Lecture 5

Part A

Two-Dimensional Arrays - Nested Loops

Nested Loops: Semantics and Tracing

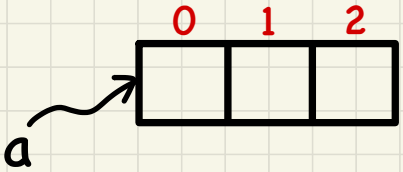
```
for(int i = 0; i < a.length; i ++) {
  for(int j = 0; j < a.length; j ++) {
    System.out.println("(" +i+ ", " +j+ ")");
  } }
```

3

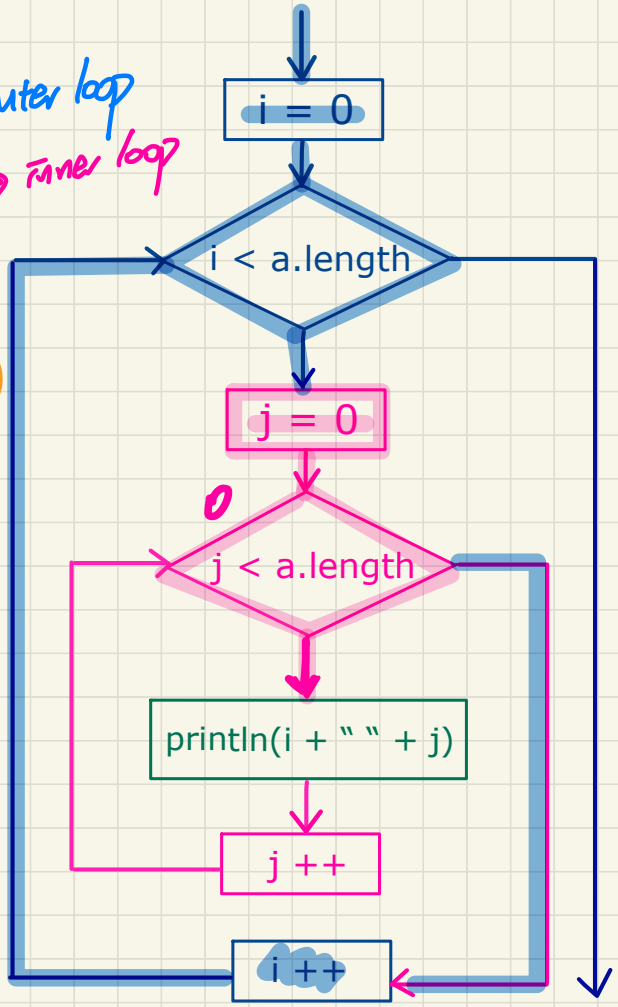
outer loop

inner loop

3 * 3 = 9



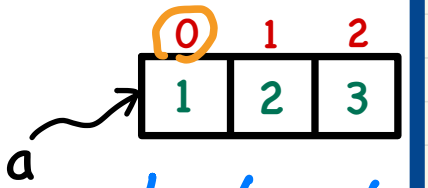
i	j
0	0
0	1
0	2
1	0
1	1
1	2
2	0
2	1
2	2



Computational Problem: Finding Duplicates

No Duplicates,
Redundant Scan

```
1  /* Version 1 with redundant scan */
2  int[] a = {1, 2, 3}; /* no duplicates */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length; i ++) {
5    for(int j = 0; j < a.length; j ++) {
6      hasDup = hasDup || (i != j && a[i] == a[j]);
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```



a.length == 100

redundant

i	j	i != j	a[i]	a[j]	a[i] == a[j]	hasDup
0	0	false	1	1	true	false
0	1	true	1	2	false	false
0	2	true	1	3	false	false
1	0	true	2	1	false	false
1	1	false	2	2	true	false
1	2	true	2	3	false	false
2	0	true	3	1	false	false
2	1	true	3	2	false	false
2	2	false	3	3	true	false

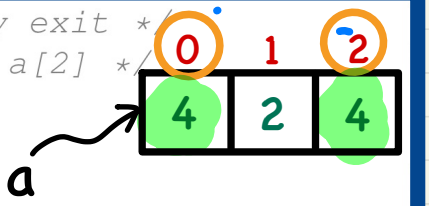
i==j

unnecessary to check

Computational Problem: Finding Duplicates

Redundant Scan, No Early Exit

