

Monday Oct. 22

Lecture 12

100 marks? 18.98%

A/A+? 33.47%

E/F 33.58%

Feedback

- Lab Test 1 marks

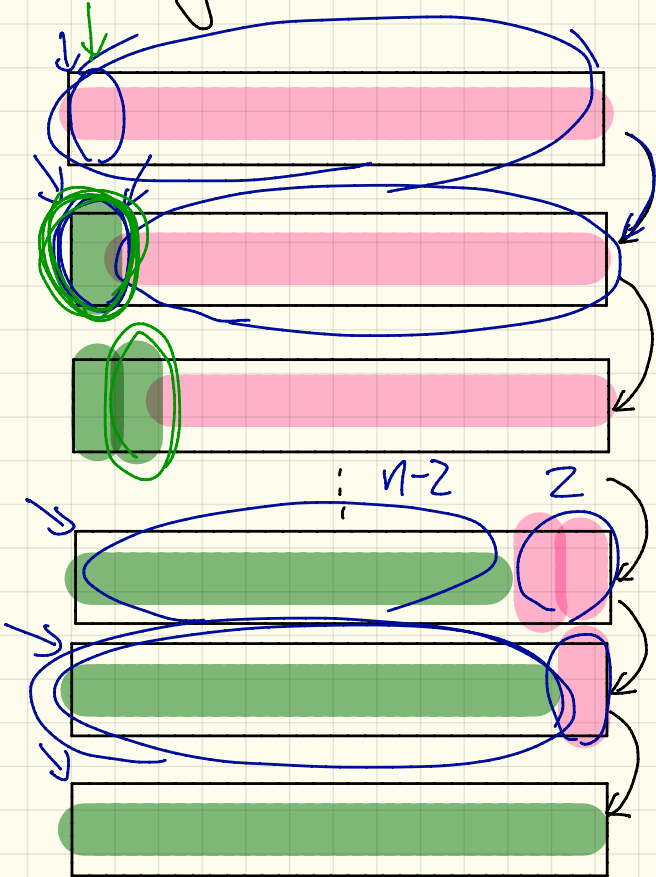
(programming)

- Lab Test 2 postponed:

Monday Nov. 5

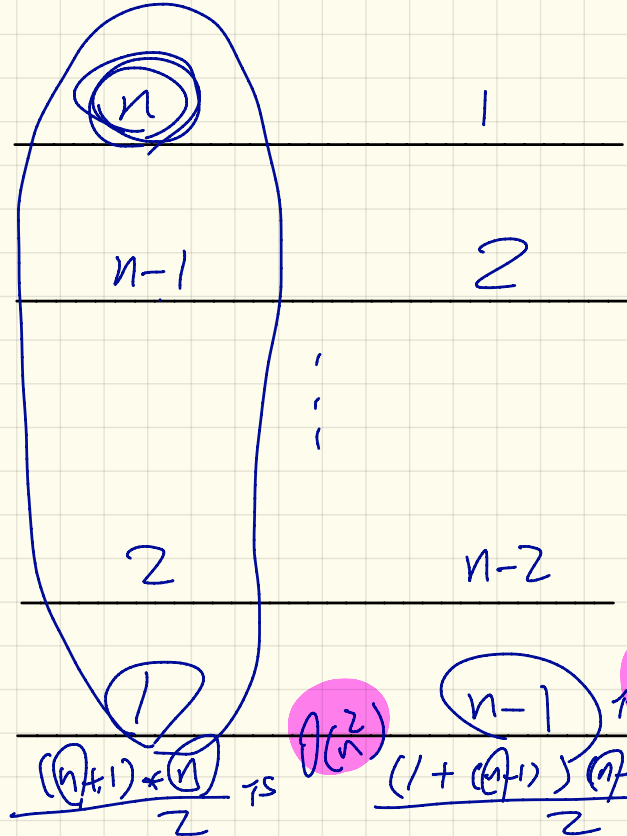
Sorting

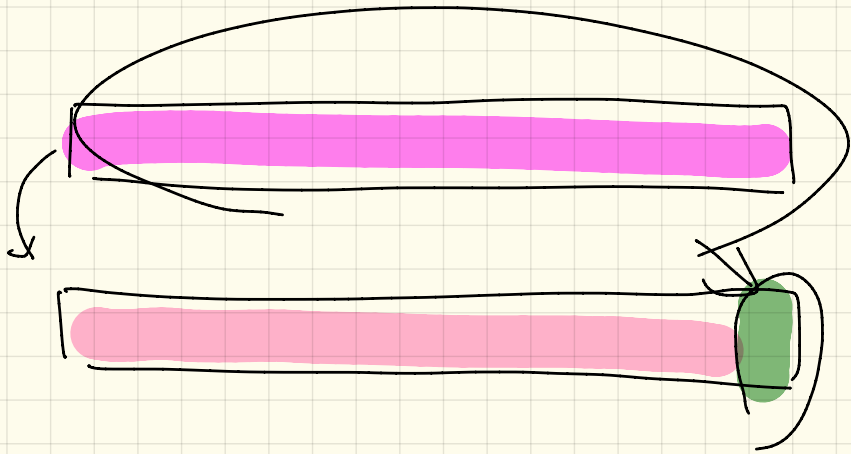
n



Selection Sort

Insertion Sort





IS $O(n^2)$

$$n = 1000 \rightarrow \begin{cases} 1M & PO. \\ (1M)^2 & PO \end{cases}$$

Merge Sort

$O(n \cdot \log n)$

$$\begin{aligned} n = 1000 &\rightarrow 1000 \cdot \frac{\log_2 1000}{\sqrt{10^3}} \\ n = 1M &\rightarrow 1M \cdot \frac{\log_2 1000}{\sqrt{10^6}} \end{aligned}$$

Selection Sort: Code



```

1 selectionSort(int[] a, int n)
2   → for (int i = 0; i <= (n - 2); i++)
3     → int minIndex = i;
4     → for (int j = i; j <= (n - 1); j++)
5       if (a[j] < a[minIndex]) { minIndex = j; }
6     → int temp = a[i];
7       a[i] = a[minIndex];
8       a[minIndex] = temp;

```

Annotations:

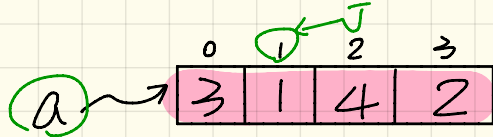
- Line 1: `selectionSort` highlighted in yellow.
- Line 2: `i = 0` circled in green. `(n - 2)` circled in green.
- Line 3: `minIndex = i` circled in green.
- Line 4: `j = i` circled in red. `(n - 1)` circled in blue.
- Line 5: `a[j] < a[minIndex]` circled in blue.
- Line 6: `int temp = a[i];` circled in red.
- Line 7: `a[i] = a[minIndex];` circled in red.
- Line 8: `a[minIndex] = temp;` circled in red.

Handwritten notes:

- SS(a, a.length)
- temp = a[0]
- a[0] = a[1]
- a[1] = 3
- green area (pointing to line 1)
- 4 a (pointing to element 3)

i	inner loop j from ? to ?	minIndex at lb	after lb ~ lg, a becomes?
0	0 1 2 3	1 a[1] 1	
1	1 2 3	3 a[3] 2	

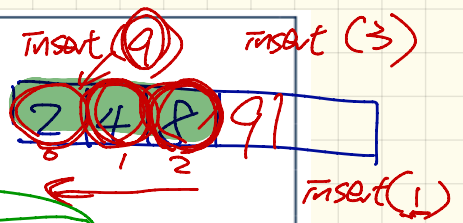
Insertion Sort : Code



```

1 insertionSort(int[] a, int n)
2   for (int i = 1; i < n; i++)
3     int current = a[i];
4     int j = i - 1;
5     while (j > 0 && a[j - 1] > current)
6       a[j] = a[j - 1];
7       j--;
8     a[j] = current;

