Today's agenda

- Sorting basics (recap)
- Selection sort
- Merge sort
  - merge 2 sorted arr[]
  - merge 2 sorted arr[] in single arr[]
  - merge sort.
- 1 Problem

- you didn't come this far only to come this far.

Sorting: arranging data in inc/dec order based on parameter.

eg: 1 7 2 9 24 : inc -> on the basis factors.

factors: 1 2 2 3 8

//Bubble Sort

↳ sort the arr in asc. order but we can swap adjacent elements only.

T.C: $O(N^2)$ S.C: $O(1)$

Selection Sort

arr[8]: 2 8 4 -1 7 10 5 6 (indices 0 1 2 3 4 5 6 7)

iter 0: min ele = -1, min idx = 3 → swap(0,3)

arr[8]: -1 8 4 2 7 10 5 6 (i = 0)

iter 1: min ele = 2, min idx = 3 → swap(1,3)

arr[8]: -1 2 4 8 7 10 5 6 (i = 1)

iter 2: min ele = 4, min idx = 2 → swap(2,2)

arr[8]: -1 2 4 8 7 10 5 6 (i = 2)

(|)

iter N-2

//Pseudo code

```
void SelectionSort(int arr[N]){
    for(int i=0; i<N-1; i++){
        int minEle = arr[i], minIdx = i;
        for(int j=i+1; j<N; j++){
            if(arr[j] < minEle){
                minEle = arr[j];
                minIdx = j;
            }
        }
        int temp = arr[i];
        arr[i] = arr[minIdx];
        arr[minIdx] = temp;
    }
}
```

T.C: $O(N^2)$

S.C: $O(1)$

Tracing

arr[8]:

0	1	2	3	4	5	6	7
~~2~~ -1	8	4	~~1~~ 2	7	10	5	6

i:

j:

```
for (int i=0; i<N-1; i++){
    int minEle = arr[i], minIdx = i;
    for (int j=i+1; j<N; j++){
        if (arr[j] < minEle){
            minEle = arr[j];
            minIdx = j;
        }
    }
    int temp = arr[i];
    arr[i] = arr[minIdx];
    arr[minIdx] = temp;
}
```

minEle: ~~2~~ -1 minIdx = ~~0~~ 3

temp = 2

Q) Merge two sorted array

↳ Given 2 sorted arrays a[N], b[M] create C[N+M] which contains overall sorted data.

Ex: a[4] : { 7 10 11 14 }
b[3] : { 3 8 9 }

C[4+3] : { 3 7 8 9 10 11 14 }

//idea 1

↳ Create a new array of (N+M) length, copy the data of a[] and b[] and sort the new array.

T.C: O((N+M) * log(N+M))

//idea 2

a[4] : { ~~7~~ ~~10~~ ~~11~~ ~~14~~ } i

b[3] : { ~~3~~ ~~8~~ ~~9~~ } j

C[4+3] : { 3 7 8 9 10 11 14 } k

0 1 2 3 4 5 6

//Psuedo Code

```
int [] merge (int a[N], int b[m]){
    int [] c = new int [n+m];
    int p1 = 0;
    int p2 = 0;
    int p3 = 0;

    while (p1<N || p2<m) {
        if (a[p1] < b[p2]){
            c[p3] = a[p1]; p3++, p1++;
        }
        else {
            c[p3] = b[p2]; p3++, p2++;
        }
    }

    while(p1<N) { c[p3] = a[p1]; p3++, p1++;}

    while (p2<m) { c[p3] = b[p2]; p3++, p2++;}

    return c;
}
```

T.C: O(n+m)

S.C: ~~O(n+m)~~ O(1)

Tracing

```
int [] merge (int a[n], int b[m]){
    int [] c = new int [n+m];
    int p1 = 0;
    int p2 = 0;
    int p3 = 0;

    while (p1<n && p2<m) {
        if (a[p1] < b[p2]) {
            c[p3] = a[p1]; p3++, p1++;
        }
        else {
            c[p3] = b[p2]; p3++, p2++;
        }
    }

    while(p1<n) { c[p3] = a[p1]; p3++, p1++;}

    while (p2<m) { c[p3] = b[p2]; p3++, p2++;}

    return c;
}
```

a[4] : { 7 10 11 14 } (0 1 2 3) ↙ p1

b[3] : { 3 8 9 } (0 1 2) ↙ p2

c[7] : { 3 7 8 9 10 11 14 } (0 1 2 3 4 5 6) ↙ p3

Q) merge 2 sorted subarray

↳ Given arr[N] elements and 3 indices s, m and e and subarray [s,m] is sorted, subarray [m+1, e] is sorted. Sort subarray from [s,e]. {s,e} is continuous.

