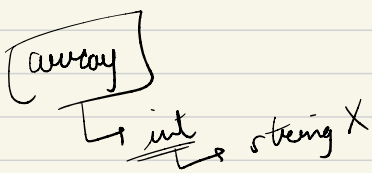
array → group of similar data type + contiguous memory

integers

int a
int b
int c
int d
int e



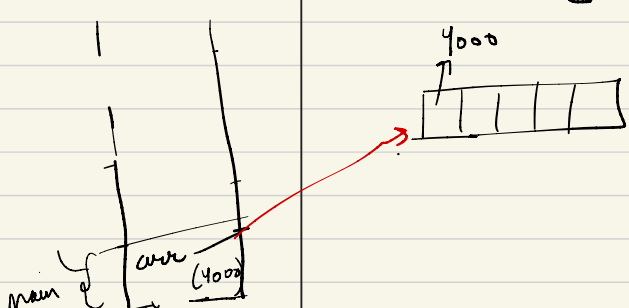
class Main {
psv fun()
int a=5
fun2()
psv main() {

int [] arr;
arr = new int[n];

}

temporary RAM
stack memory

permanent
heap memory



Syntax

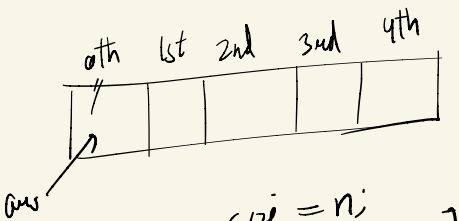
data type + [] + array name

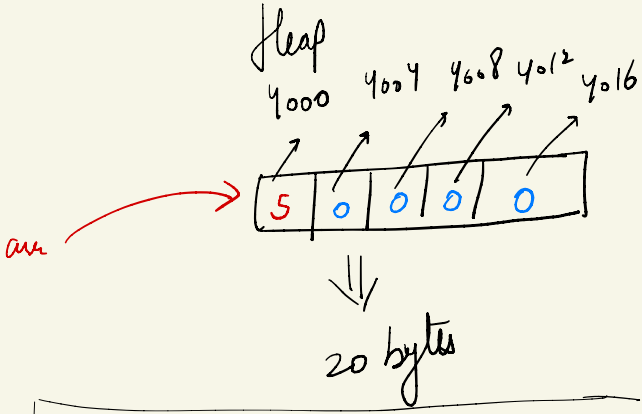
int [] arr; // declaration

jhbjhb

arr = new int[5];

indexes





arr size = n;
indexes = [0, n-1]

int () arr;
arr = new int [5]

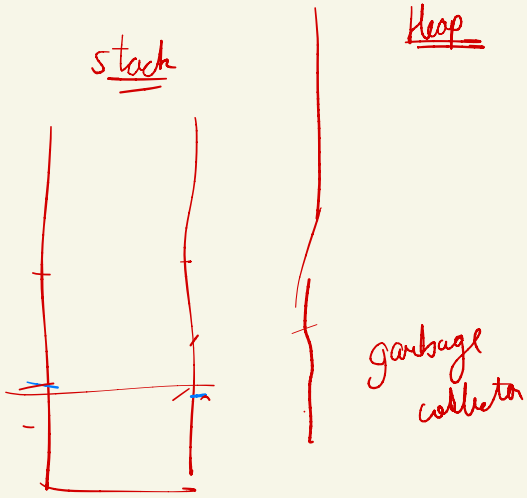
set
arr[idx] = value;

get
arr(idx)

size
int size = arr.length;

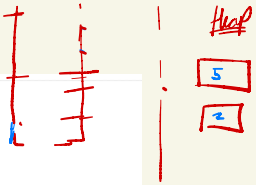
class Main {

psv main() {
int[] arr;
arr = new int(5);
}



arr = new int[5]; // initialization

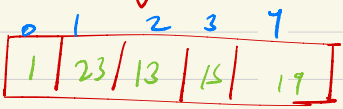

```java
public class MyClass {
    public static void swap(int[] arr1, int[] arr2){
        int temp=arr1[0];

        arr1[0]=arr2[0];
        arr2[0]=temp;
    }

    public static void main(String args[]) {
        int[] arr1;
        arr1=new int[];

        int[] arr2;
        arr2=new int[];

        arr1[0]=2;
        arr2[0]=5;

        swap(arr1,arr2);

        System.out.println(arr1[0]);
        System.out.println(arr2[0]);
    }
}
```



Ques 1) Take an input n, create an array of size n.

↳ Then Take n inputs, store it in array and print it.

Ques 2) Print the array in reverse.



n=5

i=0 ; i<n

```java
Run | Debug
public static void main(String[] args) {
    Scanner scn=new Scanner(System.in);

    int n=scn.nextInt();

    int[] arr; // declaration
    arr=new int[n]; // initialization

    for(int i=0; i<n; i++){
        arr[i]=scn.nextInt();
    }

    System.out.println("Printing array elements ");

    for(int i=0; i<n; i++){
        System.out.println(arr[i]);
    }

```

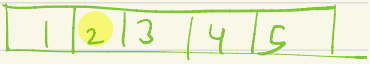
i=n-1 ; i>=0; i--

i = ~~4 3 2 1~~ 0

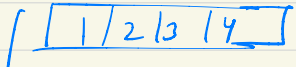
arr(i)

19
15
13
23
1

Ques 3) Print alternate elements of array



1
3
5



1
3

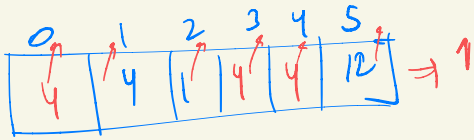
Ques 4) Print all the even numbers existing in your array and print at which index they are.

