```
sum = 0

for( i=0; i<= n;i ++){
    sum += i;
}

return sum
```

1 + n + 1 = n + 2

$O(n)$

0 1 2 3 ... n

k k k k

$k(n+1) = kn + k$

$O(n)$

```
sum(int n){
        if(n==0)return 0;
        return n + sum(n-1)
}
```

$O(n)$

$sum(n) = 1+2+3+4+\cdots+n$

Rec $sum(n) = n + sum(n-1)$ ★

```
sum(int n){
    if(n==0)return 0;
    return n + sum(n-1)
}
```

recursion

recurrence relation

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

$n-0$	$T(n) = T(n-1) + k$	
$n-1$	$T(n-1) = T(n-2) + k$	
$n-2$	$T(n-2) = T(n-3) + k$	
$n-3$	$T(n-3) = T(n-4) + k$	
⋮	⋮	⋮
$n-n$	$T(0) = k$	$n+1$ times

$T(n) = k + k + k \cdots (n+1)$ times

$T(n) = (n+1)k$

exact $= nk + k$

$g(n) = nk + k$

$= nk + nk$

$g(n) = 2k\ n$

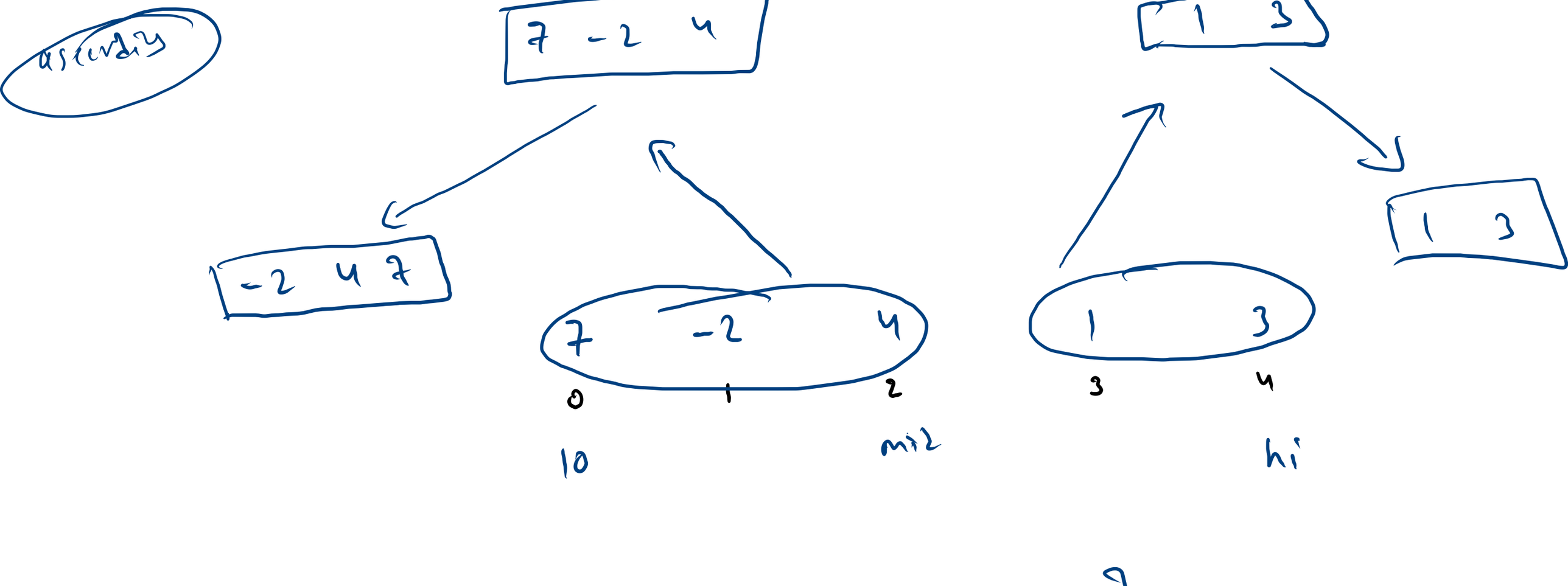
$g(n) = n$

$O(g(n))$

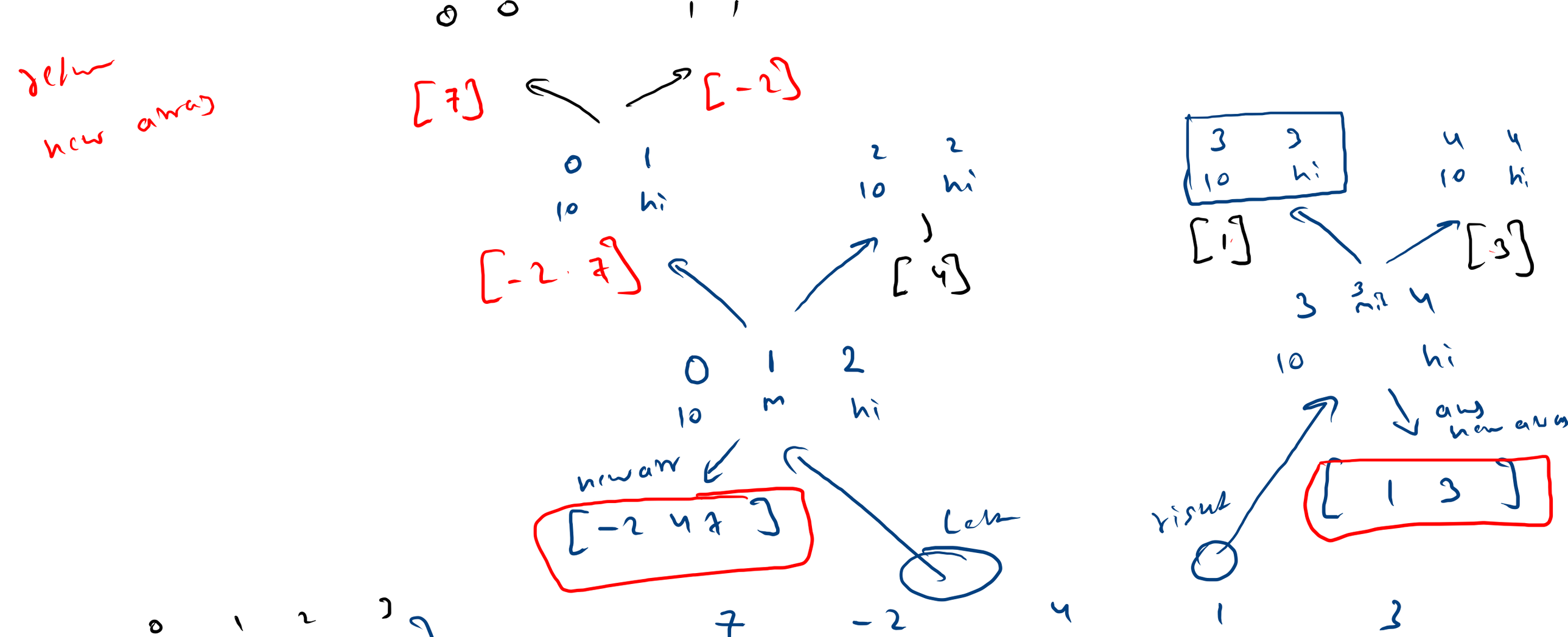
$O(n)$

$g(n) = n$

$O(n)$



ans → [-2 1, 3, 4, 7]


0 0
1 1
[7]
[-2]
0 1
10 hi
2 2
10 hi
[-2 7]
[4]
0 1 2
10 m hi
newarr
[-2 4 7]
Left
7
-2
4
3 3
10 hi
4 4
10 hi
[1]
[3]
3 3 4
10 hi
arr
newarr
[1 3]
right
1
3
0 1 2 3 4
arr → [7 -2 4 1 3]
4k
4
0
mid
4
[-2 1 3 4 7]
arr

[3] [2] [4] [7] [6] [5]

2 [3 2] 2 [4 7] 2 [6 5] [1]

[5 6] [1]

4 [3 2 4 7] 4 [6 5 1]

n/2 n/2 [1 5 6]

2 →

1 → [3 2 4 7 6 5 1]

h

span(h)

h O(h)

