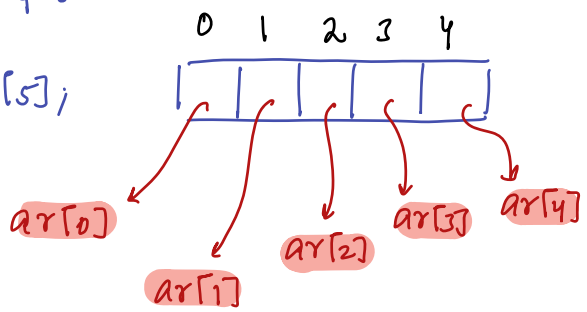
Today's Content:

- → 1D arrays content
- → 2D arrays Intro
- → How to sort 1D arrays

1D:

```
int ar[] = new int[size];
int ar[] = new int[5];
```




```
// void printar(int ar[]){
    int n = ar.length;
    for(int i=0; i<n; i++){
        print(ar[i])
    }
}
```

1) Given an array, print all unique elements. → { its frequency in array = 1 }

Ex1: ar[7] = { 2 5 7 6 7 5 9 }
(indices: 0 1 2 3 4 5 6)

→ output: 2 6 9

Ex2: ar[8] = { 6 3 2 2 7 3 5 2 }

→ output: 6 7 5

Idea: Every ar[i] element, check if its unique or not?

```
void printUnique(int ar[]){
    int n = ar.length;
    for(int i=0; i<N; i++){
        // We need to check if ar[i] is unique or not?
        // If freq of ar[i] == 1, then that can unique
        int c=0;
        for(int j=0; j<N; j++){
            if( ar[i] == ar[j] ) { c=c+1 }
        }
        if(c==1){
            // ar[i] is unique
            print(ar[i])
        }
    }
}
```

Doubts

- j = i+1
- j = i
- j < N-1

	0	1	2	3	4	5	6
ar[7] = {	2	5	7	6	7	5	9 }

i	ar[i]	{iterate on 0, N-1, get freq ar[i]}
0	2	freq of 2 = 1, print (2)
1	5	freq of 5 = 2
2	7	freq of 7 = 2
3	6	freq of 6 = 1 print (6)
4	7	freq of 7 = 2
5	5	freq of 5 = 2
6	9	freq of 9 = 1 print (9)

```
void printUnique(int ar[]){
    int n = ar.length;
    for(int i=0; i<n; i=i+1){
        // We need to check if ar[i] is unique or not?
        // If freq of ar[i] == 1, then that can unique
        int c = 0;
        for(int j=0; j<n; j=j+1){
            if( ar[i] == ar[j] ) { c = c+1 }
        }
        if(c==1){
            // ar[i] is unique
            print(ar[i])
        }
    }
}
```

