Lecture 23

Wednesday November 27

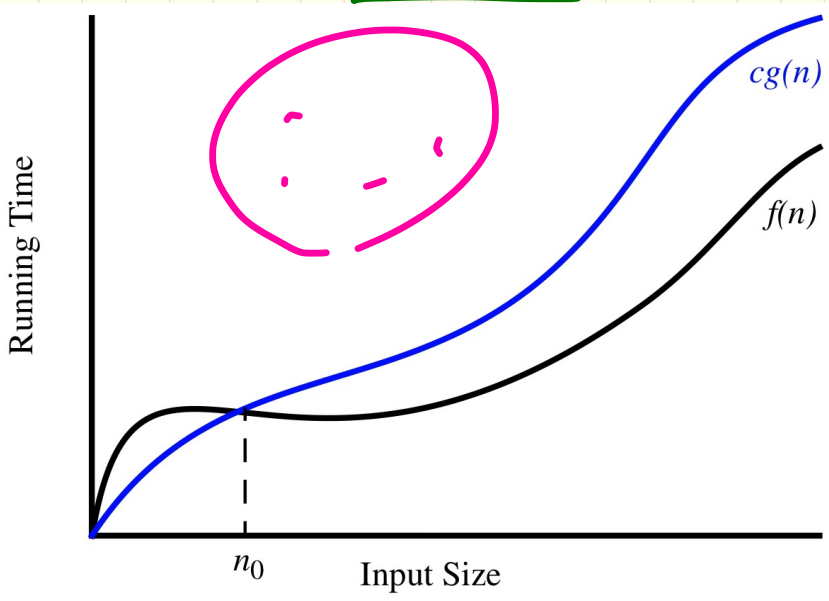
Asymptotic Upper Bound: Big-O

$f(n) \in O(g(n))$ if there are:

- A real *constant* $c > 0$
- An integer *constant* $n_0 \geq 1$

such that:

$f(n) \leq c \cdot g(n)$ for $n \geq n_0$



Example:

$f(n) = 8n + 5$

$g(n) = n$

Prove:

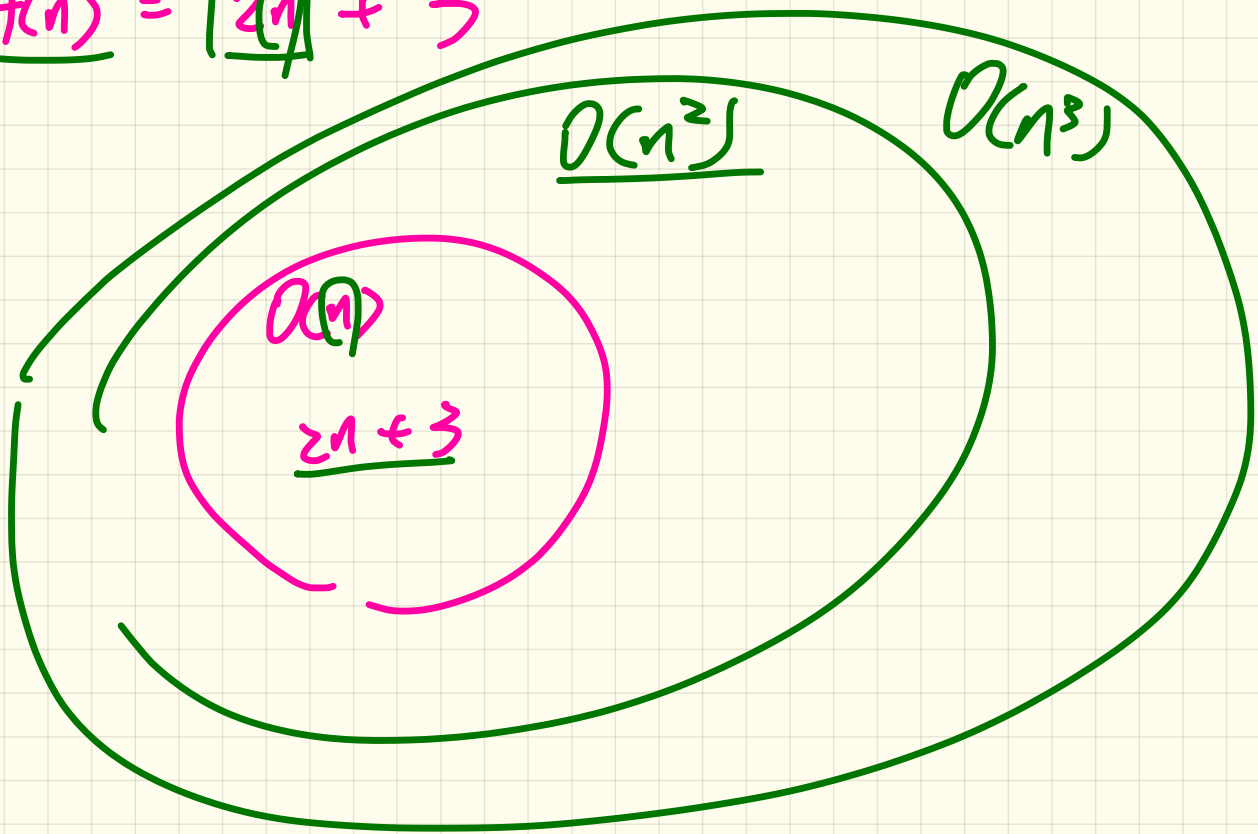
$f(n)$ is $O(g(n))$

Choose:

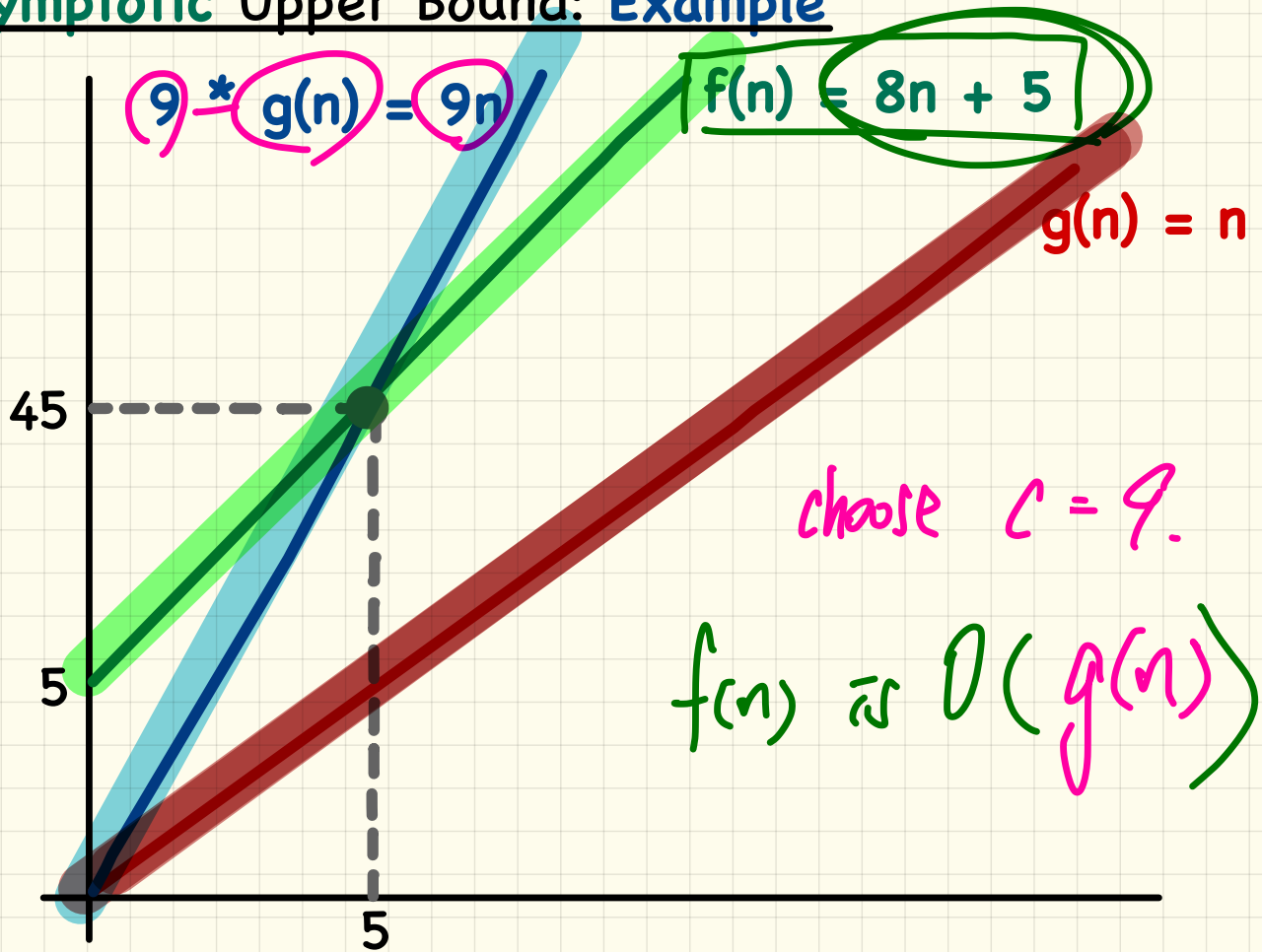
$c = 9$

What about n0?

$f(n) = 2n + 3$



Asymptotic Upper Bound: Example



choose $c = 9$.

$f(n)$ is $O(g(n))$

Proving f(n) is O(g(n))

If $f(n)$ is a polynomial of degree d, i.e.,

$$f(n) = a_0 \cdot n^0 + a_1 \cdot n^1 + \cdots + a_d \cdot n^d$$

and $a_0, a_1, \ldots, a_d$ are integers (i.e., negative, zero, or positive),

then $f(n)$ is $O(n^d)$.

We prove by choosing

$c = |a_0| + |a_1| + \cdots + |a_d|$

$n_0 = 1$

Upper-bound effect starts when $n_0 = 1$

$[f(n) \leq n^d]$

$a_0 \cdot 1^0 + a_1 \cdot 1^1 + \cdots + a_d \cdot 1^d$

$= a_0 + a_1 + \cdots + a_d$

$f(1) \leq |a_0| + |a_1| + \cdots + |a_d|$

$c \cdot 1^d$

Upper-bound effect holds?