nm arcs

h

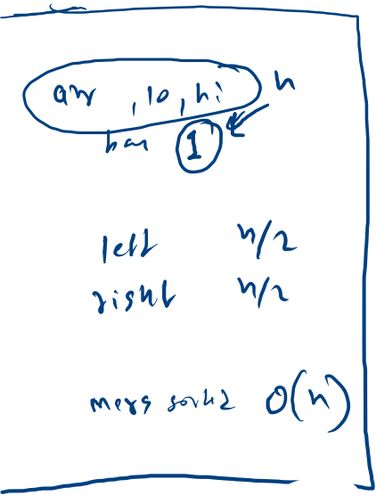
O(n) ≤

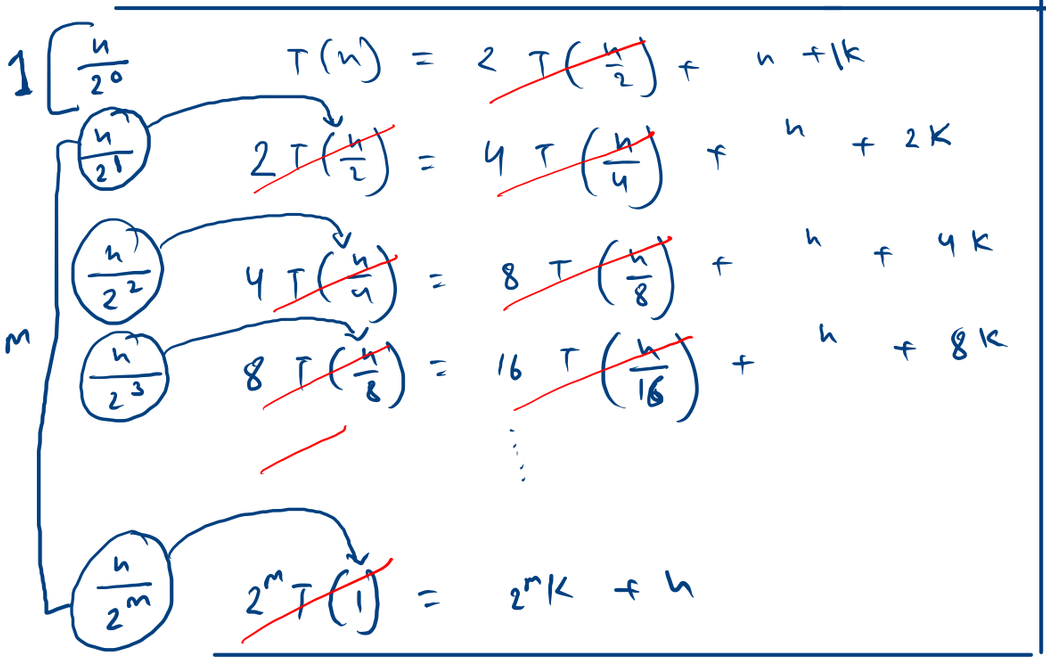
h

O(n) ←

RR $T(n) = T(n/2) + T(n/2) + n + k$

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





$\frac{n}{2^m} = 1$

$n = 2^m$

$m = \log_2(n)$

$m+1$

$T(n) = n(m+1) + k(2^{m+1} - 1)$

$= nm + n + k\,2^{m+1} - k$

$= nm + n + k\,2^{\log_2(n)} - k$

$= nm + n + kn - k$

$= n\log_2(n) + n$

$= n\log_2(n)$

$O(n\log_2(n))$

