Revision

Insertion.

Comparator

Lambda. ~~~~> Syntax. easy.

compare. → +ve (swap).
→ 0
→ -ve

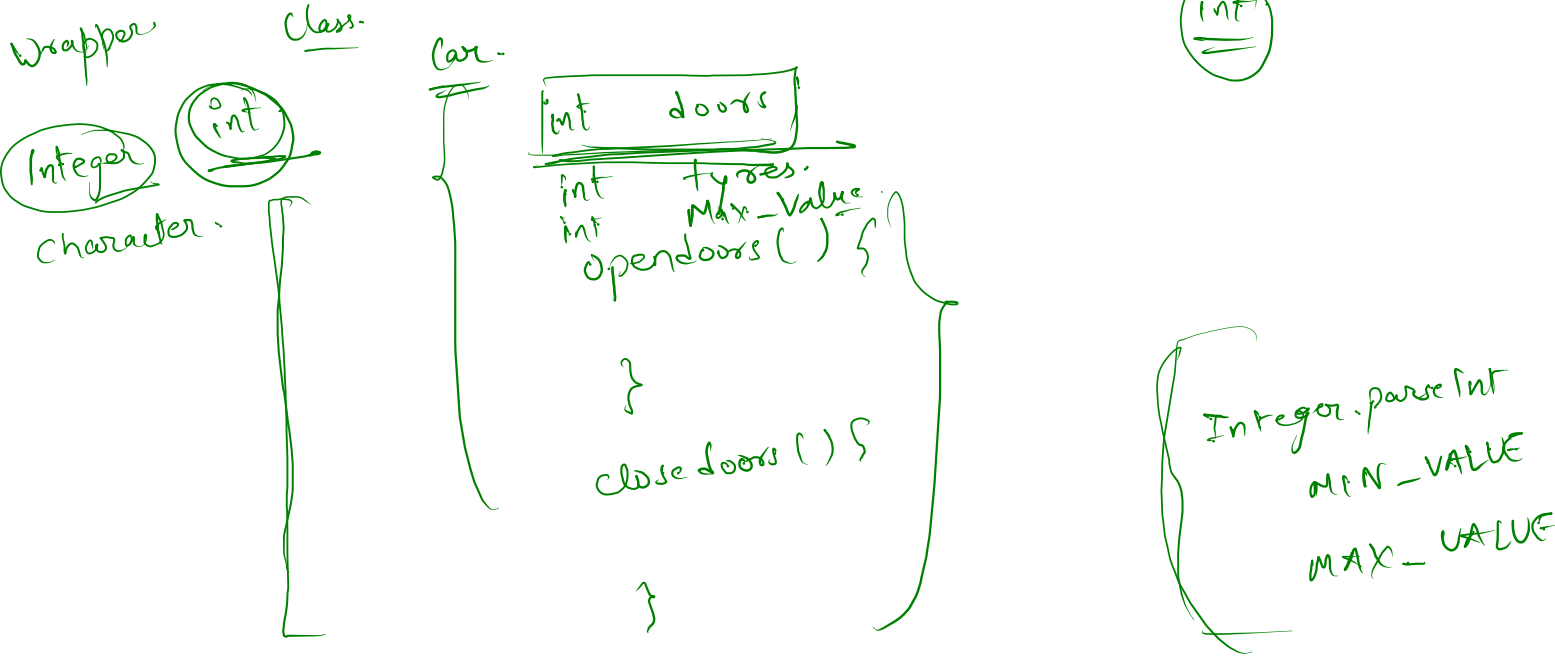
Summary

increasing. a - b.

decreasing b - a

} .

} Min Value
Max Value.
function.
use.