Ques 5) Get the sum of all the even numbers and all the odd numbers in your array and find their difference.

Ques 6) Find the number of elements with index = taken as input this value.

ans → 0 or 1.

n = 6

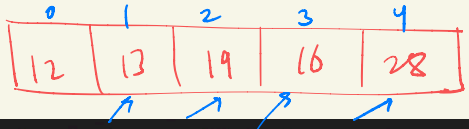


tar = 4

tar >= 0 & tar < n

↓

1



esum = 0 12 28 56
osum = 0 13 32

24

```
44
45          System.out.println("Difference of all the even and odd numbers");
46
47          int esum=0;
48          int osum=0;
49
50          for(int i=0; i<n; i++){
51              int v=arr[i];
52
53              if(v%2==0){
54                  esum=esum+v;
55              } else {
56                  osum=osum+v;
57              }
58          }
59
60          System.out.println("The difference is = "+(esum-osum));
61      }
62  }
```

Ques 1) Take an input n, input n integers in your array. Take another input "tar". Find at which index tar exist

eg.

0	1	2	3	4	5
9	45	13	4	13	9

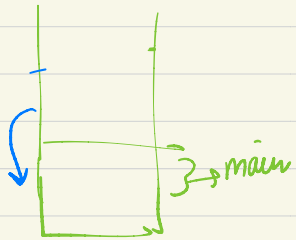
tar = 19

↓

-1

```
for(int i=0; i<n; i++){
    ↳ if(arr[i] == tar){
        return i;
}}
```

i = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5



function → employee

(input) → work/process → output

↓

parameter/argument

Ques 2 Find the maximum value in this array.
Find the minimum value in this array.

0	1	2	3	4
-10	-3	-4	-19	-5

max = -∞

↓

Integer.MIN_Value

```java
// find maximum in array
public static int max_of_array(int[] arr, int n){
    int max=0; // this is wrong

    for(int i=0; i<n; i++){
        if(arr[i]>max){
            max=arr[i];
        }
    }

    return max;
}
```

n=5

0	0	0	0	0

Math.max

int a=5;
int b=3;

b = Math.max(a, b);
b = 5

a = Math.max(a, b);
a = 85

Ques ⇒ Given a sorted array, check if it contains duplicate values or not.

0	1	2	3	4	5	6	7
9	10	12	17	19	21	25	26

(non-decreasing order)

i = 0 1 2 3 4 5 6 7

```java
//find if duplicates are there in sorted array
//n-> size of array
public static boolean containsDuplicates(int[] arr, int n){
    for(int i=0; i<n; i++){
        if(arr[i]==arr[i+1]){
            return true;
        }
    }

    return false;
}
```

arr(i) = 26 arr(7)

arr(i+1) = arr(8)

arr(8)

i<n-1

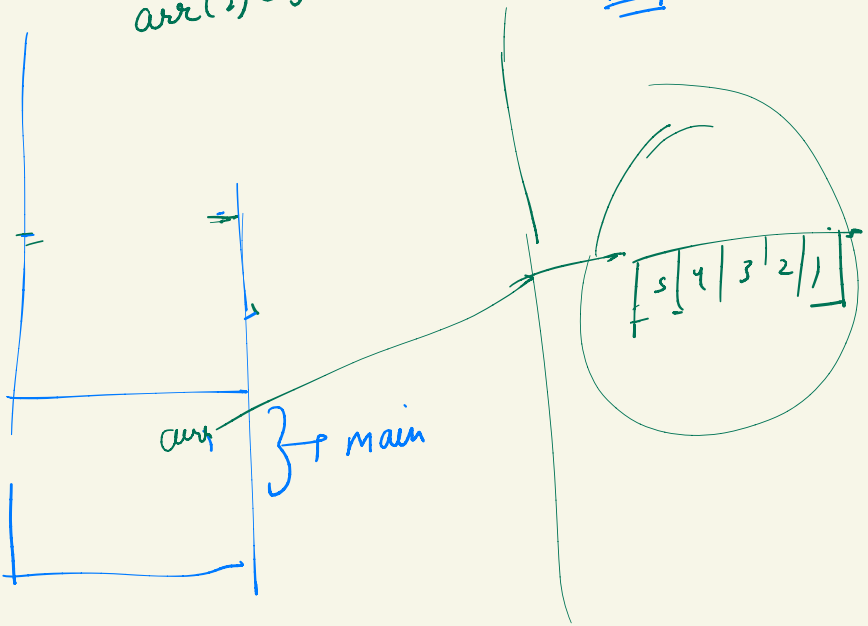
Ques) Reverse an array. (Inplace) → same array

