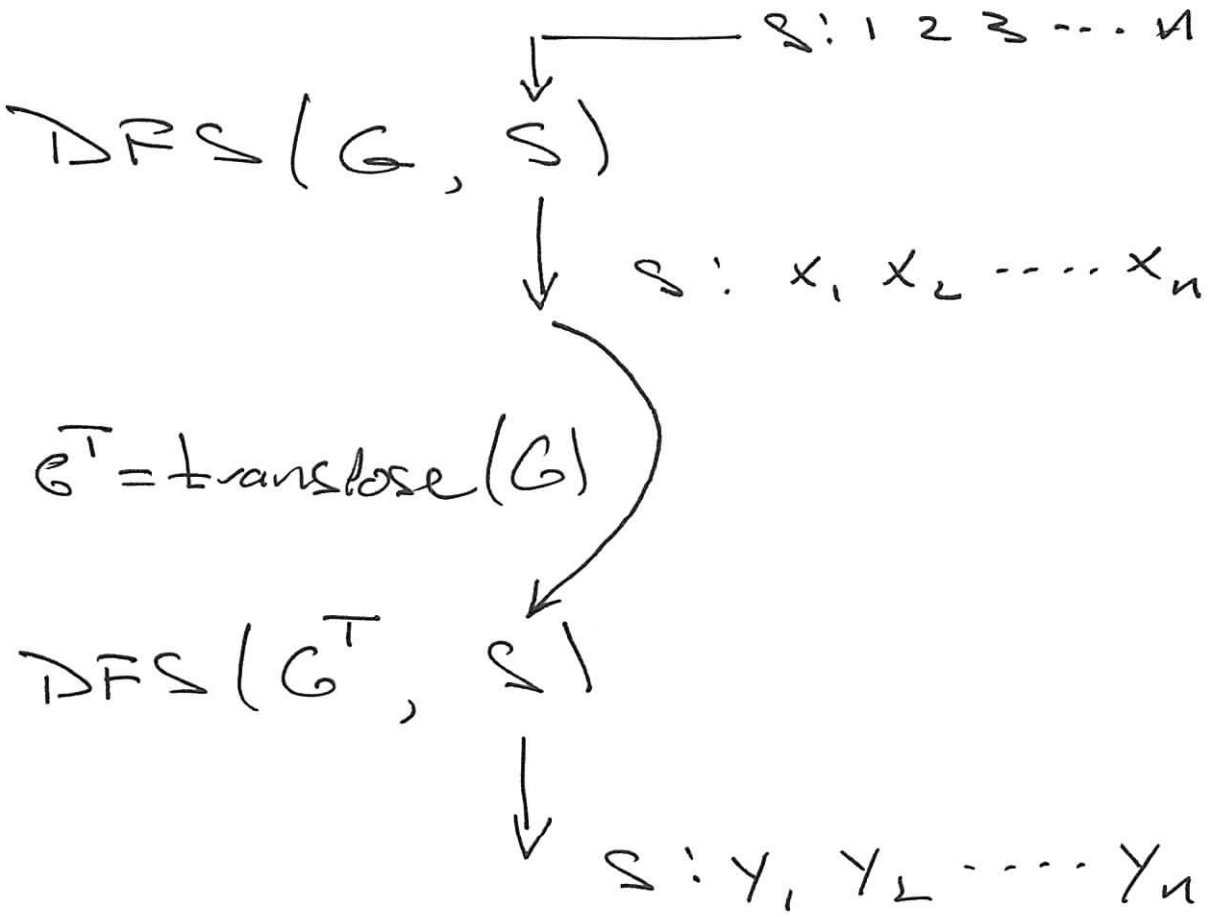
CSE 101 4-22-21

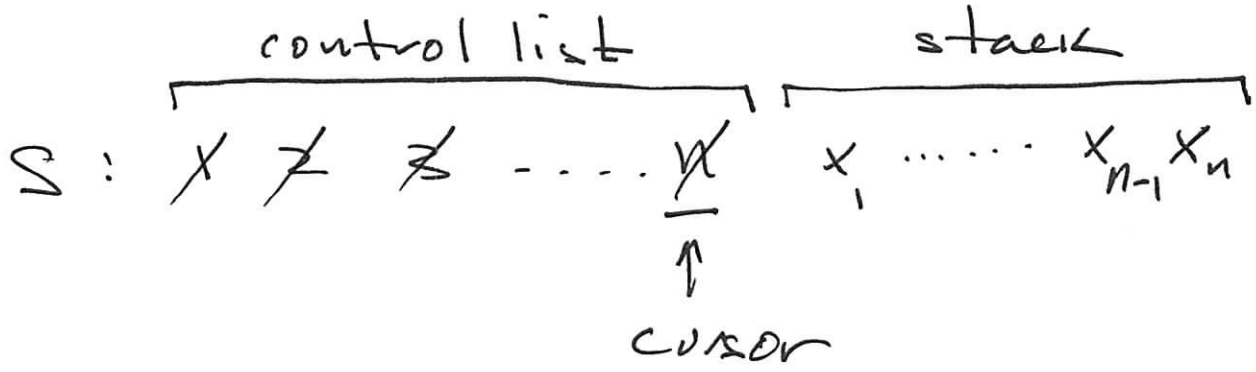
Pa3: ext. 2 days

Design of DFS in Pa3:



How to do all this with one list

- control main loop of DFS
- act as stack recording finish events.



let Push = insertAfter

Handout: Runtime efficiency

Ex.

```
OP1
for i = 1 to 10
    OP2
    for j = 1 to 10
        OP3
        for k = 1 to 10
            (OP4)
```

# of executions.

OP1 : 1

OP2 : 10

OP3 : 100

OP4 : 1000

Propose: • assume costs of ops equal ✓

• count only OP4 ?

assuming $OP1 = OP2 = OP3 = OP4 = OP$

total # of executions of OP

~~1 + 10 + 100 + 10~~

$1 + 10 + 100 + 1000 = 1,111$

why can we approx. 1,111 by 1000 ?

Ex.

```
OP
for i = 1 to n
    OP
    for j = 1 to n
        OP
        for k = 1 to n
            OP
```

total # = $n^3 + n^2 + n + 1$

Propose approx. n^3+n^2+n+1 by n^3

(for any n)

$T(n) = n^3+n^2+n+1 \quad \sim \quad n^3$?

$$\frac{T(n)}{n^3} = \underbrace{1 + \frac{1}{n} + \frac{1}{n^2} + \frac{1}{n^3}}$$

$$\lim_{n\to\infty} \frac{T(n)}{n^3} = \lim_{n\to\infty} \left(\quad \downarrow \quad \right) = 1$$

Conclusion: only count ops inside innermost loop, since this loop gives the highest order term

Ex (revised)

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            op
```

runtime: $\boxed{T(n) = n^3}$

Ex.

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            op
            op
```

