Monday Oct. 22

Lecture 12

100 marks? 18.9%

A/A+ ? 33.47%

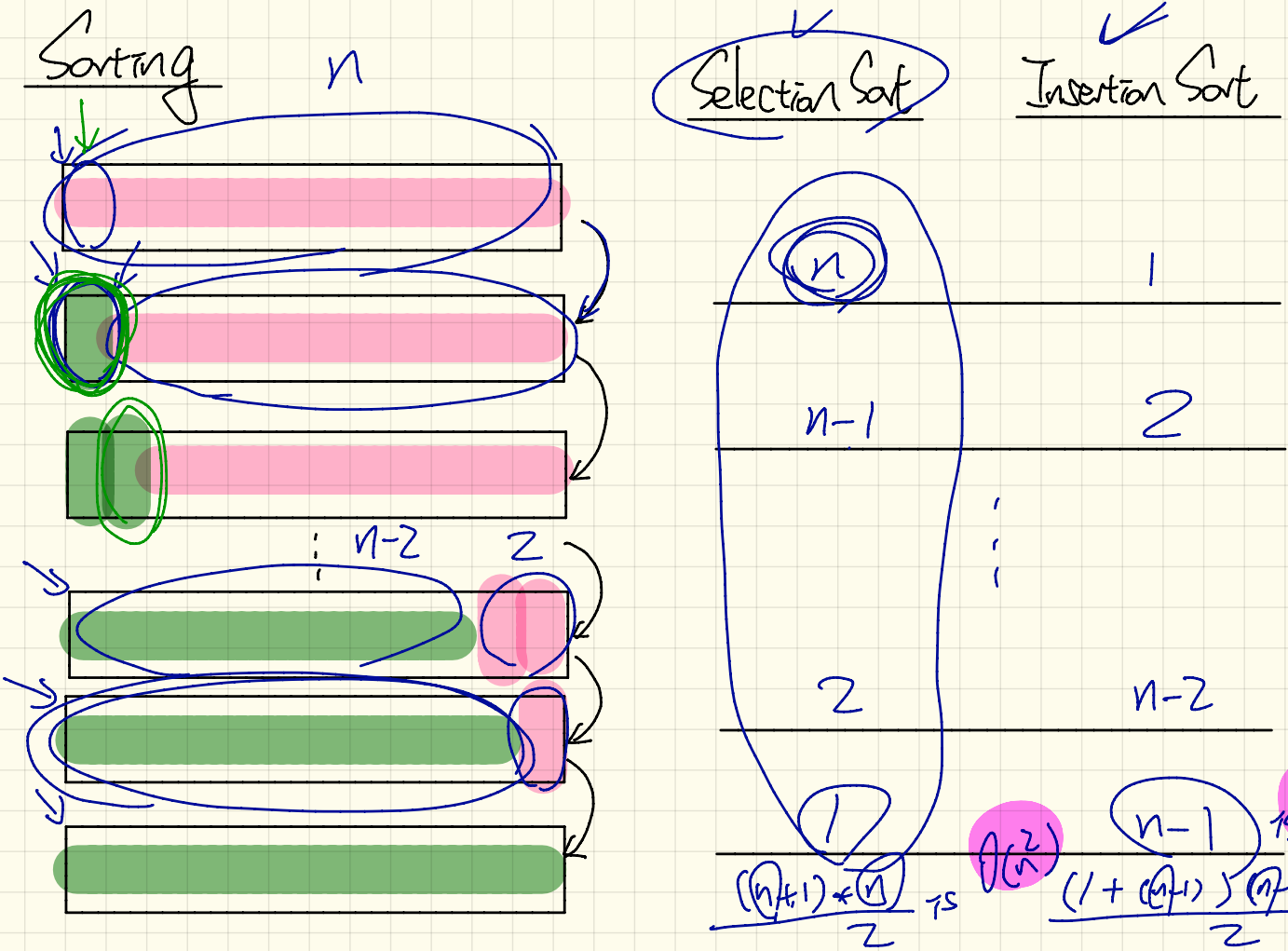
E/F 33.58%

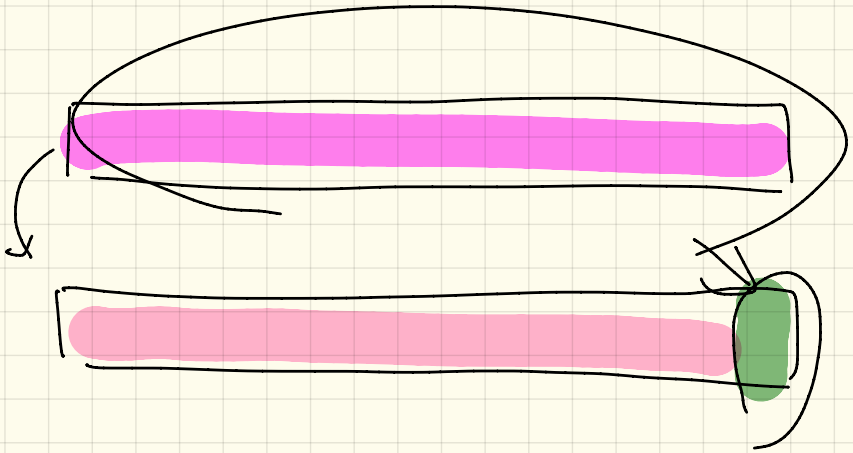
feedback

- Lab Test 1 marks (programming)

- Lab Test 2 postponed:

Monday Nov. 5


Sorting
n
n-2
2
Selection Sort
Insertion Sort
n
1
n-1
2
2
n-2
1
n-1
O(n^2)
((n+1)*n)/2 ~ n^2
(1+(n-1))(n-1)/2 ~ n^2



SS IS $O(n^2)$

$n = 1000 \rightarrow$ (1M PO.

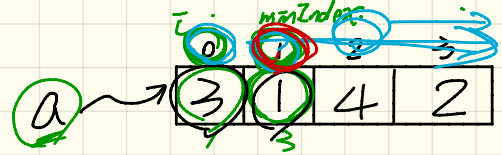
$n = 1M \rightarrow (1M)^2$ PO

Merge Sort $O(n \cdot \log n)$

$n = 1000 \rightarrow 1000 \cdot \log_2 1000 \approx 10k$

$n = 1M \rightarrow 1M \cdot \log_2 1000 \approx 20k$

