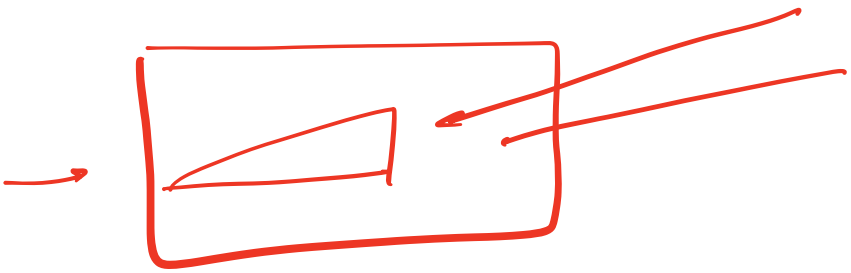
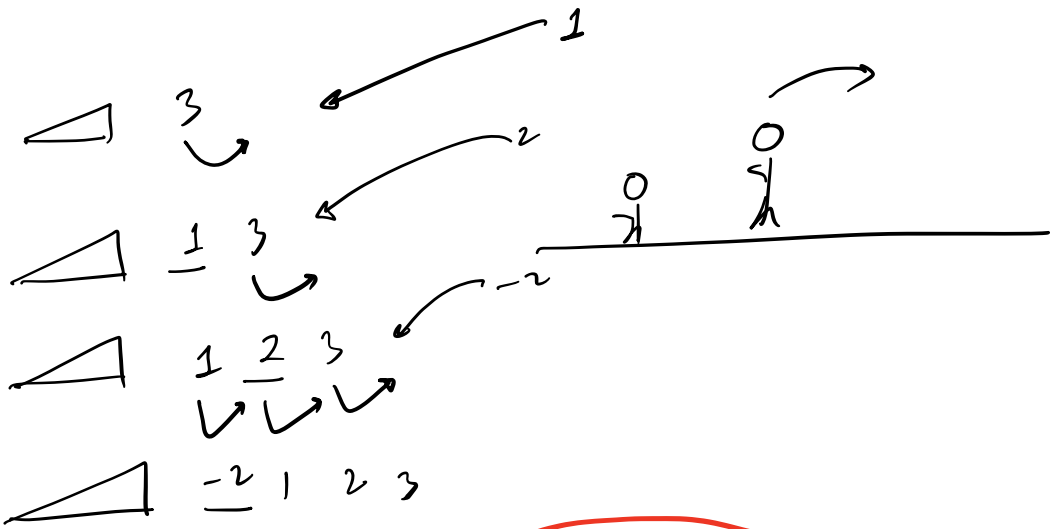
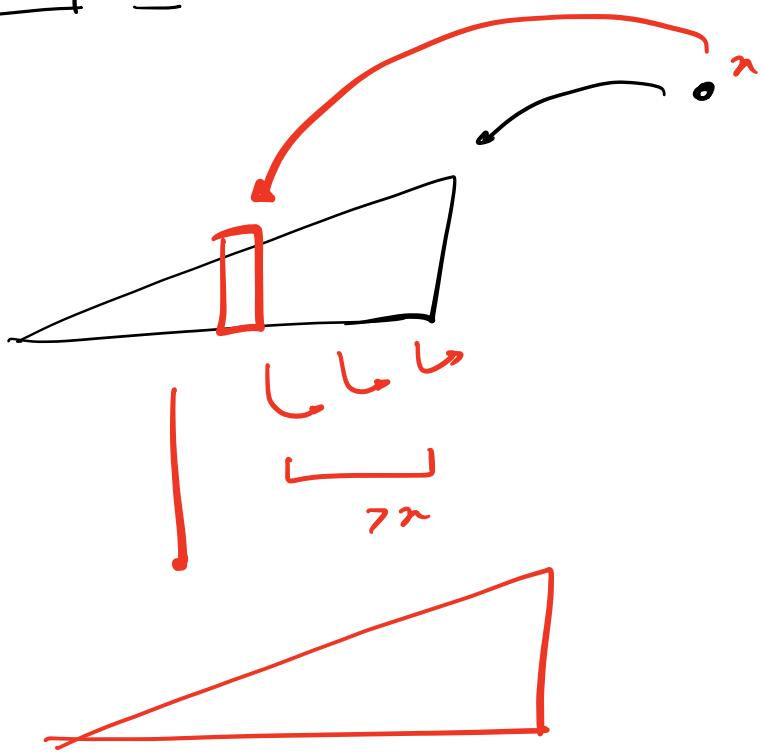
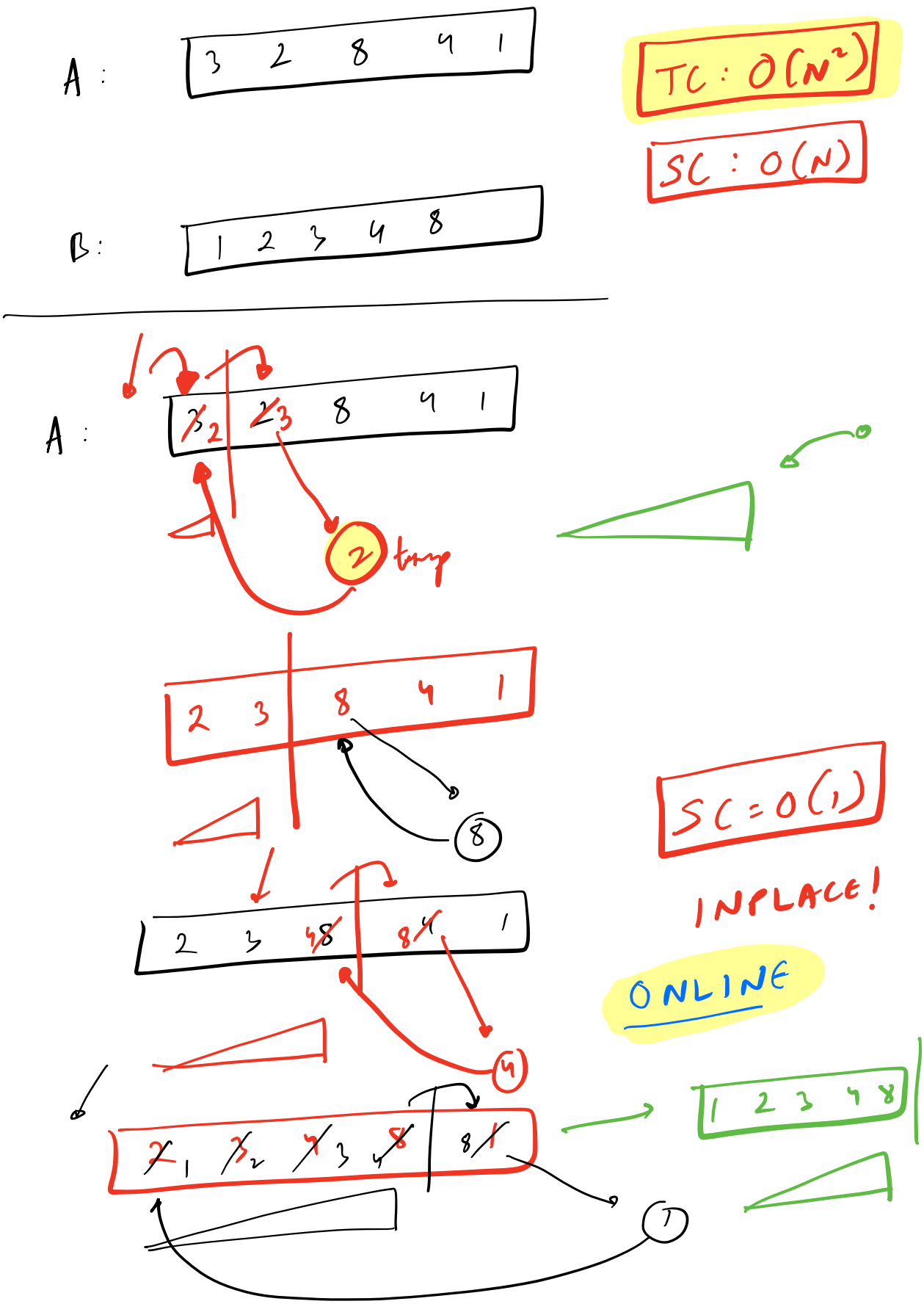
Insertion Sort