Ex: arr[12]: { 4 8 ~~-1~~ ~~2~~ ~~6~~ ~~9~~ ~~11~~ ~~3~~ ~~4~~ ~~7~~ 13 0 }

s m e

2 6 9

temp[9-2+1] = tmp[8]: { -1 2 3 4 6 7 9 11 }

//s → m
//mid → e

//Pseudo code

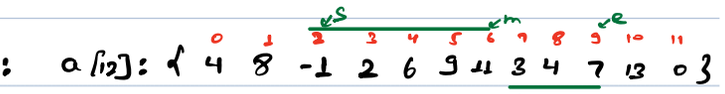
```
int[] merge(int a[N], int s, int m, int e){

    int[] temp = new int[e-s+1];
    int p1 = s;
    int p2 = m+1;
    int p3 = 0;

    while (p1<=m && p2<=e ){
        if (a[p1] < a[p2]){
        temp[p3] = a[p1]; p3++, p1++;
        }
        else{
        temp[p3] = a[p2]; p3++, p2++;
        }
    }

    while (p1<=m){ temp[p3] = a[p1]; p1++, p3++; }

    while (p2<=e) { temp[p3] = a[p2]; p2++, p3++; }

    //Copy temp[0-?] back to a[s-e] → todo

    return a;
}
```

T.C: O(N)

S.C: O(N)

```
int[] temp = new int[e-s+1];
int p1 = s;
int p2 = m+1;
int p3 = 0;

while (p1<=m && p2<=e ){
    if (a[p1] < a[p2]){
        temp[p3] = a[p1]; p3++, p1++;
    }
    else {
        temp[p3] = a[p2]; p3++, p2++;
    }
}

while (p1<=m){ temp[p3] = a[p1]; p1++, p3++; }

while (p2<=e) { temp[p3] = a[p2]; p2++, p3++; }
```

Ex: a[12] = { 4 8 -1 2 6 9 11 3 4 7 13 0 }



s m e
2 6 9

temp[8] = { -1 2 3 4 6 7 9 11 }

0 1 2 3 4 5 6 7

p3

Q) merge sort

↳ Given arr[N] sort them.