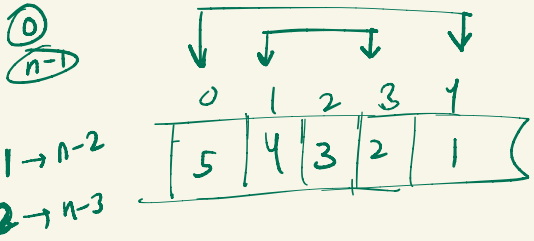
1 2 3 4 5

5 4 3 2 1

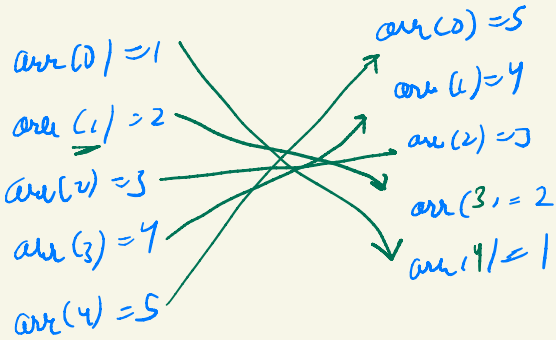
arr(2) = 50

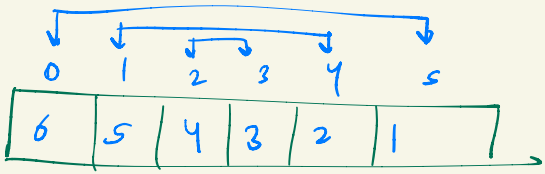
heap





first index → last index
second index → last second index
3rd " → last 3rd index





0 → n-1 (5)
1 → n-2 (4)
2 → n-3 (3)

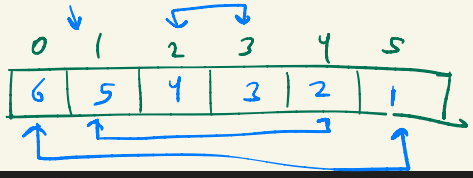
0 → (n-1) - 0 ⇒ (n-1)
1 → (n-1) - 1 ⇒ (n-2)
2 → (n-1) - 2 ⇒ (n-3)
3 → (n-1) - 3 ⇒ (n-4)

for (int i=0; i<n/2; i++)

(i, n-1-i)

i → (n-1) - i

n=6



i=0

i<3

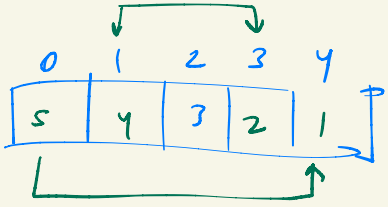
```
57
58  public static void reverse_array(int[] arr, int n){
59      for(int i=0; i<n/2; i++){
60          //swap (i) and(n-i-1);
61          int temp=arr[i];
62
63          arr[i]=arr[n-i-1];
64          arr[n-i-1]=temp;
65      }
66  }
```

i=0, n-1

$\frac{5}{2}+2$

i<2

n=5



i=0, n-1-0 → n-1

i=1, n-1-1 ⇒ n-2

int a=5
int b=6

swap(a, b) → b=5, a=6

int temp=a; (temp=5)

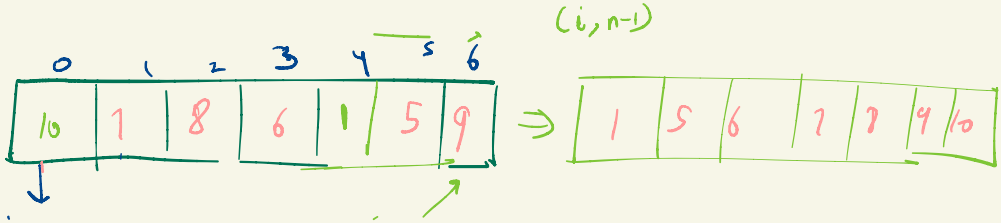
a=b (a=6)

b=temp (b=5)

a=b ✗

b=a

1) Selection Sort




```java
public static void swap(int[] arr, int i, int j) {
    int temp=arr[i];

    arr[i]=arr[j];
    arr[j]=temp;
}

public static void sort_array(int[] arr, int n){
    for(int i=0; i<n; i++){
        int min_idx=i;

        for(int j=i+1; j<n; j++){
            if(arr[j]<arr[min_idx]){
                min_idx=j;
            }
        }

        swap(arr,i,min_idx);
    }
}
```

i = 0

j = 1 2 3 4 5 6

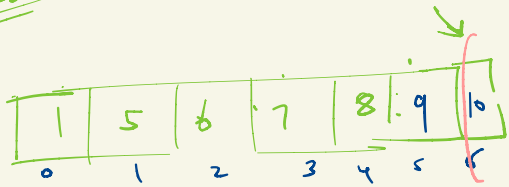
min_idx = 0 1 2 3 4

arr(min_idx) → min / min

i < n-1

Bubble Sort

n-1



0th it → n-2 (< n-1)

1st it → n-3 (< n-2)

idx < n-1-i

0th iteration → (5, 6)

1st iteration → (4, 5)

2nd iteration → (3, 4)

3rd → (2, 3)

4th → (1, 2)

5th → (0, 1)

idx = n-3

n-1

i = 0 → n-2 (idx < n-1)

i = 1 → n-3 (idx < n-2)