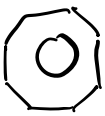







A : 3 2 8 4 1
TC : O(N²)
SC : O(N)
B : 1 2 3 4 8
A :
temp
SC = O(1)
INPLACE!
ONLINE

9 N nuts & N bolts

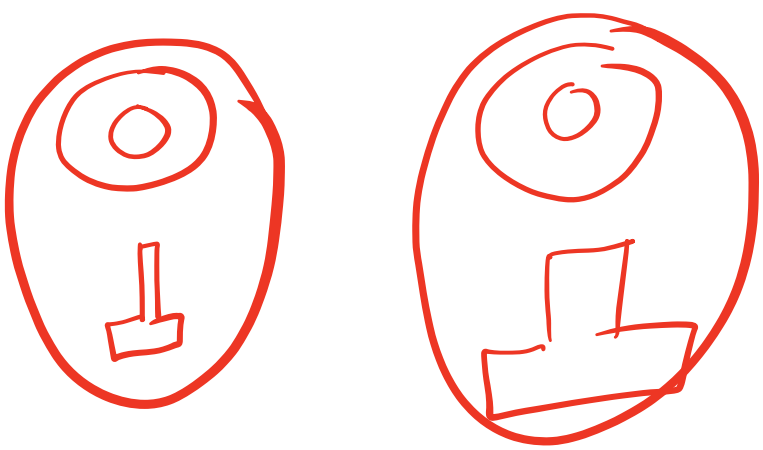


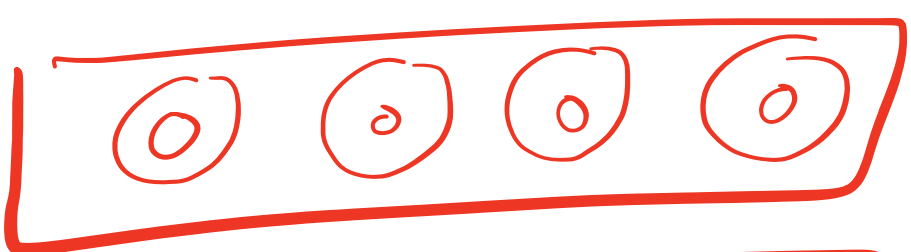
- Every nut is distinct in size.
- ~~—~~ bolt
- Every nut has a corresponding bolt!

