Doubt Clearing Session

Course on Sorting and Searching

Rohit Mazumder • Lesson 3 • Feb 20, 2021

Selection Sort

- Selects the minimum element from unsorted section and places it in the beginning of the unsorted section.

Example: [4, 3, 2, 1] => [1, 3, 2, 4] => [1, 2, 3, 4] => [1, 2, 3, 4] => [1, 2, 3, 4]

- Implement stable version of selection sort?

$[4, 5, 3, 2, 4, 1] \longrightarrow [1, 5, 3, 2, 4, 4]$

```java
public static void selectionSort(int[] A) {
    int N = A.length;
    for(int i = 0; i < N; i++) {
        int minIdx = i;
        for(int j = i + 1; j < N; j++) {
            if(A[j] < A[minIdx]) minIdx = j;
        }

        if(minIdx != i) {
            int temp = A[minIdx];
            A[minIdx] = A[i];
            A[i] = temp;
        }
    }

    System.out.println("Selection sorted : " + Arrays.toString(A));
}
```

Selection Sort is not Stable

Example: [4, 5, 3, 2, 4, 1]

Making Selection Sort Stable:

- Swapping values is what is making selection sort unstable.
- We need to do something other than swapping! ✗

sliding ✓

Making Selection Sort Stable:

[4, 5, 3, 2, 4, 1] → [1, 5, 3, 2, 4, 4] ✗

1

[4, 5, 3, 2, 4, 1] → [_, 4, 5, 3, 2, 4] → [1, 4, 5, 3, 2, 4] ✓

shift to the right

Segregate positives and negatives in an array

[50, 10, -1, 27, -19, 3, -44, -12] => [-1, -19, -44, -12, 50, 10, 27, 3]

Constraint: After segregation, the relative order of +ves and -ves must remain the same.

Solution 1: Using extra space

$O(2N) = O(N)$

A = [50, 10, -1, 27, -19, 3, -44, -12]

negatives[] = [-1, -19, -44, -12, _, _, _, _] count = 4

positives[] = [50, 10, 27, 3, _, _, _, _]

count = 4

A = [-1, -19, -44, -12, 50, 10, 27, 3]

$O(\frac{N}{2})$ $O(N)$

[50, 10, -1, ...]

[-1, 50, 10, . . .]

Worst Time Case: $O(n^2)$

Space

$O(1)$

Solution 2: Divide & Conquer?

[50, 10, -1, 27 | -19, 3, -44, -12]

[50, 10, -1, 27]

[-19, 3, -44, -12]

- To check whether we can use D&C:
  - → Assume that we have solved the left and right subproblem
  - → We will try to figure out if there's a way to merge

Solution 2: Divide & Conquer?

[50, 10, -1, 27, -19, 3, -44, -12]

[-1, 50, 10, 27]

[-19, -44, -12, 3]

Solution 2: Divide & Conquer?

[50, 10, -1, 27, -19, 3, -44, -12]

[-1, 50, 10, 27]

[N | P]

[-19, -44, -12, 3]

[N | P]

[-1, 50, 10, 27, -19, -44, -12, 3]

Solution 2: Divide & Conquer?

[50, 10, -1, 27, -19, 3, -44, -12]

[-1, 50, 10, 27]

[-19, -44, -12, 3]

[-1, 50, 10, 27, -19, -44, -12, 3]

[-1, -12, -44, -19, 27, 10, 50, 3]

Solution 2: Divide & Conquer?

[50, 10, -1, 27, -19, 3, -44, -12]

[-1, 50, 10, 27]

[-19, -44, -12, 3]

[-1, 50, 10, 27, -19, -44, -12, 3]

[-1,-12, -44, -19, 27, 10, 50, 3]

[-1,-19, -44, -12, 27, 10, 50, 3]

Solution 2: Divide & Conquer?

[50, 10, -1, 27, -19, 3, -44, -12]

[-1, 50, 10, 27]

[-19, -44, -12, 3]

[-1, 50, 10, 27, -19, -44, -12, 3]

[-1,-12, -44, -19, 27, 10, 50, 3]

[-1,-19, -44, -12, 27, 10, 50, 3]

