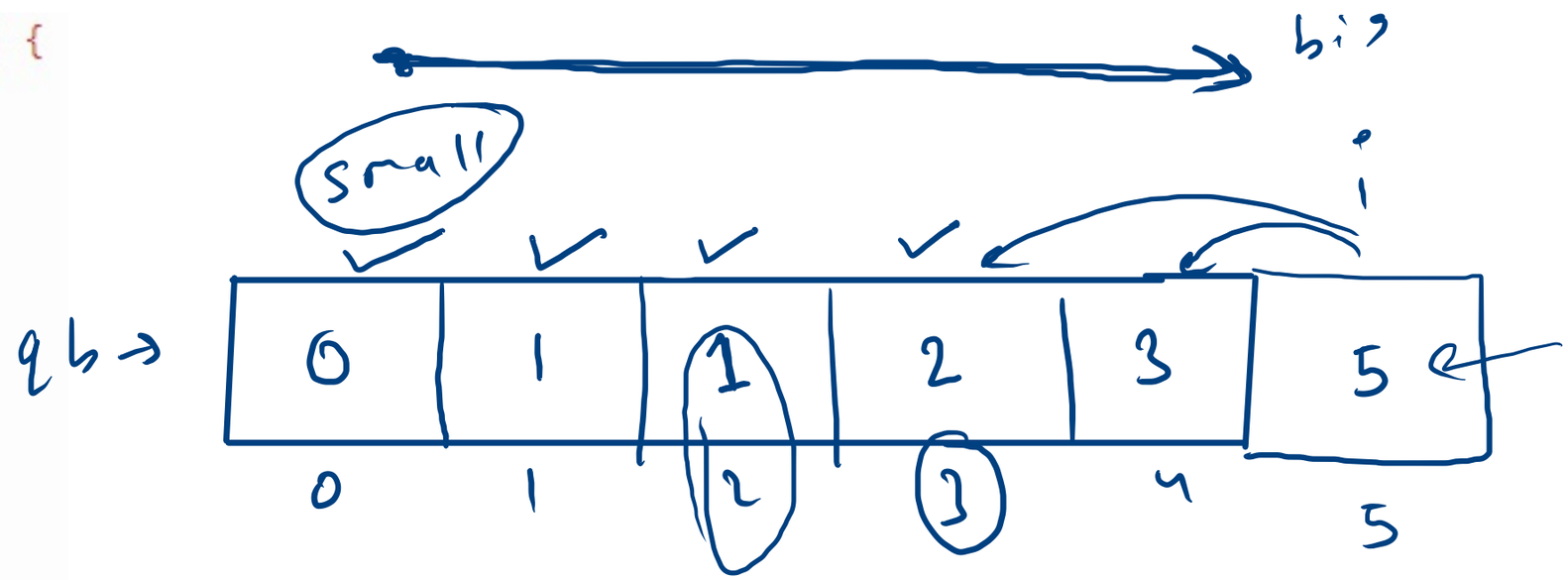
Recursion
memo
tabul

n = 5

Direction

```java
public static int fibT(int n, int qb[]) {
  for (int i = 0; i <= n; i++) {
      if (i == 0 || i == 1) {
          qb[i] = i;
          continue;
      }

      int nm1 = qb[i-1];
      int nm2 = qb[i-2];

      int nthfib = nm1 + nm2;
      qb[i] = nthfib;
      continue;
  }
  return qb[n];
}
```