Wrapper
Class.
Integer
int
character.
Car.
int doors
int types.
int Max_Value
opendoors ( ) {
}
closedoors ( ) {
}
int
Integer.parseInt
MIN_VALUE
MAX_VALUE

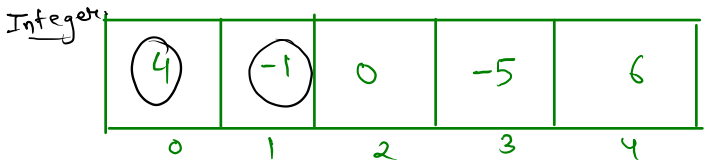
Sort the array according to their Square of each element

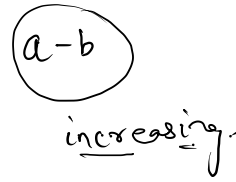
Sample Input 0

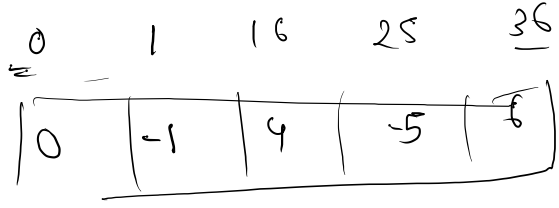
```
5
4 -1 0 -5 6
```

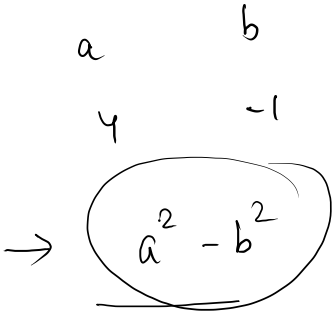
Sample Output 0

```
0 -1 4 -5 6
```









```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();

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

    Comparator<Integer> myComp=new Comparator<Integer>(){
        public int compare(Integer a, Integer b){
            return a*a-b*b;
        }
    };

    Arrays.sort(A, myComp );

    // Arrays.sort(A, (a,b) -> a*a - b*b );

    for(int i = 0; i < n; i++){
        System.out.print(A[i] + " ");
    }
}
```