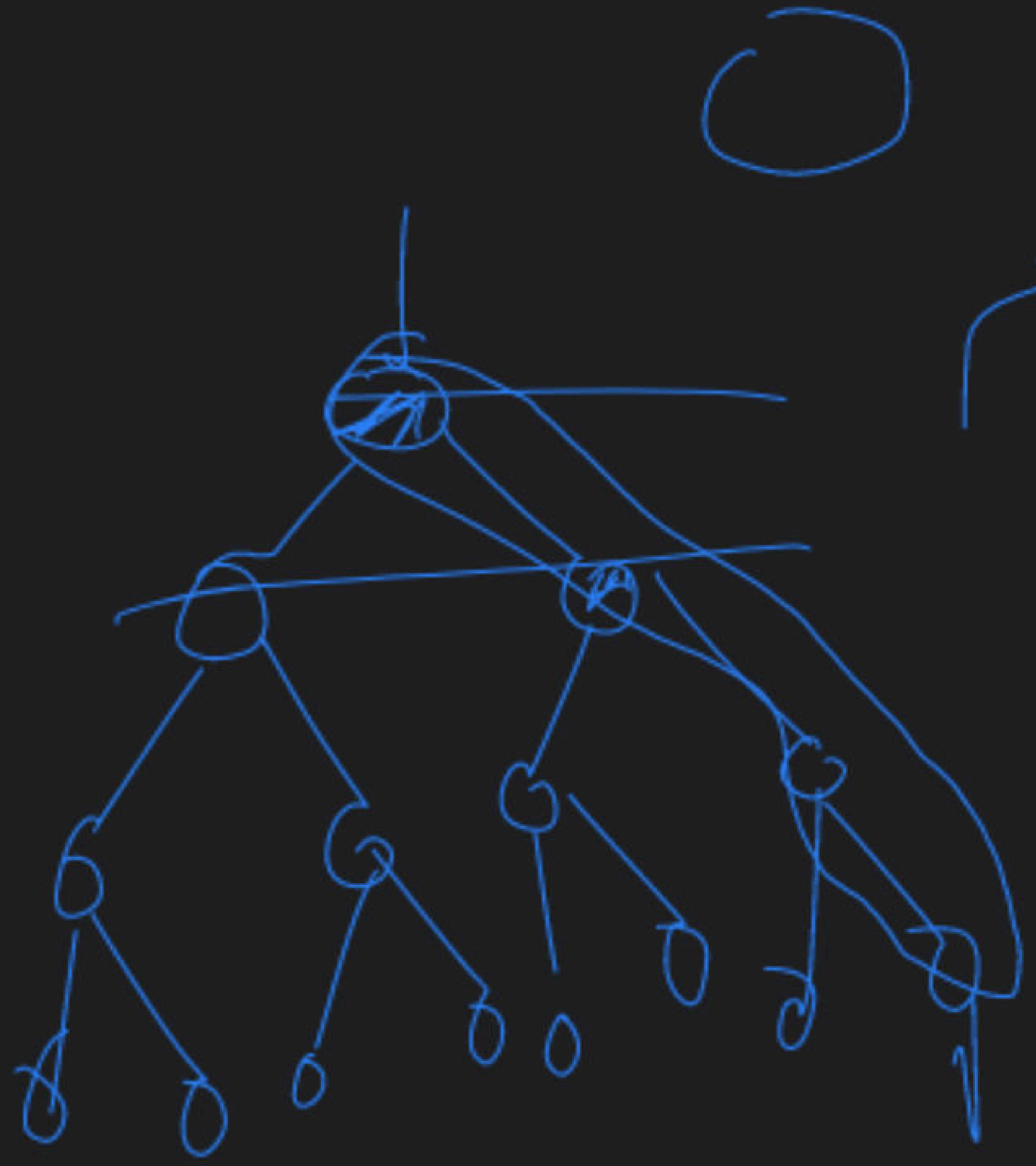
[-1,-19, -44, -12, 50, 10, 27, 3]