equal



recursion → memo

easy

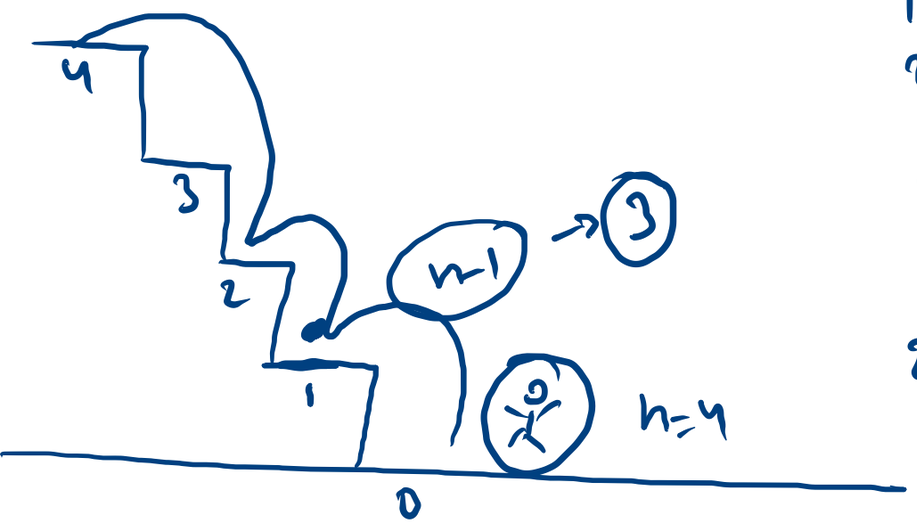
dp → memo Recursion big → small

→ tab Iteration small → big

→ tabu T

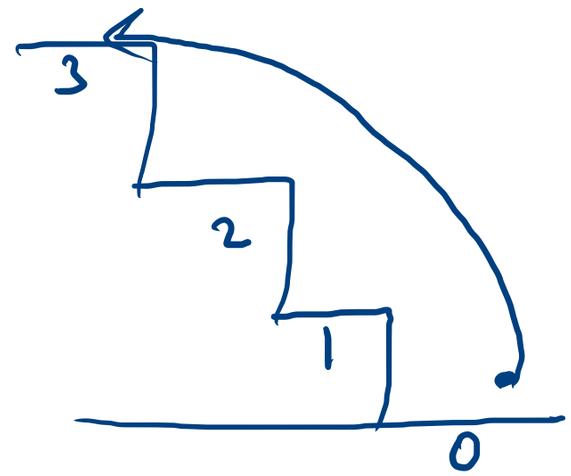
h = 4