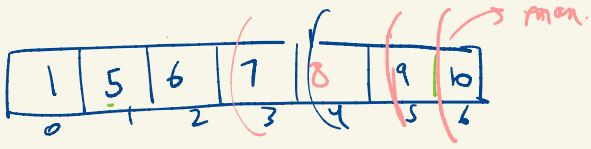
i = 2 → n-4 (idx < n-3)

i = 3 → n-5 (idx < n-4)

i = 4 → n-6 (idx < n-5)

i = 5 → n-7 (idx < n-6)

n-1-5




```java
// bubble sort ============================================================

public static void bubble_sort(int[] arr, int n){

    for(int i=0; i<n-1; i++){  // more than one iterations will be required

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

    }
}
```

6
4 5 3 8 6 1

Break Statement

```
for (int i=1; i<=10; i++){
    if (i==5){
        break;
    }
}
```

i=1
i=2
i=3
i=4
i=5

continue statement

↳ skips the current iteration.

```
for(int i=1; i<=10; i++){
    if(i==7){
        continue;
    }
    System.out.println(i);
}
```

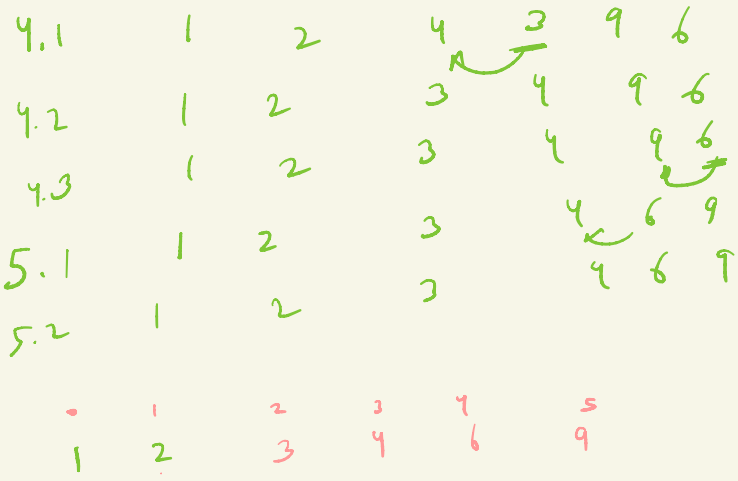
i=1 2 3 4 5 6 8 9 10

Insertion sort

iterations = n − 1

0	1	2	3	4	5
9	2	4	1	3	6
2	9	4	1	3	6
2	4	9	1	3	6
2	4	9	1	3	6
2	4	1	9	3	6
2	1	4	9	3	6
1	2	4	9	3	6

1.1
2.1
2.2
3.1
3.2
3.3



$i = 1 \times 3 \times 4 \times 5 \times$

```java
public static void insertion_sort(int[] arr, int n) {

    for(int i=1; i<n; i++){ // we need to insert arr[i] at correct position

        for(int j=i; j>0; j--){ // asking j-1 everytime
            if(arr[j-1]>arr[j]){
                swap(arr,j-1,j);
            } else {
                break;
            }
        }
    }
}
```

5 3 4 2 1

→ Read through it at a pace.
→ See the example test cases.

door
- constraints
- read the question again thoroughly
  - Read TCs
    - question and TCs simultaneously
    - Read constraints

0 1 2 3 4 5
3 6 4 3 3 9

(0,3)
(0,4)
(2,5)
(3,4)

```
public static int getLuckyPairs(int[] arr, int n){
    int count=0;

    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if(arr[i]==arr[j]){
                count++;
            }
        }
    }

    return count;
}
```

Ques Given an array, find first occurrence and last occurrence of given target value.

0	1	2	3	4	5	6	7	8	9	10	11
1	3	4	3	4	5	5	3	7	3	4	5

tar=3 , fo = 1, lo = 9 → single for loop

i = 1 2 3 4 5 6 7 8 9

```java
int lo=-1;
int fo=-1;

for(int i=0; i<n; i++){
    if(arr[i]==tar){
        if(fo==-1){
            fo=i;
        }

        lo=i;
    }
}

System.out.println("first occurence is: "+fo);
System.out.println("last occurence is: "+lo);
```

I have never discovered arr(i) before.