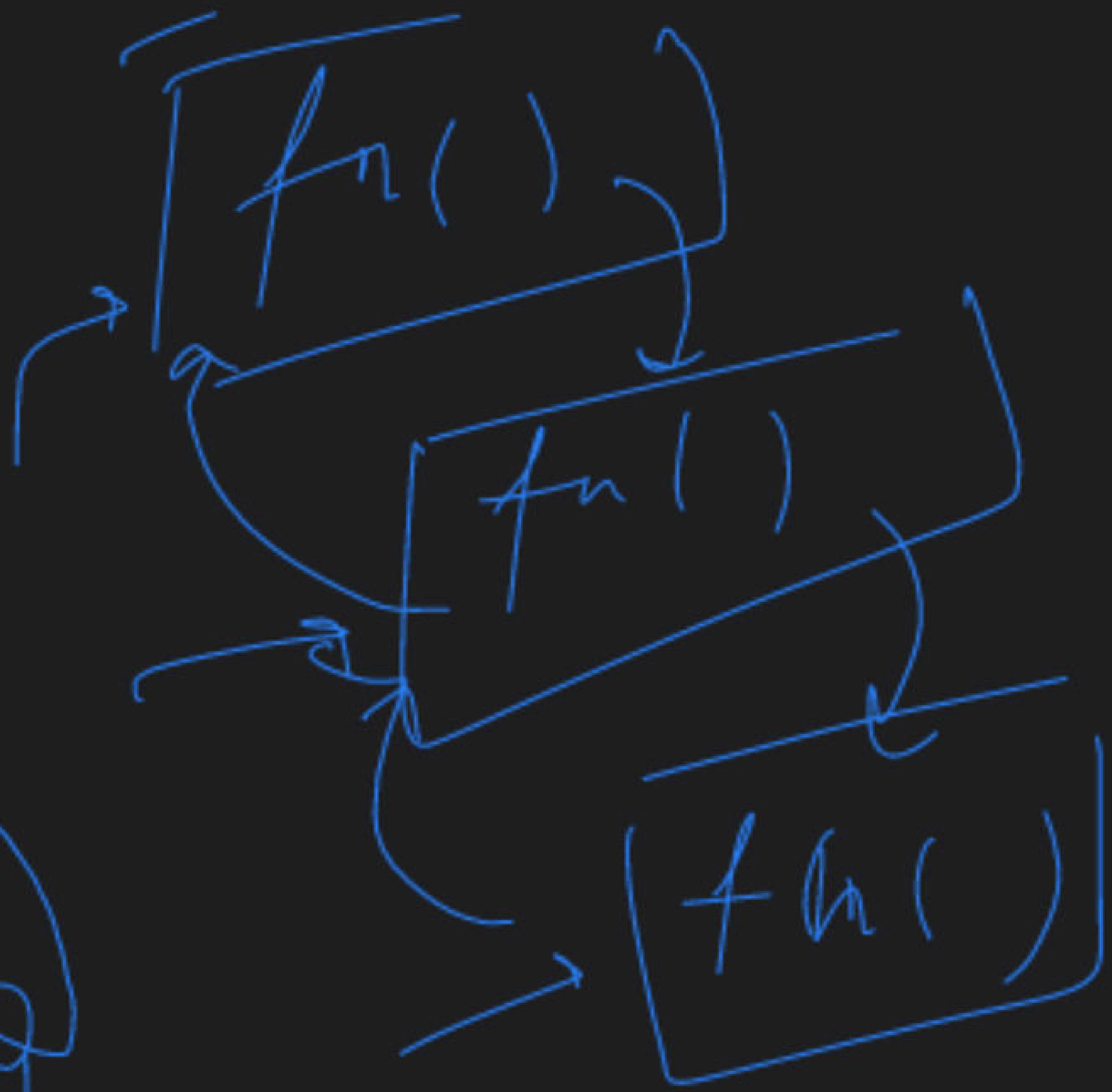
$O(\log n)$

$O(n + \log n)$

$\approx O(n)$

$\rightarrow$ ANSWER



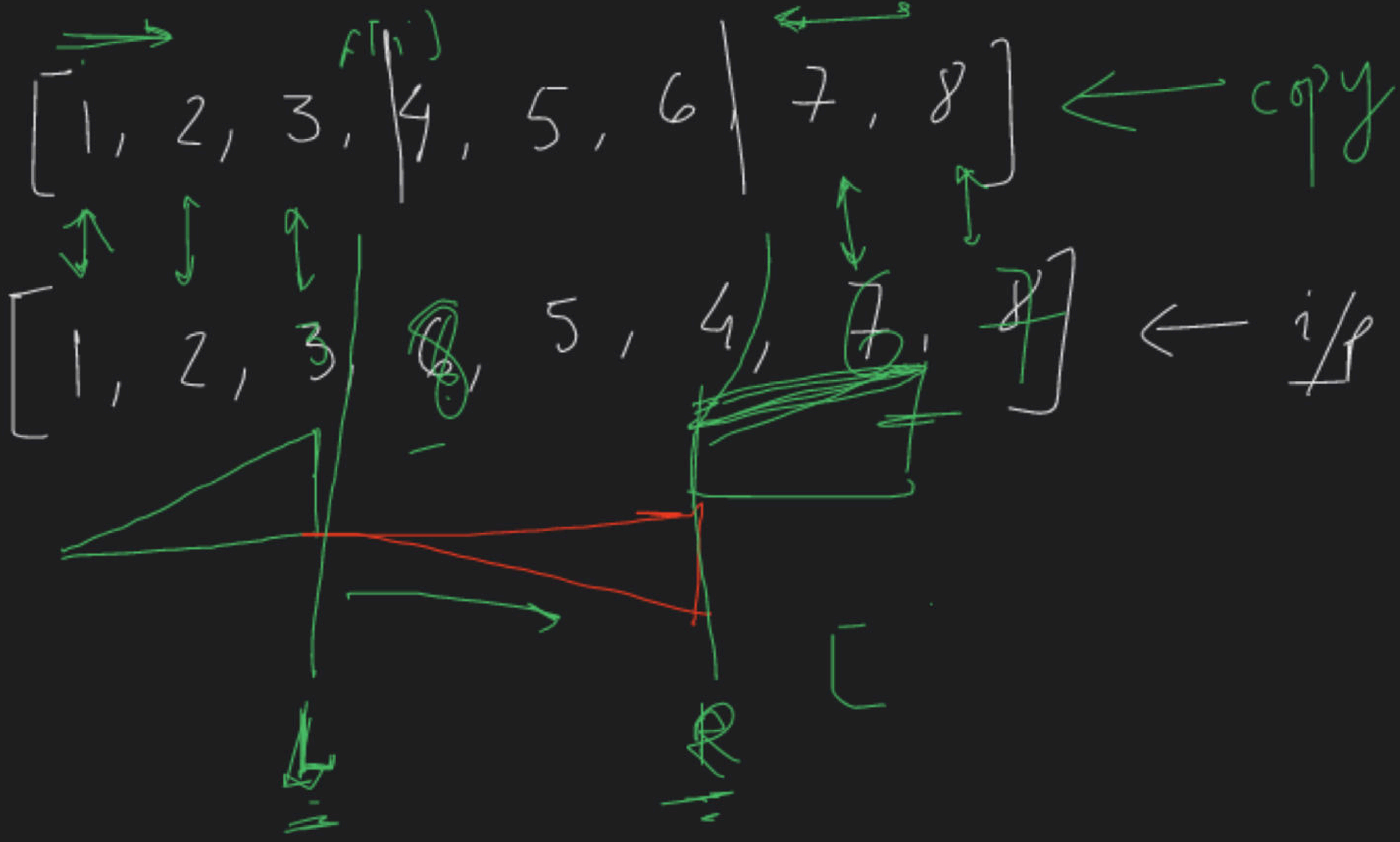

fn( )
fn( )
fn( )

Sort the array!

https://codeforces.com/problemset/problem/451/B

[1, 2, 3, 4, 5]

(1, 1)


f[i]
[1, 2, 3, 4, 5, 6, 7, 8]
copy
[1, 2, 3, 6, 5, 4, 7, 8]
i/p
L
R

1. What is the recurrence relation for worst case of merge sort and the time complexity in worst case?

A. $T(n) = T(n-2)+O(n)$, $O(n^2)$
B. $T(n) = 2*T(n/2) + O(n)$, $O(n\log n)$
C. $T(n) = 2*T(n/2) + O(1)$, $O(n^2)$
D. $T(n) = 2*T(n/2) + O(n)$, $O(n^2)$

A. $T(n) = T(n-2)+O(n)$, $O(n^2)$
B. $T(n) = 2*T(n/2) + O(n)$, $O(n \log n)$
C. $T(n) = 2*T(n/2) + O(1)$, $O(n^2)$
D. $T(n) = 2*T(n/2) + O(n)$, $O(n^2)$

Solution : We divide the array in 2 halves and do O(n) work at the merge step.

2. Which of the following is not a stable algorithm in its typical implementation?

A. Insertion Sort
B. Merge Sort
C. Selection Sort
D. Bubble Sort

A. Insertion Sort
B. Merge Sort
C. Selection Sort
D. Bubble Sort

Solution : We just discussed the implementation for stable selection sort algo.

3. Which sorting algo will take least time when all elements are identical? Consider only typical implementation.

A. Insertion Sort
B. Merge Sort
C. Selection Sort
D. Bubble Sort

A. Insertion Sort
B. Merge Sort
