Lecture 22

Monday November 25

- REVIEW SESSIONS FOR EXAM

  SURVEY ON MOODLE

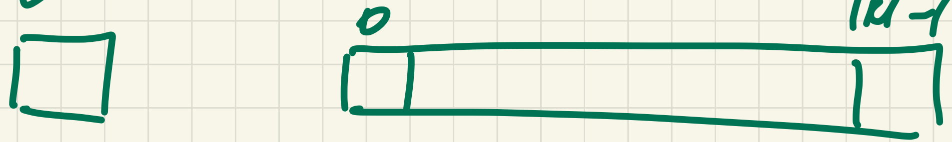
- MAKE-UP LECTURES:

  NOV. 15
  NOV. 22 } RECORDINGS

Time Efficiency of Algo.

e.g. sort an array of integers

1. size ✓



2. structure ✓

4	3	2	1

1	3	4	2

Example Experiment

Computational Problem:

- **Input**: A character *c* and an integer *n*
- **Output**: A string consisting of *n* repetitions of character *c*
  e.g., Given input `'*'` and `15`, output `***************`.

Algorithm 1 using *String* Concatenations:

```
public static String repeat1(char c, int n) {
  String answer = "";
  for (int i = 0; i < n; i ++) { answer += c; }
  return answer; }
```

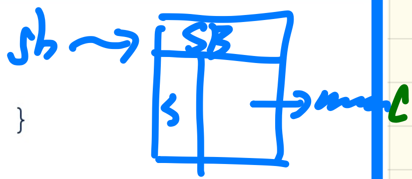
answer = answer + c;

answer
temp → c

a

Algorithm 2 using *StringBuilder* `append`'s:

```
public static String repeat2(char c, int n) {
  StringBuilder sb = new StringBuilder();
  for (int i = 0; i < n; i ++) { sb.append(c); }
  return sb.toString(); }
```



Counting the Number of Primitive Operations

```
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; }
```

10 == a.length

i = i+1

n-1 times

i	i<n	
1	1<10	T
2	2<10	T
⋮		
9	9<10	T
n 10	10<10	F

Q. # of times i < n in Line 3 is executed?

n times.

Q. # of times loop body (Lines 4 to 6) is executed?

$2 \cdot (n-1)$

Po: $\boxed{n-2}$

$\boxed{n-1 \text{ times}}$

n-1 times i<n T

nth time i<n F

findMax(int[] a, int n){

⋮

}

$6n^2 - 100$

$\hookrightarrow$ $\frac{7n - 2}{}$

n	#Bo RT
10	6s
100	69s

⋮

RT

$(7n - 2)\cancel{t}$ $(10n + 3)\cancel{t}$

$\underbrace{7n - 2}_{\text{\# of Pos}}$ $\underbrace{10n + 3}_{\text{\# of Pos}}$

relative running time.

RT.

$$2^{n} + 4n^{3} + 2n^{2} + 3n^{1}$$

$$\boxed{\cancel{4}n^{3} + \cancel{2n^{2}} + \cancel{3n^{1}} + \cancel{1 \cdot 2^{0}}}$$

n^{3}

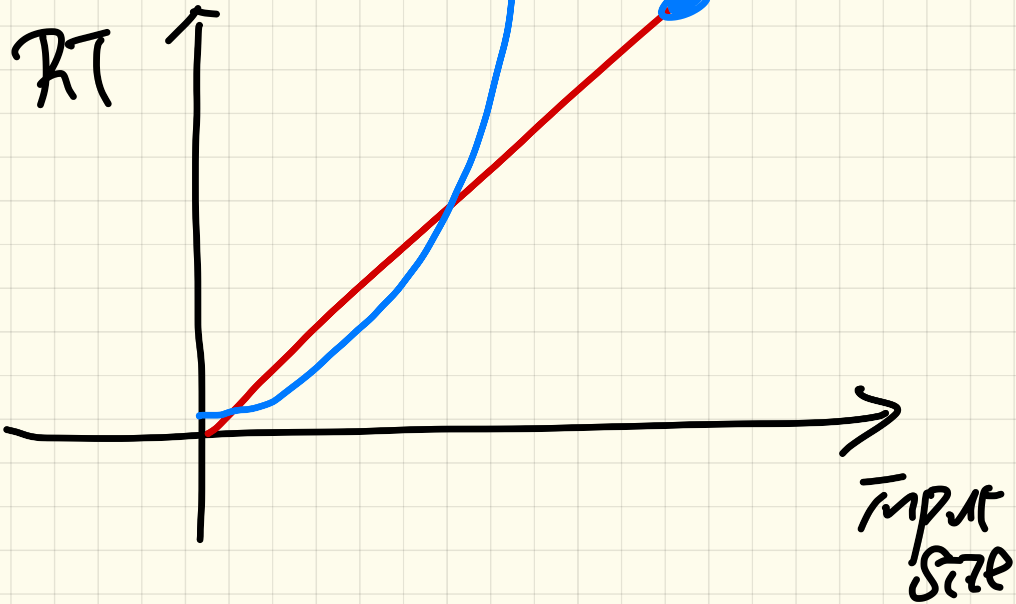
asympotic upper bound.