fo = ~~-1~~ 1

lo = ~~-1~~ ~~1~~ 3 7 9

Ques Find if the given array is mountain array or not

→ 2, 21, 45, 50, 33, 29, 25, 19 ✓

(i) 1, 81, 79, 50, 40 ✓

(i=1) 1, 2, 7, 9, 11, 5 ✓

1 2 3 4 5 → X

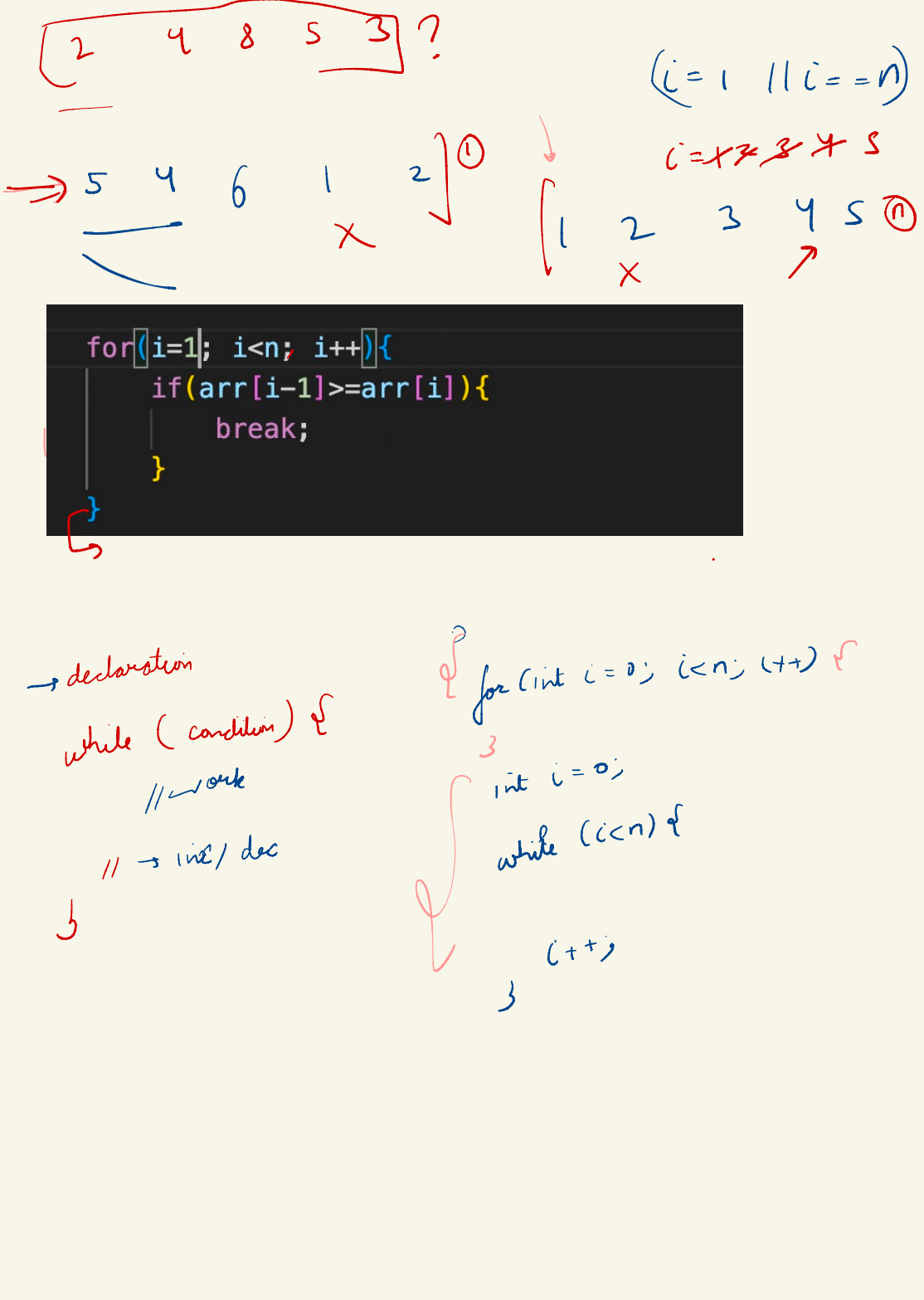
1 4 3 6 2 → X

5 4 3 2 1 → X

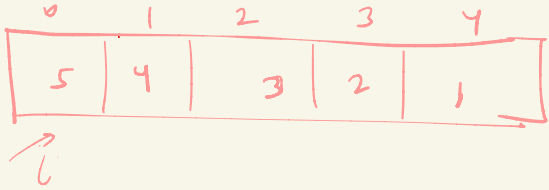
✓ arr < arr(i) < - - arr(i) > arr(i+1) > arr(i+2) - - - arr(n-1)

5 4 6 2 1 → false

i = ~~1~~ ~~2~~ 3


2 4 8 5 3 ?
(i = 1 || i == n)
5 4 6 1 2
1 2 3 4 5 n
for(i=1; i<n; i++){
if(arr[i-1]>=arr[i]){
break;
}
}
declaration
while (condition) {
// work
// → inc/ dec
}
for (int i = 0; i<n; i++) {
int i = 0;
while (i<n) {
i++;
}

Two pointers



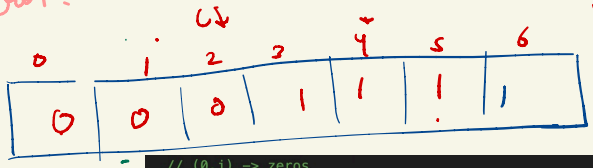
reverse

```java
public static void reverse(int[] arr, int n){
    int i=0;
    int j=n-1;

    while(i<=j){
        // swap i and j

        int temp=arr[i];

        arr[i]=arr[j];
        arr[j]=temp;

        i++;
        j--;
    }
}
```

Run | Debug

i <= j

Ques Given an array containing only 0's and 1's. Sort the array.




```java
// (0,i) -> zeros
// (i+1,j-1) -> ones
// (j,n-1) -> undefined
public static void sort_zero_one(int[] arr, int n){
    int i=-1;
    int j=0;

    while(j<n){
        if(arr[j]==0){
            i++; // increasing zeros' area
            swap(arr,i,j);
            j++;
        } else { // increasing ones' area
            j++;
        }
    }
}
```

zeros

(0, i) → zeros

(i+1, j-1) → ones

(j, n-1) → unknown

if (arr[j] == 1){ j++;

