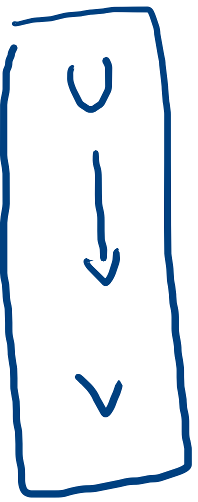
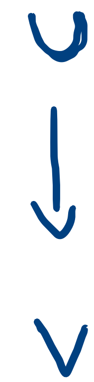
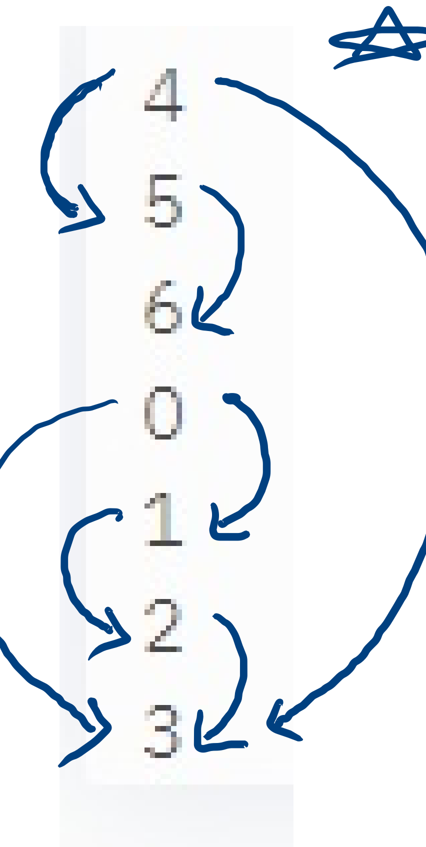
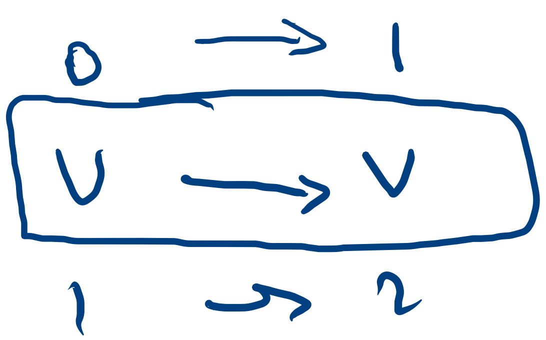
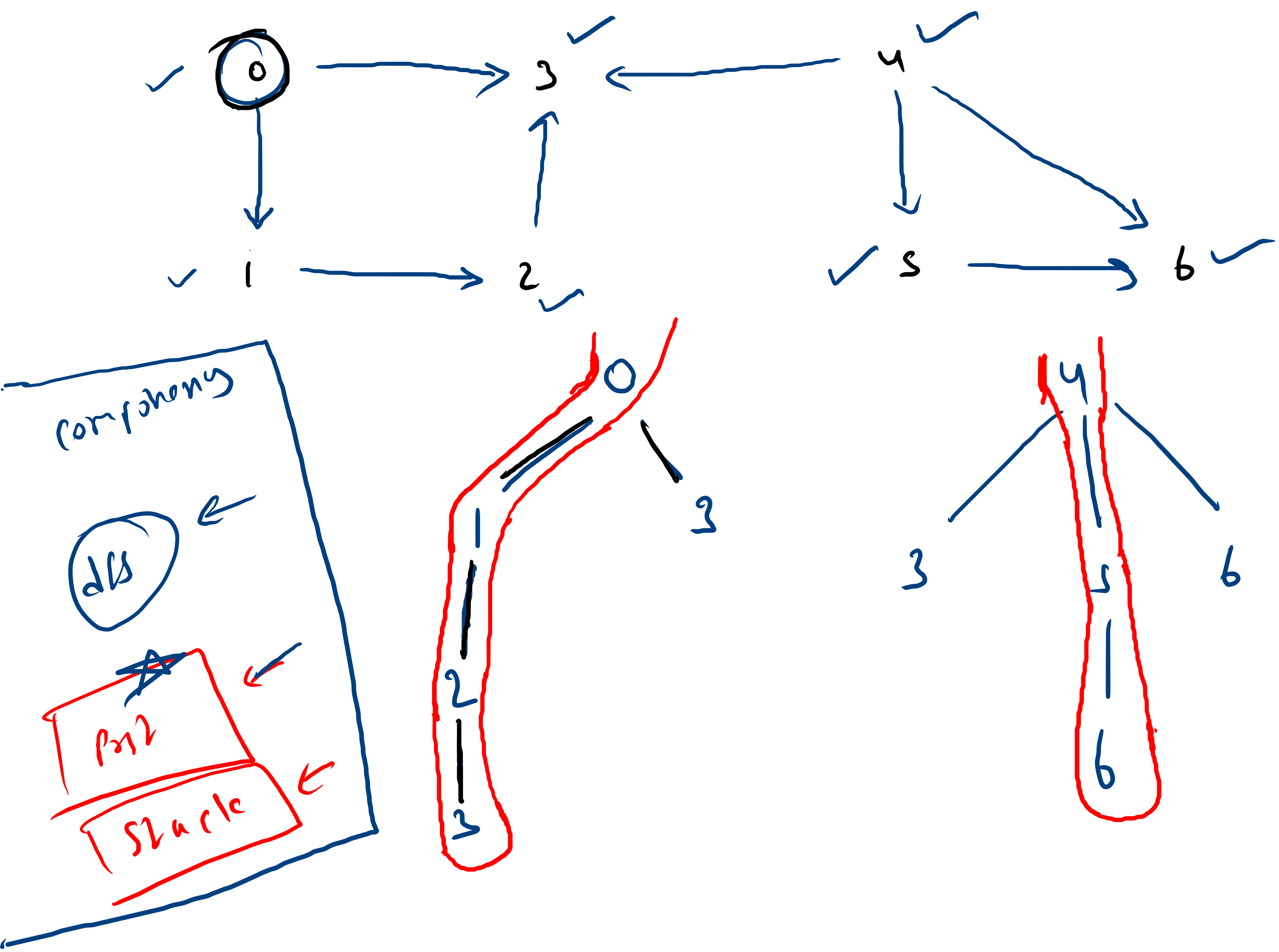