// ar[5] = { 3 2 4 2 6 }

0 1 2 3 4

i	ar[i]
0	3

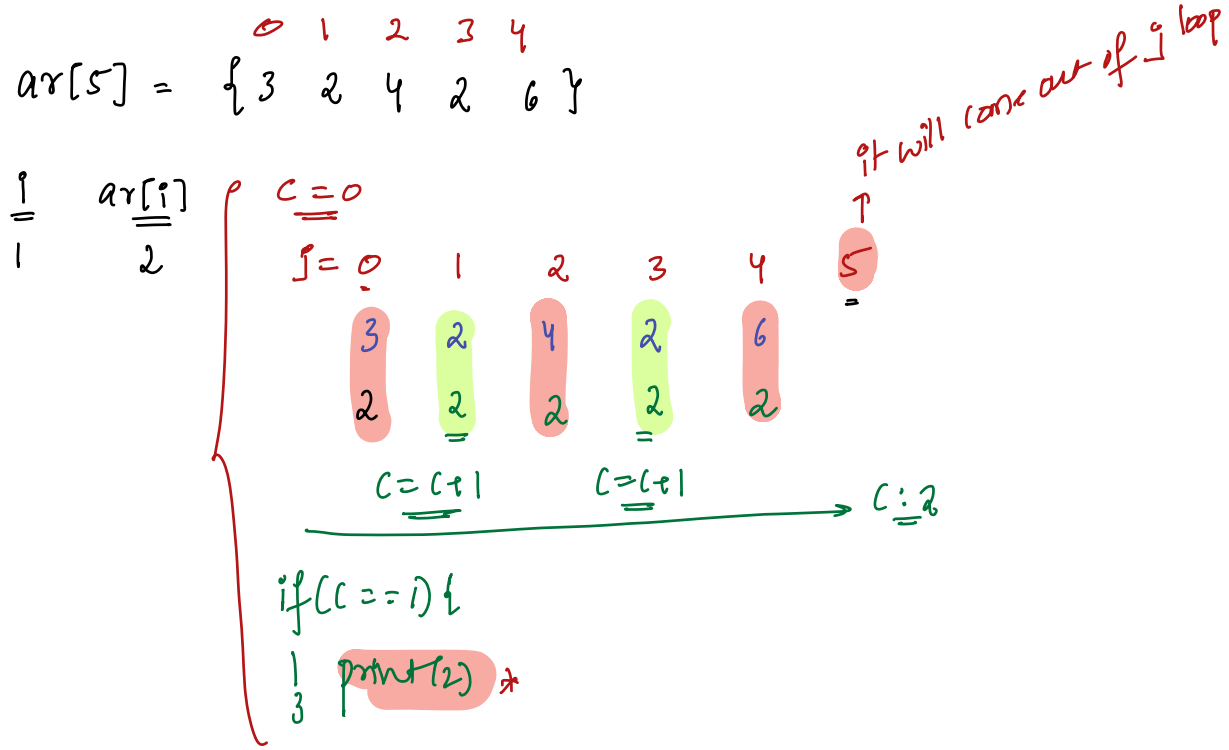
c = 0

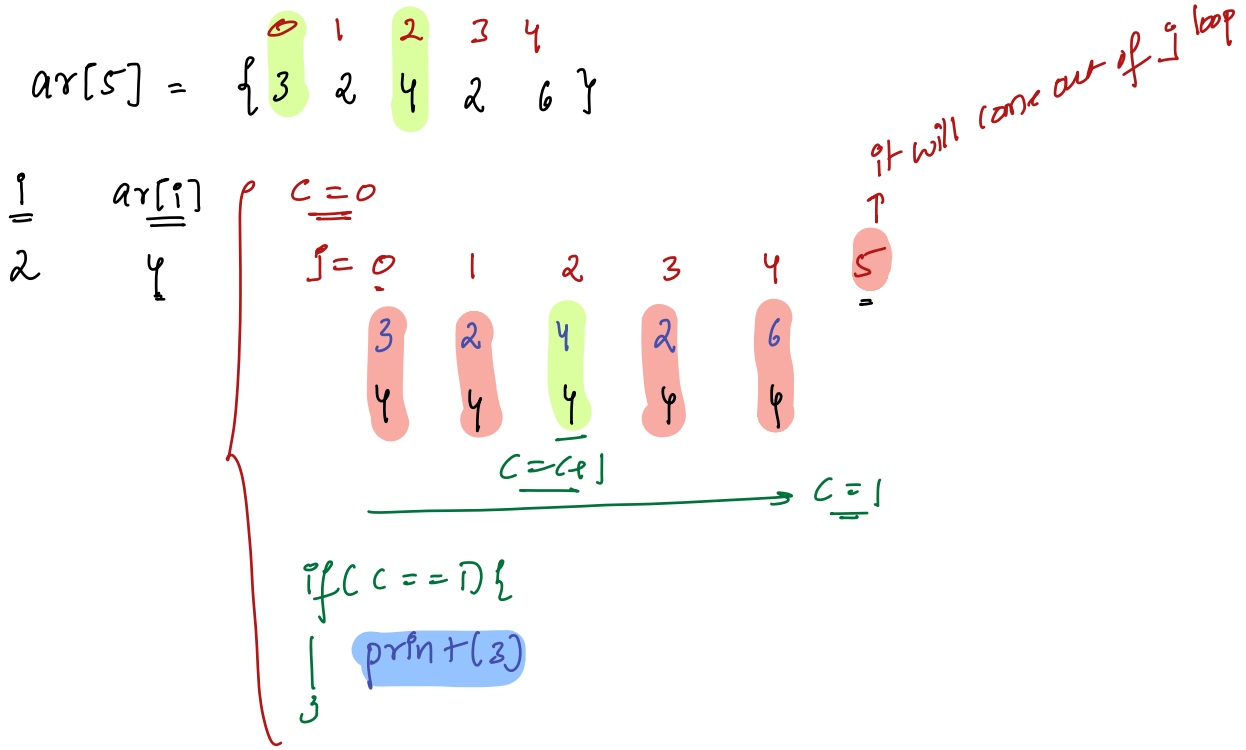
j	0	1	2	3	4	5
	3	2	4	2	6	
	3	3	3	3	3	

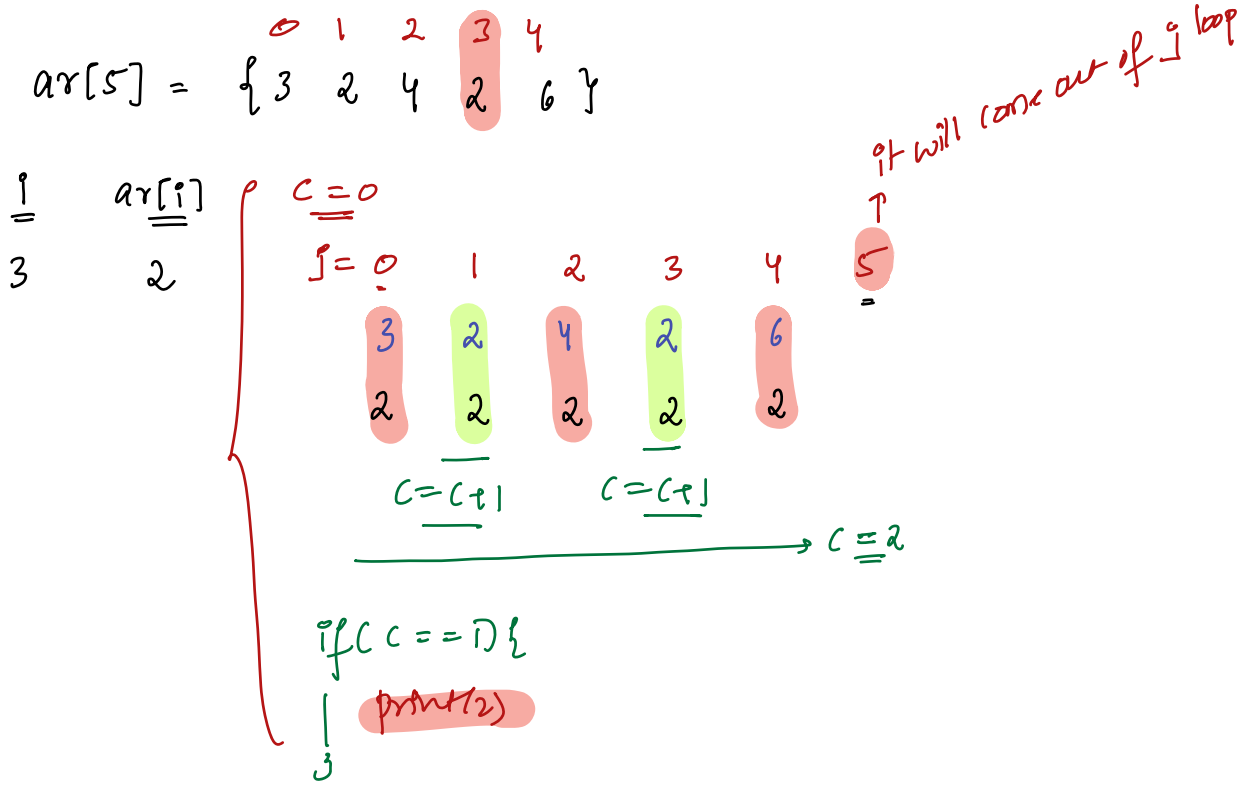
c = c+1 → c = 1

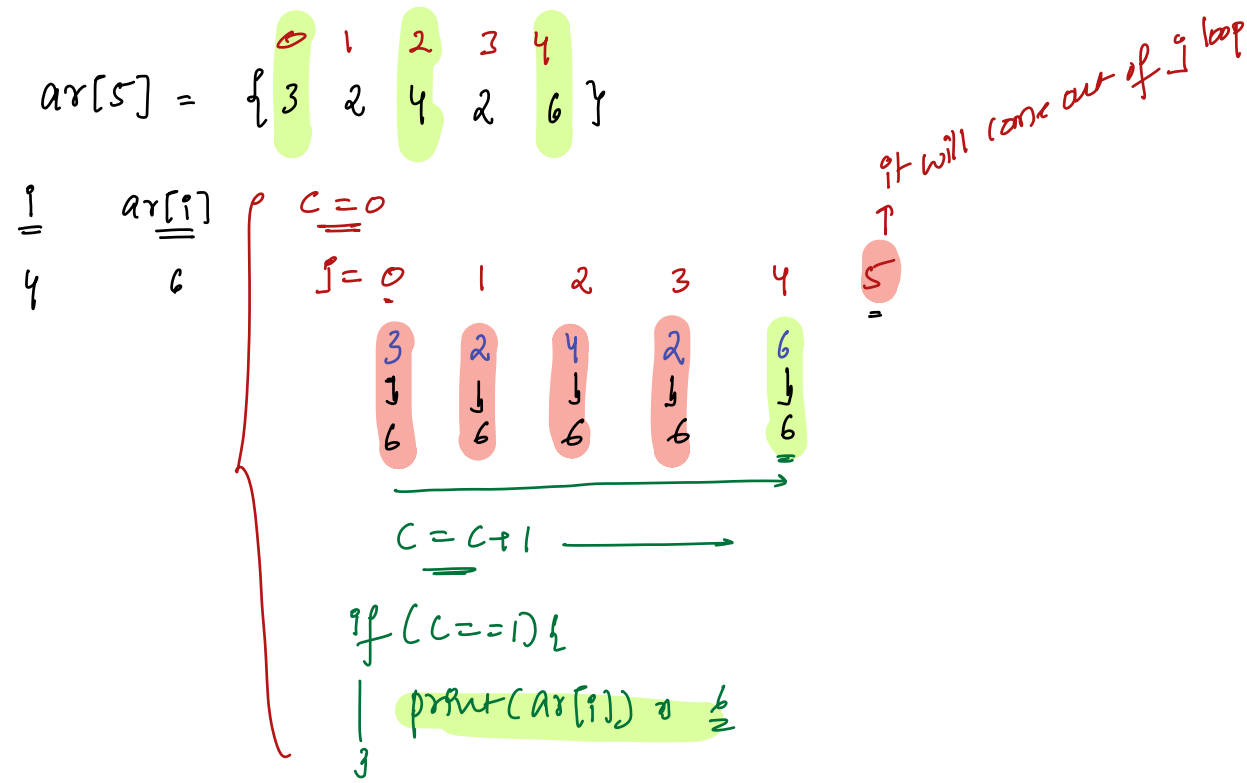
it will come out of j loop

```
if(c==1){
    print(3)
}
```


0 1 2 3 4
ar[5] = { 3 2 4 2 6 }
i ar[i]
1 2
c = 0
j = 0 1 2 3 4 5
it will come out of j loop
3 2 4 2 6
2 2 2 2 2
c = c+1
c = c+1
c : 2
if(c == 1){
print(2)
}


0 1 2 3 4
ar[5] = { 3 2 4 2 6 }
i ar[i]
2 4
c = 0