multiplication constants.

higest power (dominating over all lower terms)

$RT_1(n) = n^1$

$RT_2(n) = n^2$



$$RT_1(n) = \cancel{3}n^2 + \cancel{7n} + \cancel{18}$$

$$RT_2(n) = \cancel{100}n^2 + \cancel{3n} - \cancel{100}$$

Asymptotic Upper Bound: Big-O

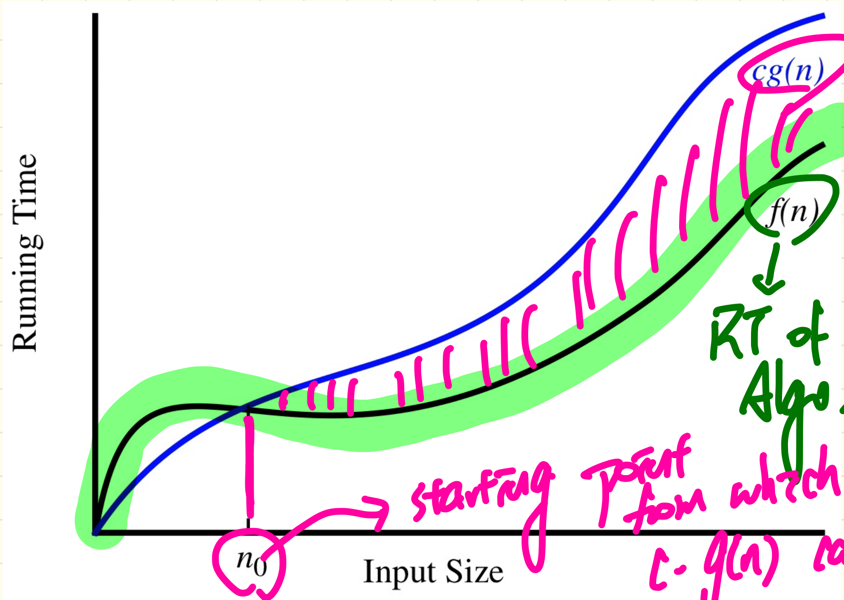
member of

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

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

such that:

$$f(n) \leq c \cdot g(n) \quad \text{for } n \geq n_0$$



Example:

$f(n) = 8n + 5$

$g(n) = n$

Prove:

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

Choose:

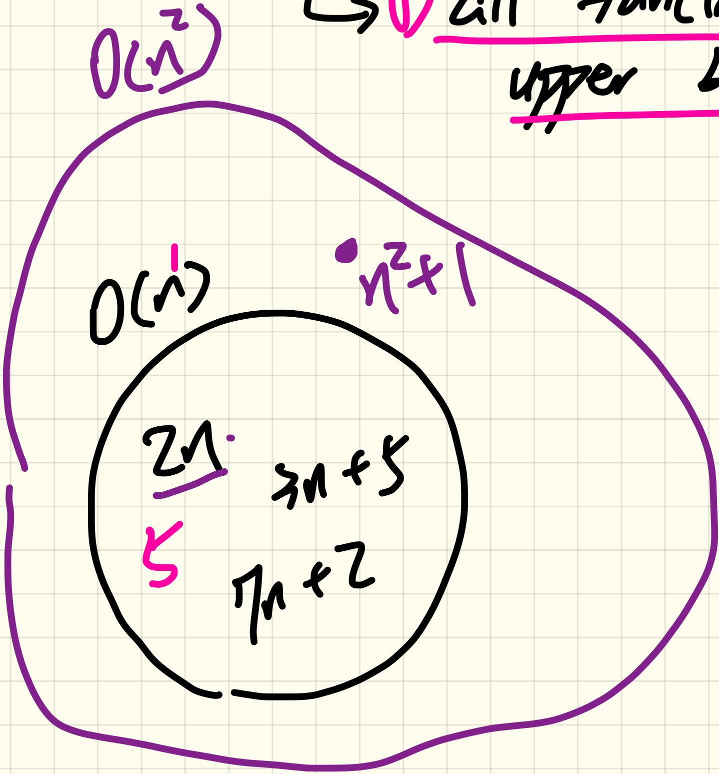
$c = 9$

What about n0?

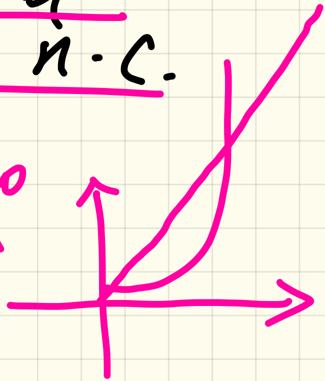
$O(n)$

↳ ① all functions that can be upper bounded by $n \cdot C$.

② all functions whose highest power ≤ 1



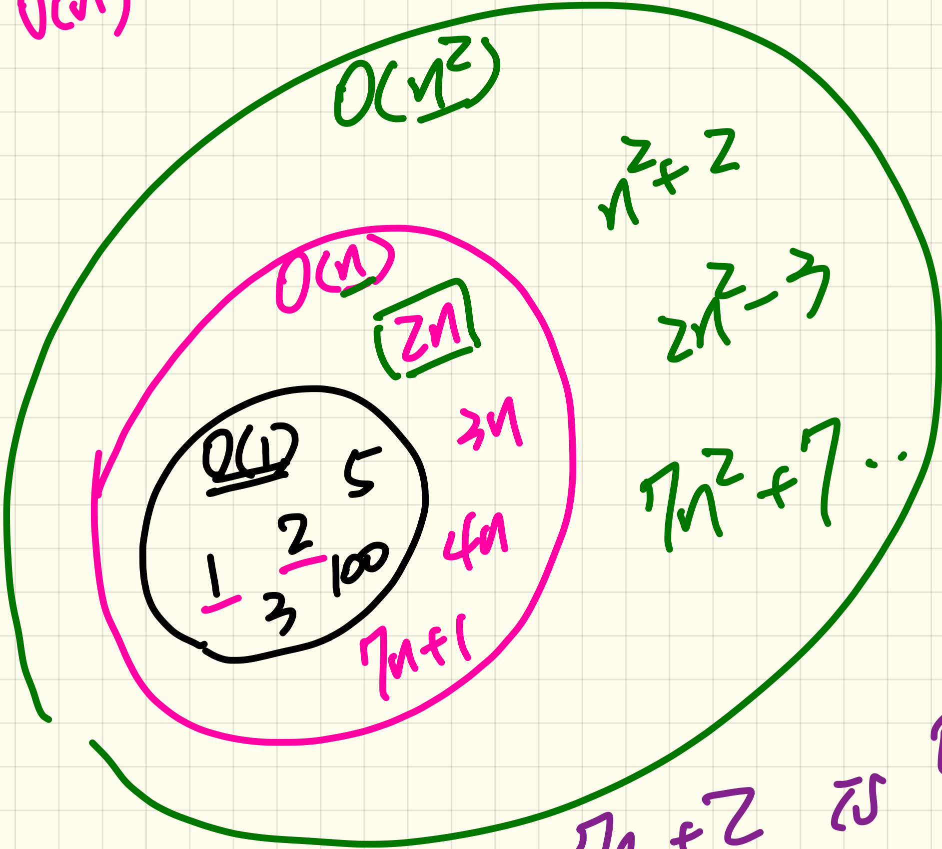
$5 \cdot n^0$



why $n^2+1 \notin O(n)$.