Selection Sort : Code

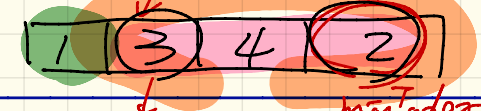
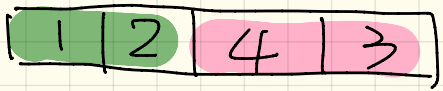


green area

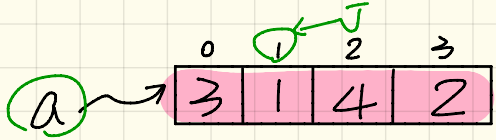
```
1  selectionSort(int[] a, int n)
2  for (int i = 0; i <= (n - 2); i ++)
3    int minIndex = i;
4    for (int j = i; j <= (n - 1); j ++)
5      if (a[j] < a[minIndex]) { minIndex = j; }
6    int temp = a[i];
7    a[i] = a[minIndex];
8    a[minIndex] = temp;
```

ss(a, a.length)

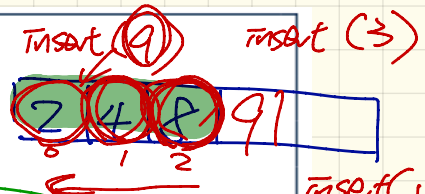
temp = a[0]
a[0] = a[1]
a[1] = 3

i	inner loop j from ? to ?	minIndex at L6	after L6~L8, a becomes?
0	0 1 2 3	1 a[1] 1	
1	1 2 3	3 a[3] 2	