```



Under what condition does while loop exit?

i	current	j at L8	a at L8	a after L8
1	(1)	0		
2				

while ($j > 0$ ~~&&~~ $a[j-1] > \text{current}$)

↳ exit: ! ($j > 0$ && $a[j-1] > \text{current}$)

|||

$j \leq 0$

||

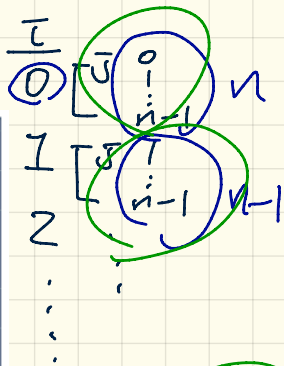
$a[j-1] \leq \text{current}$

Asymptotic Upper Bounds

f

```
1 selectionSort(int[] a, int n)
2 → for (int i = 0; i <= (n - 2); i++)
3     int minIndex = i;
4     for (int j = i; j <= (n - 1); j++)
5         if (a[j] < a[minIndex]) { minIndex = j; }
6     int temp = a[i];
7     a[i] = a[minIndex];
8     a[minIndex] = temp;
```

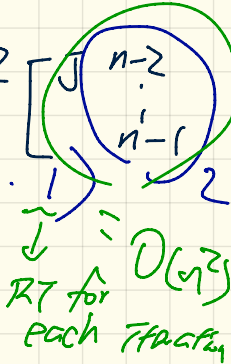
$O(n)$



$$O\left(\underbrace{(n + (n-1) + \dots + 2)}_{\text{\# of iterations}} \cdot 1\right)$$

g

```
1 insertionSort(int[] a, int n)
2 [ for (int i = 1; i < n; i++)
3     int current = a[i];
4     int j = i;
5     while (j > 0 && a[j - 1] > current)
6         a[j] = a[j - 1];
7         j--;
8     a[j] = current;
```



$O(n^2)$
RT for each iteration

Call by Value (1)

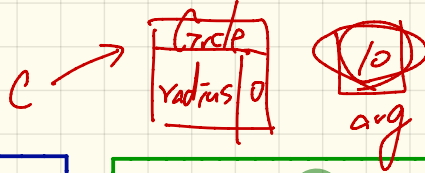
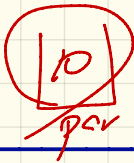
```
class Supplier {  
  void m1(T par) {  
    /* manipulate par */  
  }  
}
```

Annotations: `Supplier` is circled in green. `m1` and `T` are circled in red. `par` is circled in red. `par` and `arg` are circled in orange. A red arrow points from `par` to `arg`.

```
class Client {  
  Supplier s = new Supplier();  
  T arg = ...;  
  s.m1(arg);  
}
```

Annotations: `Client` is circled in green. `Supplier s` is circled in green. `new Supplier()` is underlined in green. `T` is circled in yellow. `arg` is circled in red. `s.m1` and `arg` are circled in orange.

T being Primitive



```
class Circle {  
  int radius; par = arg;  
  void setRadius(int par) {  
    this.radius = par;  
  }  
}
```

Annotations: `Circle` is circled in yellow. `radius` is circled in red. `par = arg` is written in red. `void setRadius` is circled in red. `int par` is circled in yellow. `this.radius = par` is circled in yellow. `par` is circled in red. A red arrow points to `void setRadius`.

```
class CircleUser {  
  Circle c = new Circle(c);  
  int arg = 10;  
  c.setRadius(arg);  
}
```