↳ to prove, choose C s.t. $C \cdot n \geq n^2+1$

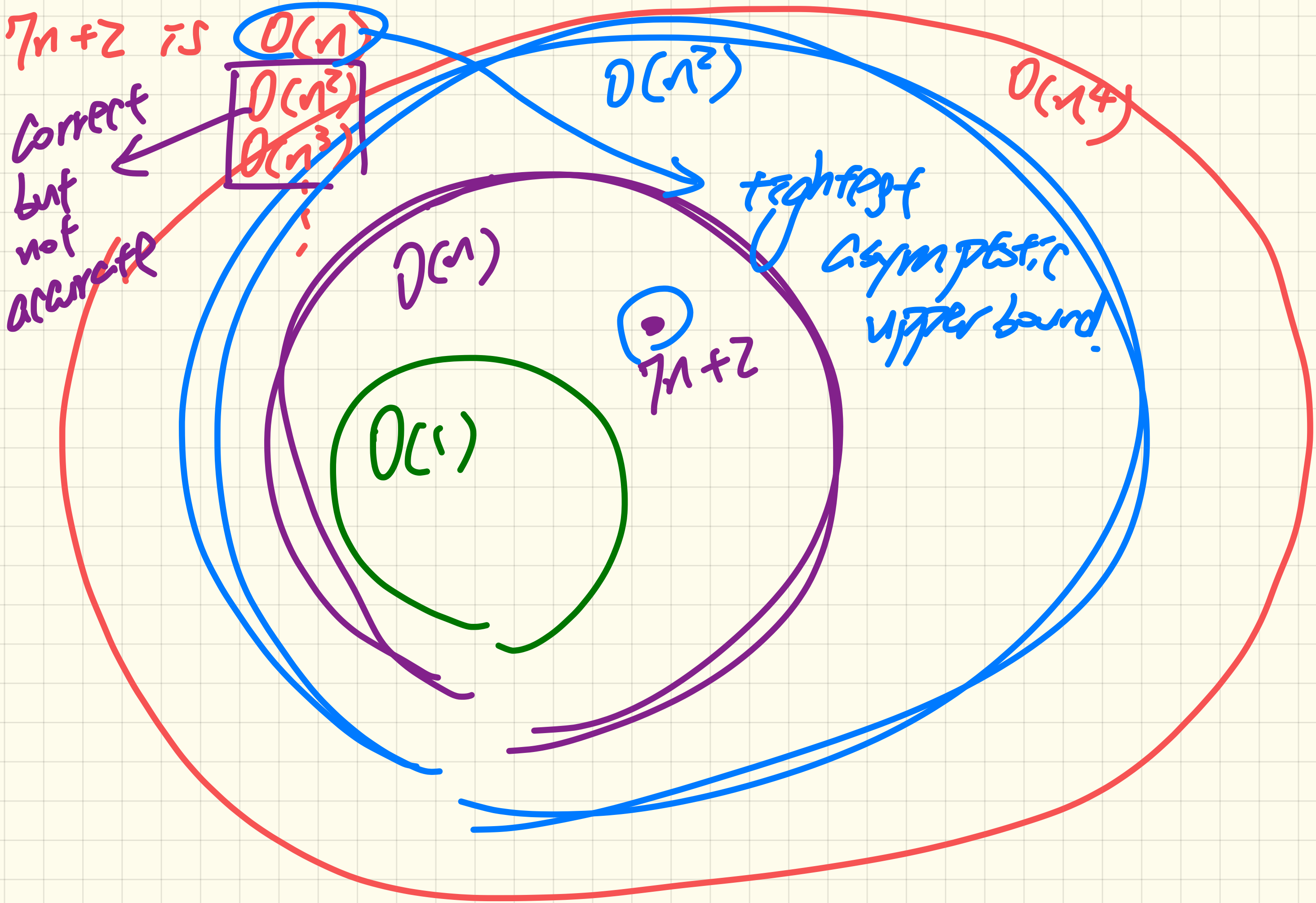
$O(1) \subset O(n)$



$7n+2$ is $O(n)$ ✓

$7n+2$ is $O(n^2)$

$\underline{7n+2}$ is $O(1)$ ✗


7n+2 is O(n)
O(n²)
O(n³)
correct but not accurate
O(n²)
O(n⁴)
tightest asymptotic upper bound
O(n)
7n+2
O(1)