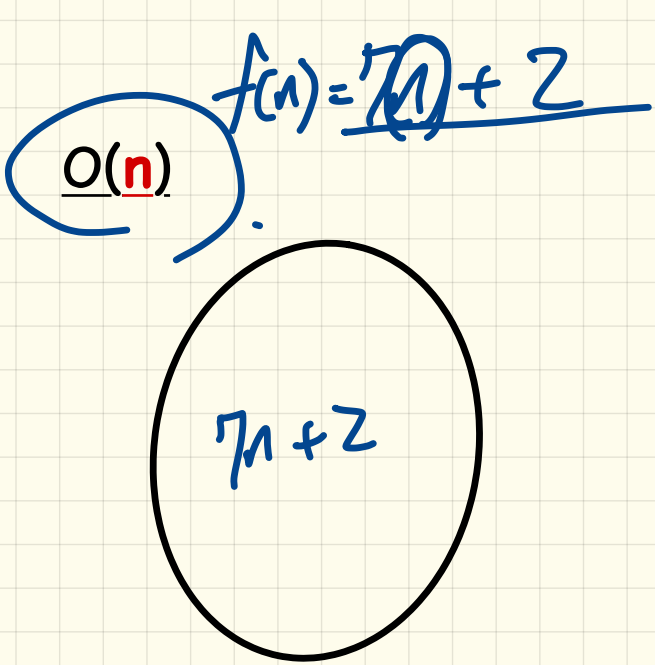
$[f(n) \leq n^d]$

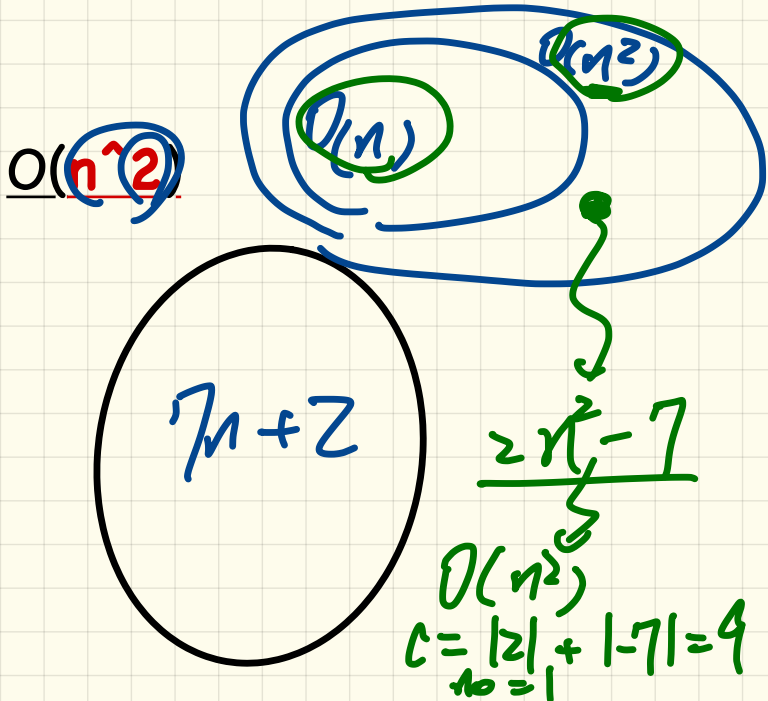
$2 + (-3) + (-7) \leq |2| + |-3| + |-7|$

O(g(n)): A Set of Functions

Each member f(n) in O(g(n)) is such that:

Higest Power of f(n) <= Highest Power of g(n)





Asymptotic Upper Bounds: Example (1)

$5n^2 + 3n \cdot \log n + 2n + 5$ is $O(n^2)$

$\log_2 1 = 0$

$2^0 = 1$

Prove.

choose $C = ? \; 5+3+2+5 = 15$

$n_0 = ? \; 1$ s.t.

$15 \cdot n^2 \geq 5n^2 + 3n \cdot \log n + 2n + 5$

$15 \geq \frac{5 + 0 + 2 + 5}{1^2}$

Asymptotic Upper Bounds: Example (2)

$20n^3 + 10n \cdot \log n + 5$ is $O(n^3)$

Prove. choose $c = 35$

$n_0 = 1$ s.t.

$c \cdot n^3 \geq 20n^3 + 10n \cdot \log n + 5$

$35 \cdot 1 \geq 20 \cdot 1 + 0 + 5$

Asymptotic Upper Bounds: Example (3)

$3 \cdot \log n + 2$ is $O(\log n)$

Prove:

$5 \cdot \log 2 \geq 3 \cdot \log 2 + 2$

$5 \geq 3 + 2$

Choose $c = 5$

$n_0 = 2$ s.t.

$$c \cdot \log n \geq 3 \cdot \log n + 2$$

$5 \cdot \log 1 \geq 3 \cdot \log 1 + 2$

$0 \geq 2$

Asymptotic Upper Bounds: Example (4)

2^{n+2} is $O(2^n)$

$2^{n+2} = 2^2 \cdot 2^n$

Prove

Choose $C = ?? \cdot 2^2 = 4$

$n_0 = 1$? s.t.

$C \cdot 2^n \geq 2^{n+2}$

Asymptotic Upper Bounds: Example (5)

$2n + 100 \cdot \log n$ is $O(n)$

Determining the Asymptotic Upper Bound (1)

```
1  maxOf (int x, int y) {
2    int max = x;
3    if (y > x) {
4      max = y;
5    }
6    return max;
7  }
```

O(1) ← O(1) ← O(1) ← O(1)

O(1).