$T(n) = n(m+1) + k(1+2+4+8+16\ldots 2^m)$

$T(n) = n(m+1) + k(2^{m+1} - 1)$

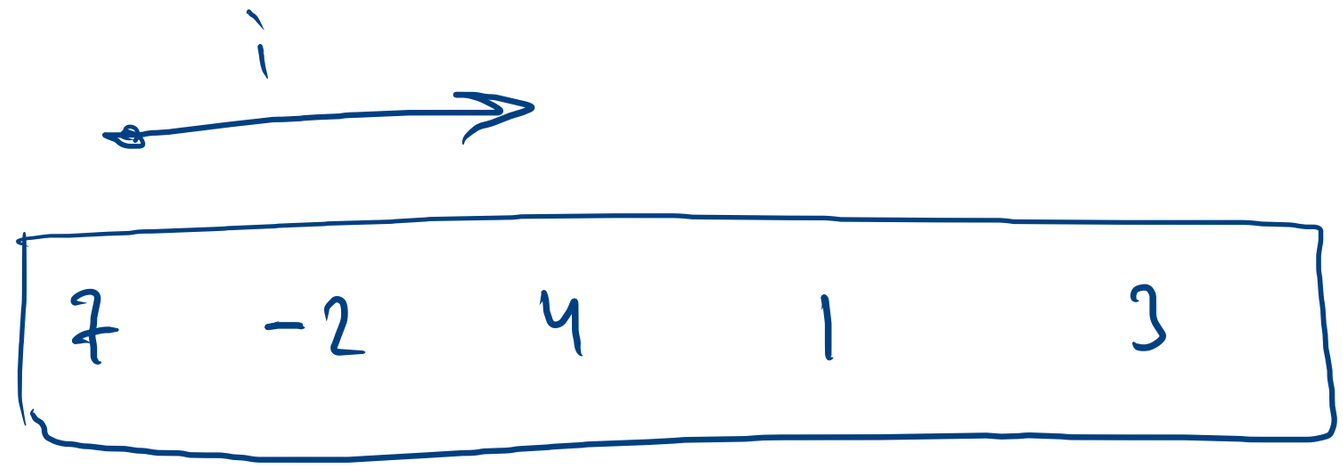
$a=1,\ r=2,\ n=m+1$

$Sum = a\left(\frac{r^n - 1}{r-1}\right)$

$1\left(\frac{2^{m+1} - 1}{1}\right)$

$2\ n\log n + n$

pivot -> 3

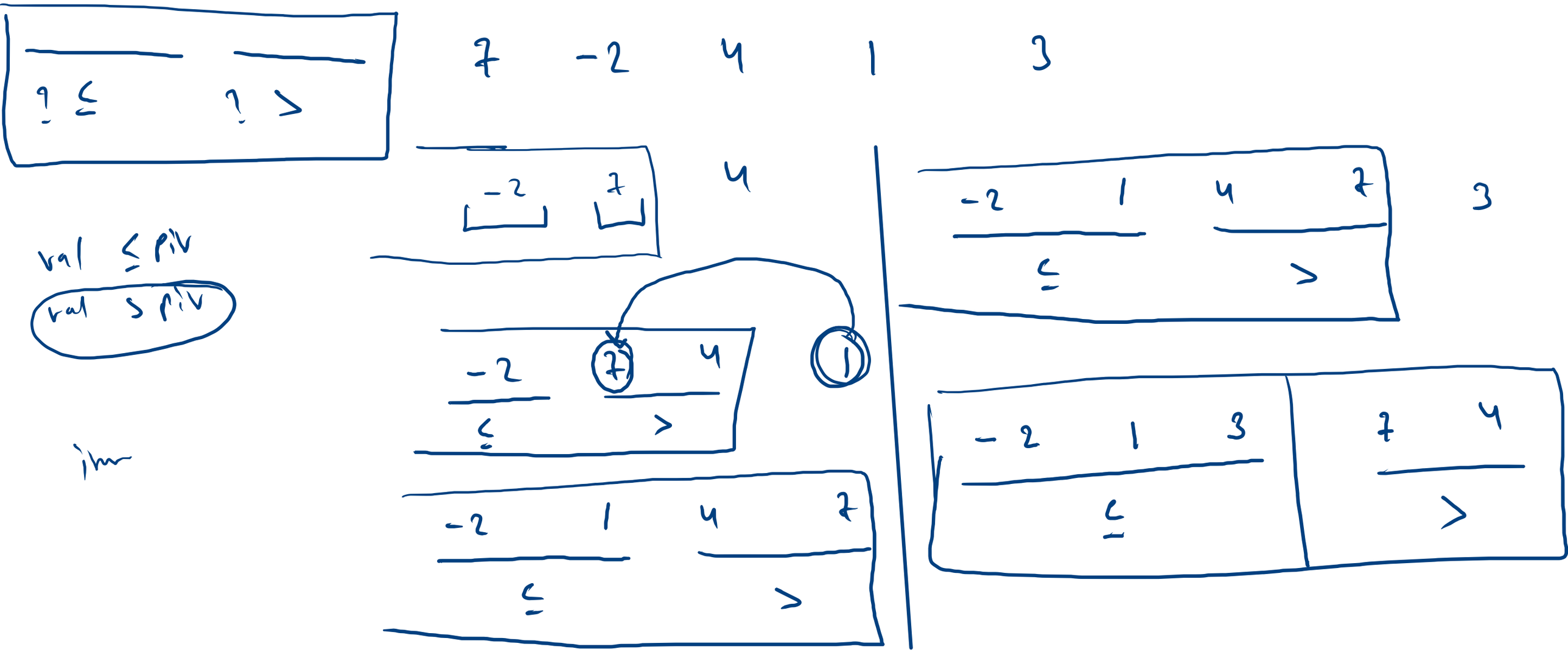


O(n)

O(n log(n))

? ≤ pivot

? > pivot


7 -2 4 1 3
? ≤ ? >
-2 7
4
val ≤ piv
val > piv
-2 7 4
≤ >
1
-2 1 4 7
≤ >
3
-2 1 3
≤
7 4
>
im