$$RT(n) = \boxed{8n} + \boxed{5}$$

$$\text{is } O(n)$$

↳ choose $c = 13$ $\quad 8+5 = 13$

s.t. $n_0 = 1$

↳ starting from $n \geq 1$

$13 \geq 13$

$13 \cdot n \geq 8n + 5$

$\boxed{5}n^2 + \boxed{3}n \cdot \log n + \boxed{2}n + \boxed{5}$

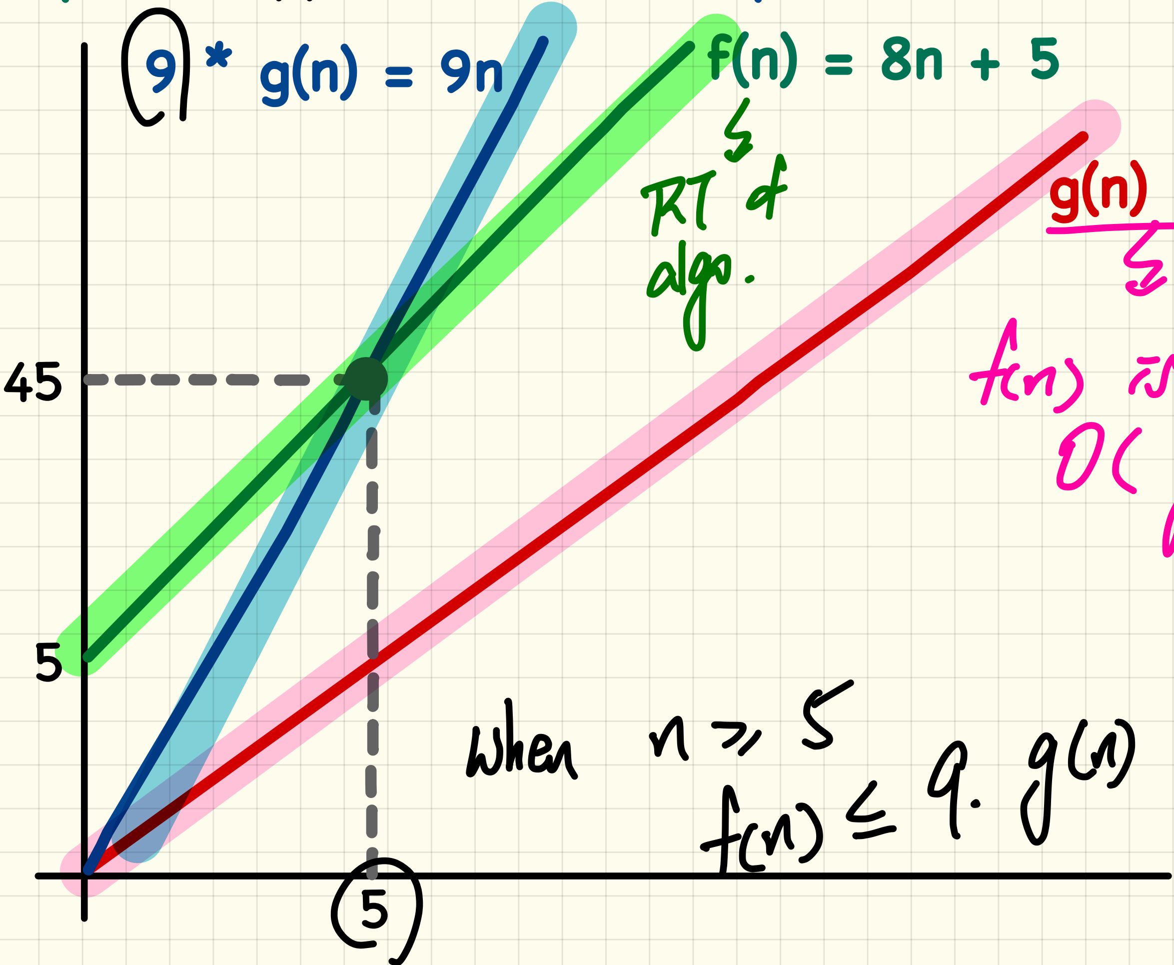
$\hookrightarrow O(n^2)$

Prove: choose $c = 15$

check: starting from $n = n_0$:

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

Asymptotic Upper Bound: Example



RT of algo.

f(n) is O(g(n)) ?

when $n \geq 5$

$f(n) \leq 9 \cdot g(n)$