(0, i) (i+1, j-1) (j, n-1)

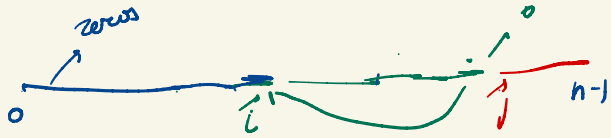
(0, i) → zeros

(i+1, j-1)

(j, n-1) → undefined

zeros = 5

ones = 4



i+1 j

Ques Given an array containing 0's, 1's and 2's. Sort the array in one traversal.

p1 = -1 ~~0~~ ~~1~~ ~~2~~

p2 = 0 ~~1~~ ~~2~~ ~~3~~ 4

p3 = 10 9 8 7 6

0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	1	1	1	1	2	2
				p1				p3	p2	

(0 - p1) → zeros

(p1+1, p2-1) → ones

(p2, p3) ⇒ unknown

(p3+1, n-1) ⇒ 2's

(10, 10)

(p2, p3)

p2 > p3

swap(p2, p3)

```java
// (0,p1) -> zeros
// (p1+1,p2-1) -> ones
// (p2,p3) -> unknown
// (p3+1,n-1) -> twos
public static void sort_zero_one_two(int[] arr, int n){
    int p1=-1;
    int p2=0;
    int p3=n-1;

    while(p2<=p3){ // unknown area alive
        if(arr[p2]==0){
            p1++; // increase zero area
            swap(arr,p1,p2); // swapping 0 and 1
            p2++;
        } else if(arr[p2]==2){
            swap(arr,p2,p3);
            p3--;
        } else {
            p2++;
        }
    }
}
```

and 1

nn area

if (arr(p2) == 2){

swap(arr, p2, p3)

p3--;

}

↓

increase

2's area

Intersection of two arrays

a = {1, 1, 2, 3, 3}
b = {1, 1, 2, 2, 3}

a(i) == a(i-1)

a(i) = 1

b →

count ++

Common = {1, 2, 3} → 3

a → sort

a(i) == a(i-1) ← skip

i = 0 1 2 3 4 5

```java
// the intersection of two arrays.
public static int NumberofElementsInIntersection(int a[], int b[], int n, int m
    int count=0;

    Arrays.sort(a);

    for(int i=0; i<n; i++){
        // search a[i];
        if(i>0 && a[i]==a[i-1]){
            continue;
        }

        for(int j=0; j<m; j++){
            if(a[i]==b[j]){
                count++;
                break;
            }
        }
    }

    return count;
}
```

O(n*m)

a = {1, 2, 2, 3, 4, 5, 6}
b = {5, 6, 3, 4, 7, 2, 3}

{2, 3, 4, 5, 6}

count = 0 1 2 3 4

if (a(i) > b(j)) {
    j++;
}

a = {1, 2, 2, 4, 11, 14, 19, 19, 100, 120}

b = {2, 11, 12, 13, 14, 21, 25, 100, 109}

i = 0 1 2 3 4 5

j = 0 1 2

a(i) = 100 120
b(j) = 25 109

if (a(i) < b(j)) {
    i++;
}

if (a(i) == b(j)) {
    count++
    i++;
    j++;
}

```java
// the intersection of two arrays.
public static int NumberofElementsInIntersection(int a[], int b[], int n, int m) {
    int count=0;

    Arrays.sort(a);
    Arrays.sort(b);

    int i=0;
    int j=0;

    while(i<n && j<m){

        if(i>0 && a[i]==a[i-1]){
            i++;
            continue;
        }

        if(a[i]==b[j]){
            count++;
            i++;
            j++;
        } else if(a[i]<b[j]){
            i++;
        } else {
            j++;
        }
    }
```

$O(m+n)$

$7+9 \Rightarrow 16$

a = { 1, 2, 4, 11, 14, 19, 120, 130, 140, 150 }

b = { 2, 11, 12, 13, 21, 22, 130, 140, 150 }

ans =

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	2	4	11	11	12	13	14	19	21	22	120	130	140	150

a(i) = ~~4~~ ~~4~~ ~~14~~ ~~19~~ 120

b(j) = ~~2~~ ~~11~~ ~~12~~ ~~13~~ ~~21~~ 22

$-10^9 \to 10^9 \to$ int

$-10^{18} \to 10^{18} \to$ long

Two Sum

tar = 9

[-1, 11, -2, 5, 13, 9, 4, -4, 15] ⇒ 3

count = 1 2 3

sort ⇒ [-4, -2, -1, 4, 5, 9, 11, 13, 15]

tar = 9

arr(i) = 4

arr(j) = 5

sum = arr(i) + arr(j) = 9 ~~13~~

if (sum > tar)
j--;

i++ → sum increase

j-- → sum decrease

```java
// count of distinct pair with given sum
public static int countPairs(int[] nums, int target){
    int n=nums.length;

    int i=0;
    int j=n-1;
    int count=0;

    while(i<j){
        int sum=nums[i] + nums[j];

        if(sum==target){
            count++;
            i++;
            j--;
        } else if(sum<target){
            i++;
        } else {
            j--;
        }
    }

    return count;
}
```

[12, 3, 6, 1, 6, 9] tar = 24

required_sum = 24 − 6 = 18

k2

[1, 3, 6, 6, 9, 12]

i j

→ i = k + 1

→ j = n − 1

count == 2

arr(i) + arr(j) + arr(k) == tar

arr(i) + arr(j) = tar − arr(k)

T

(i < j)

⇒ arr(i) + arr(j) == T

where T = tar − arr(k)

2

arr(i) + arr(j) + arr(k) + arr(l) == tar

arr(i) + arr(j) + arr(k) = T

T = tar − arr(l).