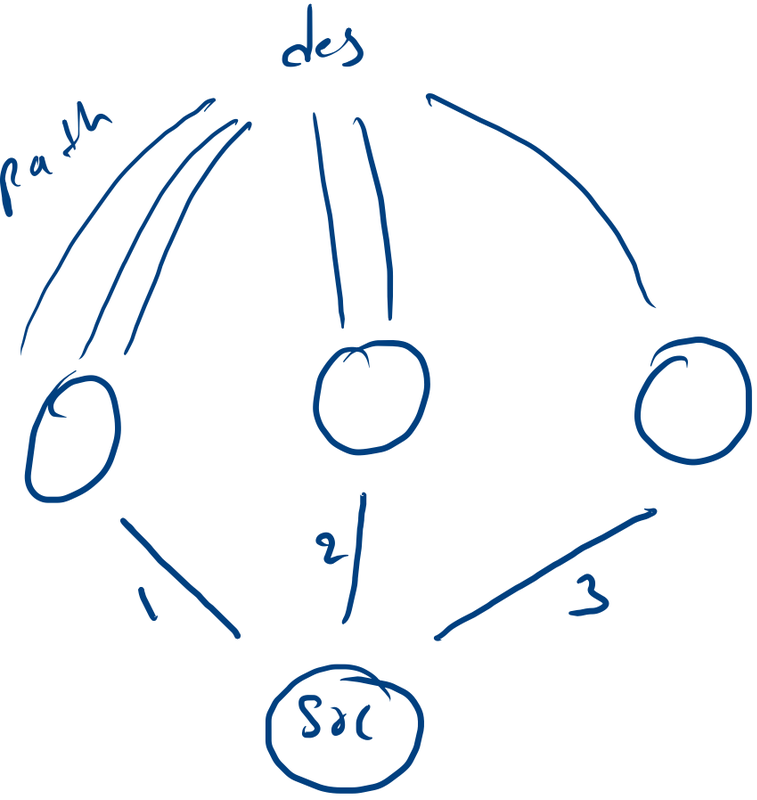
jump 1 2 3

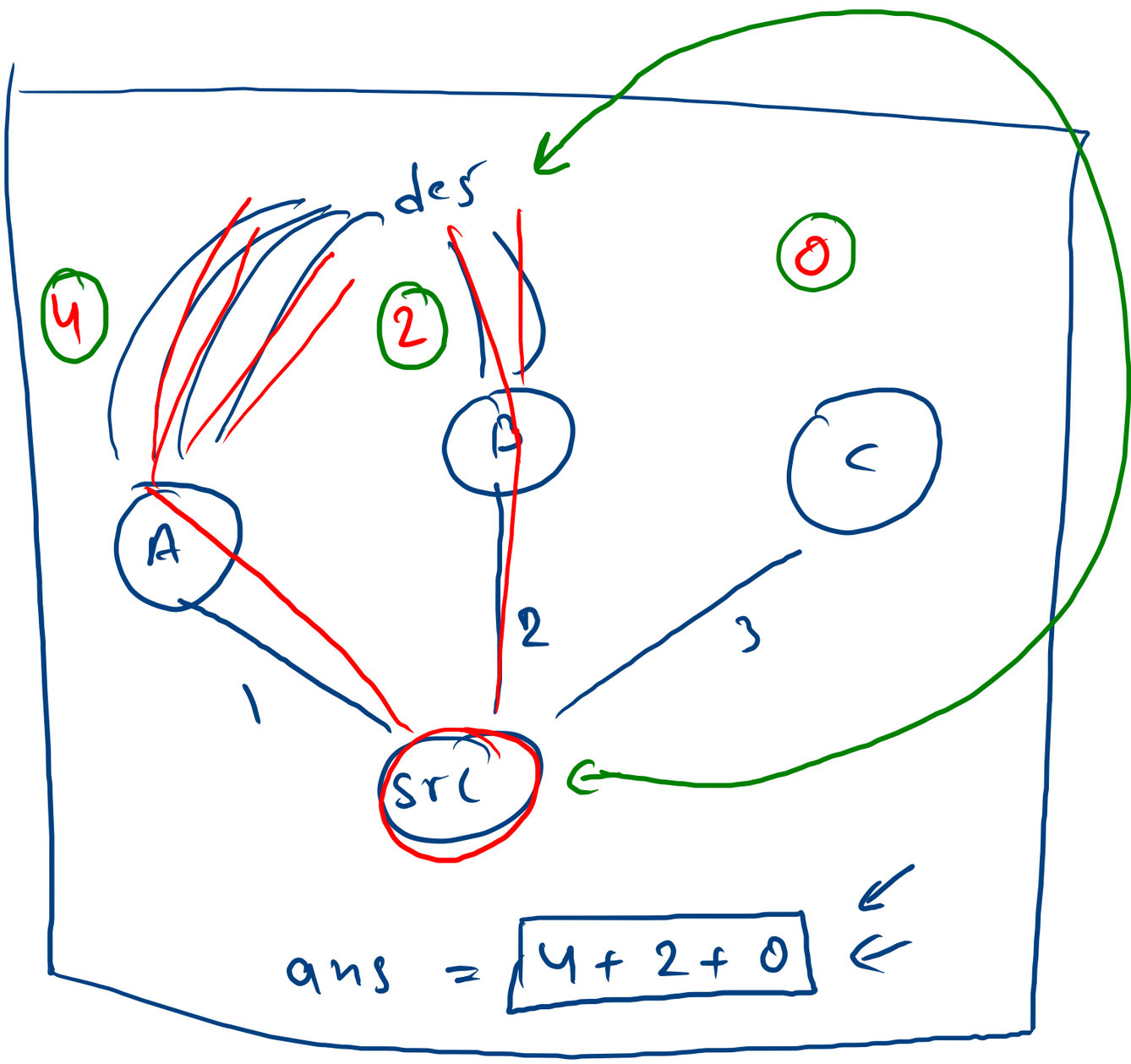


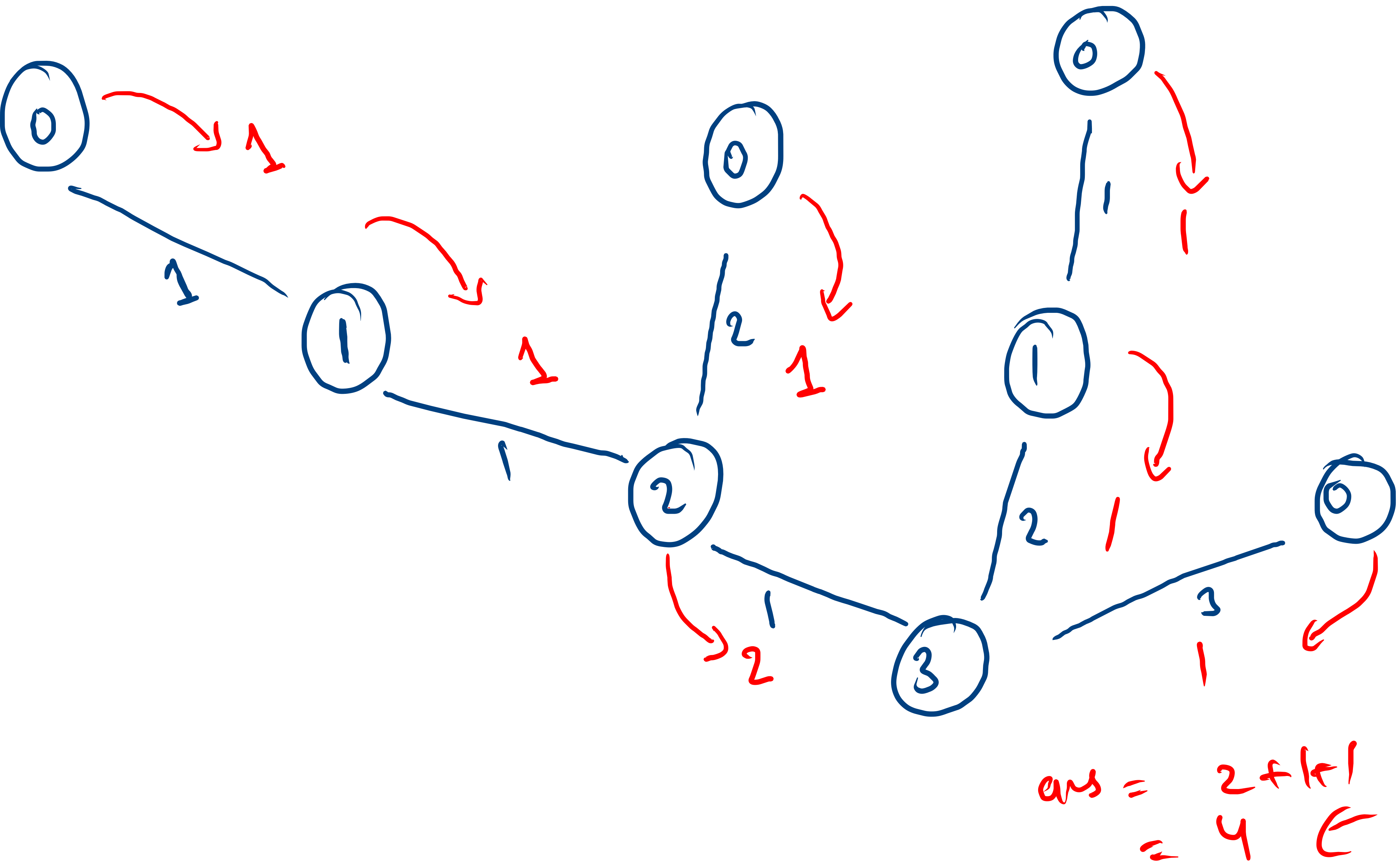
1111 — 1
112 — 2
121 — 3
211 — 4
13 — 5
31 — 6
22 — 7