RT does not depend on how large x and y are.

Determining the Asymptotic Upper Bound (2)

```
1  findMax (int[] a, int n) {
2    currentMax = a[0];
3    for (int i = 1; i < n; ) {
4      if (a[i] > currentMax)
5        currentMax = a[i]; }
6    i ++ }
7    return currentMax; }
```

→ 1 → O(1)
→ 2 → O(1)
⋮ ⋮
→ n → O(1)

$O(\frac{1 + 1 + \cdots + 1}{n})$

↳ O(n).

Will a double-nested loop give RT = $O(n^2)$? No.

```
for (int i=0; i<10; i++){
    for (int j=5; j<10; j++){
        print(...)
    }
}
```

$O(1)$ $O(1)$ $O(n)$

Determining the Asymptotic Upper Bound (3)

```
1  containsDuplicate (int[] a, int n) {
2    for (int i = 0; i < n; ) {
3      for (int j = 0; j < n; ) {
4        if (i != j && a[i] == a[j]) {
5          return true; }
6        j ++; }
7      i ++; }
8    return false; }
```

$O(1)$

$O(1)$

$j = 0 \ldots n-1$

$[i, j]$

$j - i + 1$

observe the pattern of changes on loop counter

$i = 0$ [$j = 0 \ldots n-1$] $O(n)$

$i = 1$ [$j = 0 \ldots n-1$]

$i = 2$ [$j = 0 \ldots n-1$]

$\vdots$

$i = n-1$ [$j = 0 \ldots n-1$]

n

$O(n \cdot n \cdot 1) = O(n^2)$

# of iterations for i · # of iterations for j · $O(1)$ for L4 L5

Determining the Asymptotic Upper Bound (4)

```
1  sumMaxAndCrossProducts (int[] a, int n) {
2    int max = a[0];
3    for(int i = 1; i < n; i ++) {
4      if (a[i] > max) { max = a[i]; }
5    }
6    int sum = max;
7    for (int j = 0; j < n; j ++) {
8      for (int k = 0; k < n; k ++) {
9        sum += a[j] * a[k]; } }
10   return sum; }
```

$O(1) + O(n) + O(1) + O(n^2)$

$= O(n^2)$

Determining the Asymptotic Upper Bound (5)

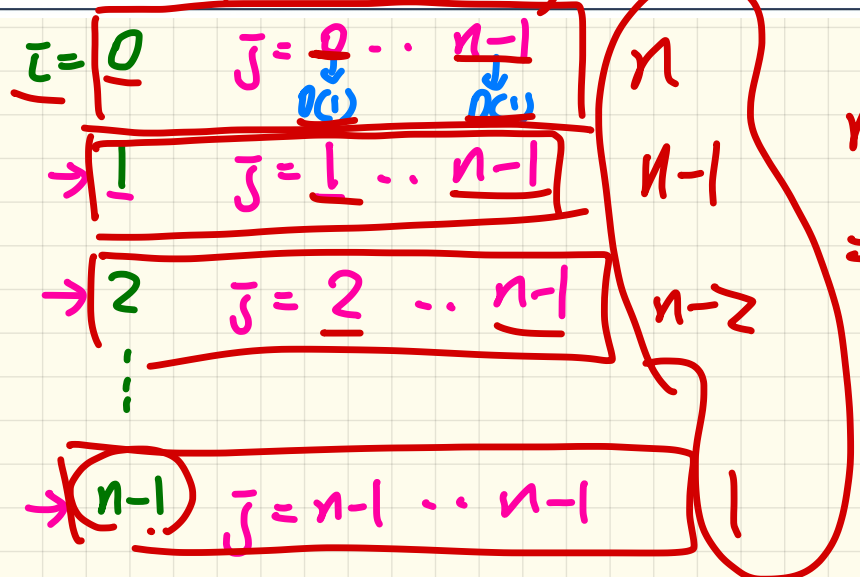
```
1  triangularSum (int[] a, int n) {
2    int sum = 0;
3    for (int i = 0; i < n; i ++) {
4      for (int j = i; j < n; j ++) {
5        sum += a[j]; } }
6    return sum; }
```