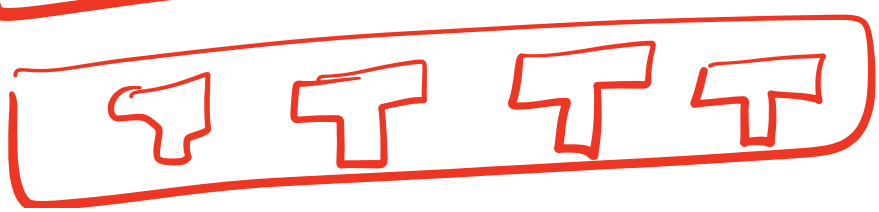
Sort the nuts & bolts in INC order of size!

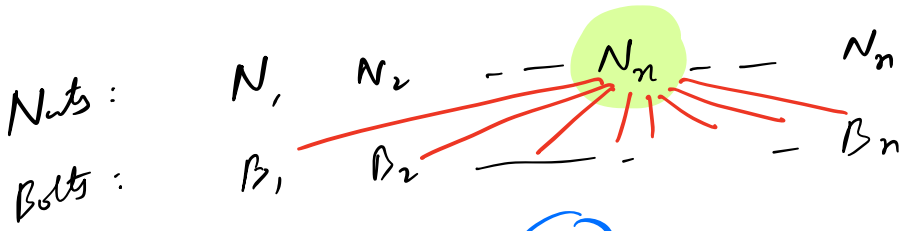
→ You cannot compare b/w the size of 2 nuts or 2 bolts

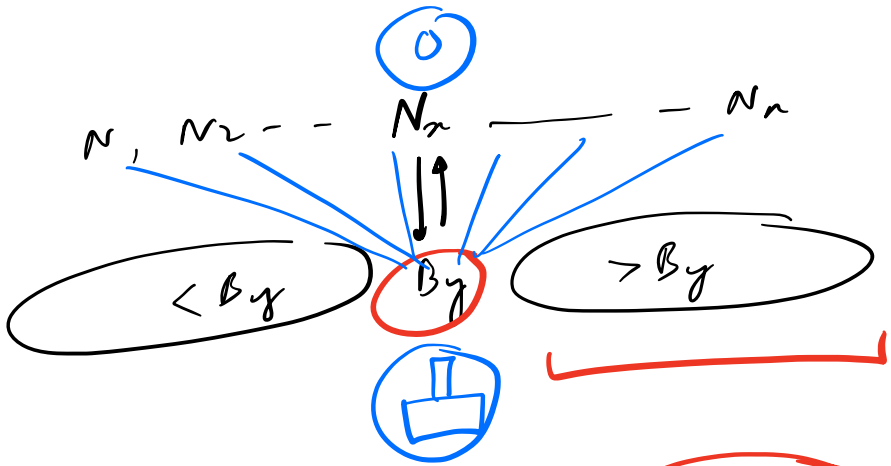
→ You can compare a nut & a bolt

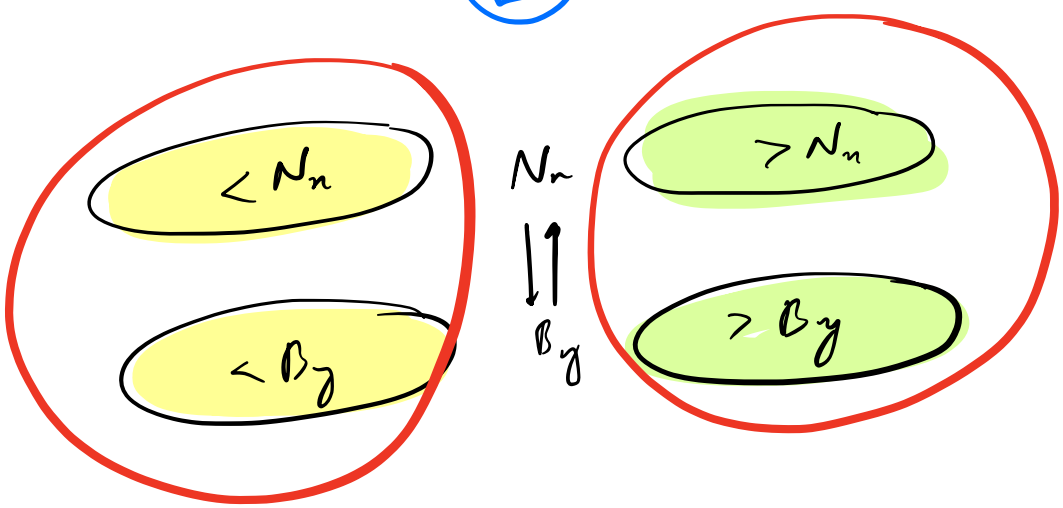












QUICK SORT : idea

Q: Given an Array.
Rearrange the array s.t.

