Monday February 25

Lecture 13

Computational Problem: Are all numbers positive?

Is there at least one positive?

Four possible solutions (`posNumsSoFar` is initialized as *true*):

1. Scan the entire array and accumulate the result.

```
for (int i = 0; i < ns.length; i ++) {
  posNumsSoFar = posNumsSoFar && ns[i] > 0; }
```

2. Scan the entire array but the result is **not** accumulative.

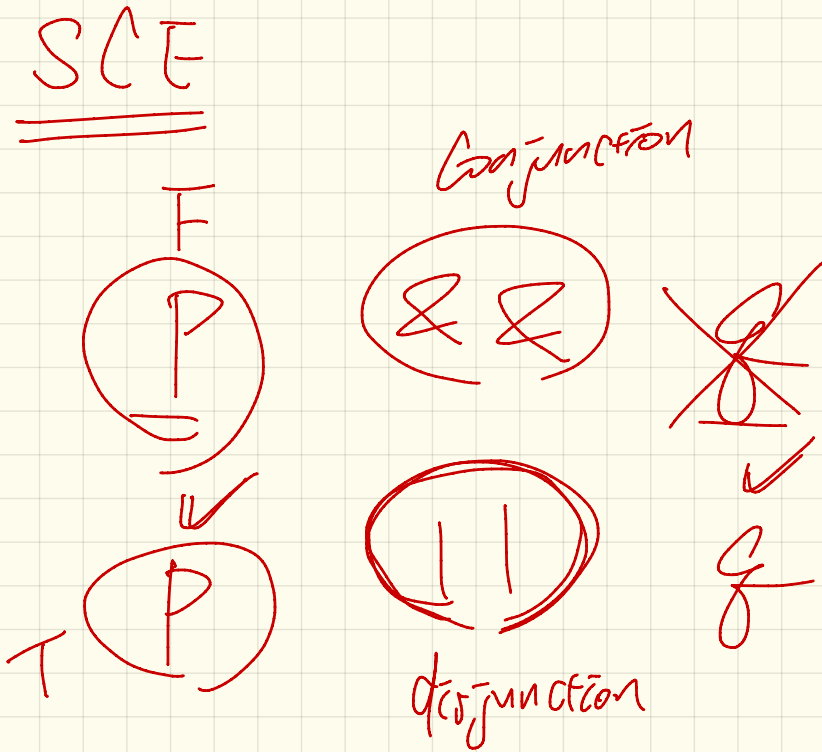
```
for (int i = 0; i < ns.length; i ++) {
  posNumsSoFar = ns[i] > 0; }  /* Not working.  Why?  */
```

3. The result is accumulative until the early exit point.

```
for (int i = 0; posNumsSoFar && i < ns.length; i ++) {
  posNumsSoFar = posNumsSoFar && ns[i] > 0; }
```

4. The result is **not** accumulative until the early exit point.

```
for (int i = 0; posNumsSoFar && i < ns.length; i ++) {
  posNumsSoFar = ns[i] > 0; }
```