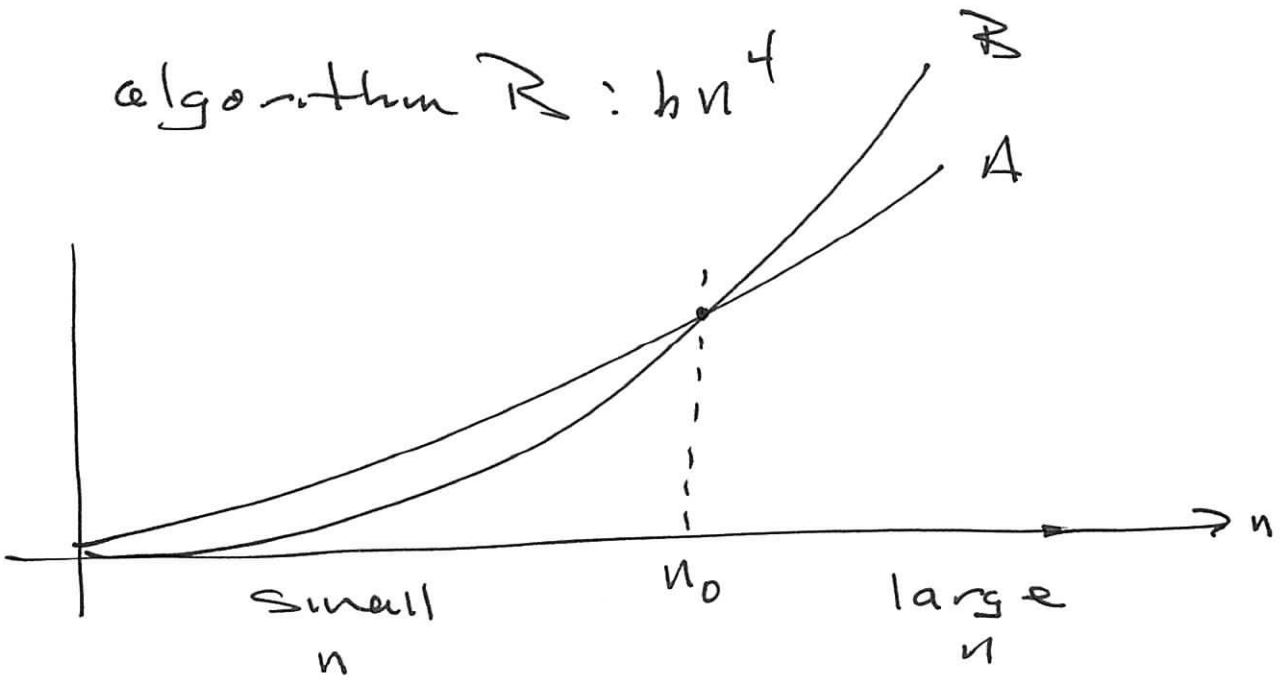
runtime $\boxed{T(n) = 2n^3}$

Propose: consider n^3 and $2n^3$ to be equivalent, i.e. previous 2 examples are equally efficient.

Goal: analyse algorithms as a technology independent of computing device.

Ex algorithm A : an^3

algorithm B : bn^4



our main concern: what happens to runtime as $n \to \infty$

Informal process for analyzing algorithms

(1) choose a basic operation occuring inside innermost loop

(2) count # of executions as a fcn. of n, the input size (consider best, worst, average cases.)

(3) simplify fcn in (1) by
- dropping lower order terms
- replace lead coeff. by 1

Asymptotic growth of fcns:

Defn

Let $f(n), g(n)$ be positive fcns.

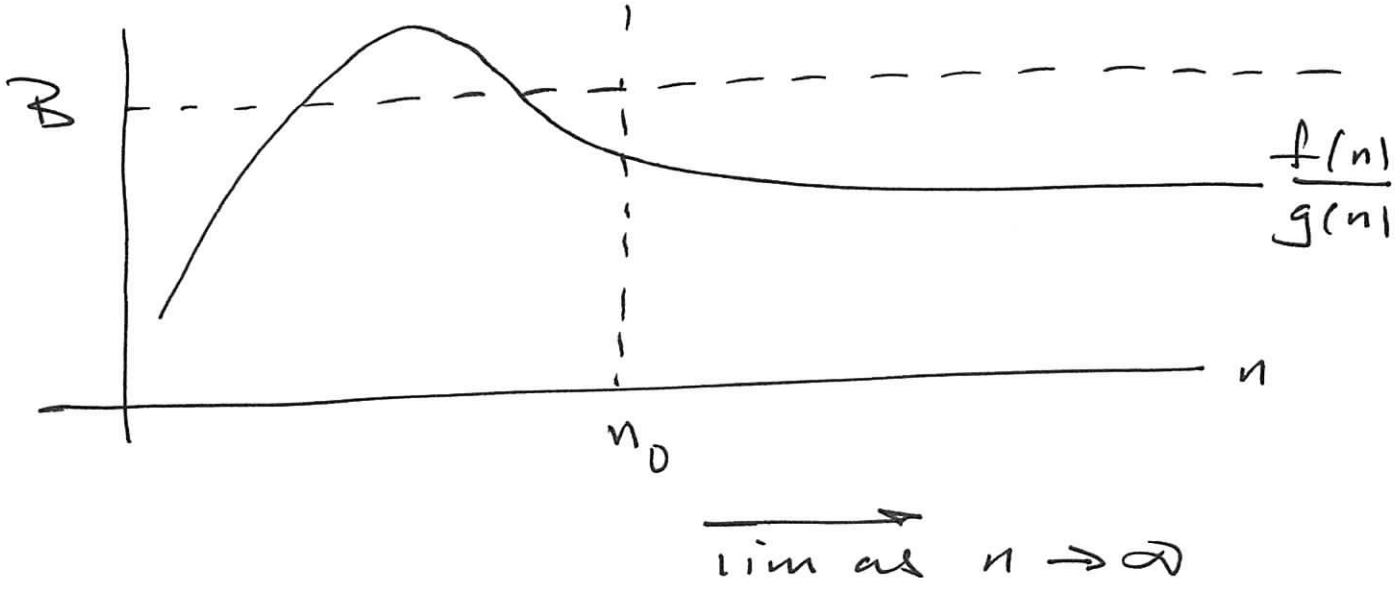
we write $f(n) = O(g(n))$, and we say $f(n)$ is 'big O' of $g(n)$ iff there exist $R > 0$, $n_0 > 0$ such that