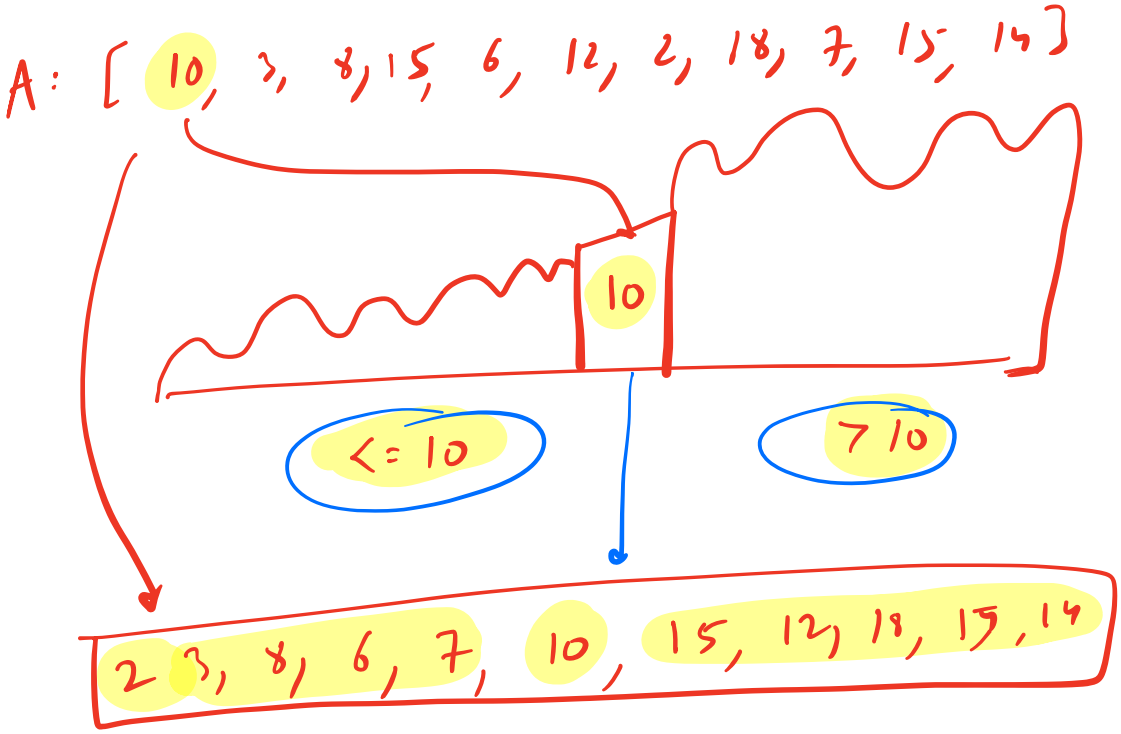
→ A[0] should go to it's right sorted position!

→ All the elements <= A[0] should go to the left of the right sorted pos of A[0]

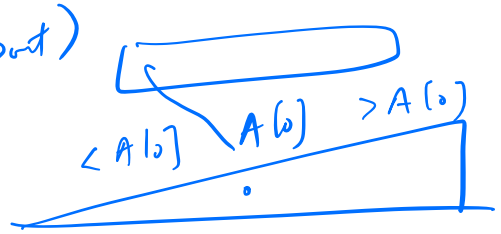
→ _____ >A[0] _____

right _____

_____

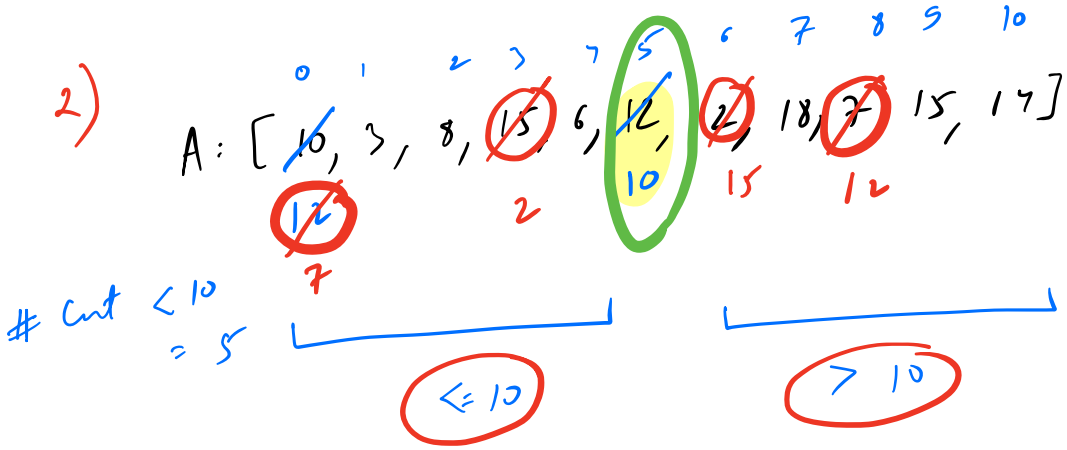


1) Sort (Merge Sort)

