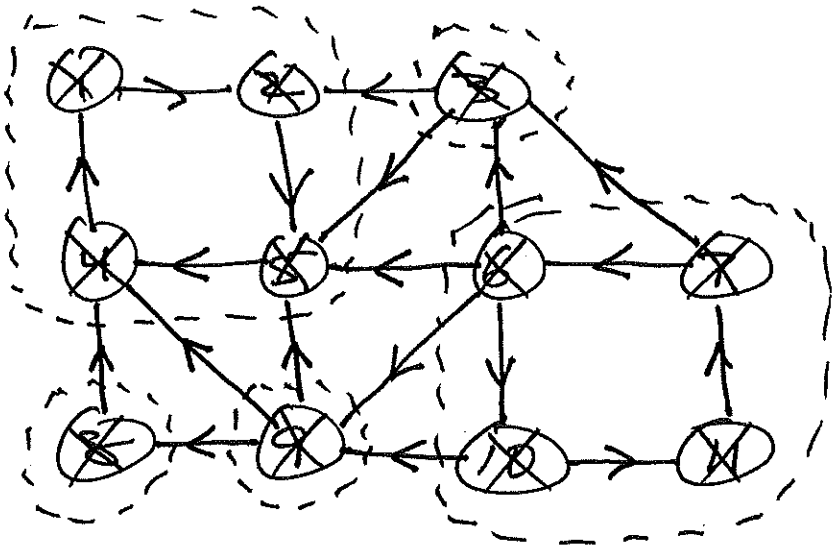
CSE 101-01 4-19-22

- Midterm 1 : Thursday 4/21
  - exam 65 min
  - break 5 min
  - lecture 25 min
- Pa3 ext. 2 days, due Fri.
- Pa4 Posted

Pa3 Questions :

