Lecture 6 - Wednesday, January 25

Announcements

- Assignment 1 released:
  + Tracing Recursion:
    - Paper: Call Stack vs. Tree
    - Debugger in Eclipse
  + Help: Scheduled Office Hours & TAs
  + Look ahead: WrittenTest1

$* \; 1^0 = 1^1 = \cdots = 1^d = 1$

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 $O(n^d)$. → $g(n)$

We prove by choosing

$$\boxed{c} = \boxed{|a_0| + |a_1| + \cdots + |a_d|}$$
$$n_0 = 1$$

(1) $f(1) \le c \cdot \cancel{n^d}\; 1$

(2) $f(n) \le c \cdot n^d$ $(n > 1)$

Upper-bound effect: $n_0 = 1$? $[f(1) \le (|a_0| + |a_1| + \cdots + |a_d|) \cdot 1^d]$

$a_0 \le |a_0|$, $a_1 \le |a_1|$, $\vdots$, $a_d \le |a_d|$ $(**)$

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

$\overset{*}{=} (a_0 + a_1 + \cdots + a_d) \cdot 1^d \overset{**}{\le} (|a_0| + |a_1| + \cdots + |a_d|) \cdot 1^d$

Upper-bound effect holds? $[f(n) \le (|a_0| + |a_1| + \cdots + |a_d|) \cdot n^d]$ $\boxed{n > 1}$

$(***)$ $n^0 \le n^d$, $n^1 \le n^d$, $\vdots$, $n^d \le n^d$

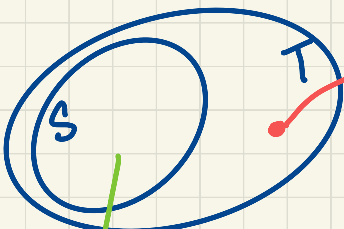
$f(n) = a_0 \cdot n^0 + a_1 \cdot n^1 + \cdots + a_d \cdot n^d$

$\overset{***}{\le} (a_0 + a_1 + \cdots + a_d) \cdot n^d \overset{**}{\le} (|a_0| + |a_1| + \cdots + |a_d|) \cdot n^d$

