STRUTTURE DATI

- contenitore
- n elementi

array

2	3	1	5	...

- alloc(n) O(n)
- dealloc() O(n)
- at(i) O(1)

vector - array din.

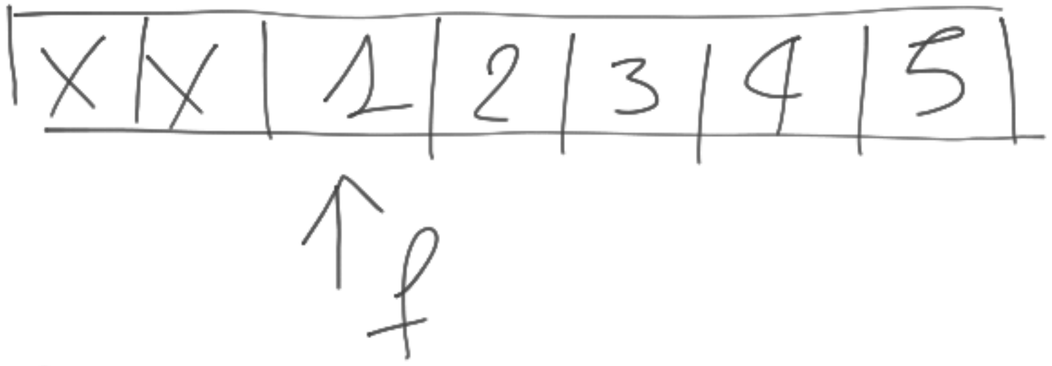
push_back(e)	O(1)	AMM
pop_back()	O(1)	
at(i)	O(1)	
resize(n)	O(n)	

$$1 + 2 + 4 + 8 + \ldots + 2^K = 2^{K+1} - 1$$

N

queue

push_back(e)	O(1)	A	→ v.push_back(e)
pop-front()	O(1)		→ ++f
front()	O(1)		→ V.at(f)

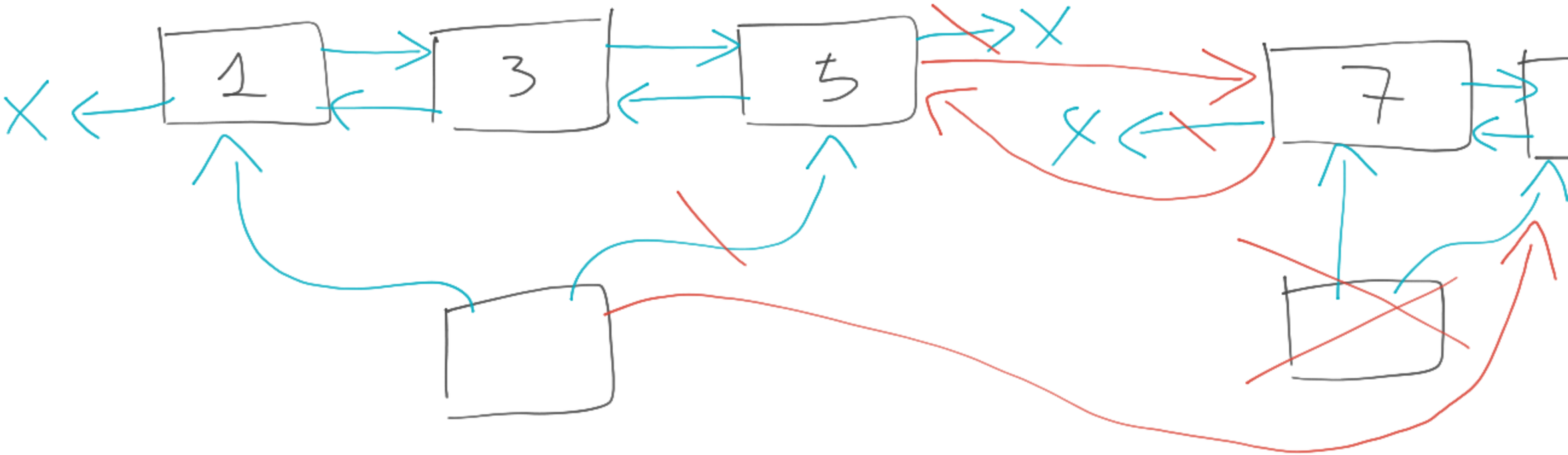


deque

push_back(e)
pop_back()
push_front(e)
pop-front()
} O(1)