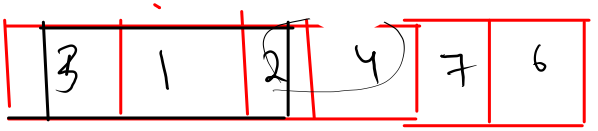
Sort Array By Parity

Sample Input 0

```
4
3 1 2 4
```

Sample Output 0

```
2 4 1 3
```

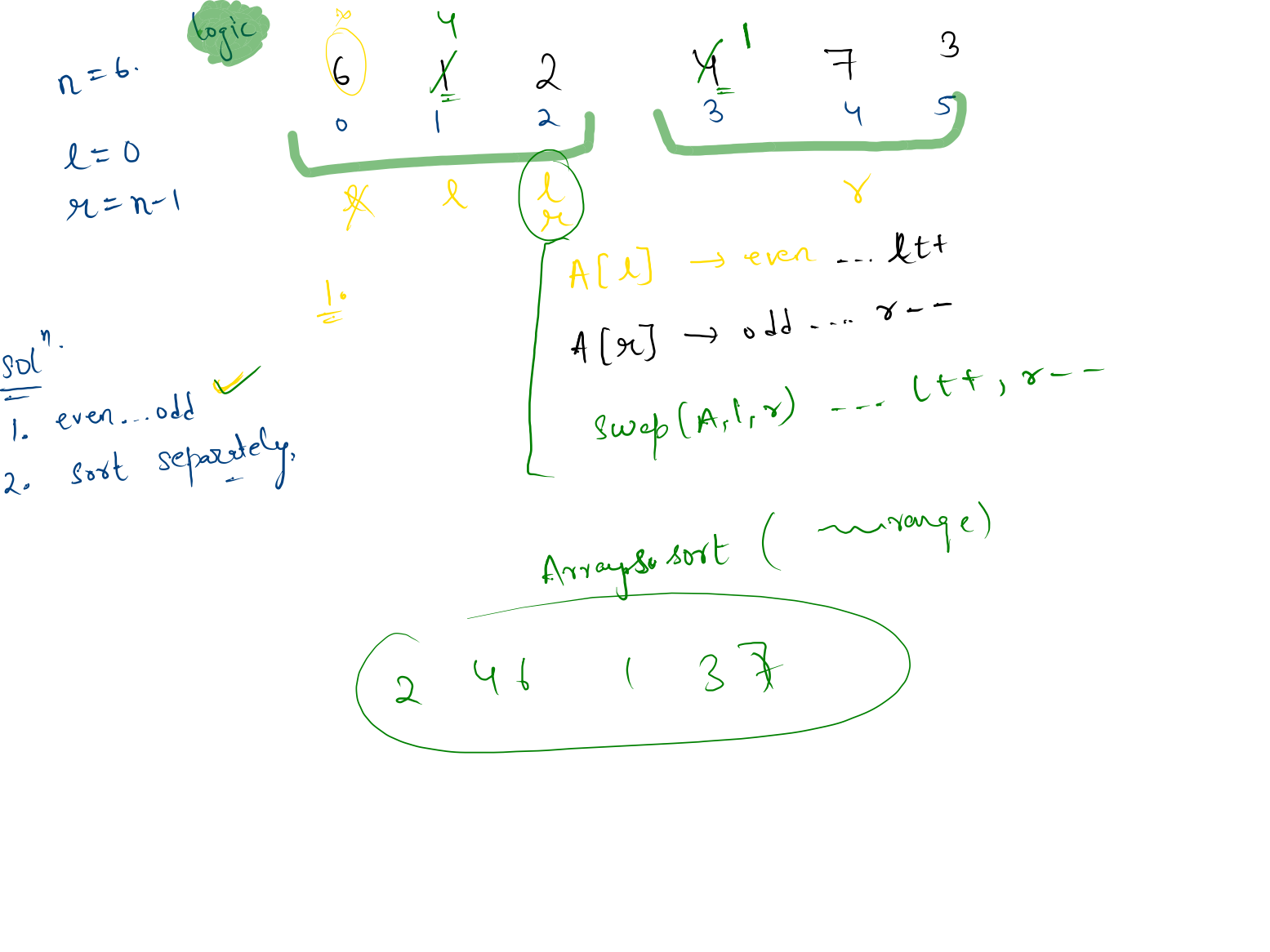


even ---- odd (increasing)

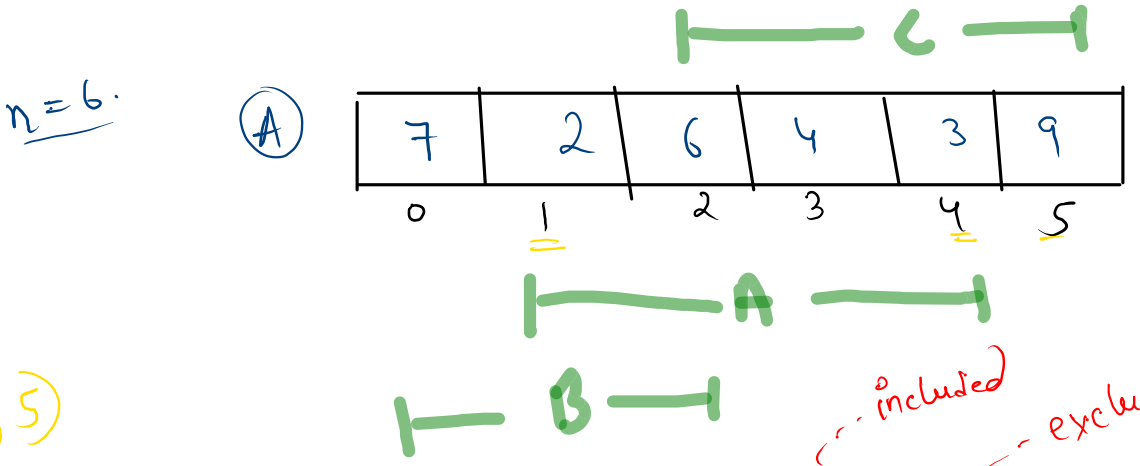
learn?