$O(n^0) \subset O(n^1) \subset O(n^2) \cdots$

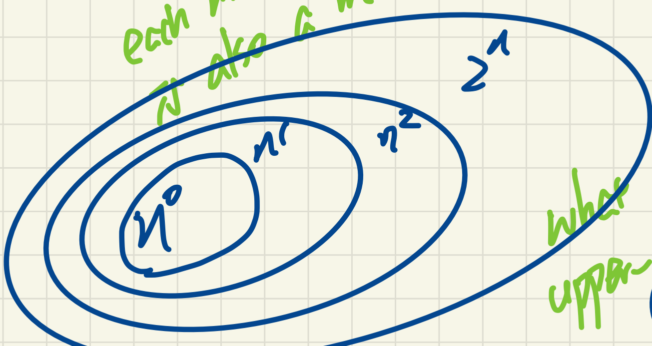
Proper Subset

$S \subset T$

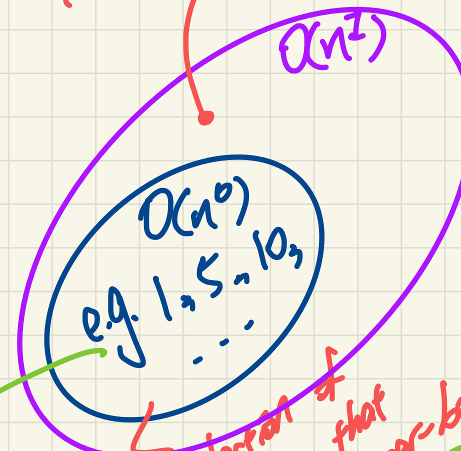


there's at least one member of T that's not a member of S

each member in S is also a member in T



there's at least one function that can be u.b. by n^1 but not u.b. by n^0



what can be upper-bounded by n^0 can also be upper-bounded by n^1

the collection of functions that can be upper-bounded by n^0

e.g. $f_1(n) = 7n - 2$

$f_2(n) = 4n^2 - 3n + 6$

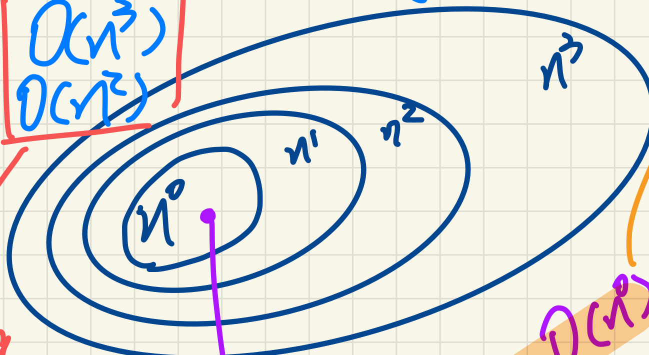
} two possible solutions to the same problem

What if inaccurate U.b. is given?

$f_1(n)$ is $\boxed{O(n^3)}$

$f_2(n)$ is $\boxed{O(n^2)}$

misleading for decision making



$f(n) = 1$

system.out.println("Hello World");

$f(n)$ is $O(n^0)$ ✓

$f(n)$ is $O(n^1)$ ✓

$f(n)$ is $O(n^2)$ ✓

⋮

$f(n)$ is $O(2^n)$ ✓

the most accurate upper bound that's correct.

Asymptotic Upper Bounds: Example (2)

$20n^3 + 10n \cdot logn + 5$ is $O($[redacted]$)$

Derive[(1)] and Prove[(2)] the most accurate asymptotic u.b. of the above function.

(1) $O(n^3)$

(2) Prove by choosing: $c = |20| + |10| + |5| = 35$

$n_0 = 1$

$c \cdot g(1) = 35 \cdot 1^3 = 35$

Verify: $\boxed{f(1) \le c \cdot g(1)}$ $f(1) = 20 \cdot 1^3 + 10 \cdot 1 \cdot \log 1 + 5 = 25$ ✓

Asymptotic Upper Bounds: Example (3)

$\boxed{3 \cdot \log n + 2 \text{ is } O(\text{[redacted]})}$ $\equiv 3 \cdot \log n + 2 \cdot n^0$

(1) $O(\underline{\log n})$

(2) Prove by choosing: $c = |3| + |2| = \underline{\underline{5}}$

$n_0 = 1$

Verify $\boxed{f(1) \overset{\text{failed}}{\leq} c \cdot g(1)}$ failed

$f(1) = 3 \cdot \log 1 + 2 = 2$

$c \cdot g(1) = 5 \cdot \log 1 = 0$

$c = |3| + |2| = 5$

$n_0 = 2$

(exercise!)

Asymptotic Upper Bounds: Example (4)

2^{n+2} is $O($[redacted]$)$

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

$$= \cancel{4} \cdot 2^n$$

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

$O(2^n)$.