C. Selection Sort
D. Bubble Sort

Solution : Since the array is sorted, insertion sort will work in O(n)

4. A list of n strings each of length n, is sorted into lexicographic order using the merge sort algorithm. The worst case running time is?

A. $O(n\log n)$
B. $O(n\log^2 n)$
C. $O(n^2\log n)$
D. $O(n^2\log^2 n)$

A. $O(n\log n)$
B. $O(n\log^2 n)$
C. $O(n^2\log n)$
D. $O(n^2\log^2 n)$

$n \times \underline{n\log n}$

Solution : Everything is same as merge sort, except while comparing two elements in merge step we need $O(n)$ time for strings.

5. Which of the following statements are false about merge sort?

A. It is stable by nature
B. It is an in-place algorithm
C. It outperforms insertion sort in best case
D. Both B and C

A. It is stable by nature
B. It is an in-place algorithm
C. It outperforms insertion sort in best case
D. Both B and C

Solution : Merge sort is not in-place. And insertion sort takes O(n) in best case while merge sort takes O(nlogn)

6. Given an array = {4,3,2,1}, how many minimum number of operations will be required to sort the array if you are only allowed to swap the adjacent elements in one operation.?

A. 2
B. 1
C. 4
D. 6

A. 2
B. 1
C. 4
D. 6

$${}^{n}C_2 = \frac{n(n-1)}{2}$$

Solution : This is equivalent to finding the number of inversions in an array which in this case if equal to n*(n-1)/2