Lecture 4

Saturday, 11 January 2020 2:44 PM

Arrays.

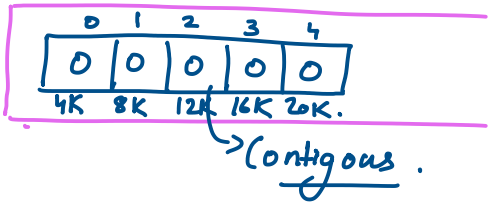
int[] arr; → non-primitive.

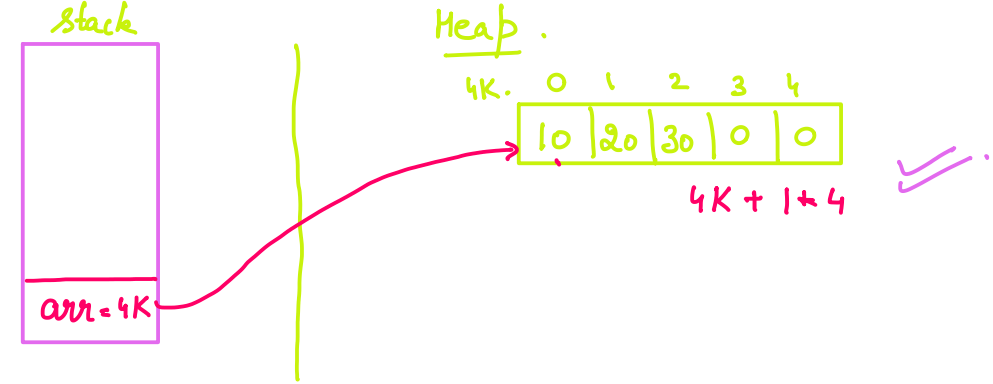
data type.

Syso(arr);

arr = new int[5]; → size.

reference





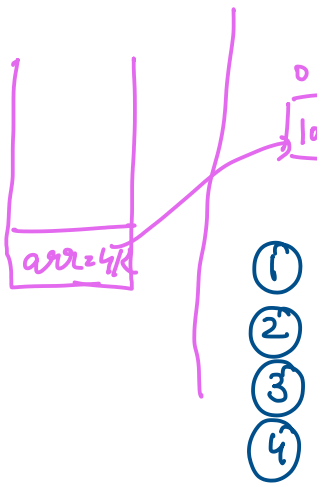
psvm(String[] args) {

```
int[] arr = new int[5];
arr[0] = 10;
arr[1] = 20;
arr[2] = 30;
for(int i=0; i<arr.length; i++){
    Syso(arr[i]);
}
```

5<5*x

0 1 2 3 4

```
int i=1; int j=2;
Syso(arr[i], arr[j]);   (1)
Swap(arr, i, j);
Syso(arr[i], arr[j]);   (4)
int[] arr2 = new int[4];
}
```