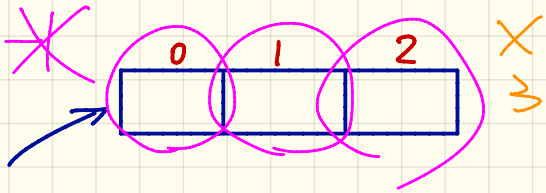

SCE
F
P
P
T
Conjunction
&&
||
disjunction

Short-Circuit and Array Indexing

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many integers?");
3  int howMany = input.nextInt();
4  int[] ns = new int[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter an integer");
7    ns[i] = input.nextInt(); }
8  System.out.println("Enter an index:");
9  int i = input.nextInt();
10 if(ns[i] % 2 == 0) {
11   System.out.println("Element at index " + i + " is even."); }
12 else { /* Error ∵ ns[i] is odd */ }
```

Test Case 1: -1

Test Case 2: 3



Short-Circuit and Array Indexing

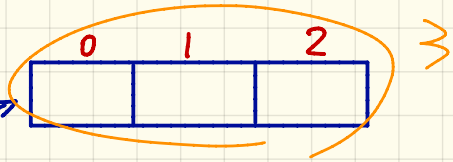
```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many integers?");
3  int howMany = input.nextInt();
4  int[] ns = new int[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter an integer");
7    ns[i] = input.nextInt(); }
8  System.out.println("Enter an index:");
9  int i = input.nextInt();
10 if(0 <= i && i < ns.length && ns[i] % 2 == 0) {
11   println(ns[i] + " at index " + i + " is even."); }
12 else { /* Error: invalid index or odd ns[i] */ }
```

Test Case 1: -1

Test Case 2: 3

Use of &&

valid conditions



0 <= -1 guard

F

0 <= 3 T

3 < 3 F

Short-Circuit and Array Indexing

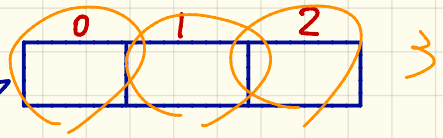
```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many integers?");
3  int howMany = input.nextInt();
4  int[] ns = new int[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter an integer");
7    ns[i] = input.nextInt(); }
8  System.out.println("Enter an index:");
9  int i = input.nextInt();
10 if (i < 0 || i >= ns.length || ns[i] % 2 == 1) {
11 /* Error: invalid index or odd ns[i] */ }
12 else { println(ns[i] + " at index " + i + " is even."); }
```

Test Case 1: -1

Test Case 2: 3

Use of ||

Invalid condition



-1 < 0 T

3 < 0 F

3 >= 3 T ✓

Short-Circuit and Array Indexing