7 -2 4 1 3

j i

-2 7 4 1 3

j i

-2 7 4 1 3

j

-2 1 4 7 3

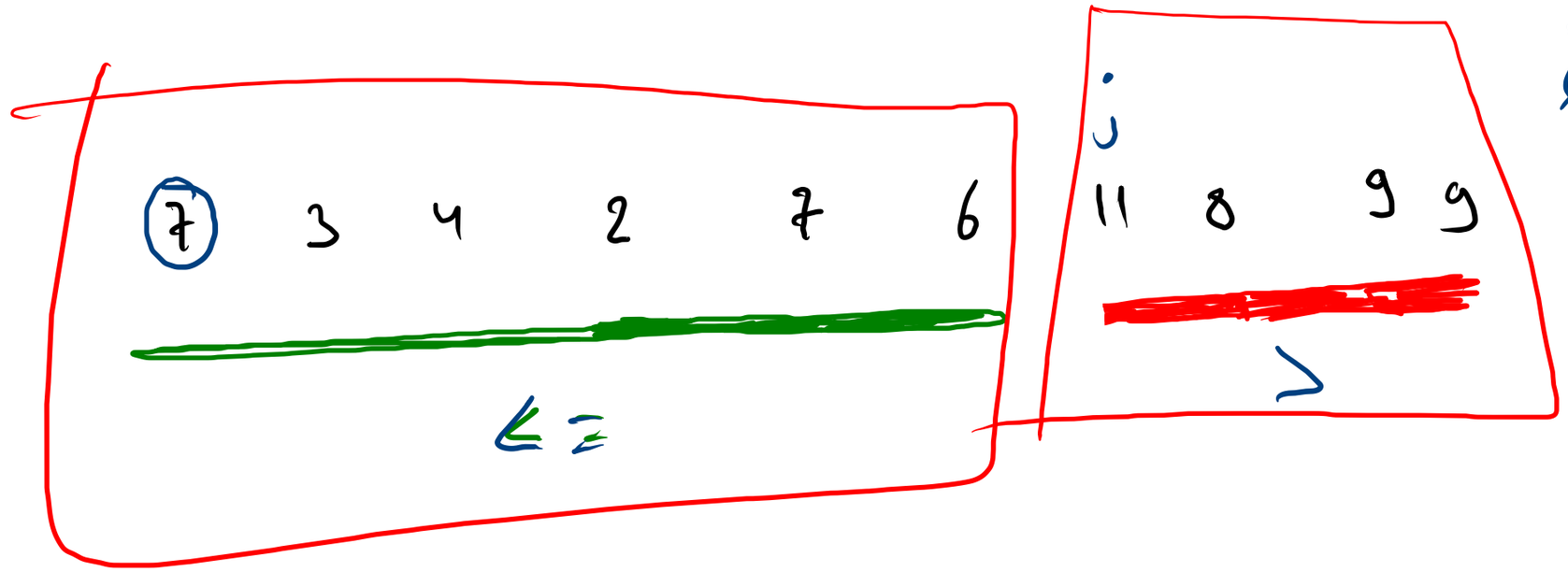
-2 1 3

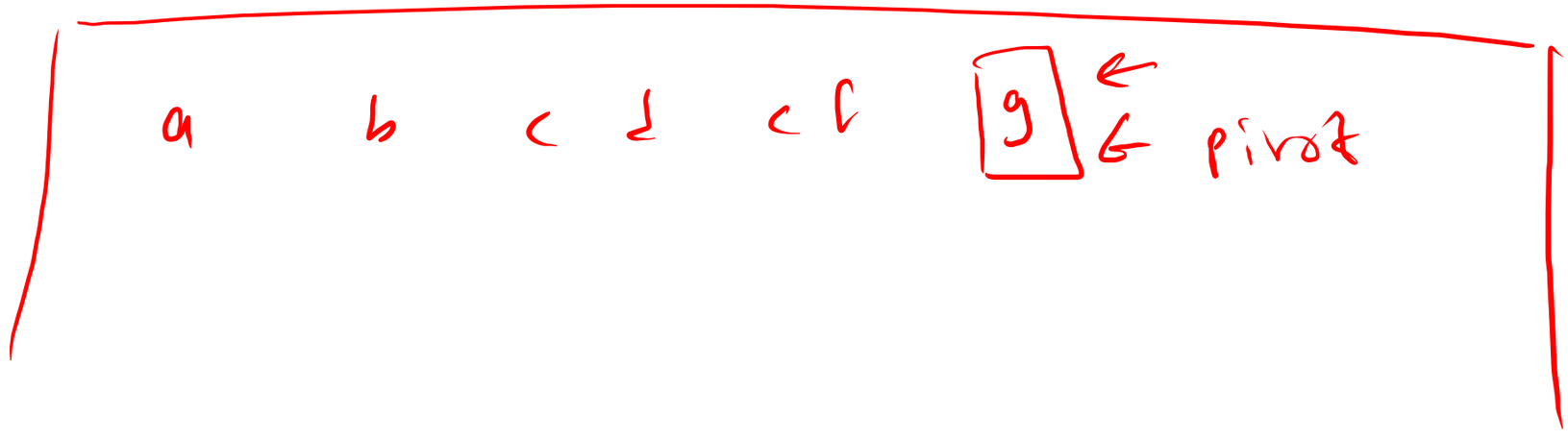
7 4

pivot = 7

i = 0
j = 0

```
while(i<arr.length){
    if(arr[i] <= pivot){
        swap(arr, i, j);
        j++;
        i++;
    }else{
        i++;
    }
}
```





in2 = partition( )