LINKED LIST

push_back(e)	O(1)	concatenate(L)	O(1)
⋮			
at(i)	O(n)		



set

insert (e)	O(log n)
erase (e)	O(log n)
find (e)	O(log n)
min()	O(1)

hashset

O(1)
O(1)
O(1)
✗

map

insert (K, V)	O(log n)
erase (K)	O(log n)
find (K)	O(log n)
min()	O(1)

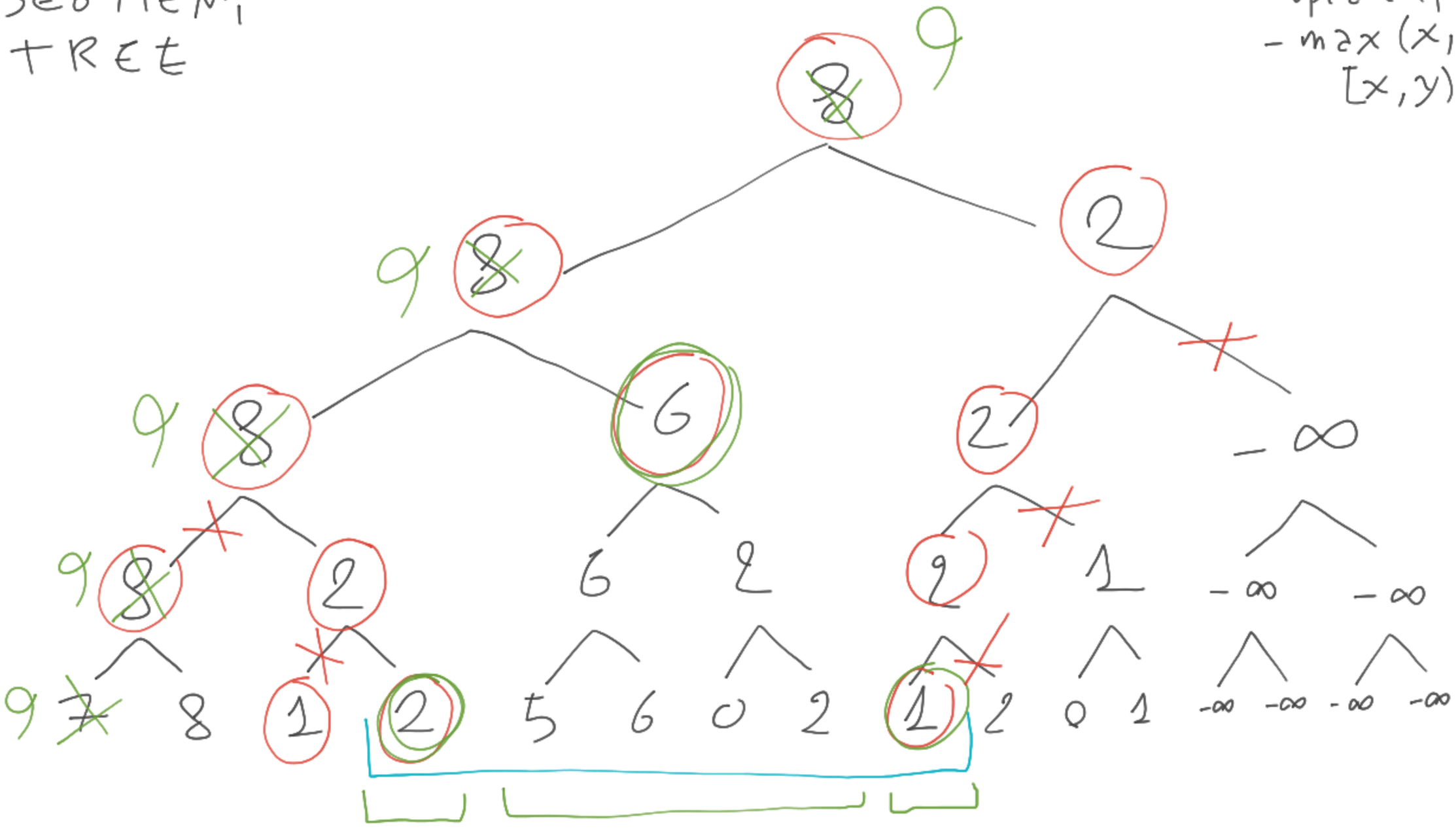
hashmap

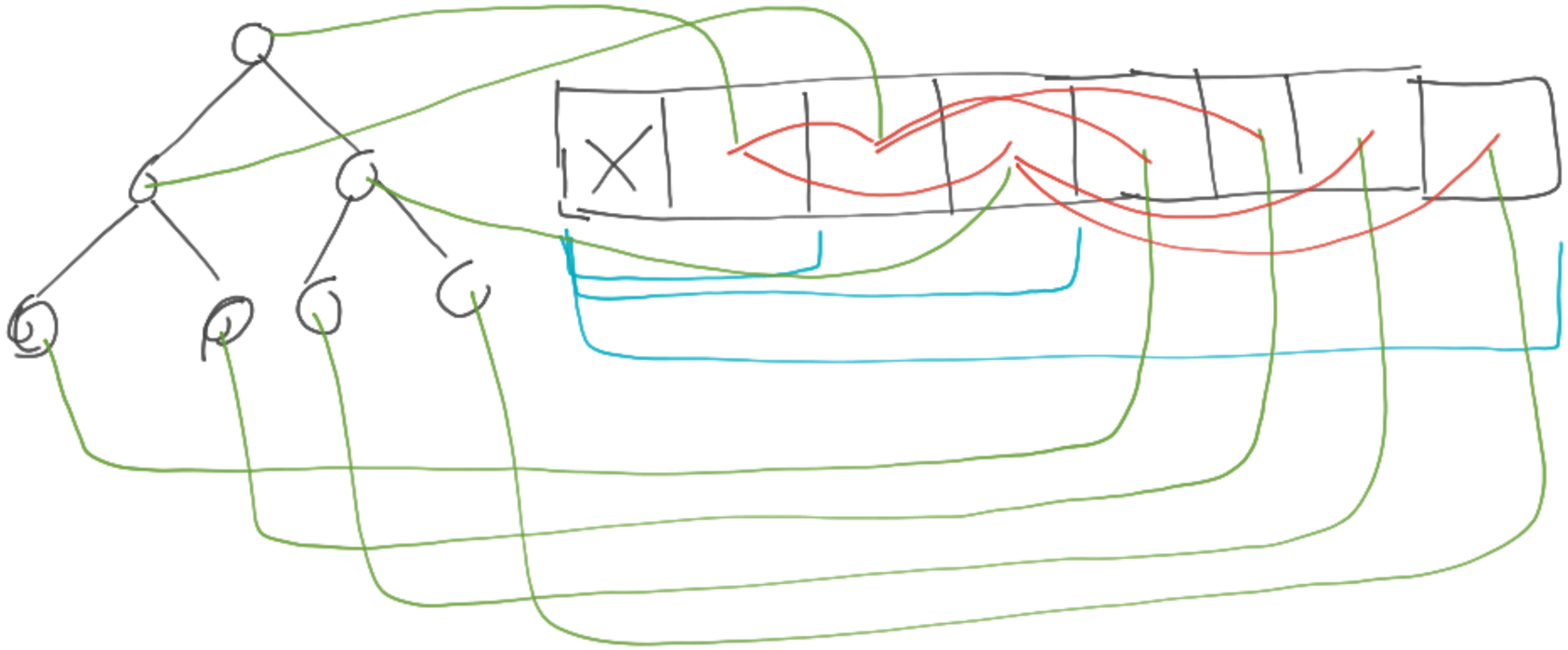
O(1)
O(1)
O(1)
✗

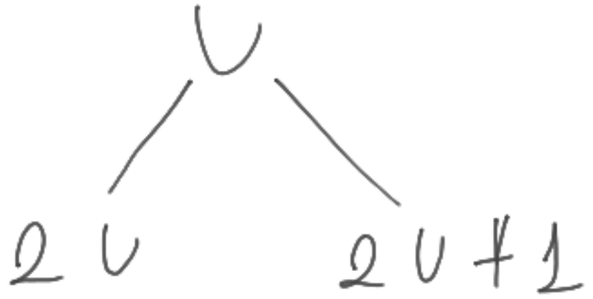
```
set<int, greater<int>> s;
```

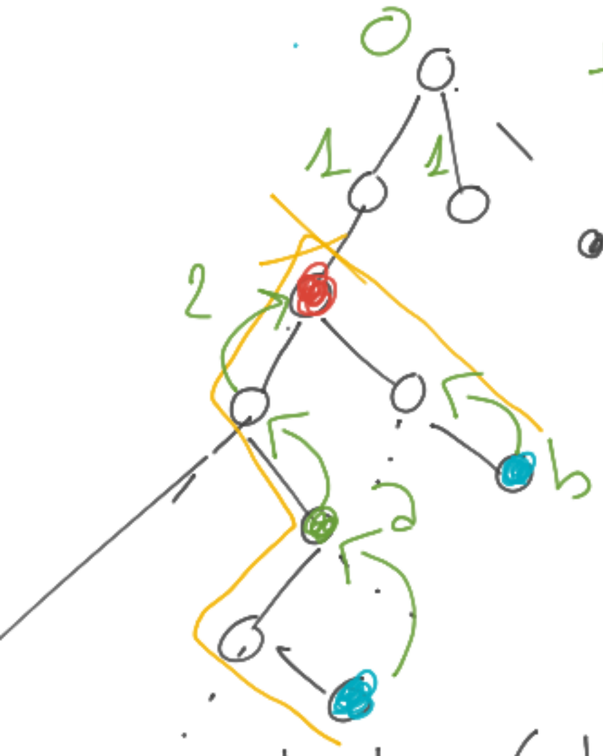
SEGMENT TREE