U
V


0
3
4
1
2
5
6

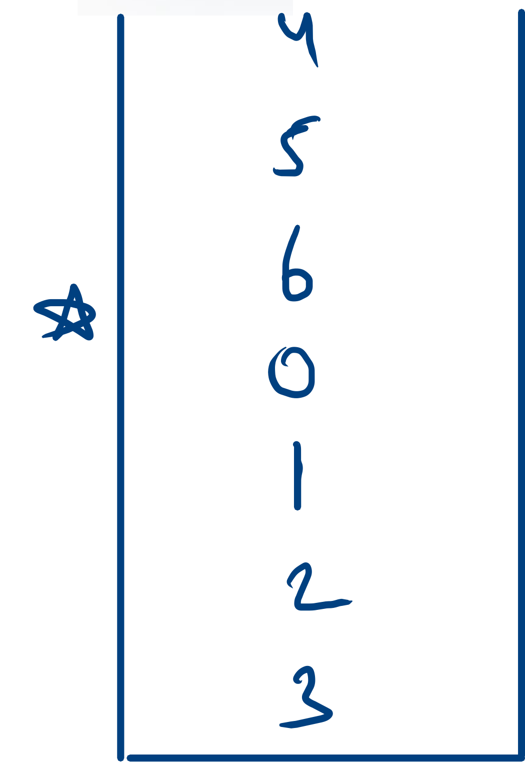

U
V

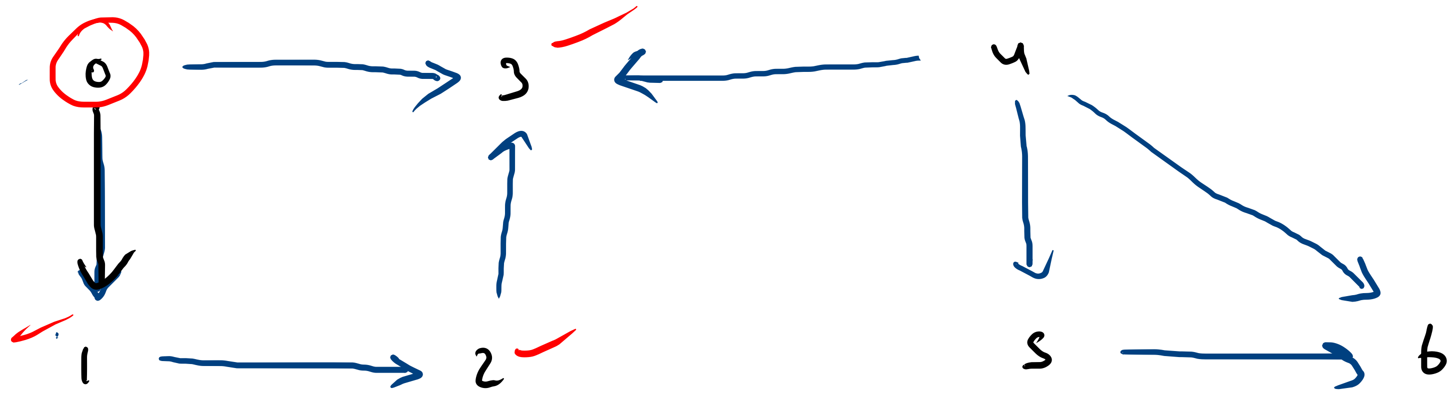

4
5
6
0
1
2
3


0
1
U
V
1
2

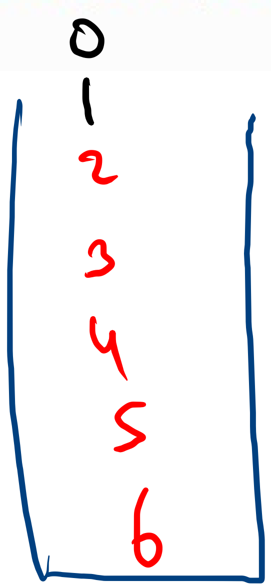


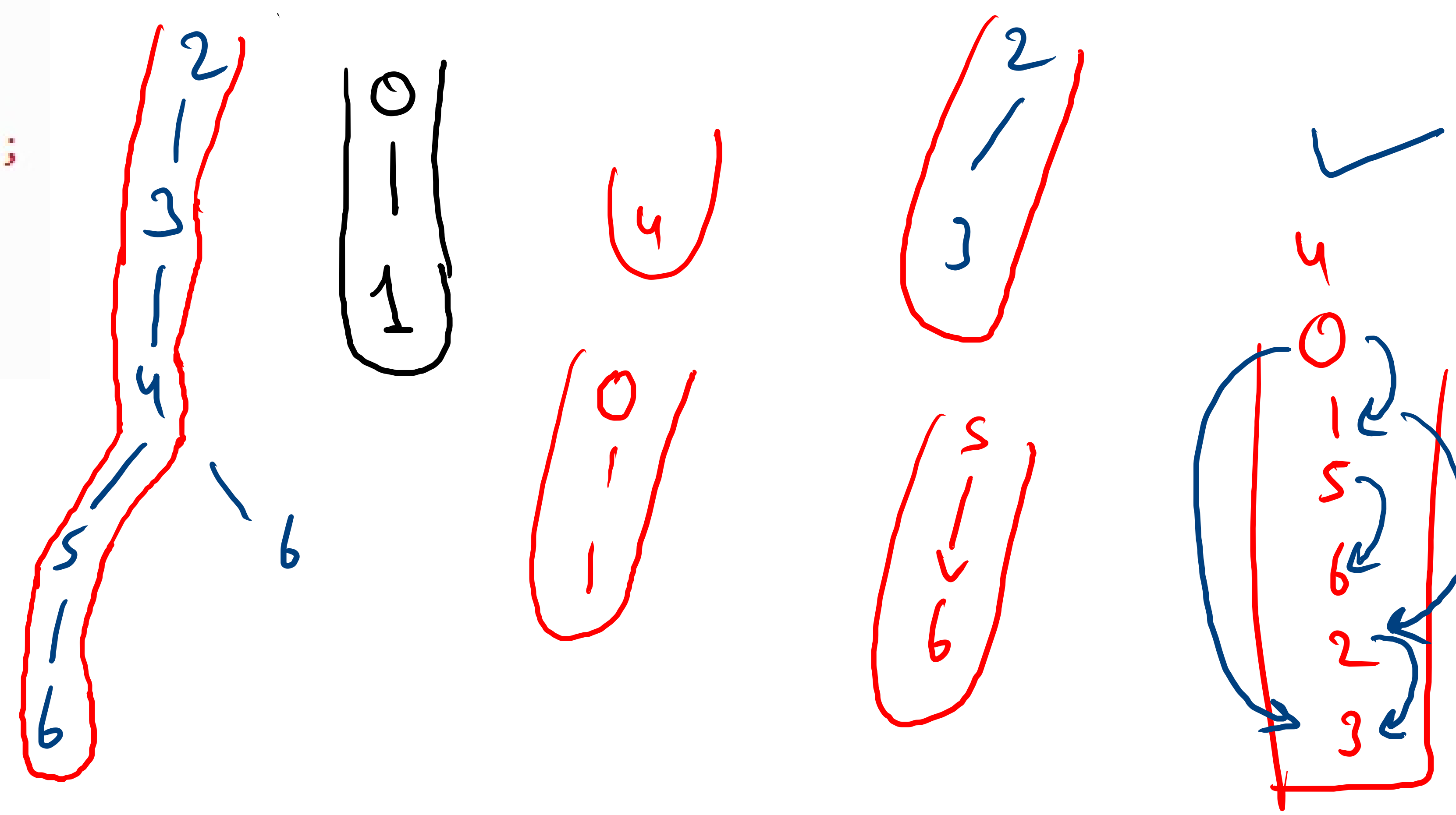
4
5
6
0
1
2
3

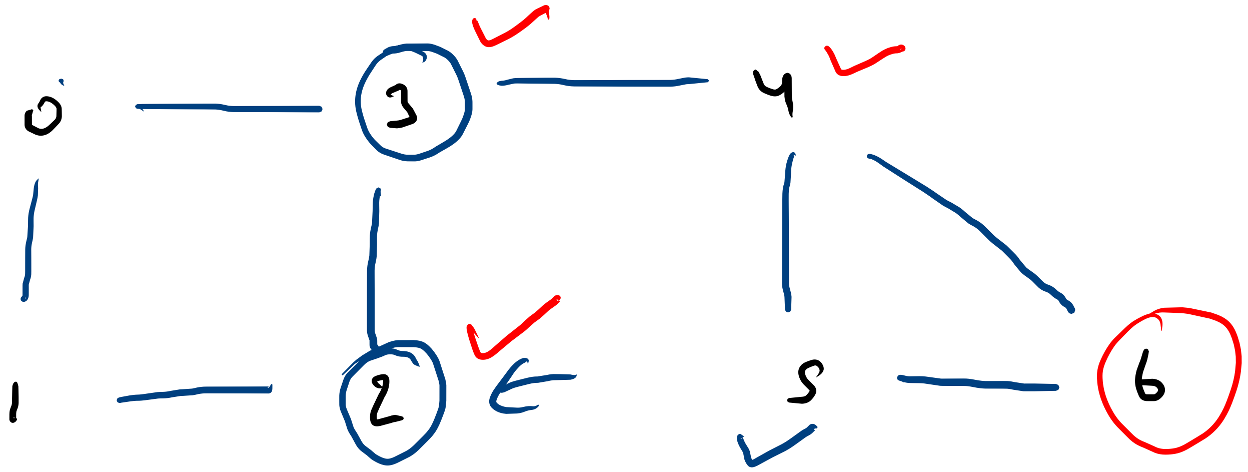