Insertion Sort : Code



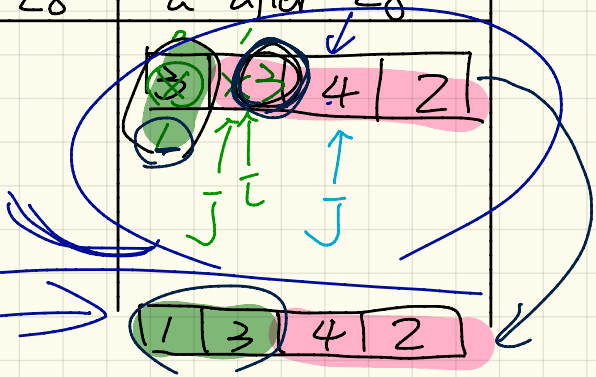

```
1  insertionSort(int[] a, int n)
2    for (int i = 1; i < n; i ++)
3      int current = a[i];
4      int j = i;
5      while (j > 0 && a[j - 1] > current)
6        a[j] = a[j - 1];
7        j --;
8      a[j] = current;
```



stay

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] > current)

↳ exit: ! (j > 0 && a[j-1] > current)

|||

j ≤ 0 || a[j-1] ≤ current

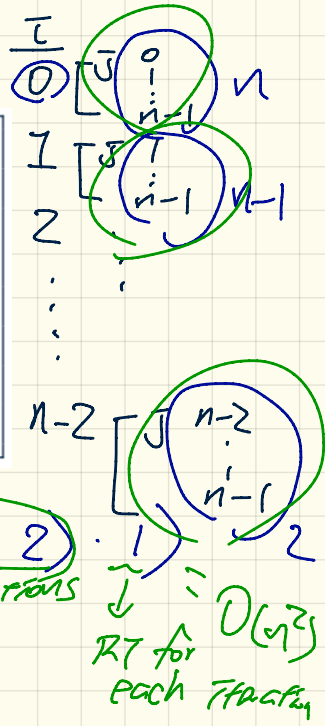
Asymptotic Upper Bounds

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(1)

O(1)



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

RT for each iteration

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;
```

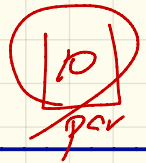
Call by Value (1)

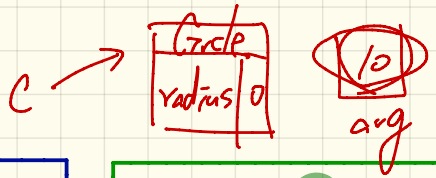
```
class Supplier {
  void m1(T par) {
    /* manipulate par */
  }
}
```

par = arg;

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

T being Primitive






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

par = arg;

10

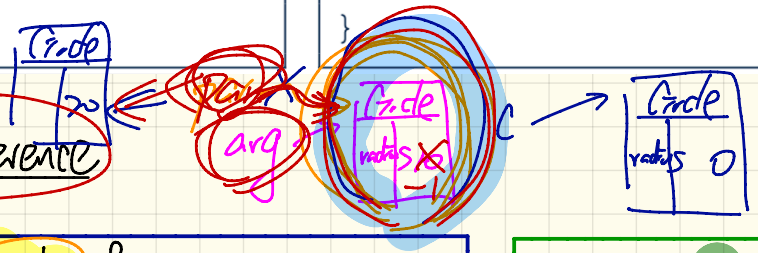
```
class CircleUser {
  Circle c = new Circle();
  int arg = 10;
  c.setRadius(arg);
}
```

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)
}
```



T being Reference

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

par = arg;

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

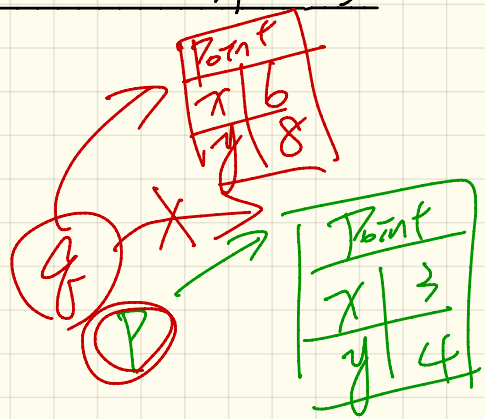
Call by Value : Primitive Type

```
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 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 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;
  }
}
```

```
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  }
```