0 ... in2 smaller

indfl ... hi bigger

a b c d e f g h i

0 1 2 3 4 5 6 7 8

i ← pivot

i = pivot

j

? <= piv

? > piv

smaller

a b c d

i

larger

f g h e ← pivot

0 1 2 3 4 5 6 7 8

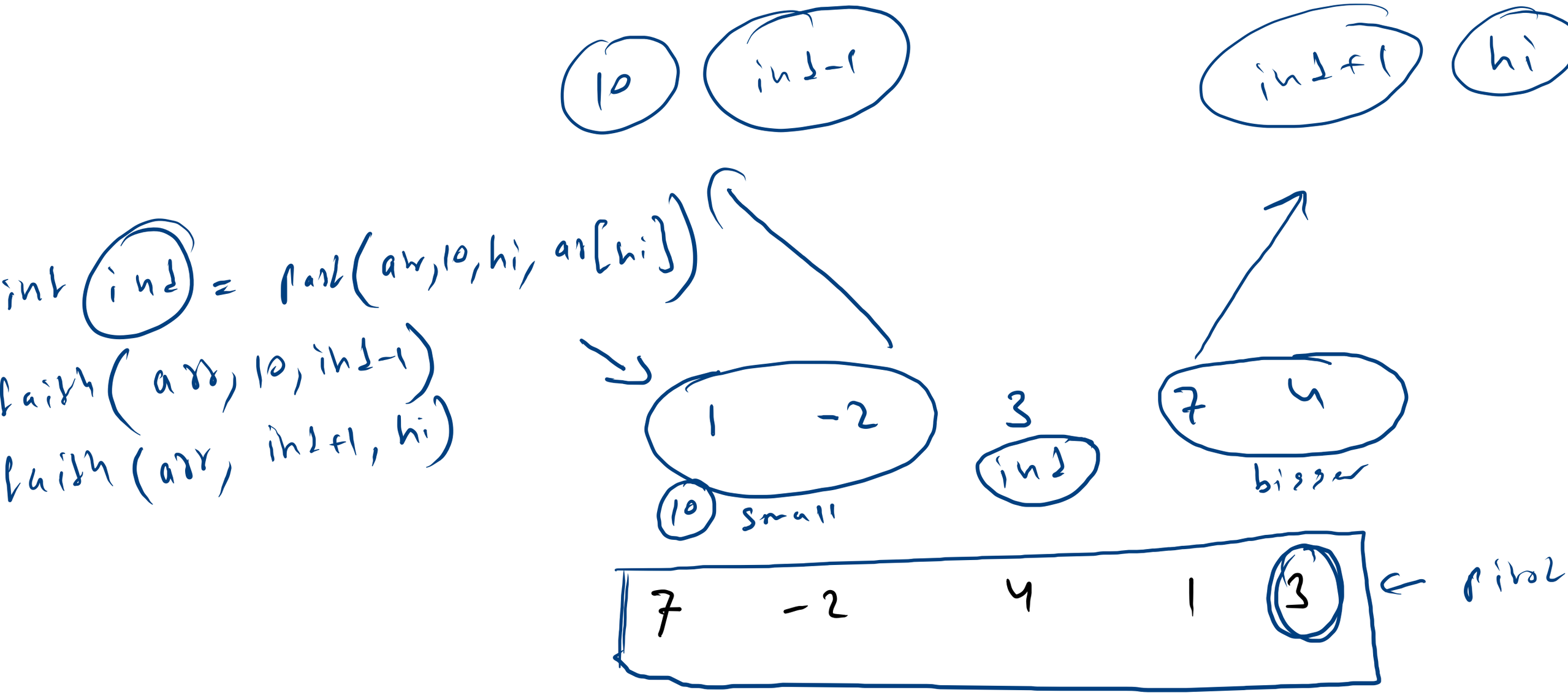
? <= piv

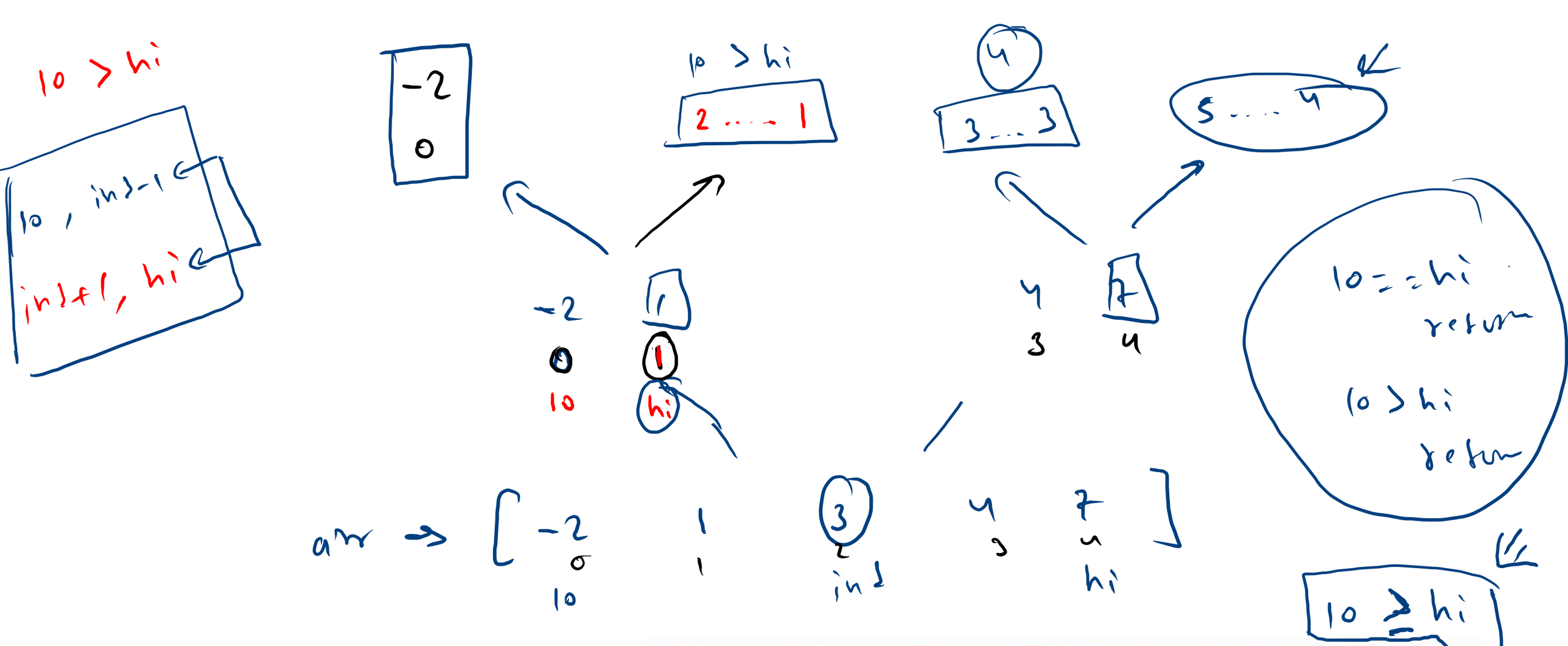
? > piv

int ind = part(ar, lo, hi, ar[hi])

part(ar, lo, ind-1)

part(arr, ind+1, hi)