→ A different way to use Arrays.sort

sorting in range


logic
n = 6.
l = 0
r = n-1
6
4
2
1
7
3
0
1
2
3
4
5
l
r
A[l] → even ... l++
A[r] → odd ... r--
swap(A, l, r) ... l++, r--
1.
Sol^n.
1. even...odd
2. sort separately.
Arrays.sort (...range)
2 4 6 1 3 7

Arrays.sort() : In a particular range.



case A → Arrays.sort(A, 1, 5);

B → Arrays.sort(A, 0, 3),

C → Arrays.sort(A, 2, 6),

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

    int n = scn.nextInt();

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

    //Step 1 .. even.. odd
    int l = 0;      //left
    int r = n-1;    //right

    while(l <= r){
        if(A[l] % 2 == 0){
            l++;
        }
        else if(A[r] % 2 != 0){
            r--;
        }
        else{
            int tmp = A[l];
            A[l] = A[r];
            A[r] = tmp;
            l++;
            r--;
        }
    }

    // Step 2: Sort in a range
    Arrays.sort(A, 0 , l);
    Arrays.sort(A, l, n);
```

Minimum difference 7

max - min.

4 n
9 4 1 7
2 k

k=2.

9 4 1 7

1 7 = 6

9 4 = 5

9 1 = 8

9 7 = 2

4 1 = 3

4 7 = 3

ans = 2.

k=3.

9 4 1 7

9 4 1

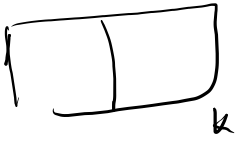
4 1 7

9 4 7

4 9 1

9 1 7

$K = 2$.



max − min = ⓓ ↓

ans = 2

9 4 1 7

9 4 = 5

4 9 = 5

9 1 = 8