```
1  /* Version 1 with redundant scan and no early exit */
2  int[] a = {4, 2, 4}; /* duplicates: a[0] and a[2] */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length; i ++) {
5    for(int j = 0; j < a.length; j ++) {
6      hasDup = hasDup || (i != j && a[i] == a[j]);
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```



i	j	i != j	a[i]	a[j]	a[i] == a[j]	hasDup
0	0	false	4	4	true	false
0	1	true	4	2	false	false
0	2	true	4	4	true	true
1	0	true	2	4	false	true
1	1	false	2	2	true	true
1	2	true	2	4	false	true
2	0	true	4	4	true	true
2	1	true	4	2	false	true
2	2	false	4	4	true	true

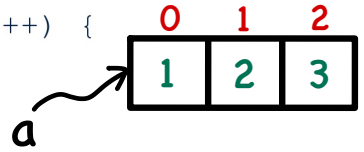
redundant to check further

we could have stopped here

Computational Problem: Finding Duplicates

No Duplicates, Redundant Scan

```
1  /* Version 2 with redundant scan */
2  int[] a = {1, 2, 3}; /* no duplicates */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length && !hasDup; i ++) {
5    for(int j = 0; j < a.length && !hasDup; j ++) {
6      hasDup = i != j && a[i] == a[j];
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```



i	j	i != j	a[i]	a[j]	a[i] == a[j]	hasDup
0	0	false	1	1	true	false
0	1	true	1	2	false	false
0	2	true	1	3	false	false
1	0	true	2	1	false	false
1	1	false	2	2	true	false
1	2	true	2	3	false	false
2	0	true	3	1	false	false
2	1	true	3	2	false	false
2	2	false	3	3	true	false

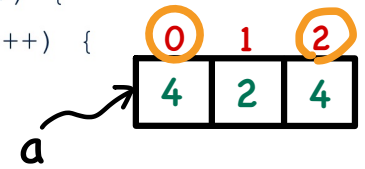
Computational Problem: Finding Duplicates

Duplicates, Early Exit

```
1  /* Version 2 with redundant scan and early exit */
2  int[] a = {4, 2, 4}; /* duplicates: a[0] and a[2] */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length && !hasDup; i ++) {
5    for(int j = 0; j < a.length && !hasDup; j ++) {
6      hasDup = i != j && a[i] == a[j];
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```

F

F



i	j	i != j	a[i]	a[j]	a[i] == a[j]	hasDup
0	0	false	4	4	true	false
0	1	true	4	2	false	false
0	2	true	4	4	true	true

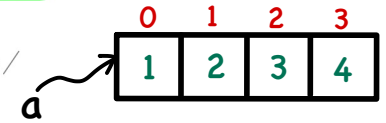
cause early exit as soon as a satisfaction witness is found.

Computational Problem: Finding Duplicates

No Duplicates,
Non-Redundant Scan

got rid of : ① 0,0
② 0,1
1,0 ×

```
1  /* Version 3 with no redundant scan */
2  int[] a = {1, 2, 3, 4}; /* no duplicates */
3  boolean hasDup = false;
4  for(int i = 0; i < a.length && !hasDup; i ++) {
5    for(int j = i + 1; j < a.length && !hasDup; j ++) {
6      hasDup = a[i] == a[j];
7    } /* end inner for */ } /* end outer for */
8  System.out.println(hasDup);
```



i	j	a[i]	a[j]	a[i] == a[j]	hasDup
0	1	1	2	false	false
0	2	1	3	false	false
0	3	1	4	false	false
1	2	2	3	false	false
1	3	2	4	false	false
2	3	3	4	false	false

$(n-1) + (n-2) + \cdots + 1$ V3

n

v1, v2
I
0

I
v1, v2
n * n

V3

J
0 ... n-1
0 ... n-1

I
0
1
...
n-1

J
I .. n-1
2 .. n-1
n-2

nx

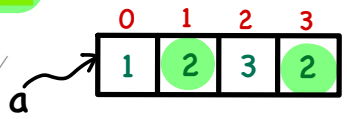
Computational Problem: Finding Duplicates

Duplicates,
Non-Redundant Scan,
Early Exit

```
1  /* Version 3 with no redundant scan:
2   * array with duplicates causes early exit
3   */
4  int[] a = {1, 2, 3, 2}; /* duplicates: a[1] and a[3] */
5  boolean hasDup = false;
6  for(int i = 0; i < a.length && !hasDup; i ++) {
7    for(int j = i + 1; j < a.length && !hasDup; j ++) {
8      hasDup = a[i] == a[j];
9    } /* end inner for */ } /* end outer for */
10 System.out.println(hasDup);
```

F

F



i	j	a[i]	a[j]	a[i] == a[j]	hasDup
0	1	1	2	false	false
0	2	1	3	false	false
0	3	1	2	false	false
1	2	2	3	false	false
1	3	2	2	true	true

→ exit from both loops.