O(1)

O(1)

O(1)

O(n^2)



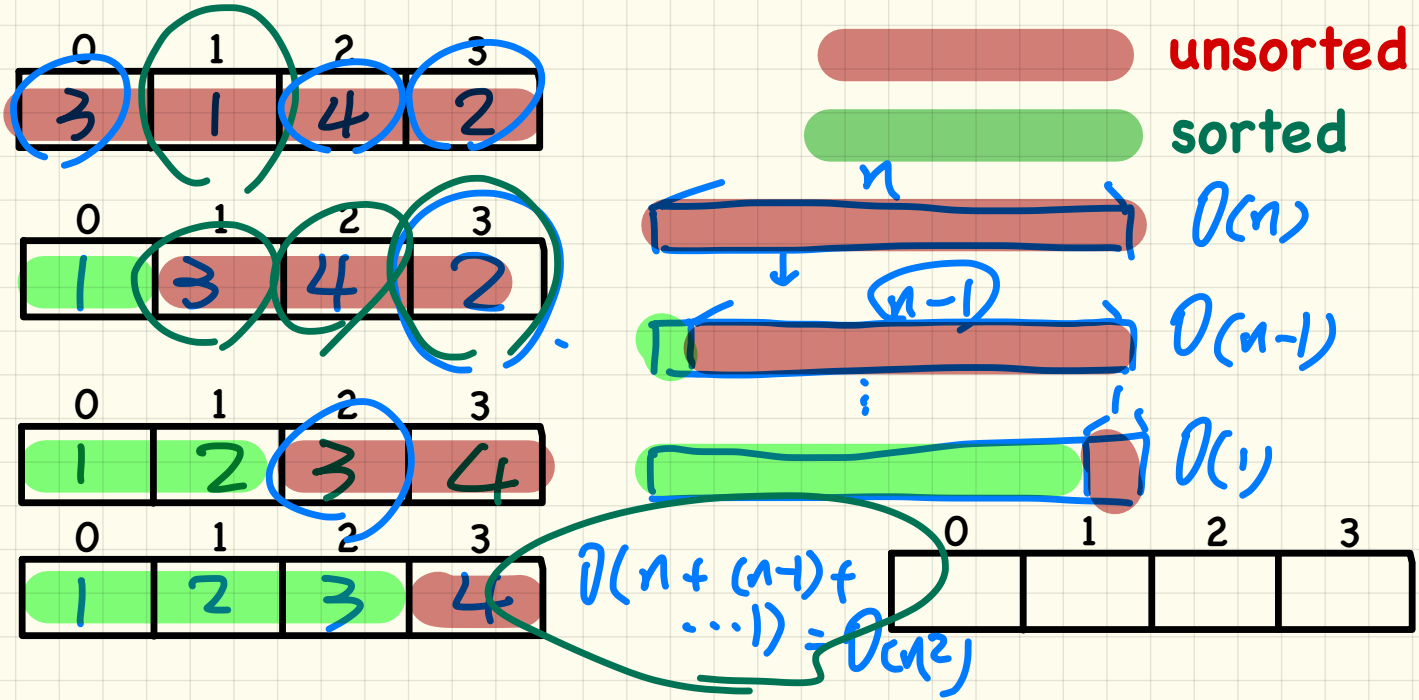
$n + (n-1) + \dots + 1$

$= \frac{(n+1) * n}{2}$

$= O(n^2)$

Selection Sort

Keep selecting minimum from the unsorted portion and appending it to the end of sorted portion.



Insertion Sort

$O(1+2+\cdots(n-2)) = O(n^2)$

Keep getting 1st element from the **unsorted** portion and **inserting** it to the **sorted** portion.