```
1  Scanner input = new Scanner(System.in);
2  System.out.println("How many integers?");
3  int howMany = input.nextInt();
4  int[] ns = new int[howMany];
5  for(int i = 0; i < howMany; i ++) {
6    System.out.println("Enter an integer");
7    ns[i] = input.nextInt(); }
8  System.out.println("Enter an index:");
9  int i = input.nextInt();
10 if( 0 <= i && i < ns.length && ns[i] % 2 == 0) {
11   println(ns[i] + " at index " + i + " is even."); }
12 else { /* Error: invalid index or odd ns[i] */ }
```

Test Case 1: -1

Test Case 2: 3

Exercise

AIOBE

0 1 2

too late

0 <= i && ns[i] % 2 == 0 && i < ns.length

Nested Loops : Flow Chart

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

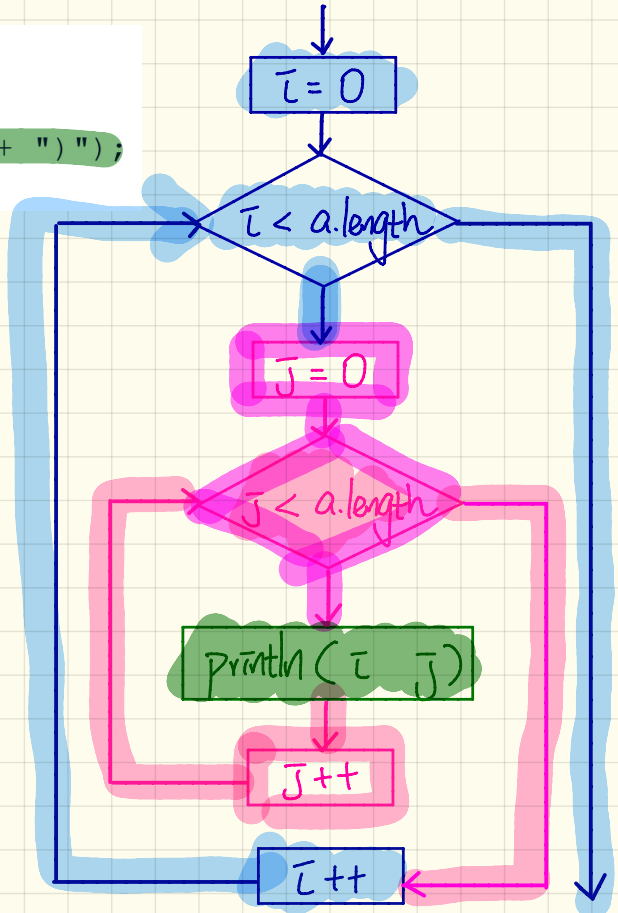
for (. .) {

  for (. .) {

    . —

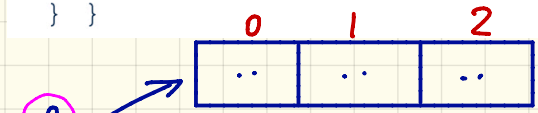
  }

}



Nested Loops : Tracing

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



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

Single iteration of outer loop

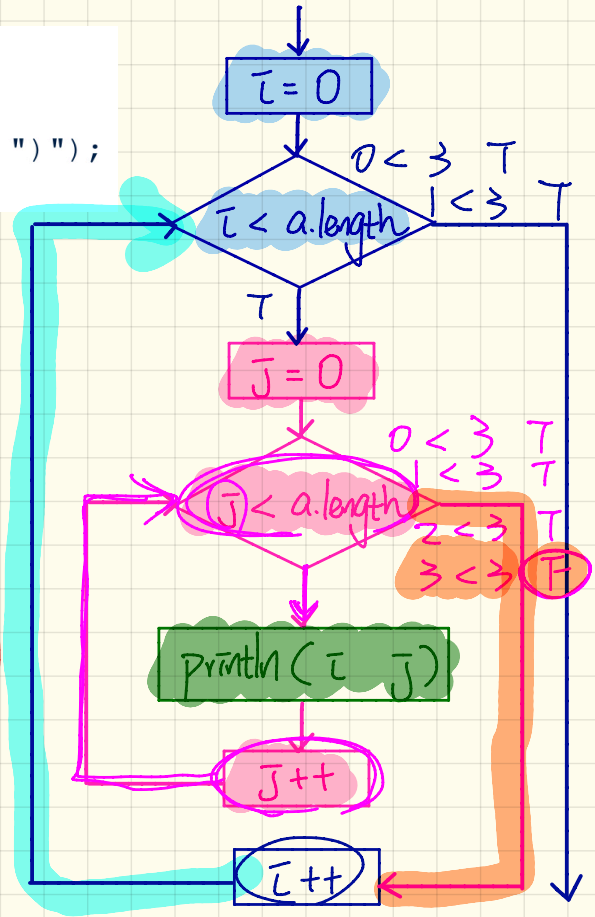
3 iterations of inner loop of outer loop

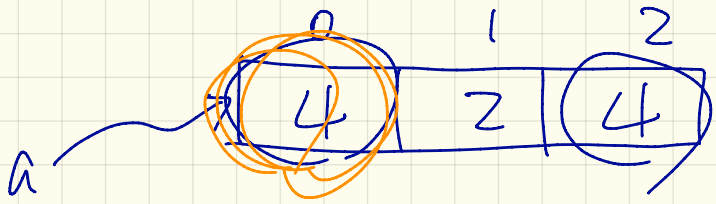
Output
(0, 0)
(0, 1)
(0, 2)
(1, 0)
(1, 1)
(1, 2)
(2, 0)
(2, 1)
(2, 2)

1st iteration of outer loop

2nd

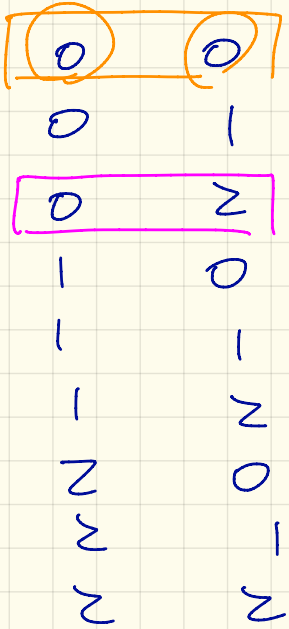
3rd





duplicates? True

a[0] == a[2]

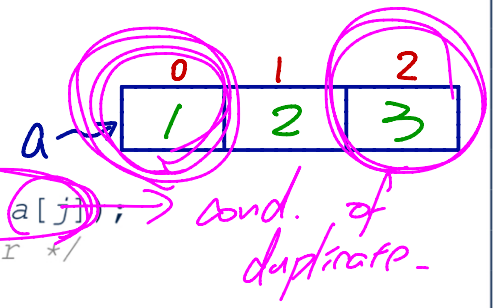


a[0] == a[0] → true

a[0] == a[2] → duplicates

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);
```



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