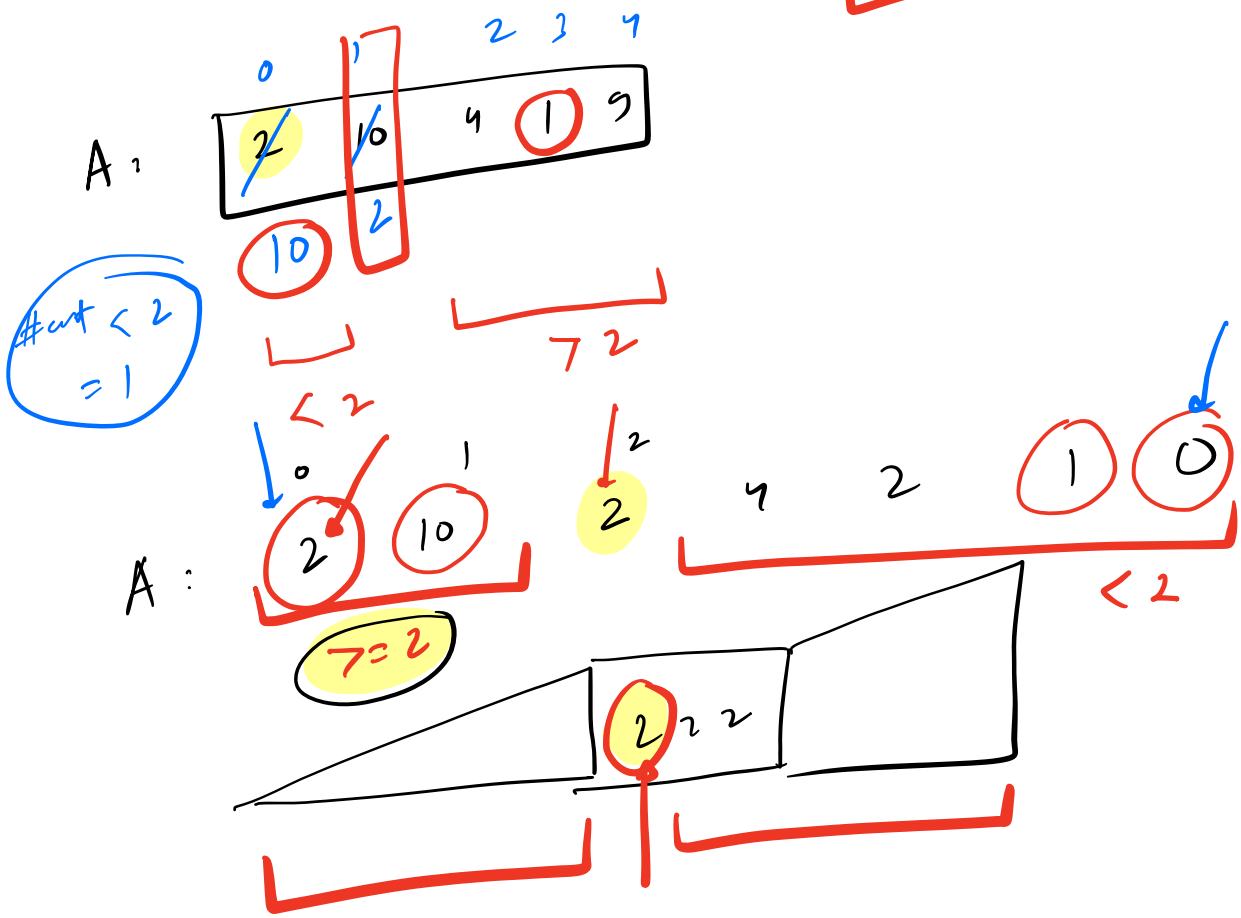


TC: O(N log N)

2)



TC = O(N)



PARTITIONING ALGO

```
// A [N];  cut = 0 , x = A[0];
for (i = 1; i < N; i++) {
    if ( A[i] < x ) cut++;
}
swap ( A[0], A[cut] );
P1 = 0,  P2 = N-1;
while( P1 < cut && P2 > cut) {
    while( A[P1] < x) P1++;
    while ( A[P2] >= x) P2--;
    swap( A[P1], A[P2]);
    P1++, P2--;
} ret cut;
```

TC = O(N)

SC = O(1)

Q Given A[], //[s, e]

move A[s] in it's right sorted place in the SA : A[s, e]