→ 0th (largest negative)

→ n-1

arr(i) * arr(i)
arr(j) * arr(j)

O(n)

i

{-6, -3, -1, 2, 4, 5}

j

ans ⇒ | 1 | 4 | 9 | 16 | 25 | 36 |

arr(0) * arr(0);
arr(n-1) * arr(n-1);

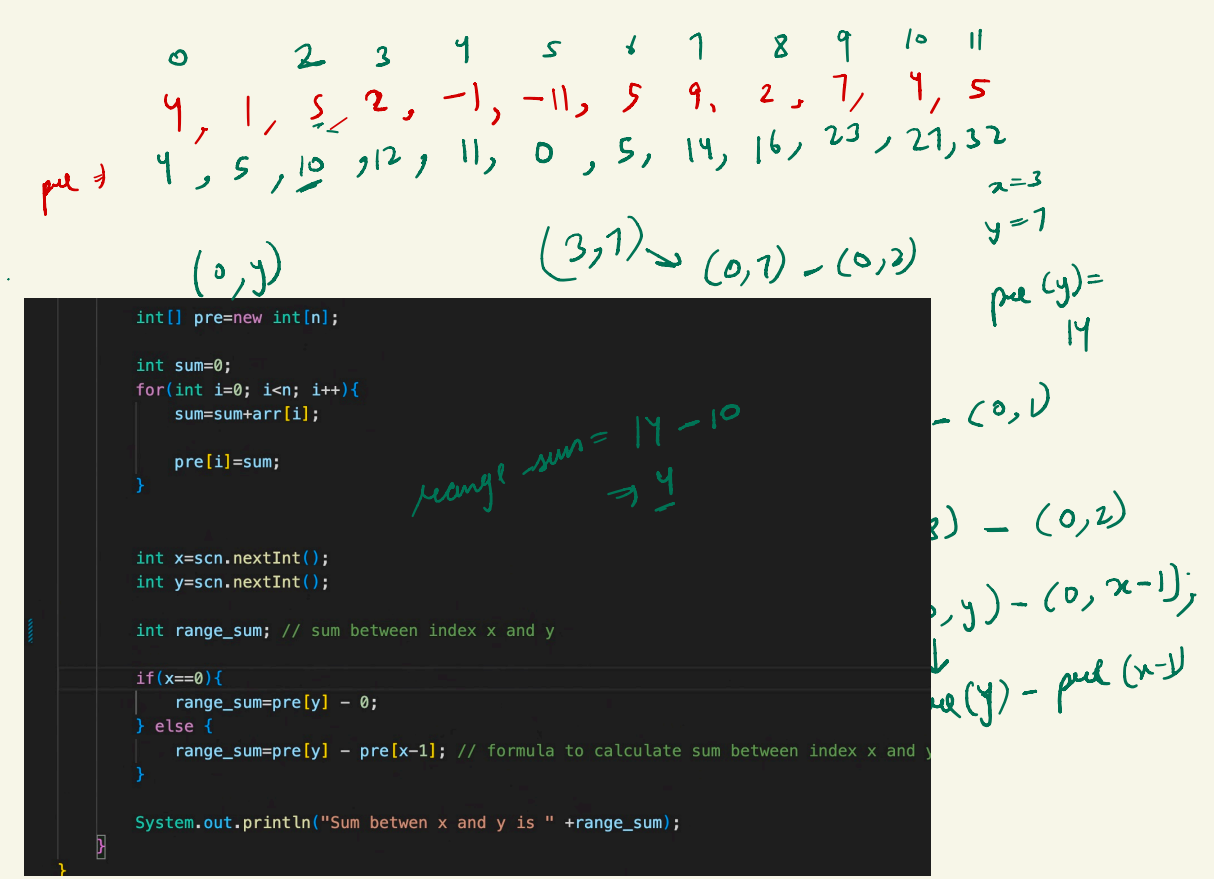
```java
public int[] sortedSquares(int[] nums) {
    int n=nums.length;

    int[] ans=new int[n];
    int k=n-1;

    int i=0;
    int j=n-1;

    while(i<=j){
        int isquare=nums[i]*nums[i];
        int jsquare=nums[j]*nums[j];

        if(isquare>jsquare){
            ans[k]=isquare;
            i++;
            k--;
        } else {
            ans[k]=jsquare;
            j--;
            k--;
        }
    }

    return ans;
}
```