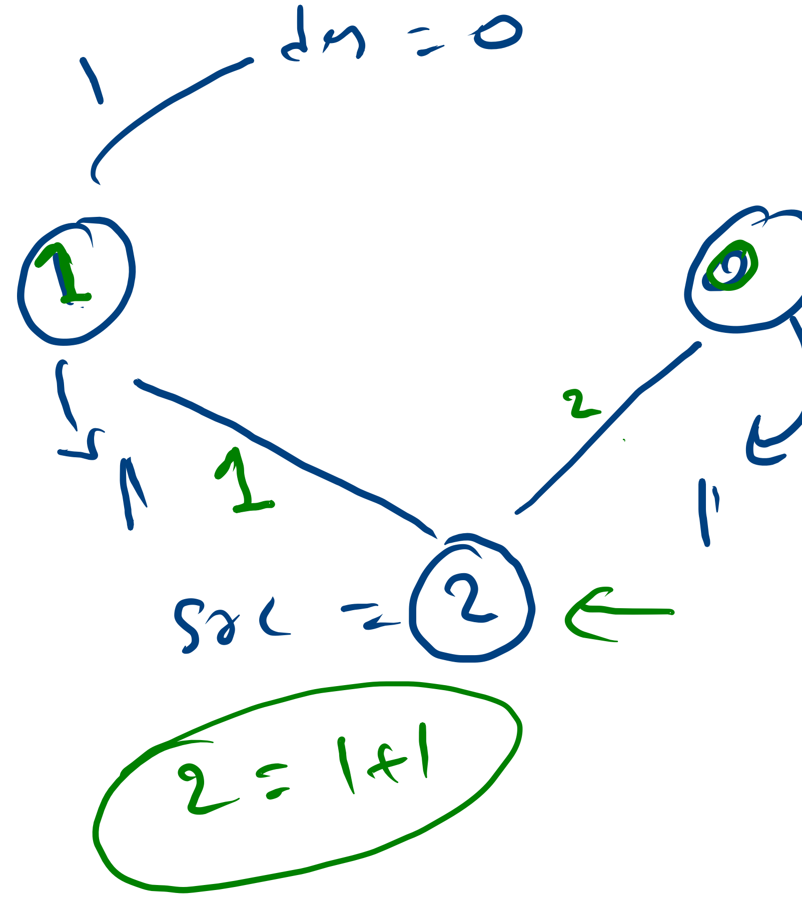
h = 3



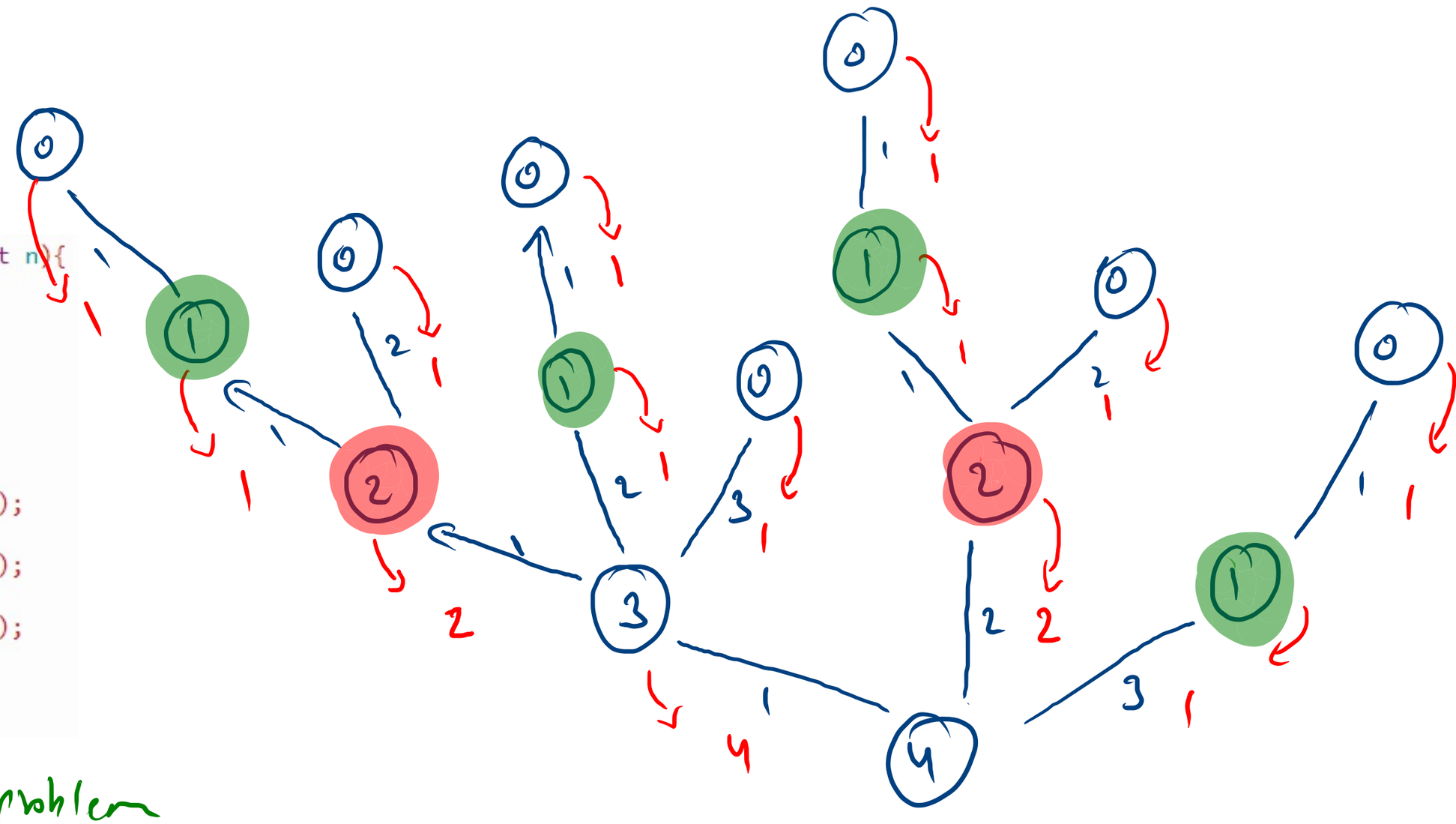

des
path
1
2
3
Src


des
4
2
0
A
B
C
1
2
3
Src
ans = 4 + 2 + 0


ans = 2+1+1
= 4

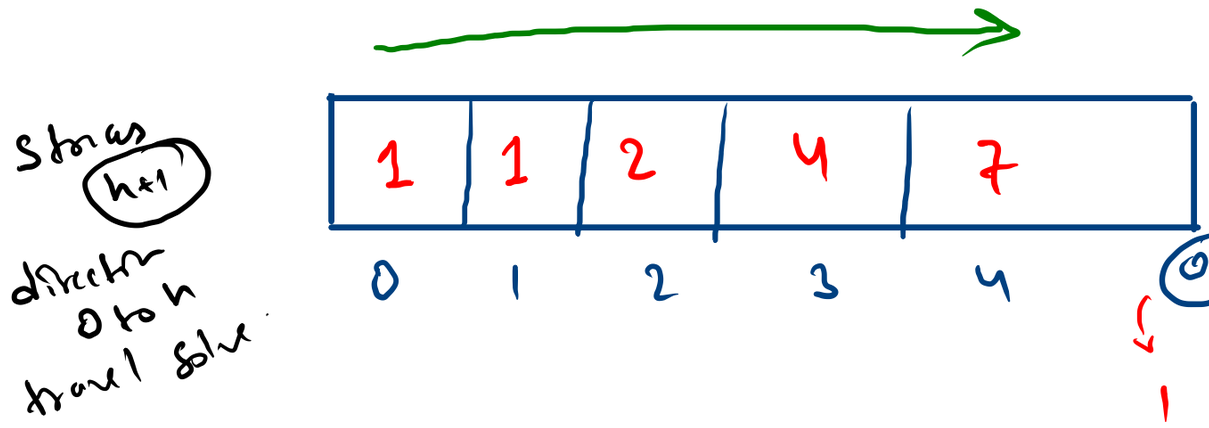

dm = 0
src = 2
2 = 1+1

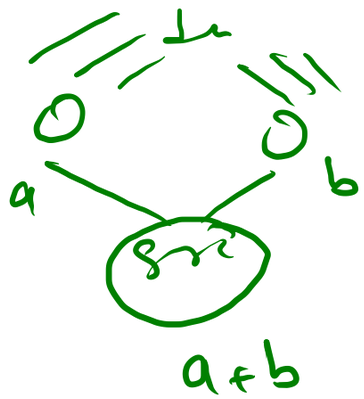
```java
public static int climbStairR(int n){
    if(n==0){
        return 1;
    }

    int total=0;

    if(n-1 >= 0)
        total += climbStairR(n-1);
    if(n-2 >= 0)
        total += climbStairR(n-2);
    if(n-3>=0)
        total += climbStairR(n-3);

    return total;
}
```