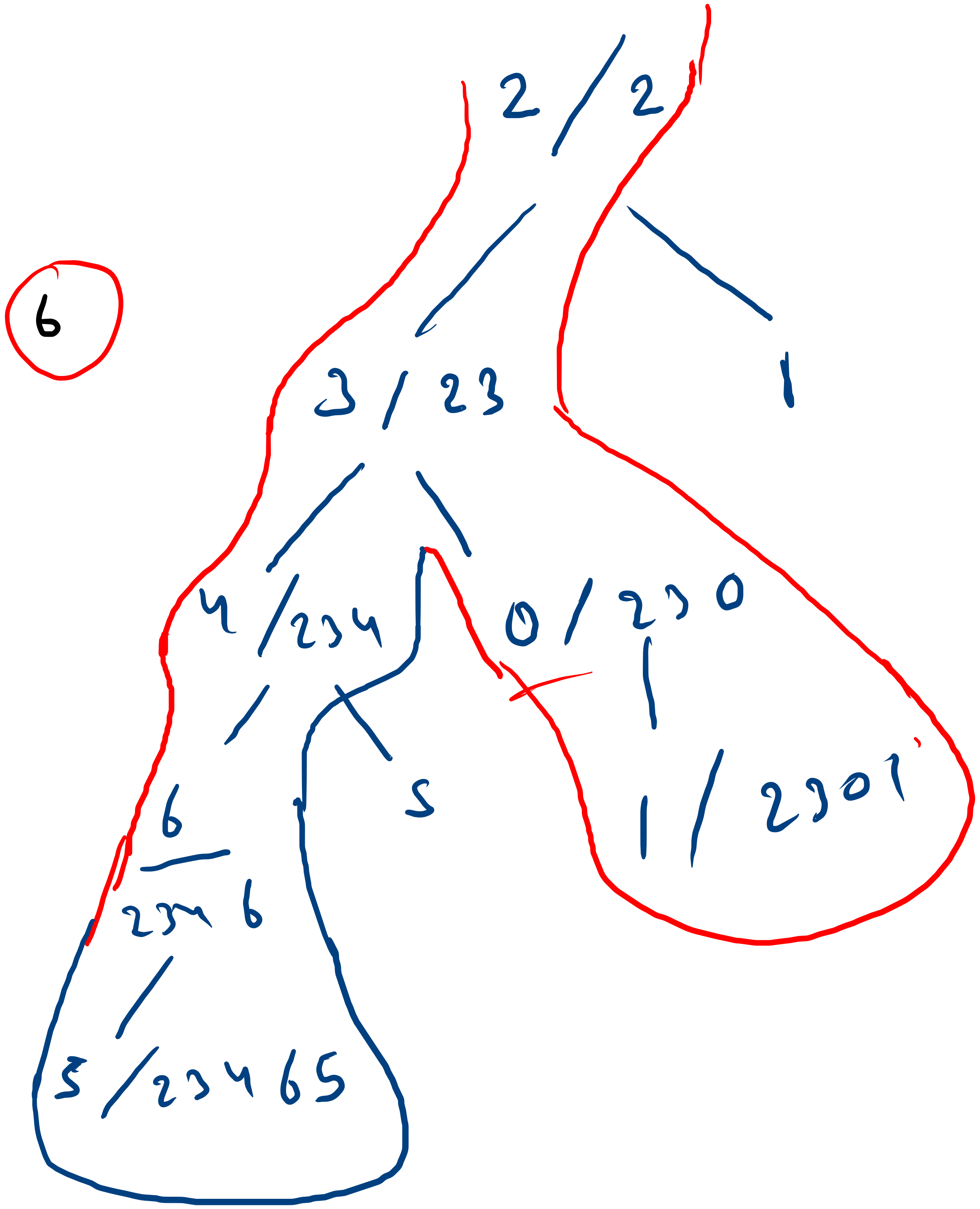


```
public static void dfs(ArrayList<Edge>[] graph,

    visited[src] = true;

    for(Edge edge: graph[src]){
        if(visited[edge.nbr] == false){
            dfs(graph, edge.nbr, visited, st);
        }
    }
    st.push(src);
}
```