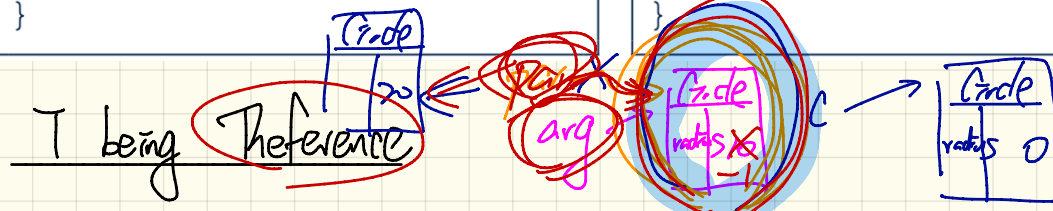
Annotations: `CircleUser` is circled in green. `Circle c` is circled in green. `new Circle(c)` is circled in red. `int arg = 10` is circled in red. `c.setRadius` and `arg` are circled in orange. A red arrow points to `new Circle(c)`.

Call by Value (2)

Is pink obj going to be changed?
① par = new Circle(20) No
② par.setRadius(-1); YES

```
class Supplier {  
    void m1(T par) {  
        /* manipulate par */  
    }  
}
```

```
class Client {  
    Supplier s = new Supplier();  
    T arg = ...;  
    s.m1(arg)  
}
```



```
class Circle {  
    int radius;  
    Circle(int radius) { this.radius = radius; }  
    void setRadius(Circle par) {  
        par = new Circle(20);  
        this.radius = par.radius;  
    }  
}
```

```
class CircleUser {  
    Circle C = new Circle(C);  
    Circle arg = new Circle(10);  
    C.setRadius(arg);  
}
```

Call by Value: Primitive Type



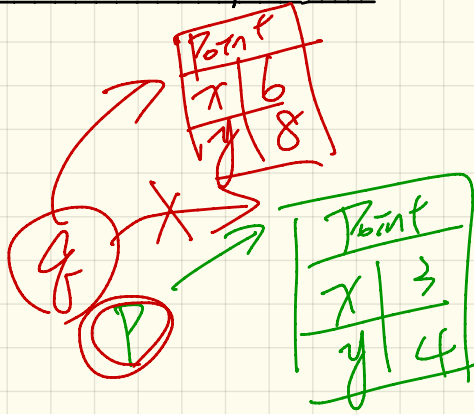
```
class Point {  
    int x;  
    int y;  
    Point(int x, int y) {  
        this.x = x;  
        this.y = y;  
    }  
    void moveVertically(int y) {  
        this.y += y;  
    }  
    void moveHorizontally(int x) {  
        this.x += x;  
    }  
}
```



```
public class Util {  
    void reassignInt(int i) {  
        i = i + 1; }  
    void reassignRef(Point q) {  
        Point np = new Point(6, 8);  
        q = np; }  
    void changeViaRef(Point q) {  
        q.moveHorizontally(3);  
        q.moveVertically(4); } }  
}
```

```
1 @Test  
2 public void testCallByVal() {  
3     Util u = new Util();  
4     int i = 10;  
5     assertTrue(i == 10);  
6     u.reassignInt(i);  
7     assertTrue(i == 10);  
8 }
```

Call by Value: Reference Type (1)



```
class Point {
    int x;
    int y;
    Point(int x, int y) {
        this.x = x;
        this.y = y;
    }
    void moveVertically(int y) {
        this.y += y;
    }
    void moveHorizontally(int x) {
        this.x += x;
    }
}
```

q = p

```
public class Util {
    void reassignInt(int j) {
        j = j + 1;
    }
    void reassignRef(Point q) {
        Point np = new Point(6, 8);
        q = np;
    }
    void changeViaRef(Point q) {
        q.moveHorizontally(3);
        q.moveVertically(4);
    }
}
```

```
1 @Test
2 public void testCallByRef_1() {
3     Util u = new Util();
4     Point p = new Point(3, 4);
5     Point refOfPBefore = p;
6     u.reassignRef(p);
7     assertTrue(p == refOfPBefore);
8     assertTrue(p.x == 3 && p.y == 4);
9 }
```