7 -2 4 1 3

```java
public static void quickSort(int[] arr, int lo, int hi) {
    if(lo > hi)return;

    int ind = partition(arr, arr[hi], lo, hi);
    quickSort(arr, lo, ind-1);
    quickSort(arr, ind+1, hi);

}
```

$$T(n) = 2T(n/2) + n \quad \Leftrightarrow \quad \Omega(n \log(n))$$

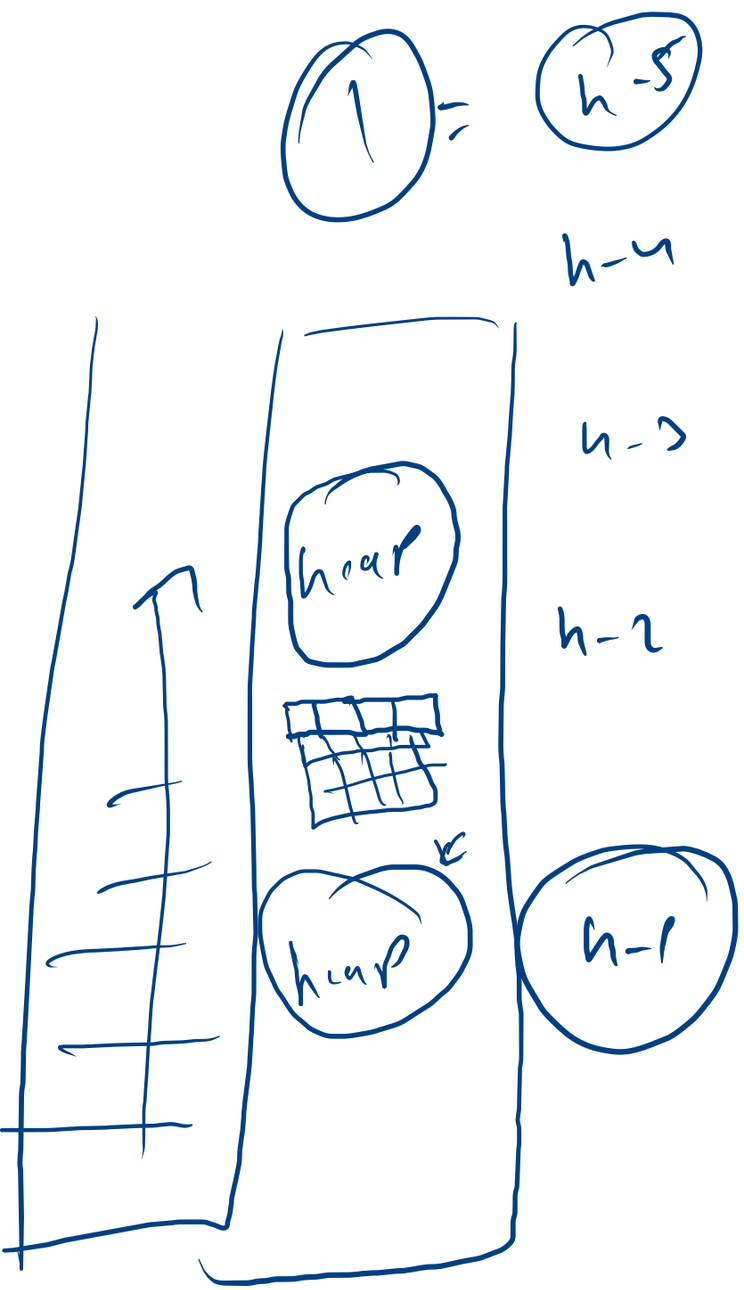
$n/2$ $n/2$

( ) k ( )

[a b c d e f g h i j k]