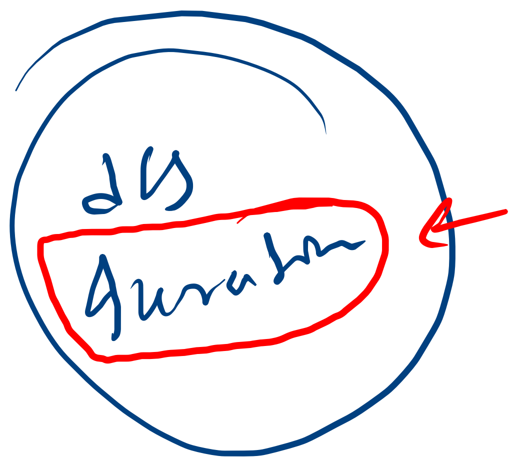




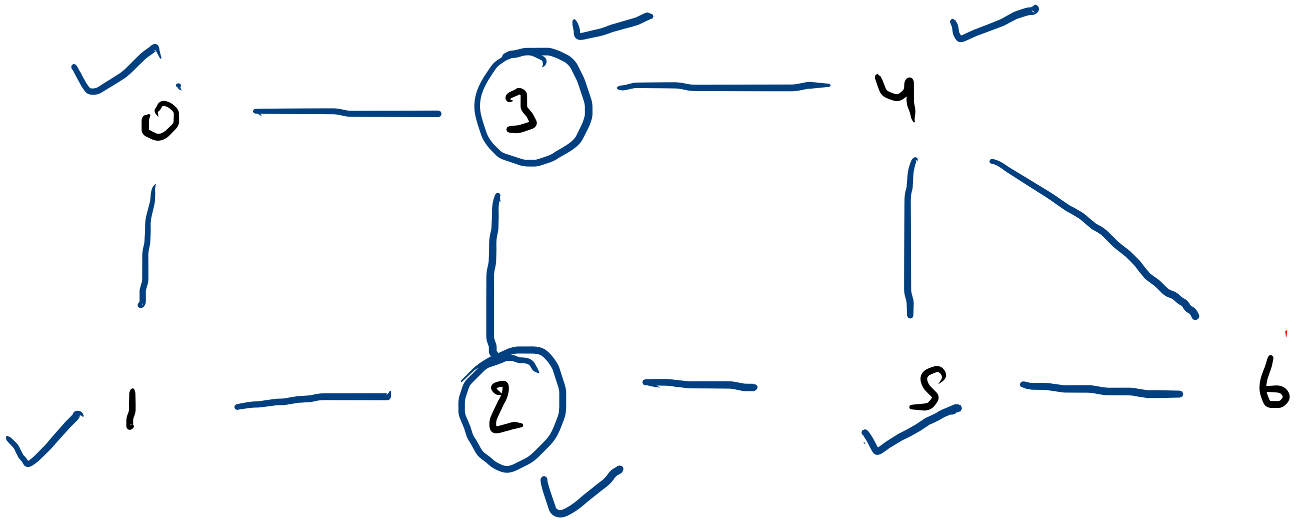


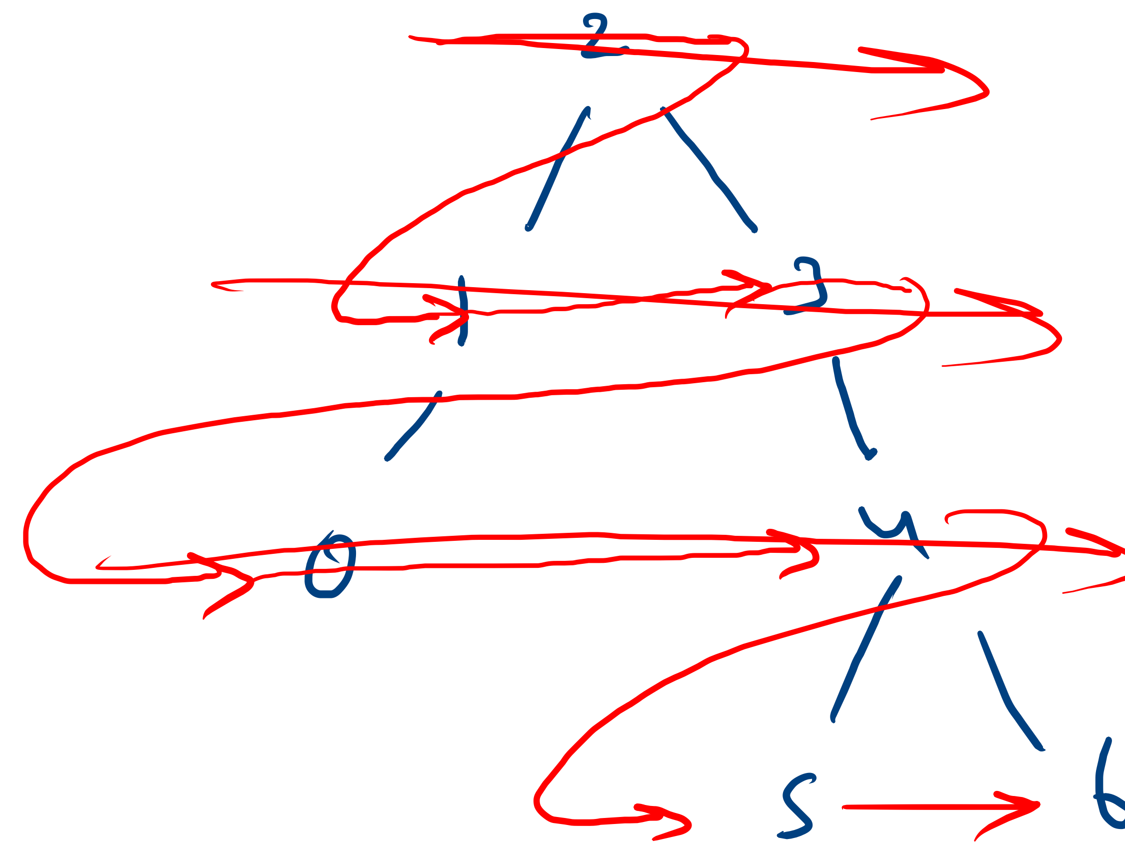


src → 2

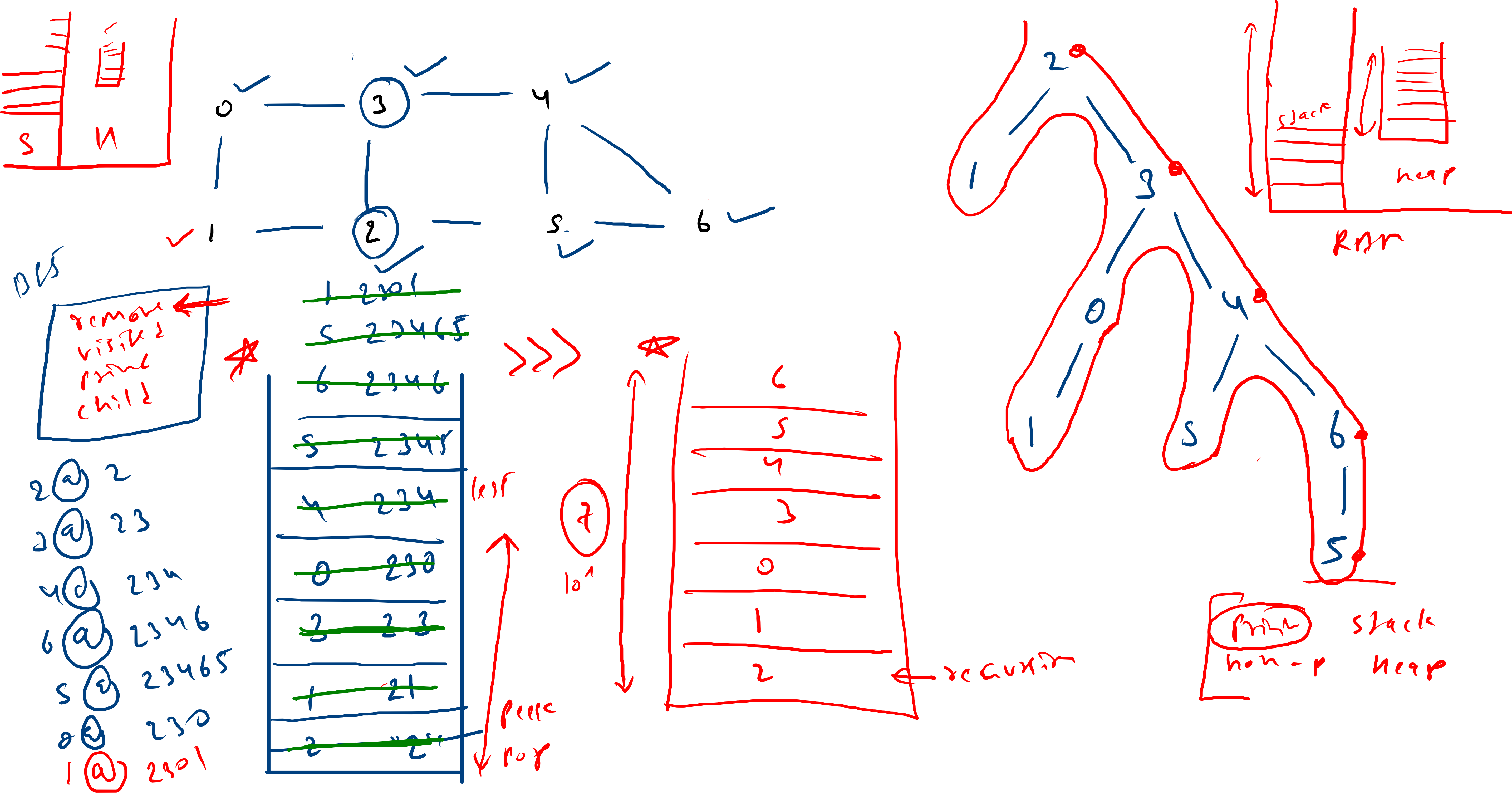


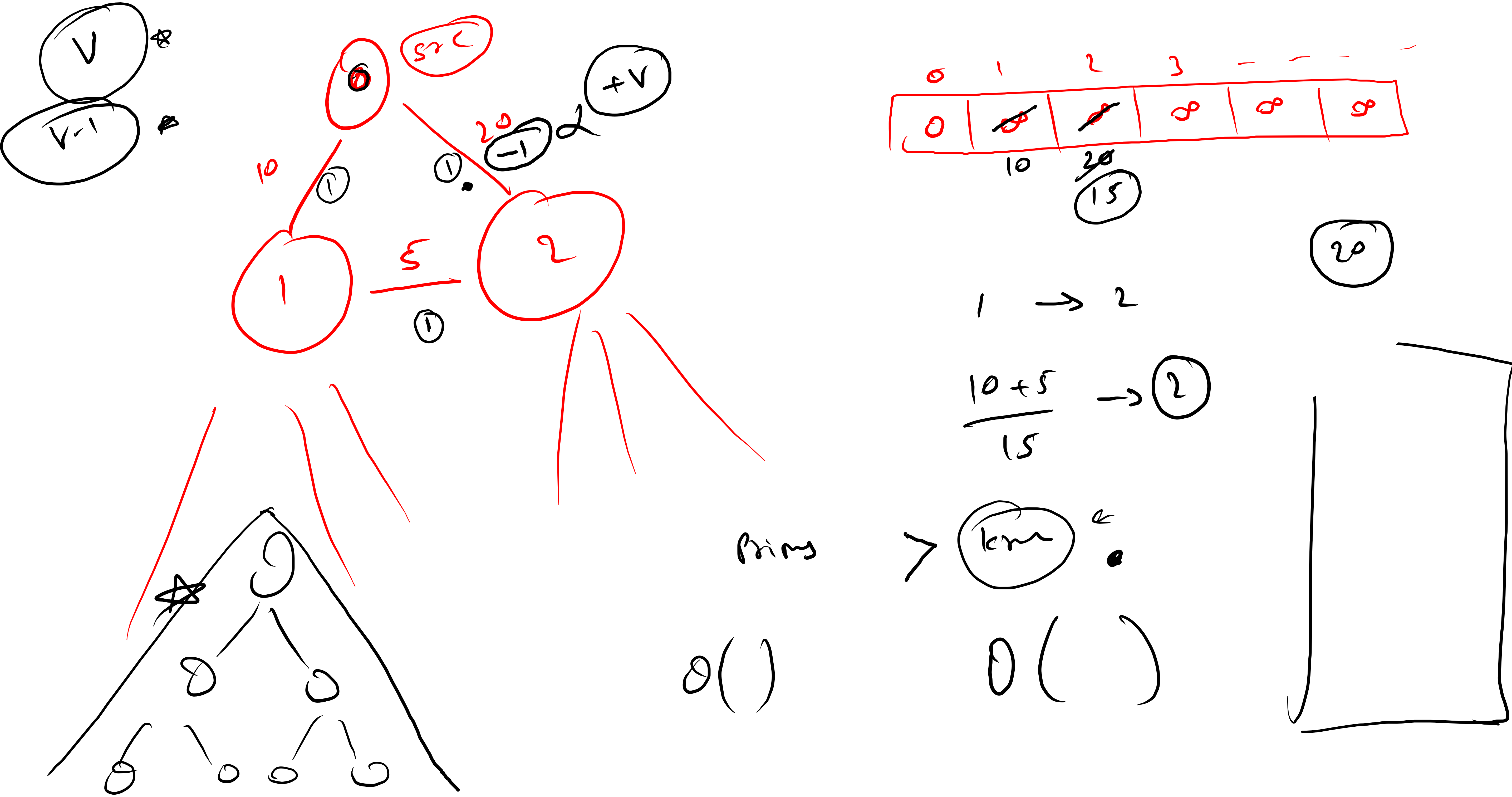