- update (p, v) O(log n)
- max (x, y) O(log
[x, y)






2 v
2 v + 1



- $ANC(v, x+y) = ANC(ANC(v, x), y)$
- $ANC(v, 2^k) = ANC(ANC(v, 2^{k-1}), 2^{k-1})$

GO_UP(v, p) p = 1 + 4 + 16 + ... 010101

```
for(i = logn; i ≥ 0; i--)
  if (p & (1<<i))
    v = ANC(v, 2^i)        ret v
```

precalcolo $O(n \log n)$
kth ancestor $O(\log n)$

LCA(v, d)
↳ v, x

LCA(a, b)

```
1. assumo d[a] ≥ d[b]
2. a = GO_UP(a, d[a] - d[b])      if(a==b) return a
3. for(i = logn; i ≥ 0; i--)
     if(ANC(a,i) ≠ ANC(b,i)):
       a = ANC(a,i)   } 2^i
       b = ANC(b,i)   }
   return ANC(a,0)
```

$ANC(v, d) \rightarrow \{a, x\}$

$ANC(v, 2^k).first = ANC(ANC(v, 2^{k-1}).first, 2^{k-1}).first$

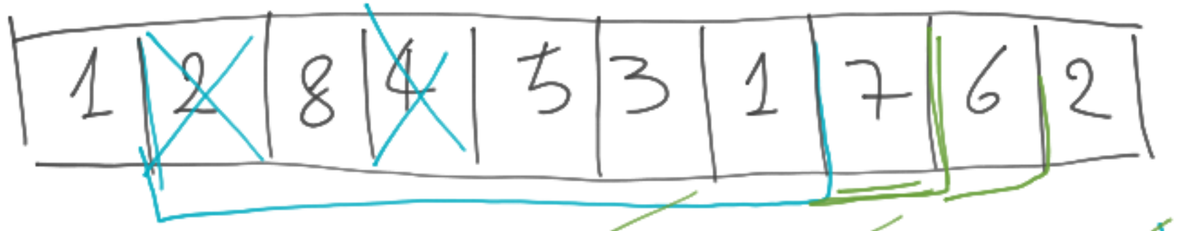
$ANC(v, 2^k).second = \max\{$

$ANC(v, 2^{k-1}).second,$

$ANC(ANC(v, 2^{k-1}).first).second$

$\}$

max queue



(8,2), (5,4), (3,5), (1,6) ⇐ (7,7)

(8,2), (7,7), (6,8)

- RIMOZIONE FINO A 1 => NON FACCIO NULLA
- " 2
- (7,7), (6,8)
- MAX 7

≤ O(1) AMM