	0	1	2	3	4	5	6	7	8
	s				m				e
arr[9]:	8	2	3	10	6	17	12	18	15

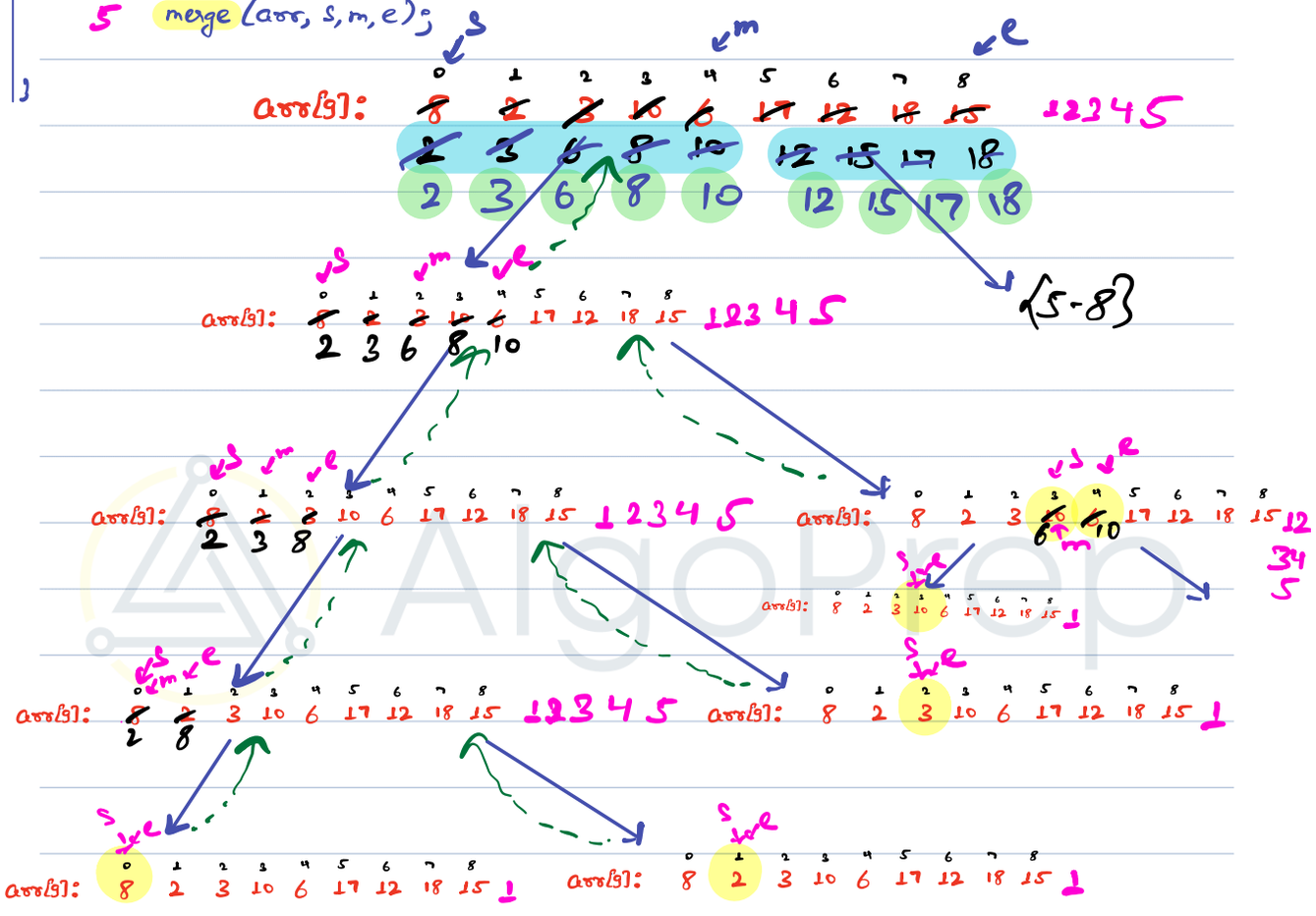
//Pseudo code

```
void mergeSort(int arr[N], int s, int e){
    if(s==e){return;}

    int m = (s+e)/2;

    mergeSort(arr, s, m);
    mergeSort(arr, m+1, e);

    merge(arr, s, m, e);
}
```

T.C: O(N*logN)

S.C: O(N)

```
void mergeSort(int arr[N], int s, int e){
1  if(s==e){ return; }
2  int m = (s+e)/2;
3  mergeSort(arr, s, m);
4  mergeSort(arr, m+1, e);
5  merge(arr, s, m, e);
}
```



N → N/2 → N/4 → ... → 1

At each level merge combined = O(N)

How many such levels ? ⇒ logN

overall T.C: O(N * logN).

Q) inversion Count

↳ Given arr[n], find the inversion count.

↳ two elements a[i] and a[j] form an inversion if a[i] > a[j] & i < j.

Ex: arr[] = { 8 4 2 1 } (0 1 2 3) → ans: 6

(8,4) (8,2) (8,1) (4,2) (4,1) (2,1)

Ex: arr[] = { 1 20 6 4 5 } (0 1 2 3 4) → ans: 5

//idea 1

↳ Check all Possible Pairs and count the inverted Pairs.