primitive → don't persist things

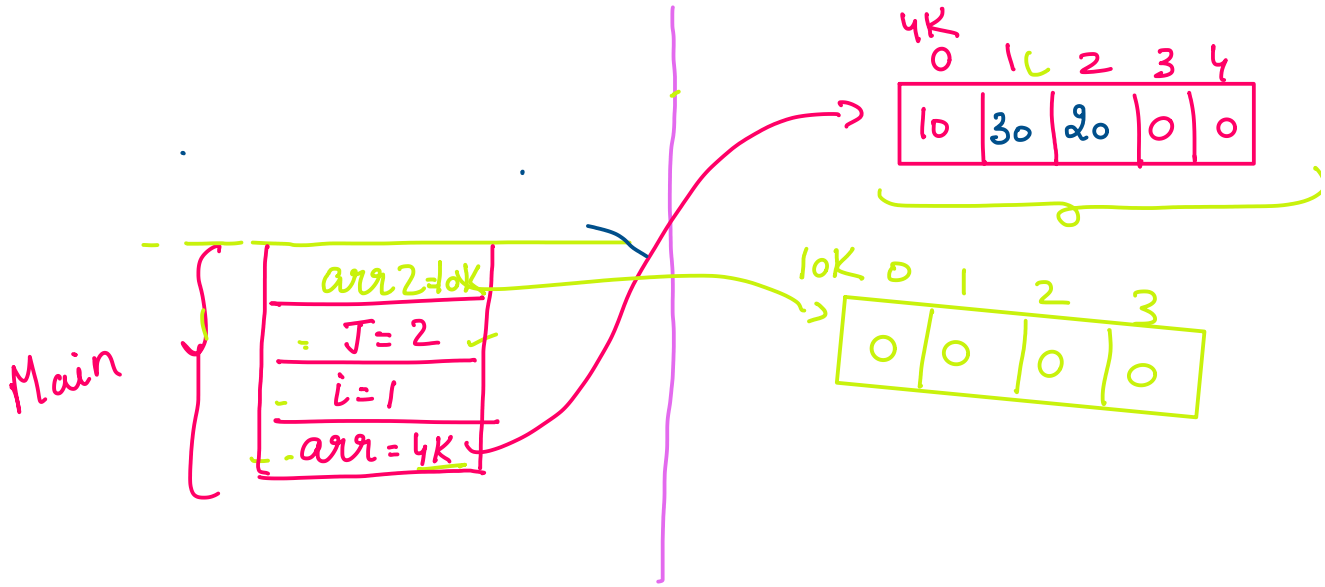
non- " → Can persist things.

public static void swap(int[] arr, int i, int j){

```
int temp = arr[i];
arr[i] = arr[J];
arr[J] = temp;
Syso(arr[i], arr[J]);   ③
```

Syso(arr[i], arr[J]) ②

```
int [] arr1 = new int[5];
swap2(arr, arr1);
```



arr[1] = 20 value

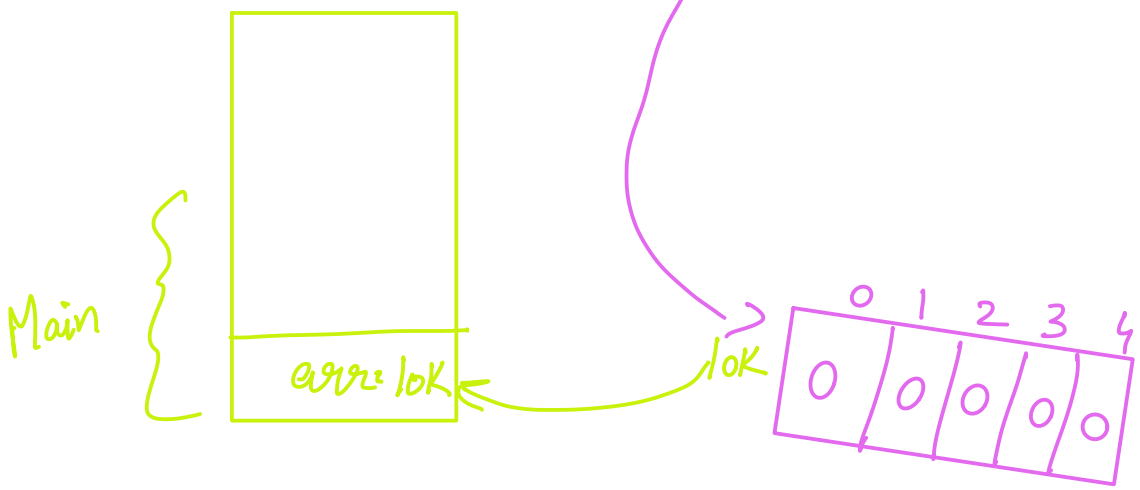
i = 1; → index.

J = 2

```
Syso(arr.length)          // 5
Syso(arr[arr.length]);    × Runtime Error.
Syso(arr[arr.length-1]);  0
```

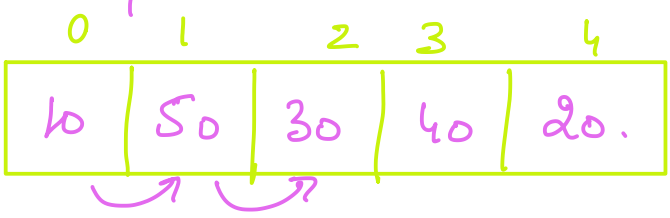
```
swap2(int[] arr1, int[] arr2) {
    int[] temp = arr1;
    arr1 = arr2;
    arr2 = temp.
```

```
int[] arr;
arr = new int[5];
Syso(arr[2]);   // 0
arr = new int[5];
```



Sorting Algorithms.