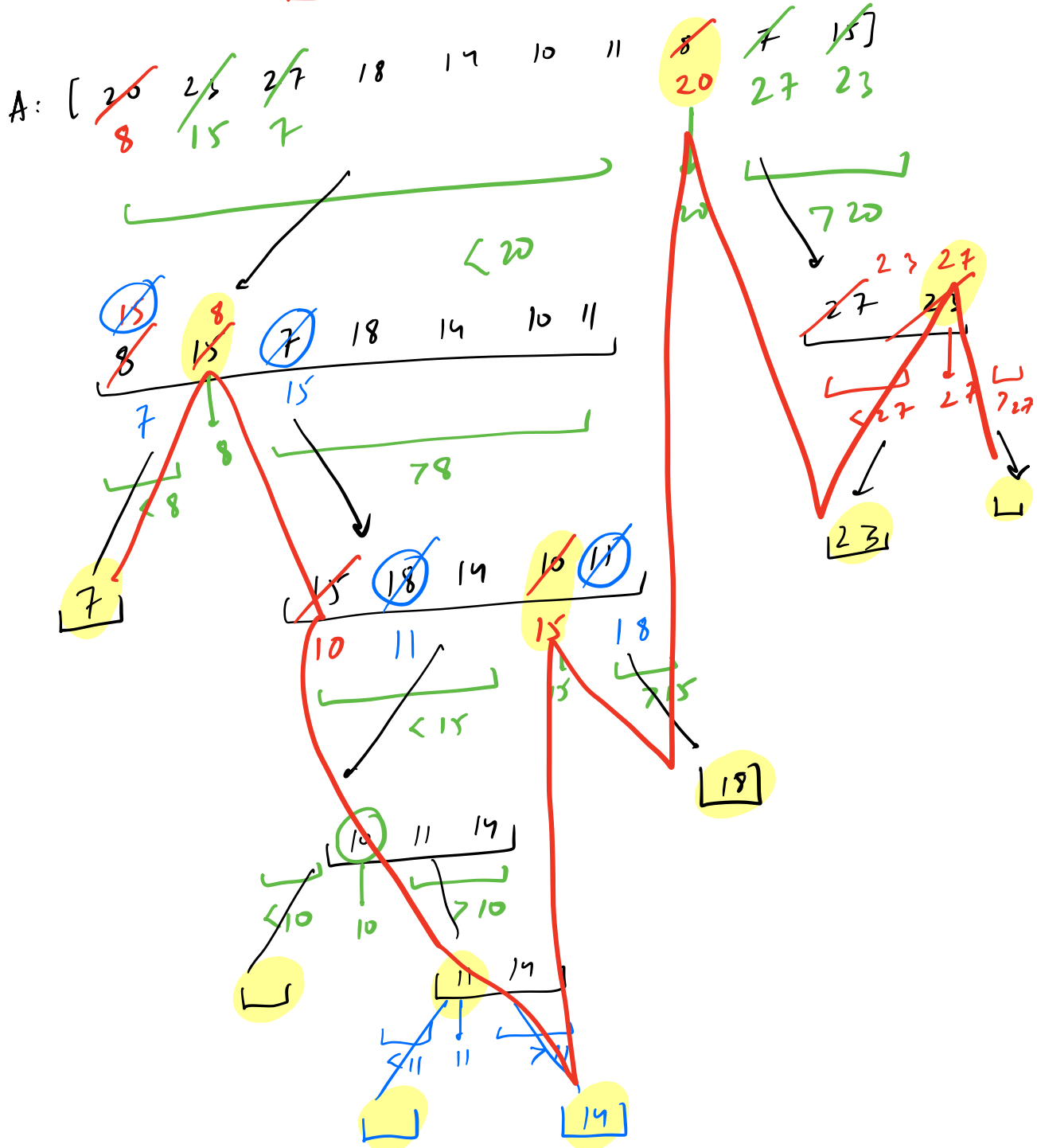
move left <= A[s]

— right > A[s]

```
// s, e
// A [N];  cnt = 0,  x = A[s];
for ( i = s+1; i <= e; i++) {
    if ( A[i] < x ) cnt++;
}
swap ( A[s], A[s+cnt]);
P1 = s,  P2 = e;
while( P1 < s+cnt && P2 > s+cnt) {
    while( A[P1] < x) P1++;
    while ( A[P2] >= x) P2--;
    swap( A[P1], A[P2]);
    P1++, P2--;
}
ret s+cnt;
```

int rearrange(A[], s, e)

QUICK SORT

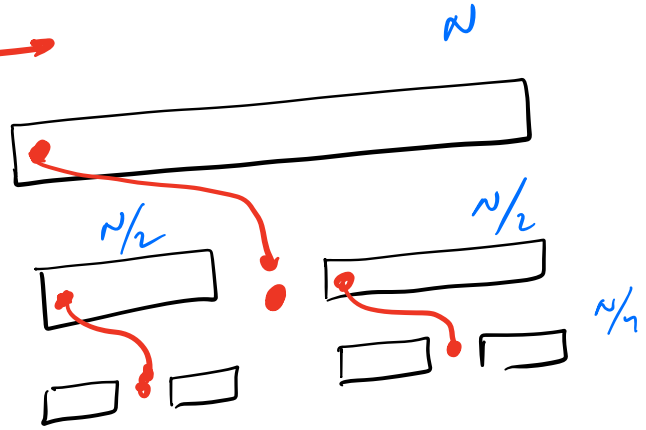

A: [20 23 27 18 14 10 11
8 15 7
20
27 23
< 20
> 20
15 8 7 18 14 10 11
8 15 7
7
< 8
8
15
> 8
23 27
27 23
< 27
> 27
23
7
15 18 14 10 11
10 11
< 15
15
18
> 15
18
10 11 14
< 10
10
> 10
11 14
< 11
11
> 11
14

```
void qSort( A[], s, e) {          A[], 0, N-1)
  if( s >= e) ret;
  p = rearrange(A[], s, e);
  qSort( A[], s, p-1);
  qSort( A[], p+1, e);
}
```