$$\frac{f(n)}{g(n)} \leq R$$

for all $n \geq n_0$.

we also say $g(n)$ is an asymptotic upper bound for $f(n)$.

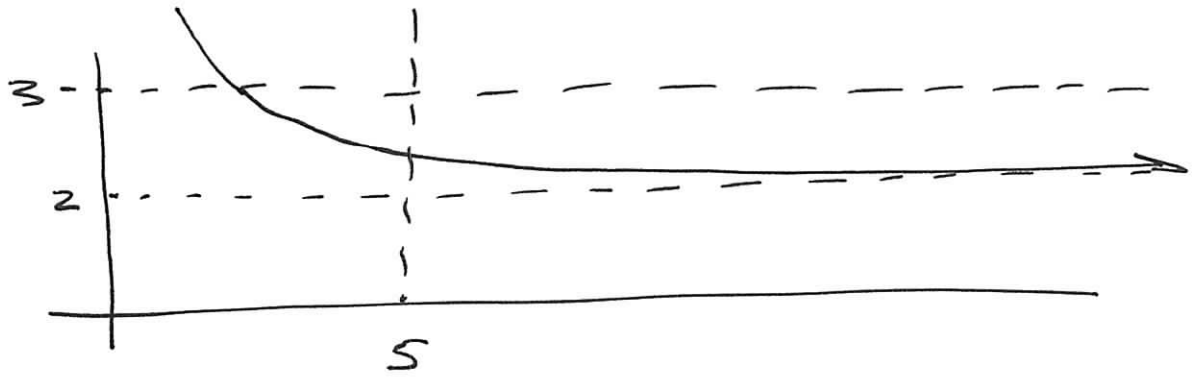
Picture :



Ex. let $f(n) = 2n+5$ $g(n) = n$. Then

$\boxed{f(n) = O(g(n))}$. why ?

$\frac{f(n)}{g(n)} = 2 + \frac{5}{n} \leq 3$ for all $n \geq 5$



[Proof:
$n \geq 5 \Rightarrow \frac{1}{n} \leq \frac{1}{5} \Rightarrow \frac{5}{n} \leq 1 \Rightarrow 2 + \frac{5}{n} \leq 3$]

we've shown : $2n+5 = O(n)$

In fact : $an+b = O(n)$

for any $a>0$, $b>0$. why

$$\frac{an+b}{n} = a + \frac{b}{n} \longrightarrow a \quad \text{as } n \to \infty$$

can pick $\boxed{R = a+1}$, $n_0 = ?$

↑ exercise algebra

$$a + \frac{b}{n} \leq R = a+1$$

$$b \leq n$$

so pick $\boxed{n_0 = b}$