j = 0 1 2 3 4 5
it will come out of j loop
3 2 4 2 6
4 4 4 4 4
c = c+1
c = 1
if(c == 1){
print(3)
}


ar[5] = { 3 2 4 2 6 }
0 1 2 3 4
i = 3
ar[i] = 2
c = 0
j = 0 1 2 3 4 5
it will come out of j loop
3 2 4 2 6
2 2 2 2 2
c = c+1
c = c+1
c = 2
if(c == 1){
print(2)
}


ar[5] = { 3 2 4 2 6 }
0 1 2 3 4
i = 4
ar[i] = 6
c = 0
j = 0 1 2 3 4 5
it will come out of j loop
3 2 4 2 6
6 6 6 6 6
c = c+1
if(c == 1){
print(ar[i]) // 6
}

```
void printUnique(int ar[]){
    int n = ar.length;
    for(int i=0; i<N; i=i+1){
        // We need to check if ar[i] is unique or not?
        // If freq of ar[i] == 1, then that can unique
        int c = 0;
        for(int j = i+1; j<N; j=j+1){
            if(ar[i] == ar[j]) {c=c+1}
        }
        if(c==1){
            // ar[i] is unique
            print(ar[i])
        }
    }
}
```

may change `j=i+1` → { Code won't work }

0 1 2 3

Eg: ar[4] = {3 2 3 4} j = i+1 .. N-1 —

i = 0 ar[i] : 3

c=0 j = 1 2 3 → c=1

1	2	3
2	3	4
3	3	3

c=c+1

if(c==1){ print(3) → {not unique element}

//

→ given ar[], print all unique elements in sorted order

↳ Sorted order?