T.C: $O(N^2)$ S.C: $O(1)$

Count = 0 + 5 + 3 + 2

{2,1} {4,1} {10,1} {13,1} {18,1}

{10,5} {13,5} {18,5}

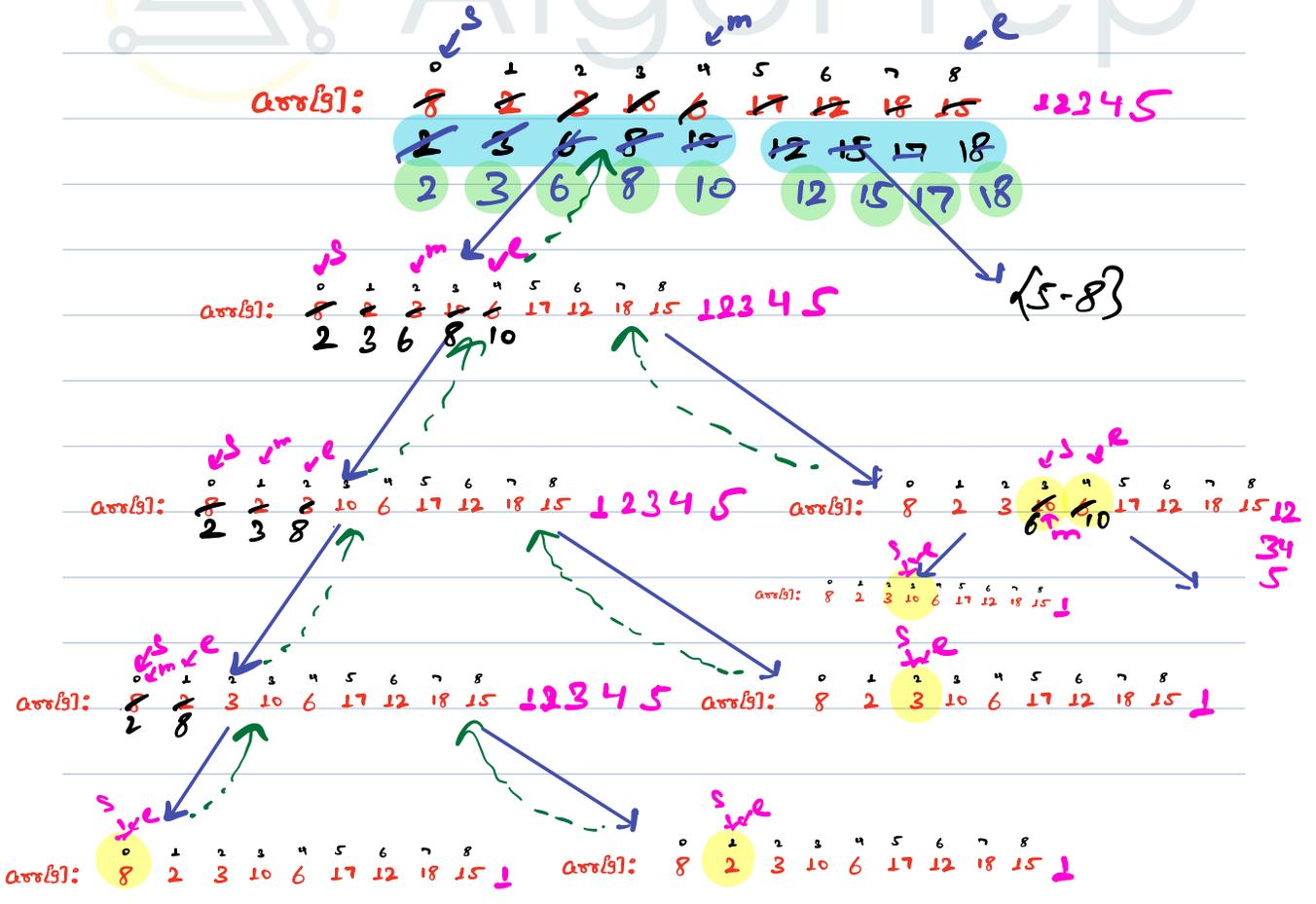
{13,12} {18,12}

Idea 2

	0	1	2	3	4
	2	4	10	13	18
	1	5	12	15	20

p1 → 0, m → 4, p2 → 0

```
if (arr[p1] > arr[p2]) {
    update count
    ans += (m - p1 + 1);
}
else { // arr[p1] <= arr[p2])
}
```