- 2@2
- 3@23
- 4@234
- 6@2346
- 5@23465
- 0@230
- 1@2301

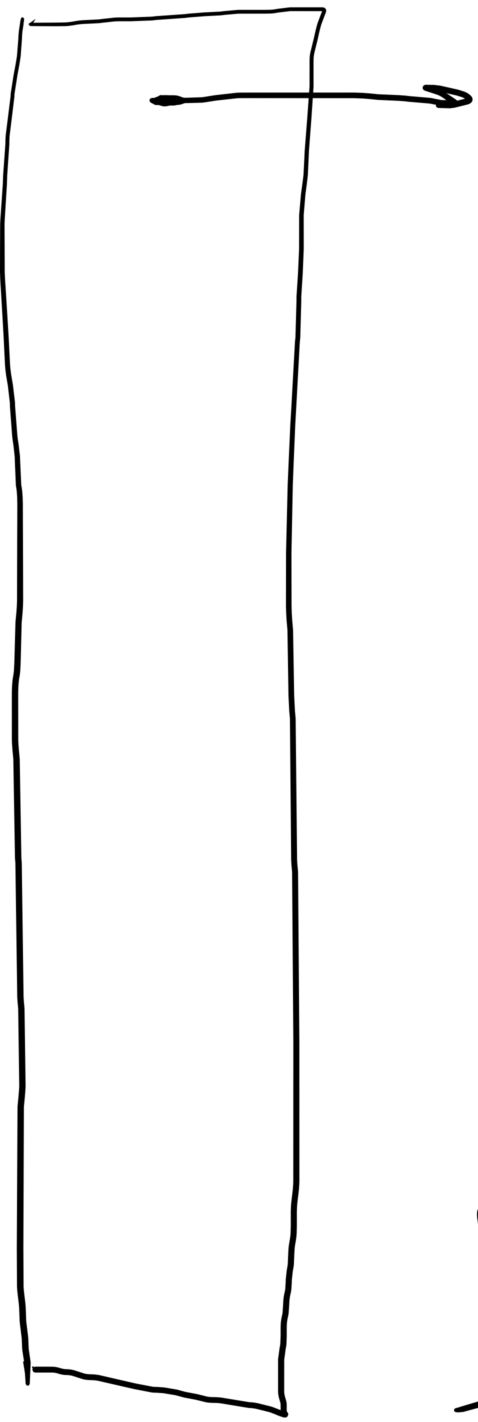




remove
visited
print
child


DFS
remove
visited
print
child
last
peek
pop
recursion
stack
heap
RAM
print
non-p