ar[7] = { 3 2 6 7 7 4 1 }

output: 1 2 3 4 6

ar[6] = { 2 4 8 4 5 -3 }

output: -3 2 5 8

Approach1:

Step1: Sort given array

Step2: Apply above unique function

ar[6] = { 2 4 8 4 5 -3 }

↓

After Sorting = { -3 2 4 4 5 8 }

↓ freq: 1 1 2 2 1 1

↓ find unique

Output: { -3 2 5 8 }

// given ar[]

Arrays.sort(ar)

printUnique(ar)

// To Sort array elements

Arrays.sort(ar)

// Assignment: Print all unique elements in increasing order.

2Q): Given a ar[], check if there exists (i,j) such that ar[i] + ar[j] == k && i != j

Index value

{ Both index should be different }

$$ar[5] = \{3\ 2\ 9\ 6\ 1\}$$ (indices 0 1 2 3 4)

k = 8 → ar[1] + ar[3] == 8 : return True

$$ar[6] = \{3\ 2\ 6\ 4\ 7\ 5\}$$ (indices 0 1 2 3 4 5)

k = 9 → ar[0] + ar[2] == 9, ar[1] + ar[4] == 9, ar[3] + ar[5] == 9 : return True

$$ar[5] = \{3\ 2\ 9\ 6\ 1\}$$ (indices 0 1 2 3 4) → { Return False }

k = 13 → ar[0] + ar[2] + ar[4] = 13 ✗
(3, 9, 1)

10:35pm
- 10:45pm →
- please go for your break

option

1) Wed — 2D arrays; Friday — ArrayList; Sat — String; Monday — Oops basics

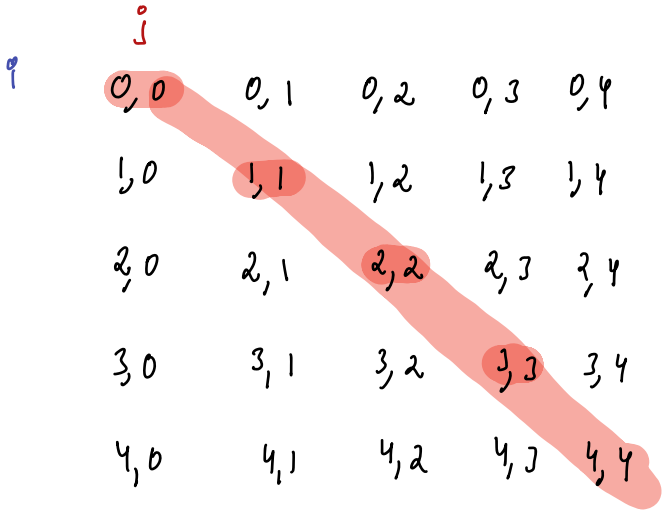
2) Tue — 2D arrays; Thurs — ArrayList; Sat — Strings; Mond — Oops basics

→ Wednesday → Exam

Approx: 27th

```
//  ar[5] = { 3  2  9  8  6 }
              0  1  2  3  4
```

All pairs:



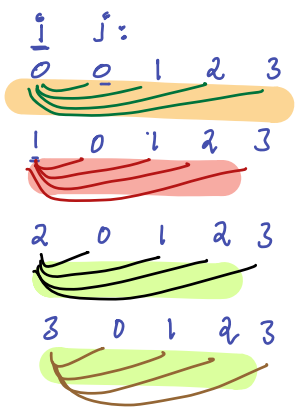

```
// boolean checkPairSum(int ar[], int k)
```