9 7 = 2

4 1 = 3

4 7 = 3

1 7 = 6

k=3.

9 4 1 7

max − min

9 4 1 = 8

9 4 7 = 5 ✓

4 1 7 = 6

9 1 7 = 8

d

ans = 5.

$K = 4$

9 4 1 7

max - min

9 4 1 7 = 8

ans = 8

$K = 2$.

$K = 3$

$K = 4$

for

for - - 3

. . . -

logic → optimized soln

$k = 3$

9 1 4 7

↳

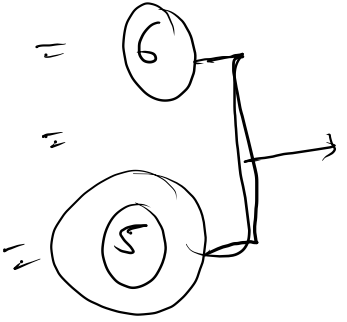
1 4 7 9

max - min

last - first = 6

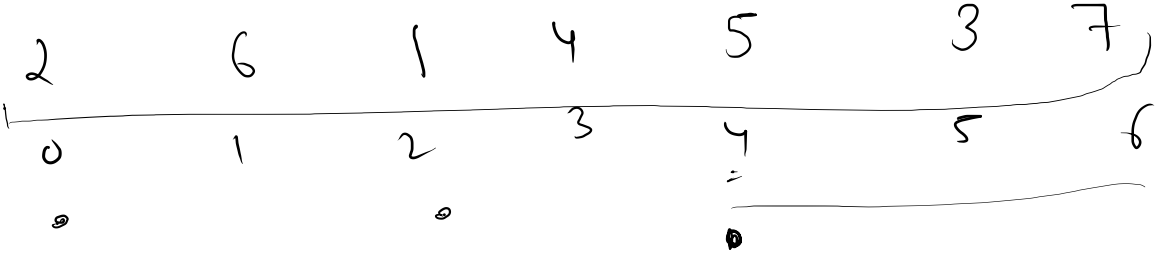
last - first = 5

9 - 4 =



n = 7

- 3

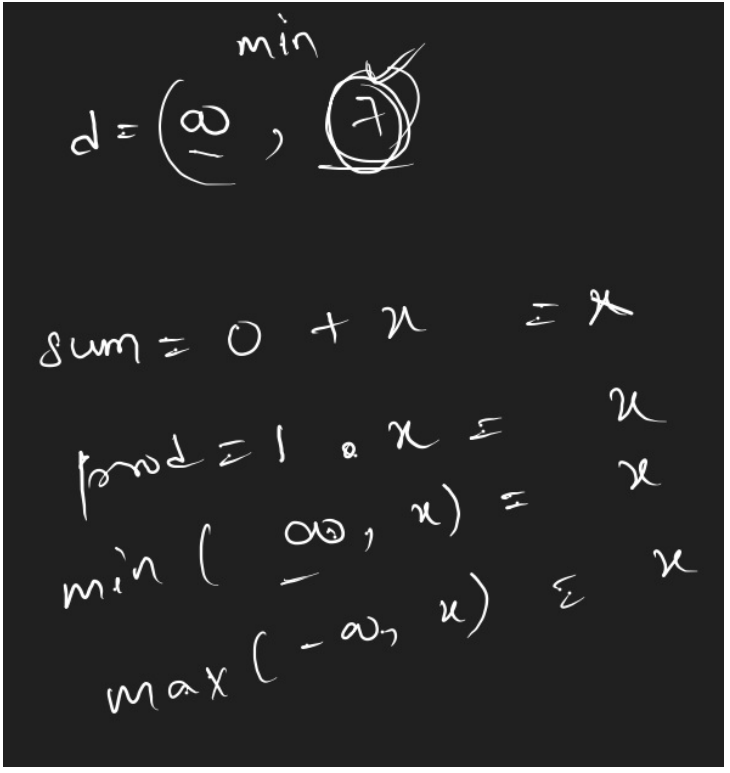


K = 3

d = 3

i + k - 1

1 + 3 - 1


min
d = (∞ , 7)
sum = 0 + x = x
prod = 1 . x = x
min (∞, x) = x
max (-∞, x) = x

4 2 9 5 12

① 4 2 9 = 7

② 4 2 . 5 = 3

③ 4 2 . 12 = 10

④ 4 9 5

⑤ 4 9 12

⑥ 2 9 5

⑦ 2 9 12

⑧ 2 5 12

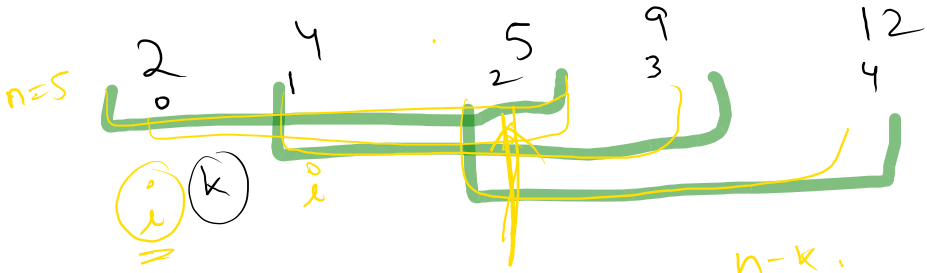
⑨ 9 5 12

⑩ 4 5 12

4 2 9 5 12

↓

k=3.



to sort

✓		✓	=	3
2	4	5		
4	5	9	=	5
5	9	12	=	7

$i = [0 \dots n-k]$

n = 5

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

    int n = scn.nextInt();

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


    int k = scn.nextInt();

    int d = Integer.MAX_VALUE;      // very large int value
    Arrays.sort(A);

    for(int i = 0; i <= n-k; i++){
        int min = A[i];
        int max = A[i+k-1];
        d = Math.min(d, max-min);
    }

    System.out.println(d);

}
```