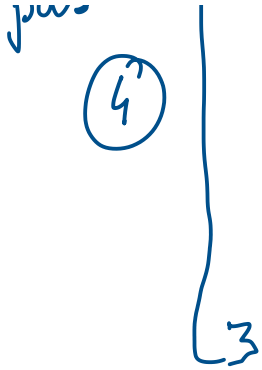
1. Bubble Sort
2. Selection
3. Insertion



Bubble Sort. → Passes => n-1 = arr.length-1

```
for (int i = 1; i<=arr.length-1; i++) {
```

Passes




```
for(int J=0; J<arr.len
    if(arr[J] > arr[J+
        swap(J, J+1)
```

Pass 1 → 10, 50, 30, 40, 20

10, 30, 50, 40, 20

10, 30, 40, 50, 20

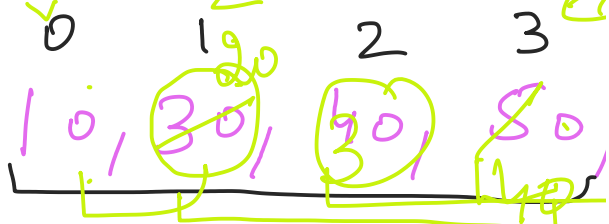
10, 30, 40, 20, 50

Pass 3 → 10, 30, 20, 40, 50

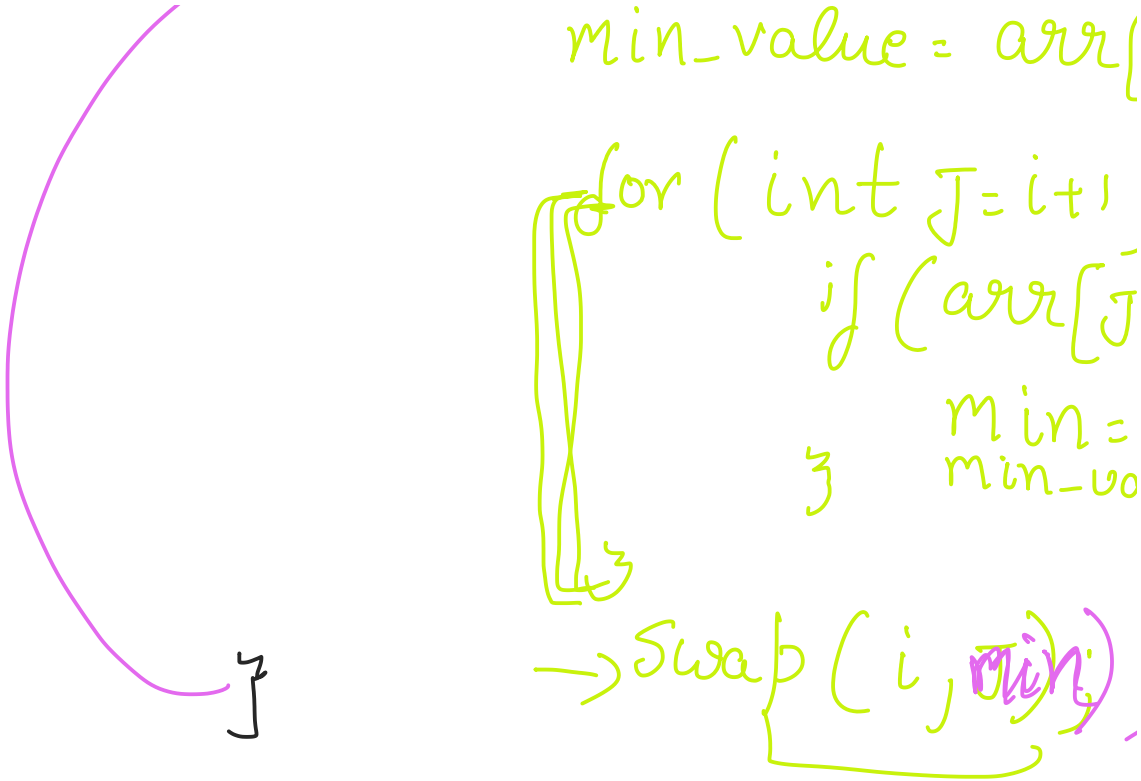
10, 30, 20, 40, 50

10, 20, 30, 40, 50

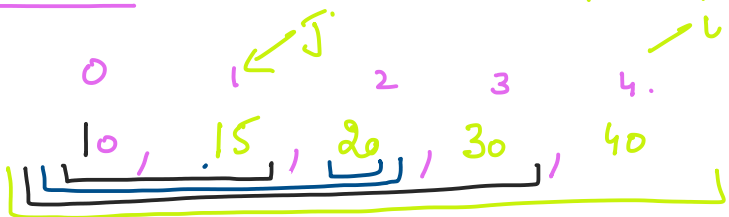
Selection Sort




```
for(int i=0; i<arr
    min=i; //1
```



Insertion Sort




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

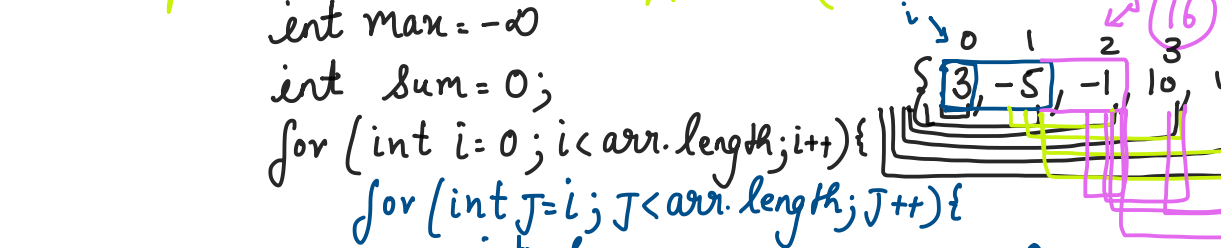
arr = { 3, -5, -1, 10, 4, 2 }

Max Subarray Sum

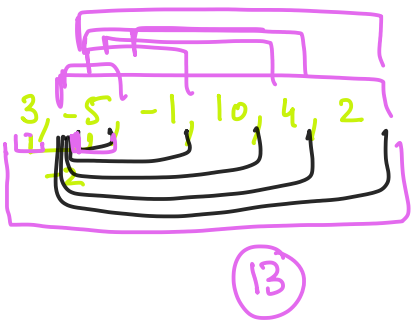
3	-5, -1
3, -5	-5, -1, 10, 4
3, -5, -1	3, -5, -1, 10, 4, 2

3, -5 =>
3, -5, -
3, -5,

```
public static int approach1(int[] arr){
    int max = -∞
    int sum = 0;
    for(int i=0; i<arr.length; i++){
        for(int j=i; j<arr.length; j++){
            int sum = 0;
            for(int k=i; k<=j; k++){
                sum = sum + arr[k];
            }
            if(sum > max){
                max = sum;
            }
        }
    }
}
```

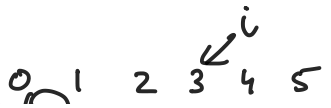



```
public static int approach2(int[] arr){
    int sum=0
    int max = -∞
    for(int i=0; i<arr.length; i++){
        for(int j=i; j<arr.length; j++){
            sum = sum + arr[j];
            if(sum > max){
                max = sum
            }
        }
        sum = 0;
    }
    return max;
}
```




```
public static int approach3(int[] arr){
    int max = -∞
    int sum = 0;
    for(int i=0; i<arr.length; i++){
        sum = sum + arr[i];          //3
        if(sum > max){
```

Kadane's




```
        max = sum
    }
    if (sum < 0){
        sum = 0;
    }
}
```

//3 11

//0