---

④ Complexity

1) CASE

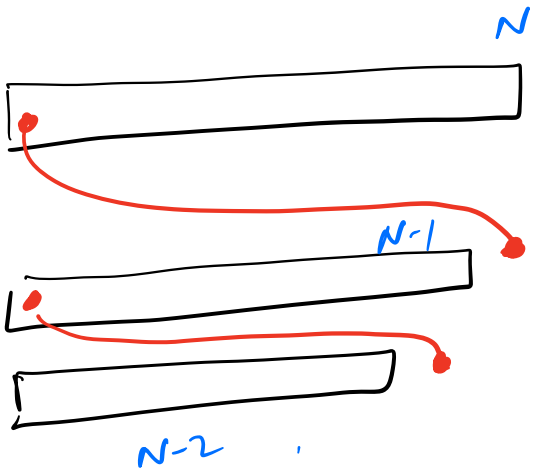


$$T(N) = N + 2\,T(N/2)$$

TC → N lg N ✓

2

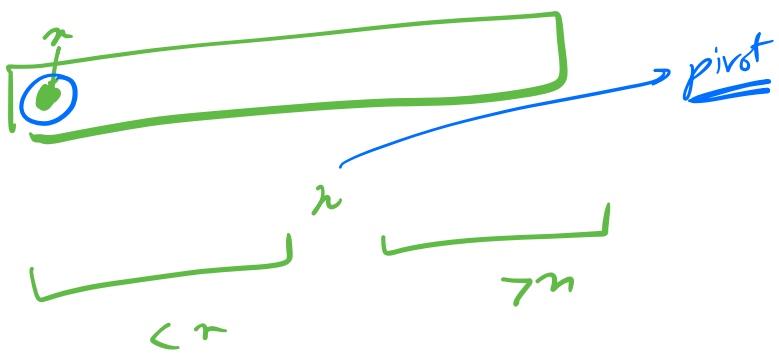
CASE



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

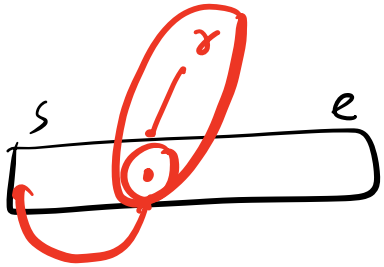
TC → N^2

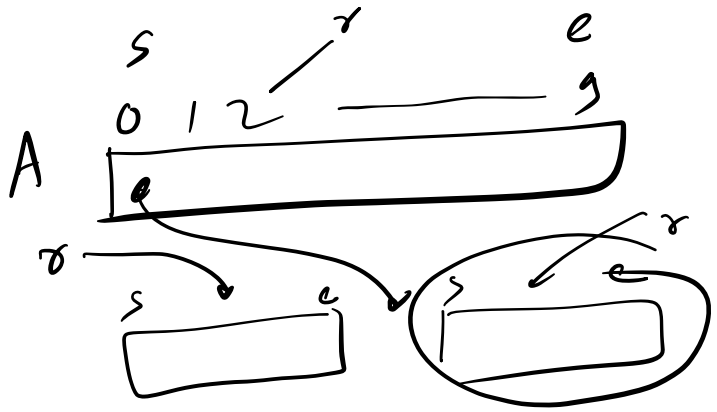
⊙ How to improve it?