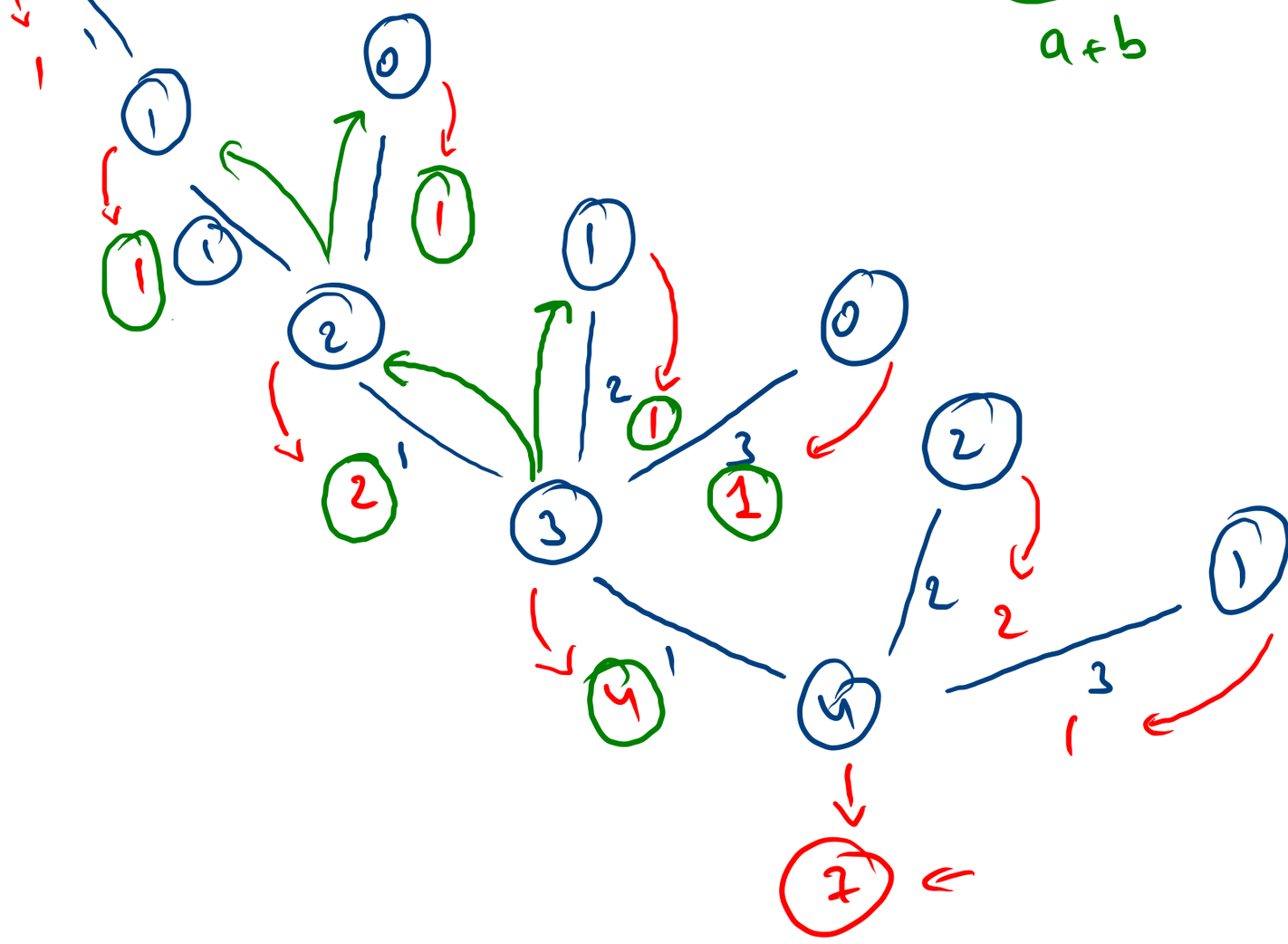
overlaping sub problem

same answer



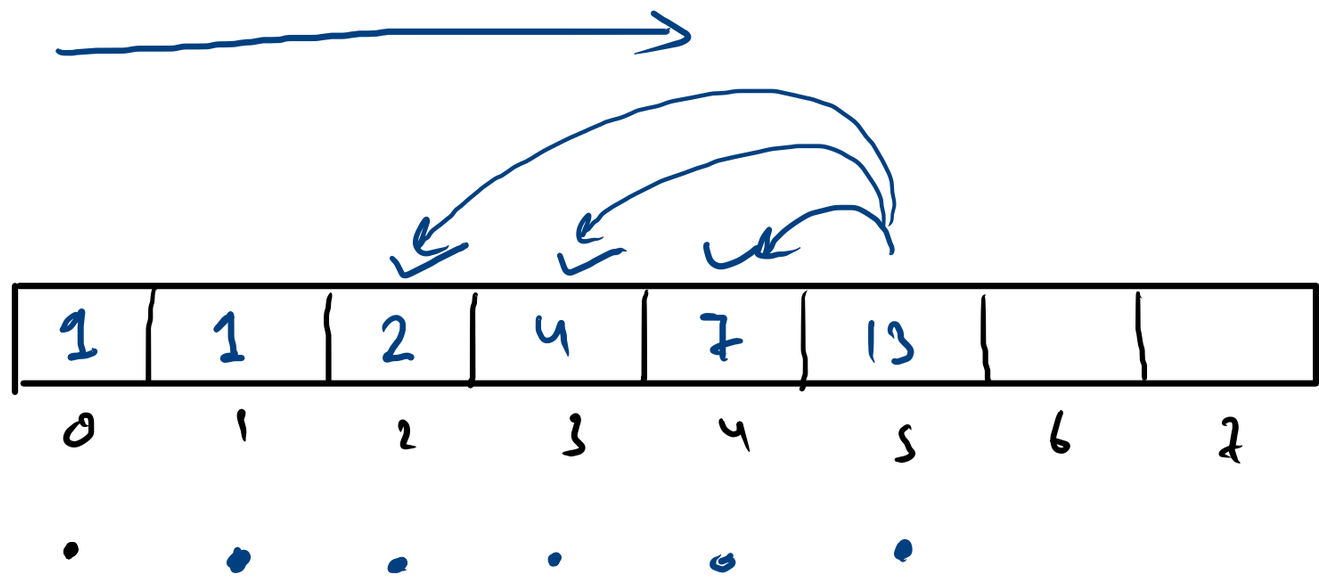



```java
public static int climbStairDpM(int n, int qb[]){
    if(n==0){
        qb[n] = 1;
        return 1;
    }
    if(qb[n] != 0){
        return qb[n];
    }

    int total=0;

    if(n-1 >= 0)
        total += climbStairDpM(n-1, qb);
    if(n-2 >= 0)
        total += climbStairDpM(n-2, qb);
    if(n-3>=0)
        total += climbStairDpM(n-3, qb);

    qb[n] = total;
    return total;
}
```