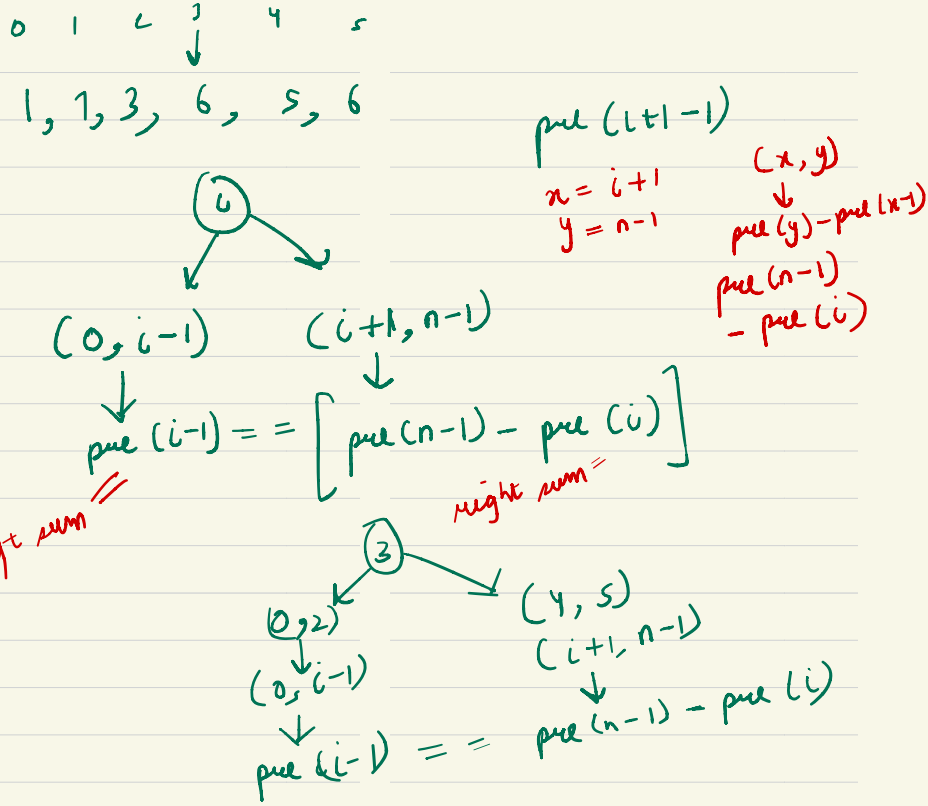
10^9

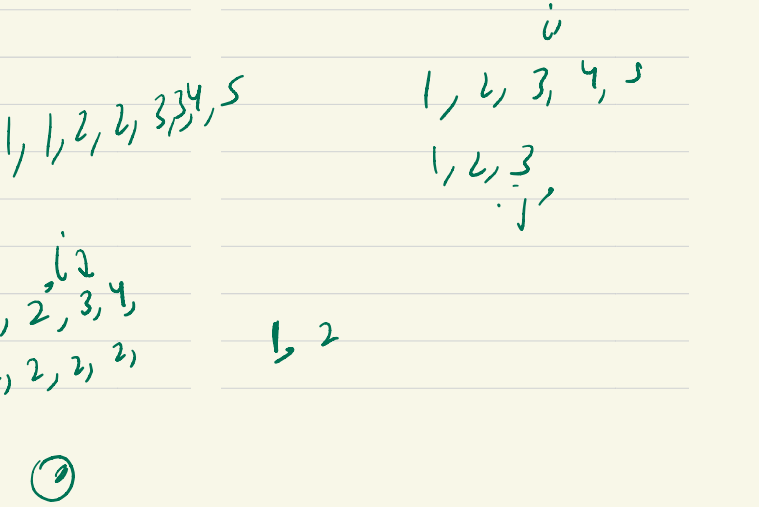
10^4 → n^2

10^5, 10^6 → $n \log n$

10^7, 10^8, 10^9 → $O(n)$


0 2 3 4 5 6 7 8 9 10 11
4, 1, 5, 2, -1, -11, 5 9, 2, 7, 4, 5
pre → 4, 5, 10, 12, 11, 0, 5, 14, 16, 23, 27, 32
x=3
y=7
pre(y)= 14
(0,y)
(3,7) → (0,7) - (0,2)
range_sum = 14 - 10
⇒ 4
- (0,1)
) - (0,2)
, y) - (0, x-1);
e(y) - pre(x-1)
int[] pre=new int[n];
int sum=0;
for(int i=0; i<n; i++){
sum=sum+arr[i];
pre[i]=sum;
}
int x=scn.nextInt();
int y=scn.nextInt();
int range_sum; // sum between index x and y
if(x==0){
range_sum=pre[y] - 0;
} else {
range_sum=pre[y] - pre[x-1]; // formula to calculate sum between index x and y
}
System.out.println("Sum betwen x and y is " +range_sum);
}
}


0 1 2 3 4 5
1, 1, 3, 6, 5, 6
pre(i+1-1)
x = i+1
y = n-1
(x, y)
pre(y) - pre(x-1)
pre(n-1)
- pre(i)
i
(0, i-1)
(i+1, n-1)
pre(i-1) == [pre(n-1) - pre(i)]
left sum
right sum
3
(0, 2)
(4, 5)
(0, i-1)
(i+1, n-1)
pre(i-1) == pre(n-1) - pre(i)


1, 1, 2, 2, 3, 3, 4, 5
i
1, 2, 3, 4, 5
1, 2, 3
j
i, 2
1, 1, 1, 1, 2, 2, 2, 3, 4,
1, 1, 1, 1, 2, 2, 2, 2, 2,
1, 1, 1, 1, 2, 2
j
1, 2
0