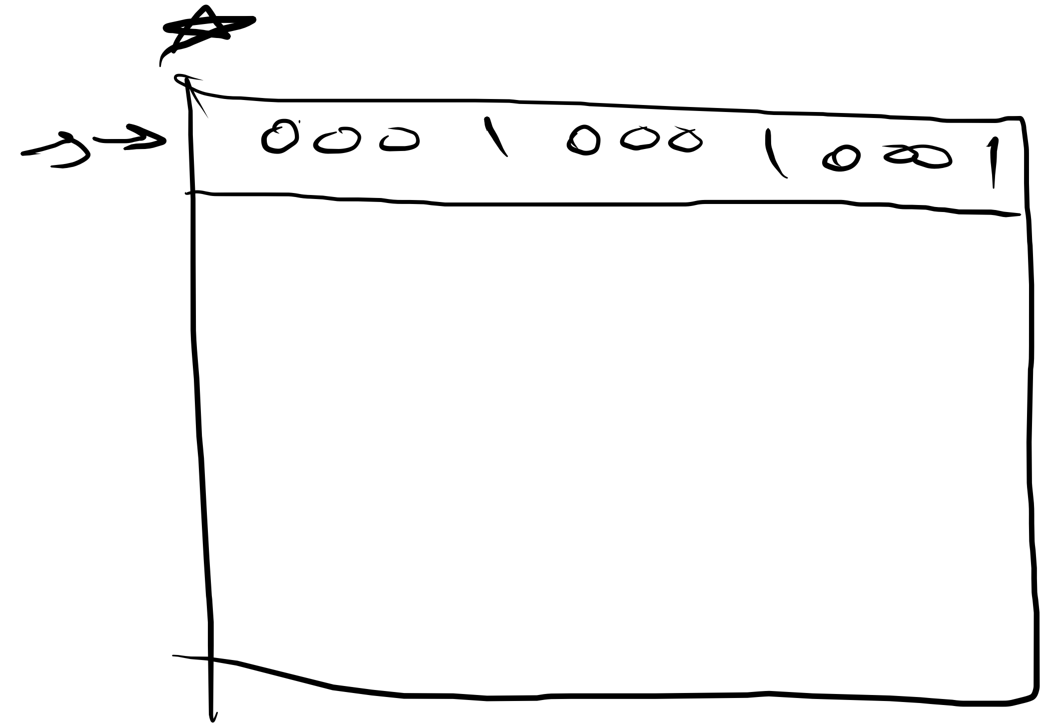

V
V-1
src
0
+V
20
-1
10
1
5
2
0 1 2 3
0 ∞ ∞ ∞ ∞ ∞
10
20
15
20
1 → 2
10+5
15
→ 2
Prims
O( )
O( )



[[else] [ ], [ ]]

array exact O(1)