n = 7

```java
public static int climbStairDpT(int n, int qb[]){

    for(int i=0;i<=n;i++){
        if(i==0){
            qb[i] = 1;
            continue;
        }
        int total=0;
        if(i-1 >= 0)
            total += qb[i-1]; //climbStairDpM(n-1, q
        if(i-2 >= 0)
            total += qb[i-2]; //climbStairDpM(n-2, q
        if(i-3>=0)
            total += qb[i-3];//climbStairDpM(n-3, qb

        qb[i] = total;
    }

    return qb[n];
}
```

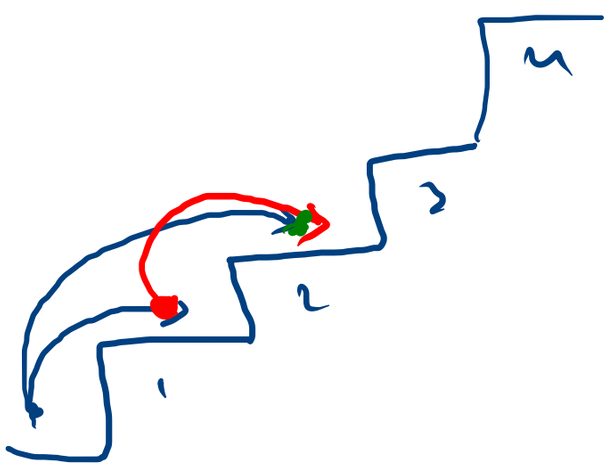


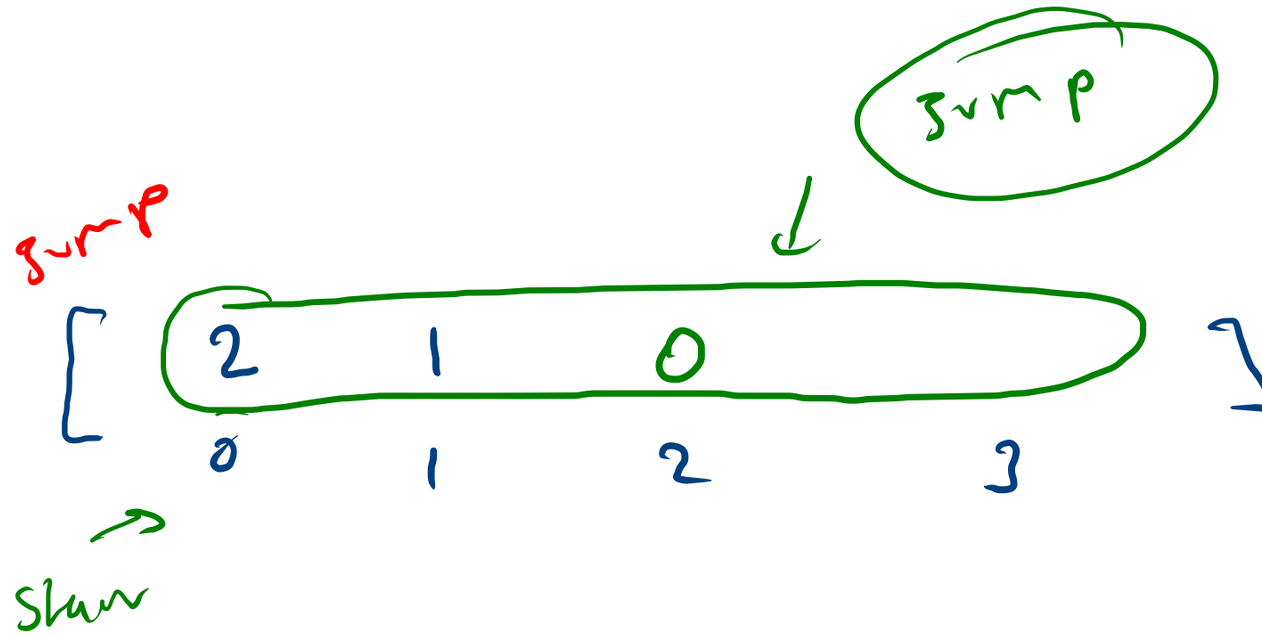
recurs → qb → memo

→ Tabul ←

n = 4