input array

Value sum we need to check

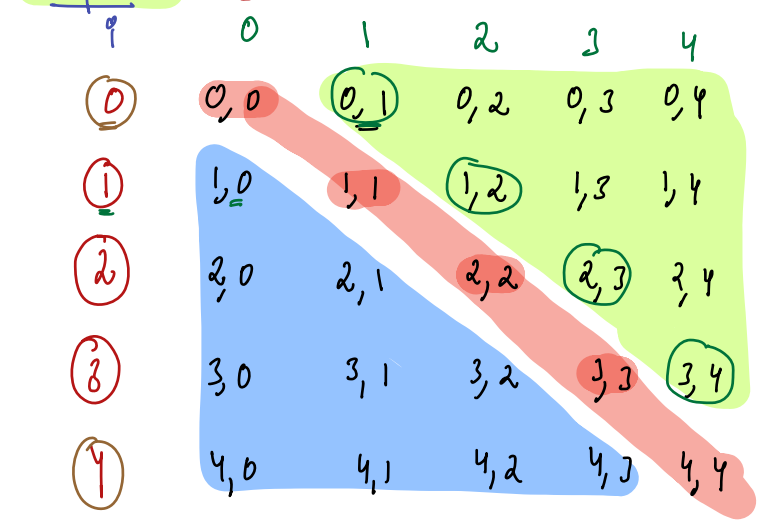
```
int n = ar.length;
// Pseudocode to get all Pairs
for(int i=0; i<n; i++){
    for(int j=0; j<n; j++){
        {i, j}
        if( i!=j && ar[i]+ar[j]==k)
            return True
    }
}
return False
}
```

True : n = 4



```
//  ar[5] = { 3  2  9  8  6 }
            0  1  2  3  4
```

All pairs:



i=0 j=1 : ar[0]+ar[1] == k

Idea: Either iterate in upper Triangle or lower Triangle

Iterate in upper part:

i=0; j=1
i=1; j=2
i=2, j=3
i=3, j=4
i=4 j=5 → { come out }

// input array
// boolean checkPairSum(int ar[], int k) Value sum we need to check

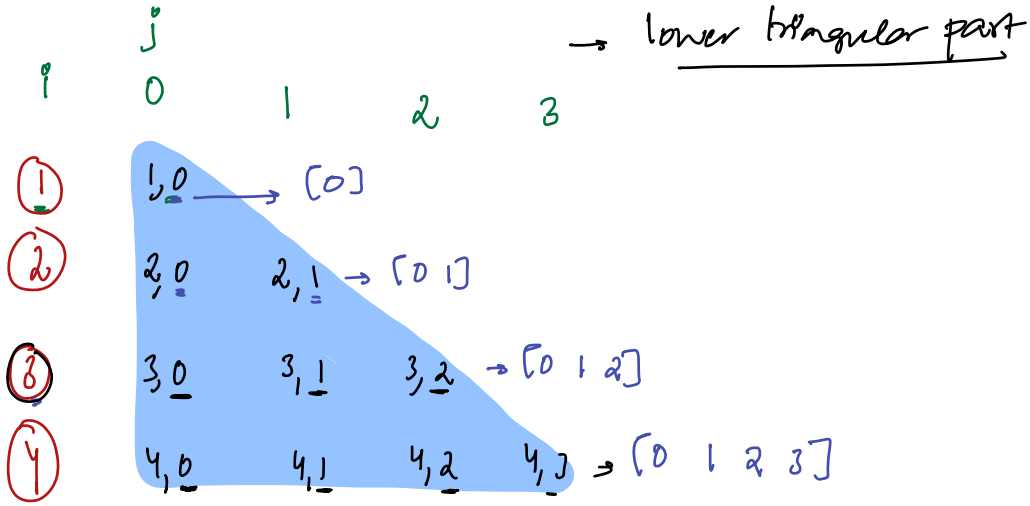
```
int n = ar.length;
// Pseudocode to get all Pairs
for(int i=0; i<n; i++){
    for(int j=i+1; j<n; j++){
        {i, j}
        if (ar[i]+ar[j]==k)
            return True
    }
}
return false
```

i = N-1

```
j = N; j<N; j++) {
    // won't go inside loop
}
```

condition fails

```
}
```




```
// for(int i=1; i<N; i++){
    for(int j=0; j<i; j++){
        if(ar[i]+ar[j]==k){
            return True
        }
    }
}
return False
```

Nearest sqrt → Given N → +ve Number

N = 24 → 4 :

```
ans = 0
for(int i=1; i*i<=N; i=i+1)
    if(i*i > N) {        // i is not sqrt,
        break;
    }
    ans = i;
print(ans)
```

N = 24, ans = 0

i	i*i <= N :	i*i > N : break	
1	1*1 <= 24 →	1*1 > 24 :	ans = 1
2	2*2 <= 24 →	2*2 > 24 :	ans = 2
3	3*3 <= 24 →	3*3 > 24 :	ans = 3
4	4*4 <= 24 →	4*4 > 24	ans = 4
5	5*5 <= 24 →	5*5 > 24	break