mem level

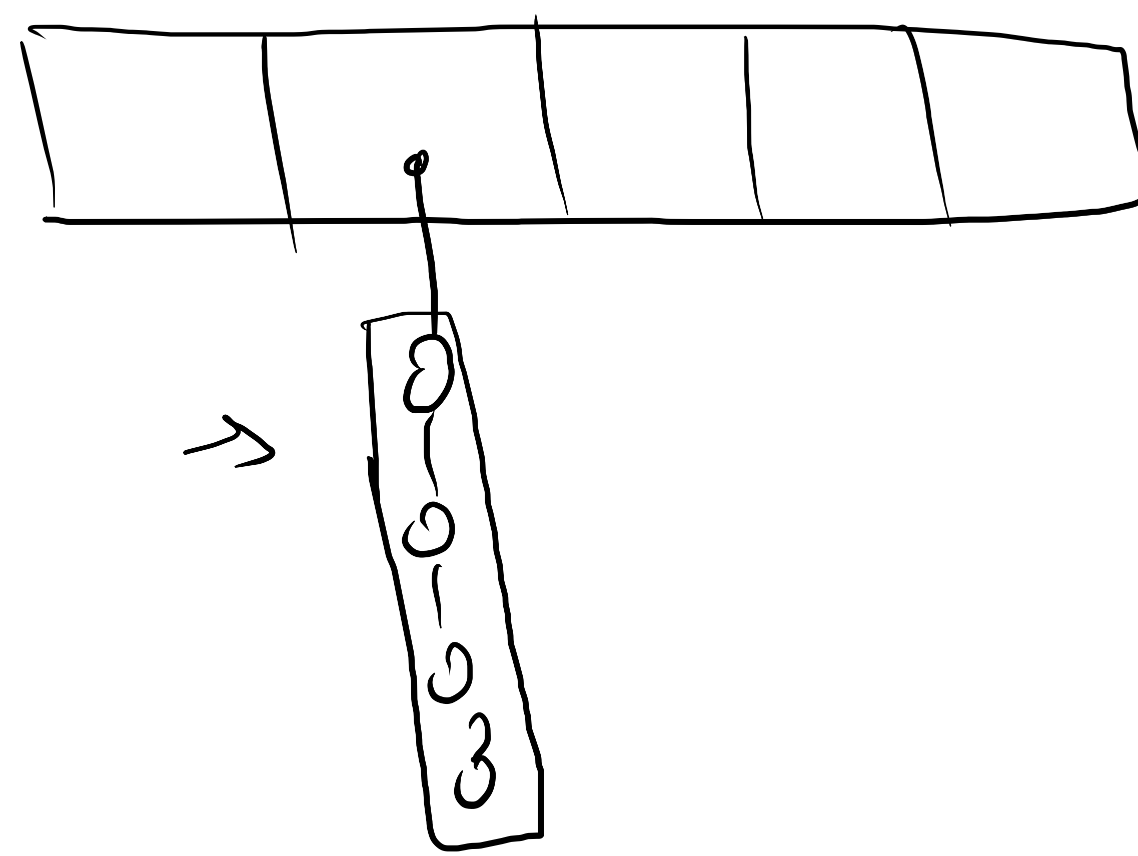


$O(\lambda)$

$O(1)\ \alpha$

key → value

< space
< time



Problem

A, B C D

resource

time

space

B code

metric

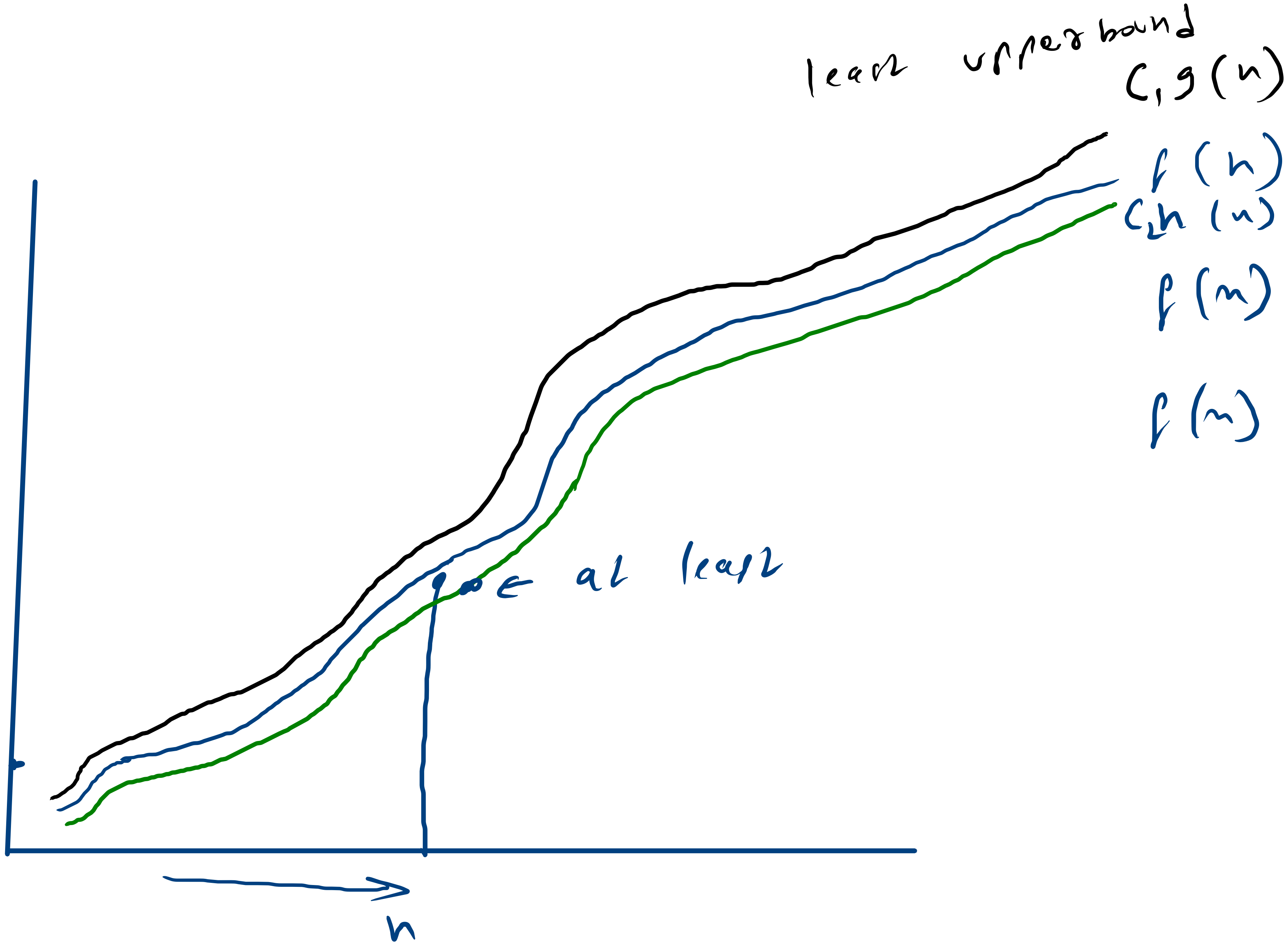
O Ω Θ

worst best average

resour

$$f(n) \geq c_2 h(n)$$

$$f(n) = \Omega(h(n))$$



$$f(n) \leq c_1 g(n)$$

$$f(n) = O(g(n))$$

Linear Search

[3 1 5 2 8 10 9 7] n

$d == 3$ 1

$d = 3$

comp → 1

$\Omega(1)$

$d = 20$

$O(n)$

$\Theta\left(\frac{n}{2}\right)$

TCS

Iteration

recursion

[code] repeat

n -> 10   100 sec

n -> 20   100 sec $\boxed{100 \text{ second}}$

n -> 10^9   100 sec

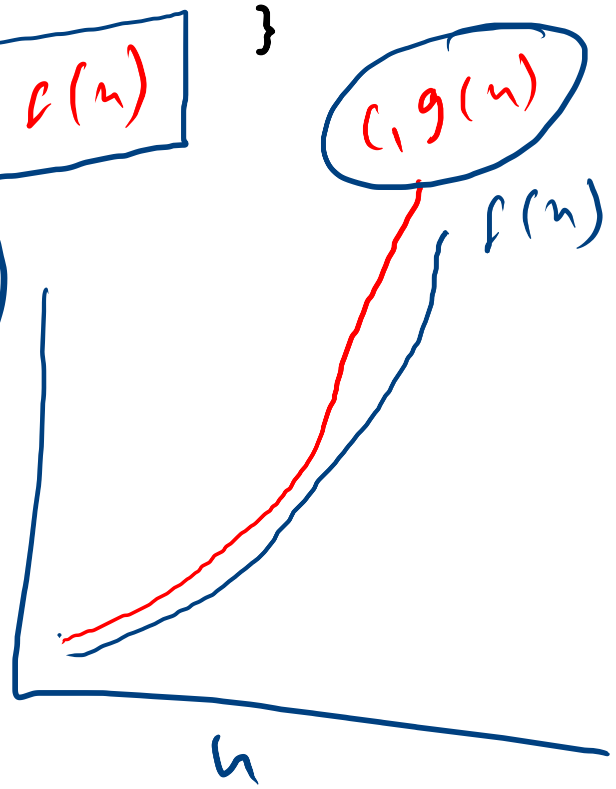
```
for( i=1 ; i<=100 ; i++ ){
    print("hello"); }    K
}
```

$K \rightarrow O(1)$

```
for( i=1; i<= n; i++){
    for( j = 1; j <= n; j++){
        print ("hello") ; } K
    }
}
```

$\boxed{c_1 g(n) \geq f(n)}$

$O(g(n))$



i	j works
1	$n \times k$
2	$n \times k$
3	$n \times k$
⋮	⋮
n	$n \times k$

n

$n \times k + n \times k + n \times k \ldots$ n times

$n(n \times k)$

$f(n) \Rightarrow n^2 k$

$O(n^2)$

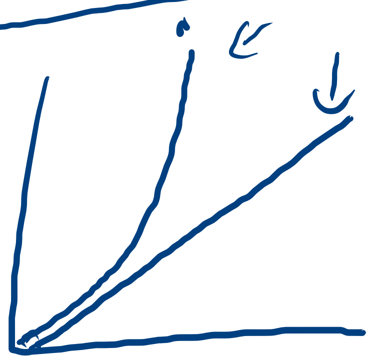
$n^2 k \leq c_1 g(n)$

$c_1 = k$

$g(n) = n^2$

$O(n^2) \to 89\%$

$O(1) \to 11\%$



```
for( i=1; i<= n; i++){
    for( j = 1; j <= i; j++){
        ≡  }  K
    }
}
```

n times

i	j	work
1	1	1K
2	2	2K
3	3	3K
⋮		
n	n	nK

$1k + 2k + 3k \cdots\cdots nk$

$k(1+2+3+4+\cdots\cdots n-1+n)$

$k\left(\frac{n(n+1)}{2}\right)$

$f(n) = \frac{K}{2}(n^2+n)$

$c_1 g(n) = \frac{K}{2}(n^2+n^2) = \frac{K\cancel{2}}{\cancel{2}} n^2$

$c_1 g(n) = k n^2$

$c_1 = k$

$g(n) = n^2$

$\boxed{O(n^2)}$

n^2 , $O(n^2)$

$f(n)$ $O(g(n))$

$f(n) \le c_1\, g(n)$

$O(n^2)$

n^2+n n^2+5n $n^2+10^{16}n$

i = 0*2

0

```
for (i=0; i<= n; i=i*2){
    = k
}
```

i

0

0

0

:

0

:

infinite

$k \infty$

infinite

none

K

k

k

k

O(n/2) → 25%

log(n) → 25%

O(n) → 13%

nlog(n) → 13%

none dah → 25%

$n/2 \to$ 43 f

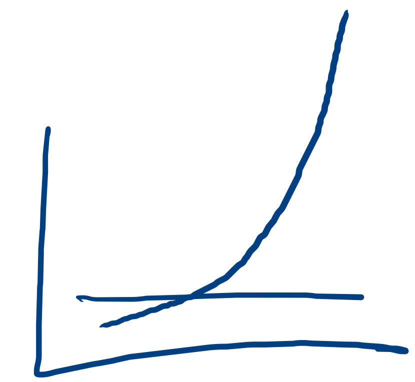
$\log(n) \to$ 49 f

$n \to$ 14 f

```
for (i=1; i<= n; i=i*2){
    ≡ } k
}
```

	i	work
$2^0 \to$	1	k
2^1	2	k
2^2	4	k
2^3	8	
$\vdots$	$\vdots$	
2^m	$n \le n$	k

m



$$2^m = n$$

$$\log_2 2^m = \log_2 n$$

$$m \log_2 2 = \log_2 n$$

$$\boxed{m = \log_2 n}$$

$$\log_2 n \to (m+1)$$

$$k + k + k \ldots\ldots \quad (m+1)$$

$$k(\log_2 n + 1)$$

$$f(n) \le k\log_2 n + k$$

$$c_1 g(n) = \boxed{2k} \log_2 n$$

$$c_1 g(n) \ge k\log_2 n + k$$

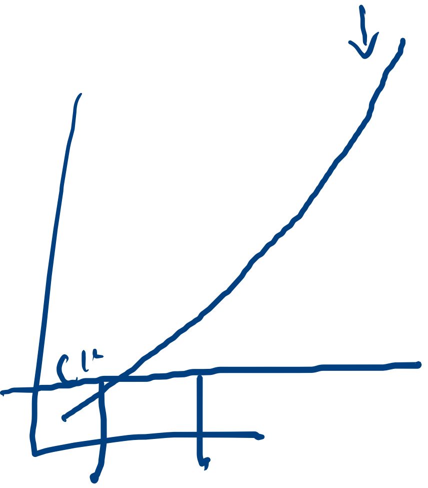
$$c_1 g(n) = k\log_2 n + k\log_2 n$$

$$O(\log_2 n)$$

n → variable (0 ∞)

k → constant

```
for( i=0; i<=n; i += k){
    for( j = 1; j<= k; j++){
        C
    }
}
```



	i	j	work
1	0	k	c k
m	1k	k	kc
	2k	k	kc
	3k	k	kc
	⋮		
	mk ≤ n	k	kc

$(m+1)\,k\,c$

$g(n) = nc + kc$

$= nc + nc$

$= 2nc$

$= \underbrace{2c}\ n$

$f(n) =$

$f(n) = (m+1)\,k\,c$

$\left(\frac{n}{k} + 1\right) kc$

$nc + kc$

$mk = n$

$m = \frac{n}{k}$

n/k 17

nk/2 17

nlog(n) 17

none of them 33%

infinite 17

- comparison
  - Bubble
  - select
  - Insertion
- Divide & conq
  - merge sort
  - quick sort
- —
  - count sort
  - radix sort

n = 5

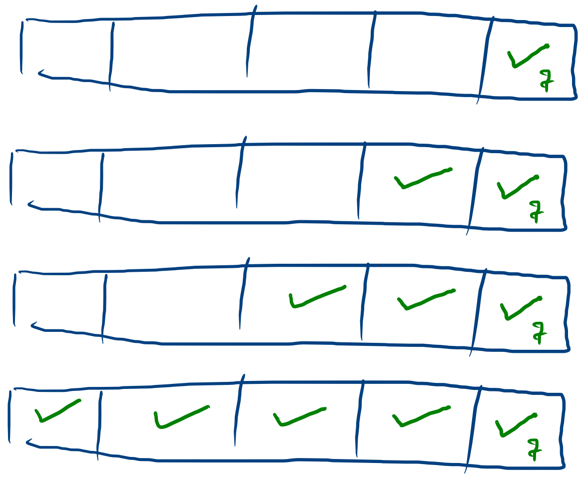
7 -2 4 1 3

7 -2 4 1 3 → -2 1 3 4 7

Order Ascending

iter 1 2 3 4

n-1 →



7 -2 4 1 3

-2 7 4 1 3

-2 4 7 1 3

-2 4 1 7 3

-2 4 1 3 7

1 iter

-2 4 1 3 7

-2 1 4 3 7

-2 1 3 4 7

2

---

3 iter

-2 1 3 4 7

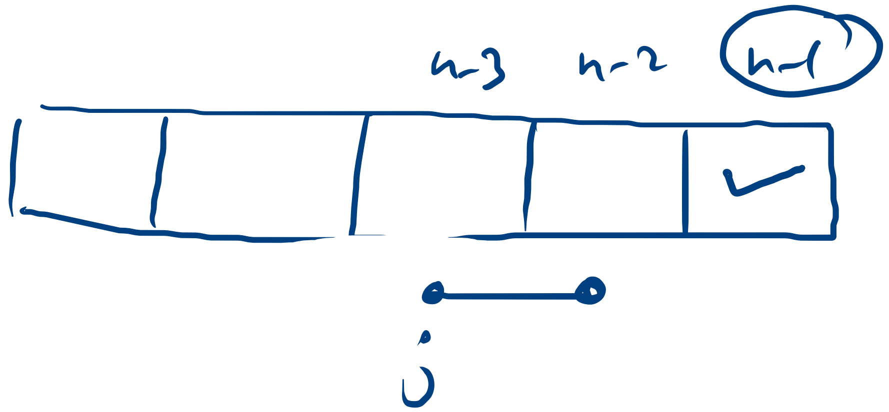
-2 1 3 4 7

-2 1 3 4 7

-2 1 3 4 7

4 iter

iter
1



iter 1 to n-1

j 0 ... n-iter-1

iter	j
1	0 .... n-2
2	0 ... n-3
3	0 ... n-4
iter	0 .... n-(iter+1)
	n-iter-1

$\boxed{O(n^2) = \Omega(n^2)}$

```java
int n = arr.length;
for(int iter=1;iter<=n-1; iter++){
    for(int j=0;j<=n-iter-1; j++){
        if(isSmaller(arr, j+1, j)){
            swap(arr, j+1, j);
        }
    }
}
```

k

$0 \ldots n-(n-1)-1$

$0 \ldots \cancel{n-n+1-1}$
0

iter		work
	$0 \ldots n-iter-1$	
1	$0 \ldots n-2$	$(n-1)k$
2	$0 \ldots n-3$	$(n-2)k$
3	$0 \ldots n-4$	$(n-3)k$
$n-1$	$0 \ldots 0$	$1k$

$1k + 2k \ldots\ldots (n-3)k + (n-2)k + (n-1)k$

$k\big(\underbrace{1+2+3 \ldots (n-2)+(n-1)\;+n}\big) - n$

$O(n) = \frac{k}{2}\left[\frac{n(n+1)}{\cancel{2}} - 2n\right] = \frac{k}{2}\left[n^2 - n \cancel{\;}\right]$

$= \cancel{\frac{k}{2}}\left[n^2\right]$

$\boxed{O(n^2)}$