jump = 1 2 3 ← list2





6

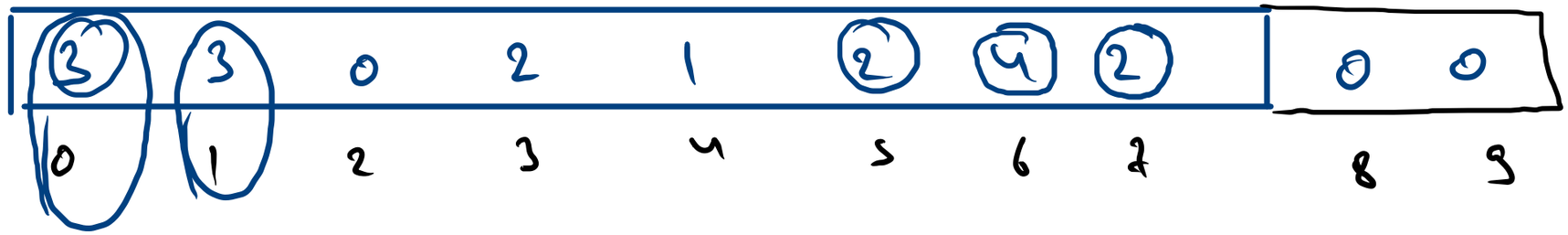
h=10

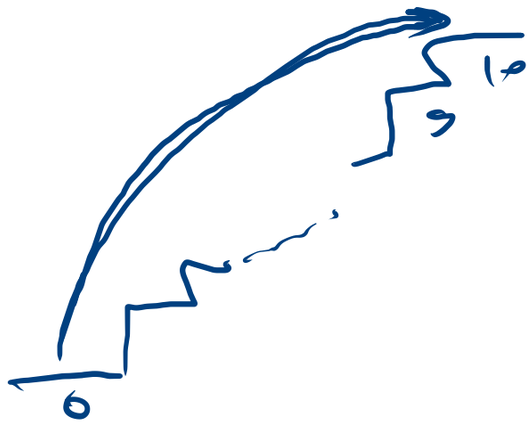
h+1

storage

meaning

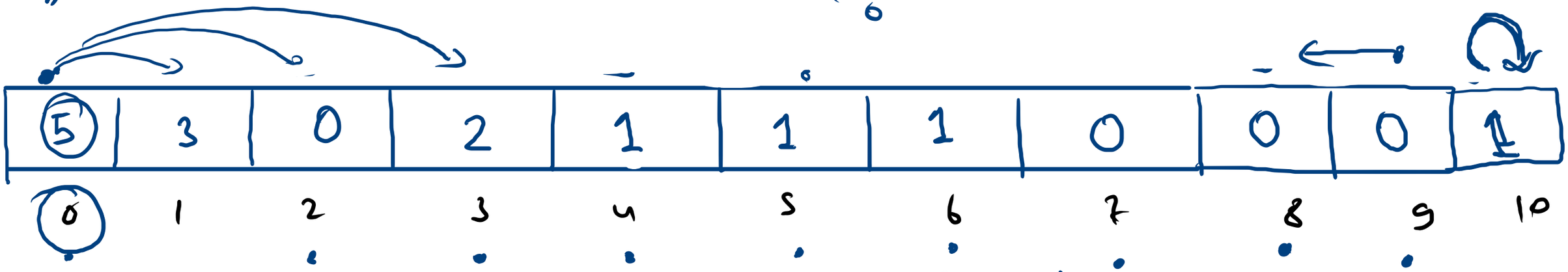
direction





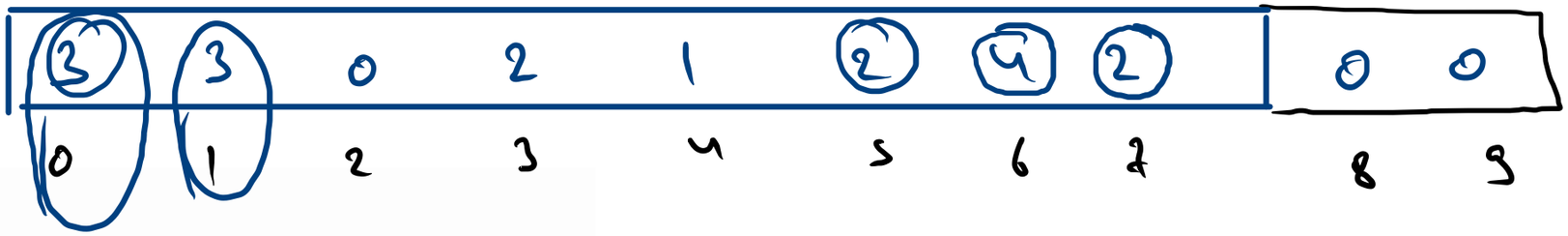
smaller

bigger



qb[ind] = number of way from ind(stair) to des

n=10




```java
// storage
int qb[] = new int[n+1];
qb[n] = 1;
for(int i=n-1;i>=0;i--){

    int totalways = 0;
    for(int jump=1;jump<=jumpLimit[i] && i+jump <= n ; jump++){
        totalways += qb[i+jump];
    }
    qb[i] = totalways;
}

System.out.println(qb[0]);
```

i = 0