$n = 2$

$O(n^2)$



1 — h-5

h-4 — 1 2

h-3 — 1 2 3

h-2 — 1 2 3 4

h-1 — 1 2 3 4 5

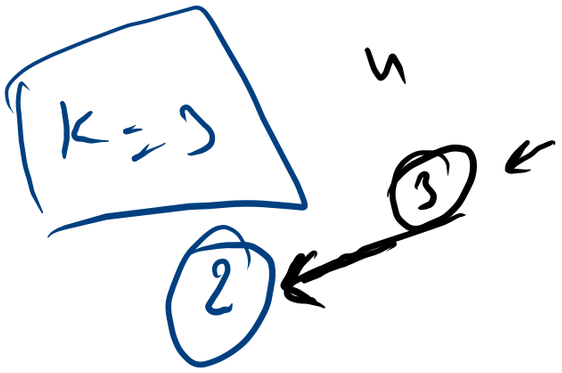
1 2 3 4 5 6 h

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

worst case → $O(n^2)$

back

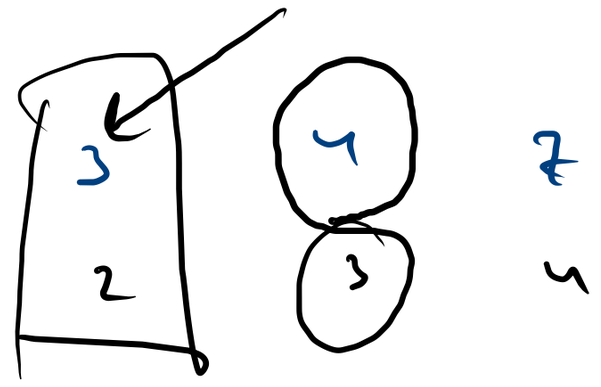
7 -2 4 1 3 3

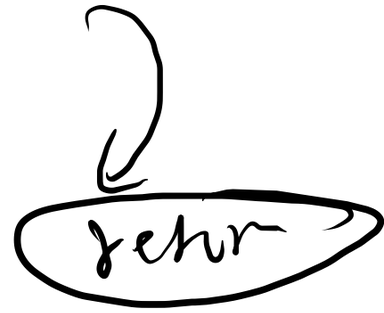


$O(n \log n)$

-2 1

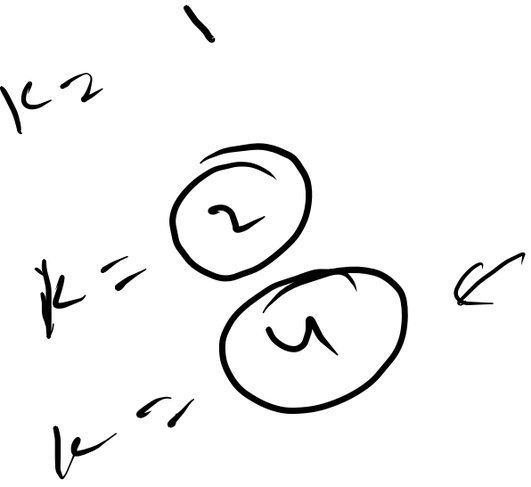
0 1

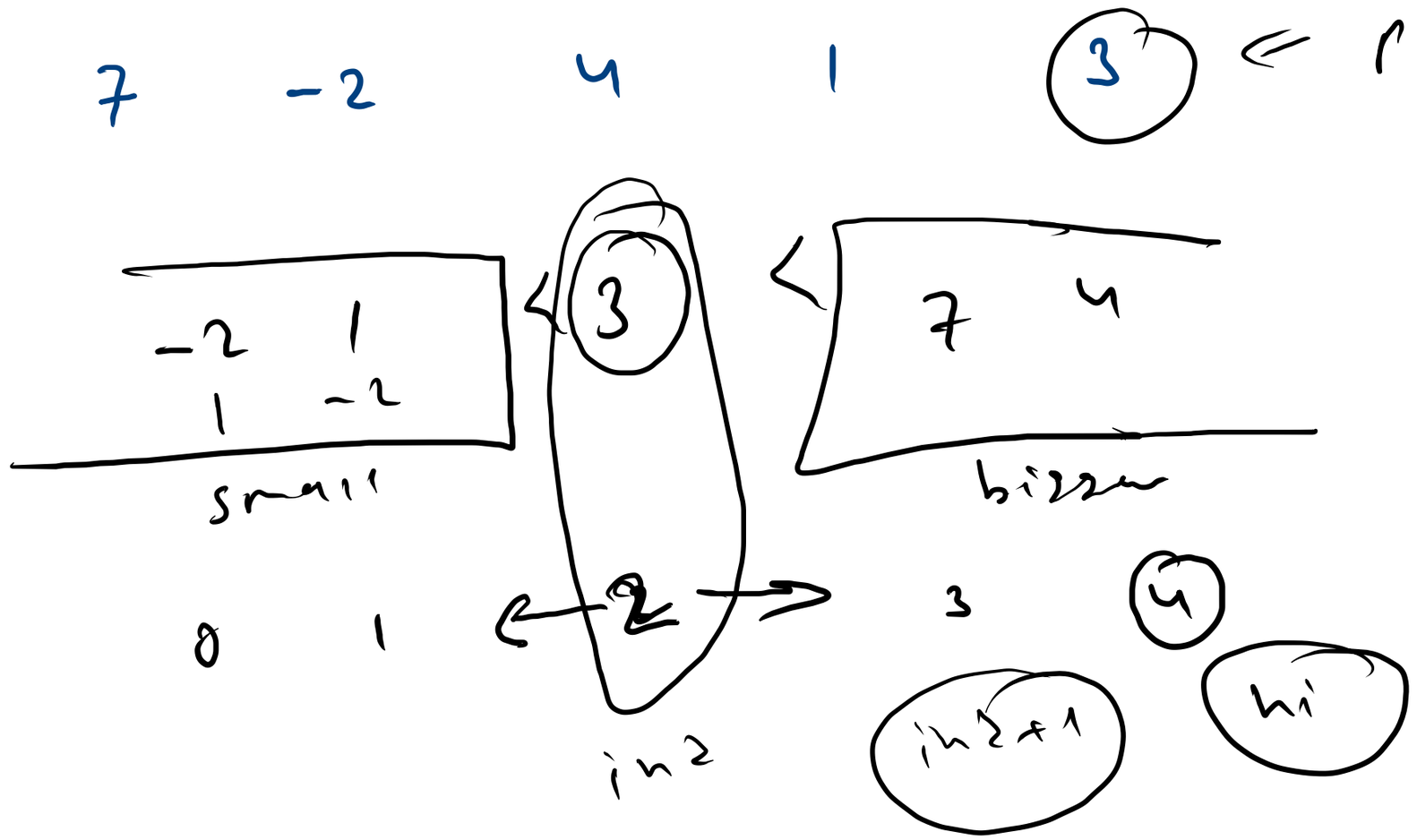




7 -2 4 1 3

H.W





9 -48 100 43 84 74 86 34 -37 60 -29 44

target = 160

100, 60

86 + 74

60 100

74 86

60 100
74 86

a + b = 160

-48 -37 -29 9 34 43 44 60 74 84 86 100

j i

160

h^2

ith + jth

100 + 60 = tar

9

100 + 74 = 174

74 + 86 = 160

100 + 9 = 109

100 + 44 = 144

sum < tar

i++

O(n log(n))

O(n)

sum > tar

j--

i<j

i>j ← duplicate

sum = tar

print

i++;

j--;