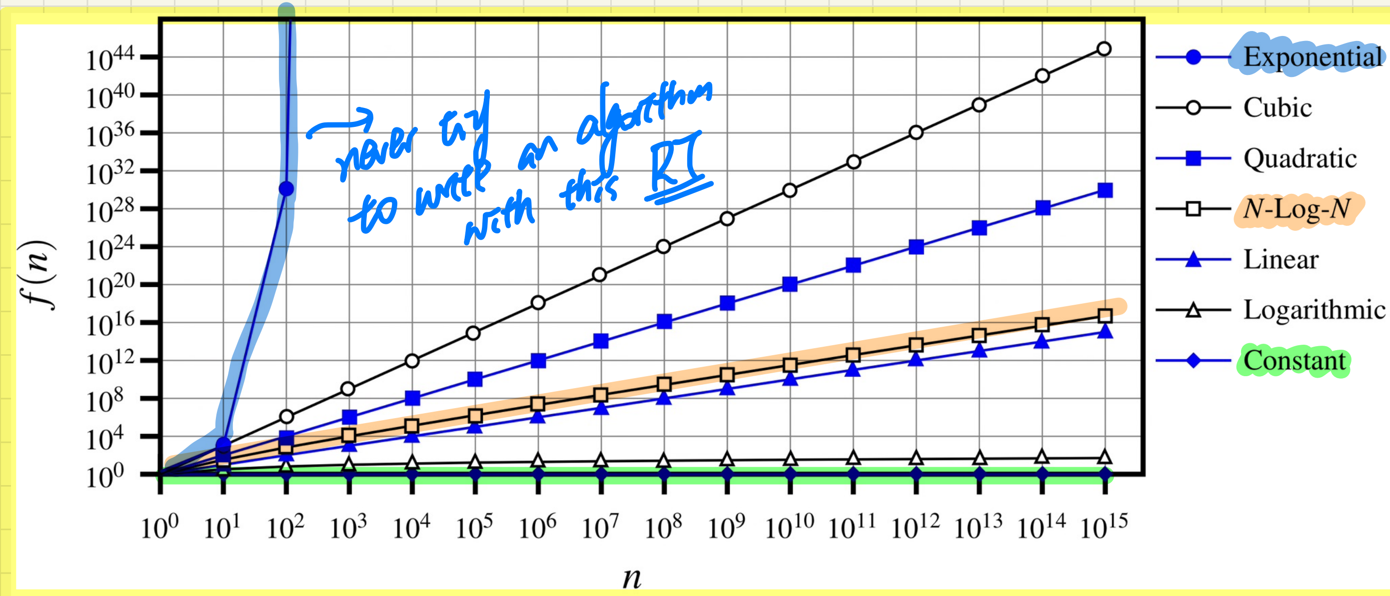
Asymptotic Upper Bounds: Example (5)

$2n + 100 \cdot logn$ is $O($[redacted]$)$

Exercise

$n! = n \cdot (n-1) \cdot \cdots$

Running Time vs. Input Size: Common Rates of Growth



2^n vs. 3^n $(2^?)^n$

$O(2^n)$

<u>**Lecture**</u>

<u>**Asymptotic Analysis of Algorithms**</u>

Asymptotic Upper Bounds
of Implemented Algorithms

Determining the Asymptotic Upper Bound (1)

```
int maxOf (int x, int y) {
  int max = x;   1.
  if (y > x) {   1.
    max = y;   1.
  }
  return max;   1.
}
```

$$O(1+1+1+1) = O(\underline{4}) = \boxed{O(1)}.$$

$4 \cdot n^0$

Determining the Asymptotic Upper Bound (2)

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

body of loop

$$O(\underbrace{1}_{\text{L2}} + \underbrace{n}_{\text{header of loop}} + \underbrace{n}_{\text{\# iterations}} \cdot (\underbrace{1+1+1}_{\text{each iteration}}) + \underbrace{1}_{\text{return}}) = \boxed{O(n)}$$

Determining the Asymptotic Upper Bound (3)

$[a, b] = b - a + 1$

$[0, n-1] = (n-1) - 0 + 1 = n$

```
boolean containsDuplicate (int[] a, int n) {
  for (int i = 0; i < n; ) {
    for (int j = 0; j < n; ) {
      if (i != j && a[i] == a[j]) {
        return true; }
      j ++; }
    i ++; }
  return false; }
```

body of inner loop: 1

i ++ → 1

Pattern of loop Counters

i	j
0	0 1 2 ... (n-1)
1	0 1 2 ... (n-1)
2	⋮
⋮	
n-1	0 1 2 ... (n-1)

outer loop runs for **n times**

inner loop runs for **n times**

(Continued on next)

```
boolean containsDuplicate (int[] a, int n) {
  for (int i = 0; i < n; ) {
    for (int j = 0; j < n; ) {
      if (i != j && a[i] == a[j]) {
        return true; }
      j ++; }
    i ++; }
  return false; }
```

10,000

O(n)

Constants

```
boolean containsDuplicate (int[] a, int n) {
  for (int i = 0; i < n; ) {
    for (int j = 0; j < n; ) {
      if (i != j && a[i] == a[j]) {
        return true; }
      j ++; }
    i ++; }
  return false; }
```

1M

1B

O(1)