1) Randomly choose the pivot!

```
r = random(s, e)
swap(A[s], A[r]);
```





MEAN

A : 3 2 8 4 6 1 10 5

5

4.875

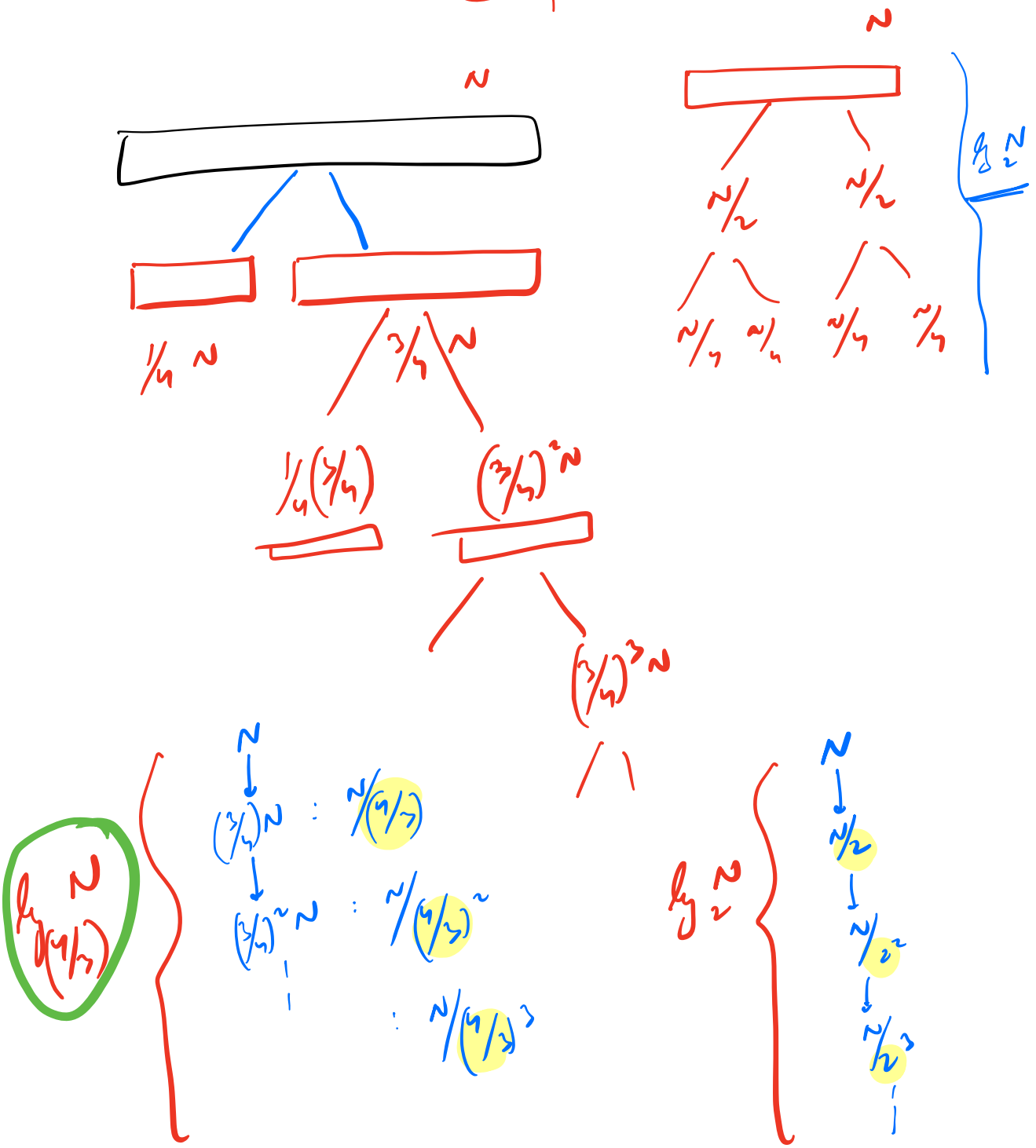
2 4 1 (5) 8 6 10

MODE

MEDIAN

AVERAGE :

```
r = random(s, e)
swap( A[s], A[r]);
recurr (  -1  )
```



Av. Case: # levels : $\log_{(4/3)} N$

Amt of Work done on every level

$= O(N)$

$TC = N \cdot \log_{(4/3)} N$

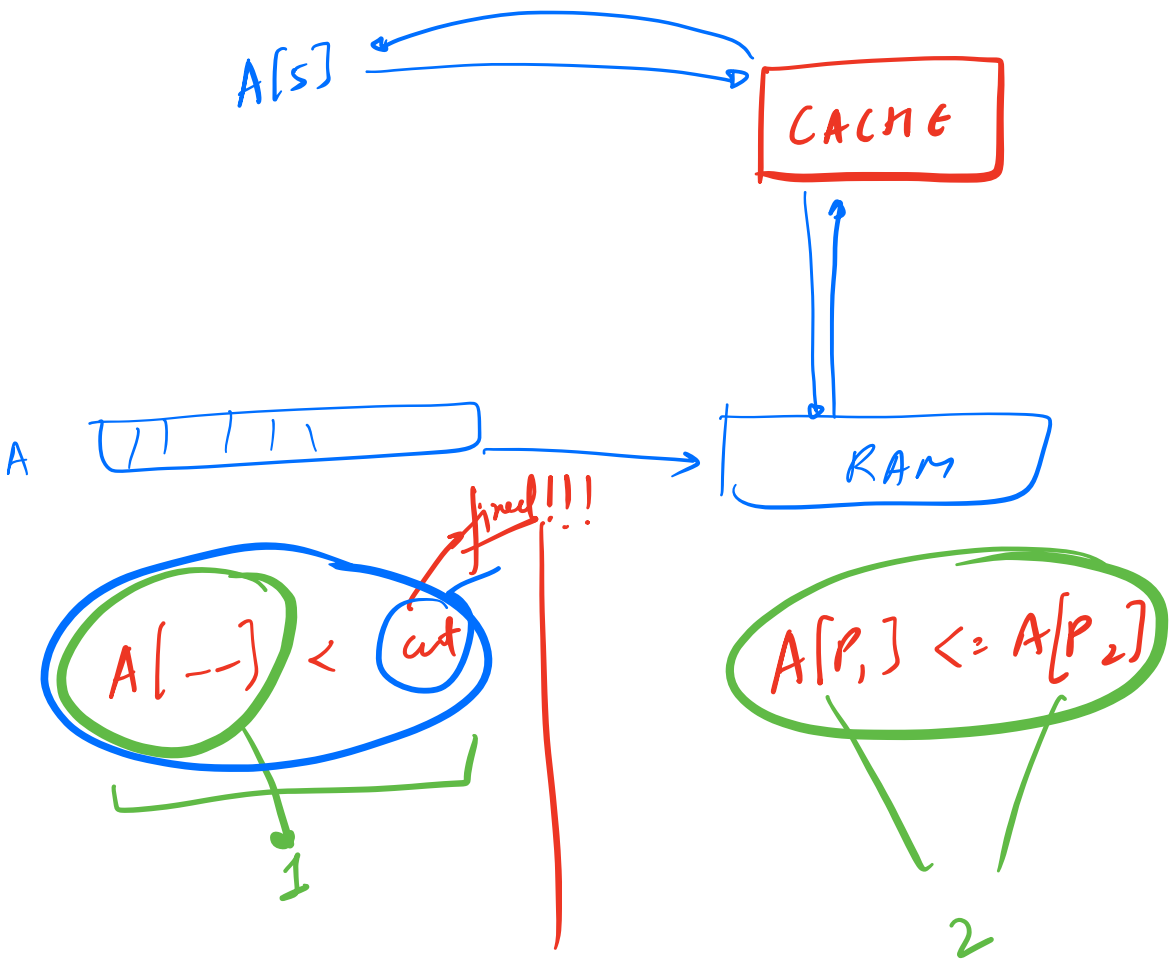
$TC = O(N \log N)$ ✓

$SC = O(\log N)$ ✓

Why Quick Sort over Merge Sort?

1) Space: $O(\log N)$ vs $O(N)$

2) Quick's Runtime is less than Merge's —



(3) Q Sort is CACHE friendly!