LECTURE 1
Monday 24/10/22 — 12:00

Array → Overhead
→ JVM, JRE, JDK
→ Platform Independence

arr: { 3, 1, 5, 8, 3, 6 }.

Q. Sorted the arr.

arr={ 3, 1, 5, 8, 3, 6 } → Unsorted.

Q. Sort the arr. → Algorithm

```
void sort(int arr[], int size){
    for (int i = arr.length; i>=0; i--){
        int j = 0;
        while (j<i){
            if (arr[j] > arr[j+1]){
                swap(arr[j], arr[j+1]);
            }
            j++;
        }
    }
}
```

0	3
1	1
2	5
3	8
4	3
5	6

arr = { 1, 3, 5, 8, 3, 6 }

int i = 0;
int j = arr.length;

while (i<=j){ i++;
int temp = arr[i]; // 1
arr[i] = arr[j]; //
arr[j] = temp; // 3
i++;
j--;
}

Q. { 3, 6, 8, 1, 5, 10, 6. }.

data = 5 → linear search

```
for (int i=0; i< arr.length; i++){
    if (arr[i] == data){
        return (i);
    }
}
```

→ return -1;

Sorting Algo. → arrangement of elements in form "order".
↳ increasing form → ↑
↳ decreasing

① Bubble Sort.
② Selection Sort
③ Insertion Sort

① Bubble Sort.

arr: { 0, 1, 2, 3, 4, 5 }.

Passes → n-1 passes. i=0

Pass 1. → largest value of array should be placed at last index.

```
for (int pass=1; pass<n; pass++){
    for (int j=0; j<n-pass-1; j++){
        if (arr[j] > arr[j+1]){
            swap(arr[j], arr[j+1])
        }
    }
}
```

Pass 2

Pass 3

Pass 4.

② Selection Sort

```
for (int i=0; i< arr.length-1; i++){
    int min = i;
    for (int j=i+1; j< arr.length; j++){
        if (arr[j] < arr[min]){
            min = j;
        }
    }
    swap(arr, i, min);
}
```

Q. Insertion Sort

arr: { 3, 1, 5, 8, 3, 6 }.

3>0 && arr[3]<arr[2] && 1>0 && arr[1] < arr[0]

```
for (int i=1; i< arr.length; i++){
    for (int j=i; j>0 && arr[j]<arr[j-1]; j--){
        swap(arr, j, j-1);
    }
}
```

4>0 && arr[4] < arr[3] && 3>0 && arr[3] < arr[2] && 1>0 && arr[1] < arr[0]

Reverse at ↑ step.

JRE → Java Runtime Environment.
JDK → Java development Kit.
JVM → Java virtual machine.

Platform - independence → write once execute anywhere.

Executable → Machine OS → windows

exe → windows.
windows, Mac, Linux → executes

Windows — Mac — Linux — independent of machine.

Compile.

Bytecode (.class)

JVM — JVM — JVM

platform dependent.

Java → OS

arr: { 3, 1, 5, -2, -4, 6. }.

max sub array sum

sub array → contiguous part in array.

3 → 3	1
3,1 → 4	1,5
3,1,5 → 9	1,5,-2
3,1,5,-2 → 7	1,5,-2,-4
3,1,5,-2,-4 → 3	1,5,-2,-4,6
3,1,5,-2,-4,6	

-2	
-2,-4	
-2,-4,6 → 0	
6	
-4,6	
5,-2	

```
int sum=0, int max = Integer.MIN_VALUE;
for (int i=0; i< arr.length; i++){
    for (int j=i; j< arr.length; j++){
        sum=0;
        for (int k=i; k<=j; k++){
            sum = sum + arr[k];
        }
        if (sum > max){
            max = sum;
        }
    }
}
```

Stack mem.

Heap mem.

Stack → Heap

Stack → Heap

min(a,b) = min(a,b)
min(a,b,c) = min(min(a,b), c)

min(5, 7) → 5

min(4,3) + min(min(1,5), min(5,3)) → 8
(3 + 3 + 3), 3) → 9

total Cost → ((2+5), 19) → 18

Monday → 2 days
Tuesday → 3 days

Coefficients

Questions → Homework →