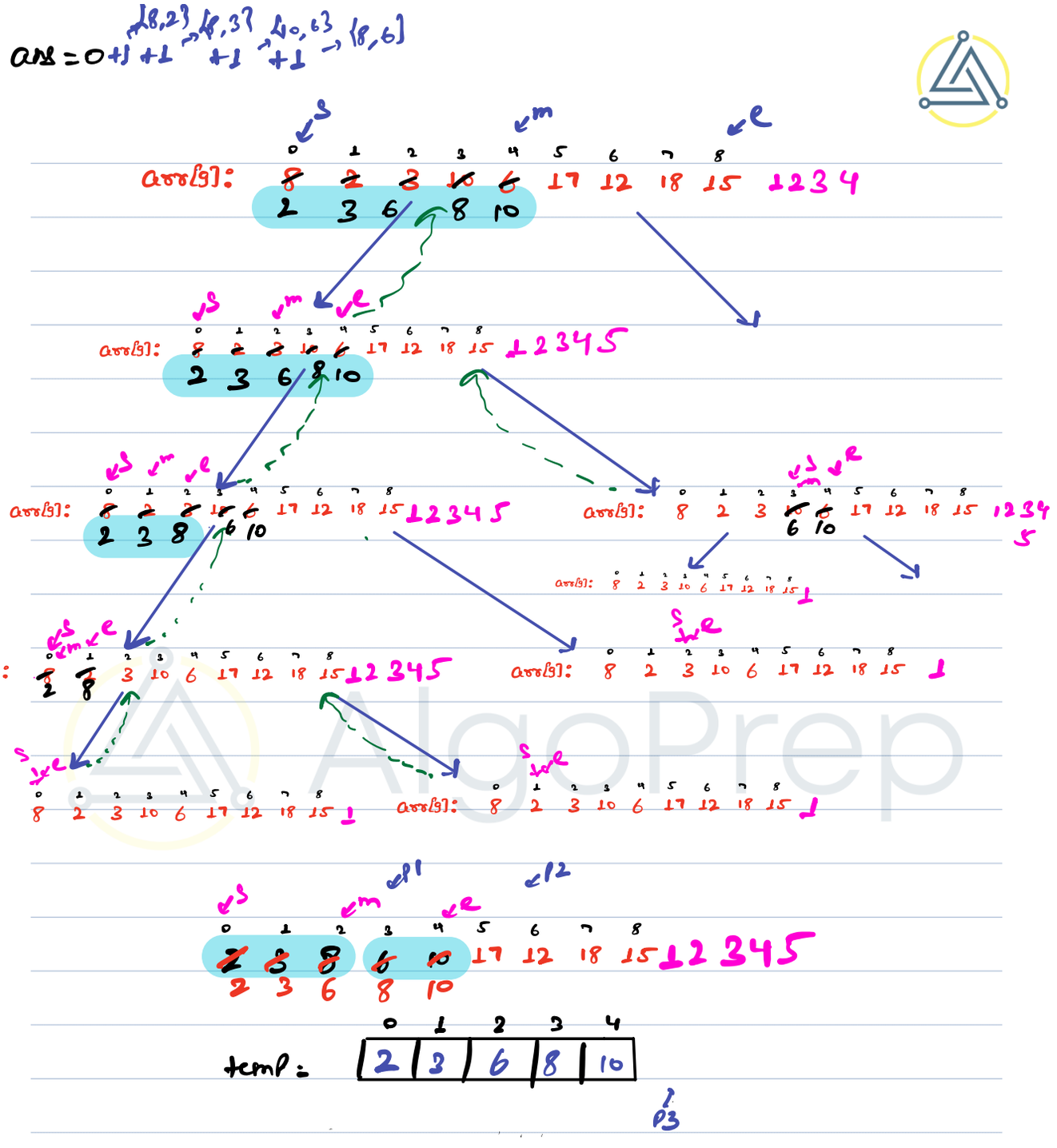


```
void mergeSort(int arr[n], int s, int e){
    if(s==e){ return; }

    int m = (s+e)/2;

    mergeSort(arr, s, m);
    mergeSort(arr, m+1, e);

    merge(arr, s, m, e);
}
```

T.C: O(n*logn)

S.C: O(n)

```
int ans = 0;

int[] merge(int a[N], int s, int m, int e){

    int[] temp = new int[e-s+1];
    int p1 = s;
    int p2 = m+1;
    int p3 = 0;

T.C: O(N)   while (p1<=m && p2<=e ){
S.C: O(N)       if (a[p1] <= a[p2]){
            temp[p3] = a[p1]; p3++, p1++;
        }
        else { // a[p1] > a[p2]
            ans = ans + (m-p1+1);
            temp[p3] = a[p2]; p3++, p2++;
        }
    }

    while (p1<=m){temp[p3] = a[p1]; p1++, p3++;}

    while (p2<=e) {temp[p3] = a[p2]; p2++, p3++;}

    //copy temp[0-7] back to a[s-e] → todo

    return a;
}
```


ans = 0+1 +1 +1 +1 +1
[8,2] [8,3] [10,6] [8,6]
arr[]: 8 2 3 10 6 17 12 18 15 12345
2 3 6 8 10
arr[]: 8 2 3 10 6 17 12 18 15 12345
2 3 6 8 10
arr[]: 8 2 3 10 6 17 12 18 15 12345
2 3 8 6 10
arr[]: 8 2 3 10 6 17 12 18 15 12345
2 8
arr[]: 8 2 3 10 6 17 12 18 15 1
arr[]: 8 2 3 10 6 17 12 18 15 1
arr[]: 8 2 3 10 6 17 12 18 15 1
arr[]: 8 2 3 10 6 17 12 18 15 1
arr[]: 8 2 3 10 6 17 12 18 15 12345
2 3 6 8 10
temp: 2 3 6 8 10