DFS Example : find SCC

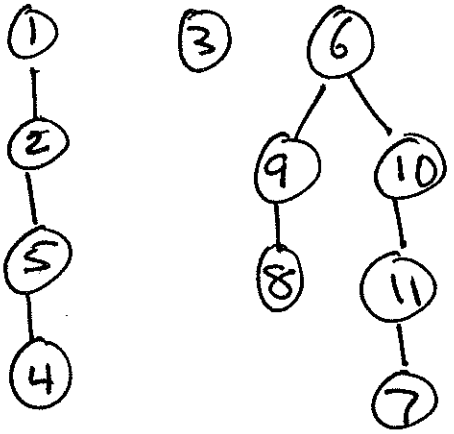
G



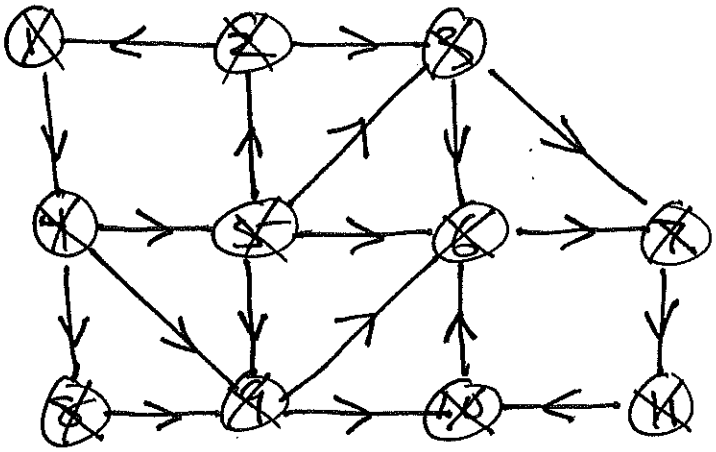
S

6 ✓
10 ✓
11 ✓
7 ✓
9 ✓
8 ✓
3 ✓
1 ✓
2 ✓
5 ✓
4 ✓

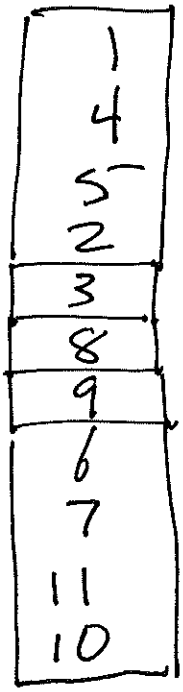
forest



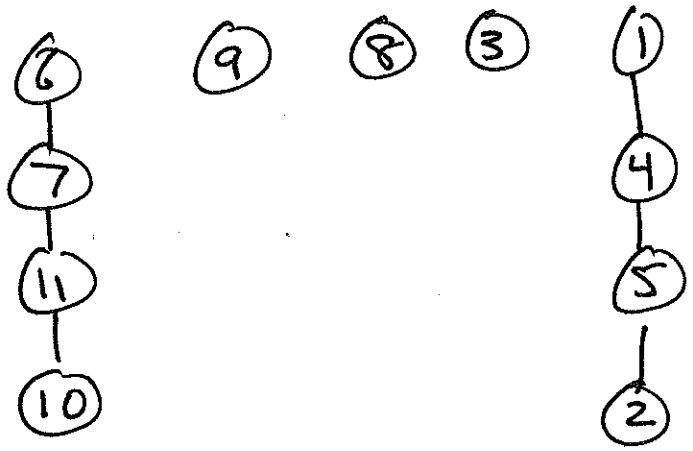
G^T



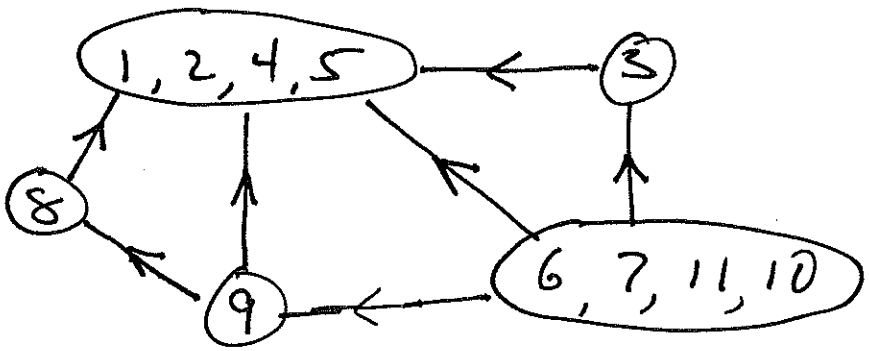
S

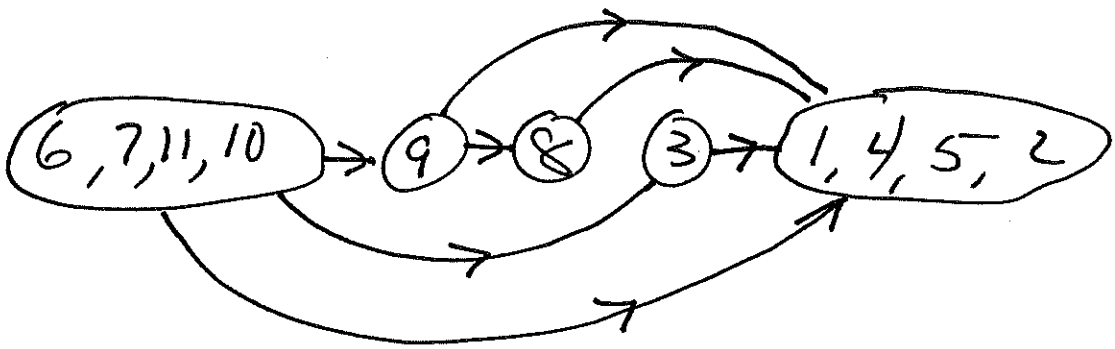


forest :



G^{SCC}

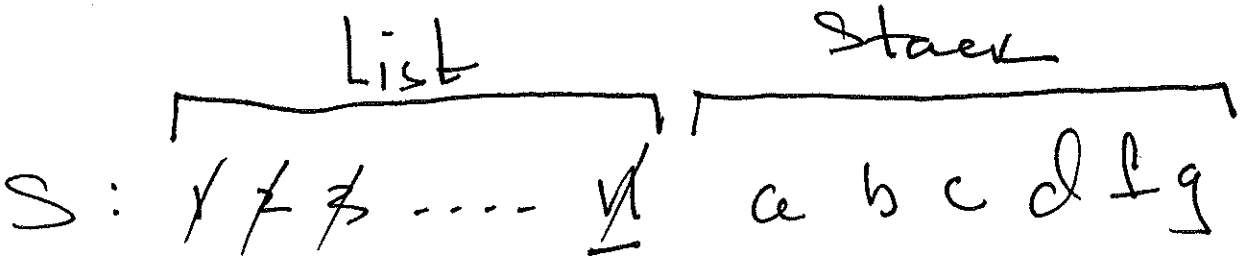




output file:

1 : 6 7 11 10
2 : 9
3 : 8
4 : 3
5 : 1 4 5 2

How to use List S as a stack?



Push = insertAfter

Handout: Algorithm Runtime

How to measure runtime of an algorithm?

- iterative (here)
- recursive (CSE 102)

Ex.

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

Measure "time" by counting operations.

Operation	# executions	
OP1	1	
OP2	10	
OP3	100	
OP4	1000	← count only this.

Questions

(1) what if OP1 - OP4 not (roughly) same cost.
decompose OPs into more primitive operations & count them.

(2) why not count all the operations.

Ex.

```
OP
for i = 1 to 10
    OP
    for j = 1 to 10
        OP
        for k = 1 to 10
            OP
```

ignore

OP ← count this only

answers: 1000

1 + 10 + 100 + 1000 = 1111

more common: loops depend on input

Ex.

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

ignore

only count this

answers:

n^3

$n^3 + n^2 + n + 1 = T(n)$

What happens for large n?

$$\lim_{n\to\infty} \frac{T(n)}{n^3} = \lim_{n\to\infty} \left(\frac{n^3+n^2+n+1}{n^3} \right)$$

$$= \lim_{n\to\infty} \left(1 + \frac{1}{n} + \frac{1}{n^2} + \frac{1}{n^3} \right)$$

$\frac{1}{n} \to 0$, $\frac{1}{n^2} \to 0$, $\frac{1}{n^3} \to 0$

$$= 1$$

<u>lesson:</u> drop lower order terms, i.e. ignore peripheral operations, i.e.

those outside inner-most loop.

simplified example:

Ex.

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

runtime n^3

what about differences inside inner loop?

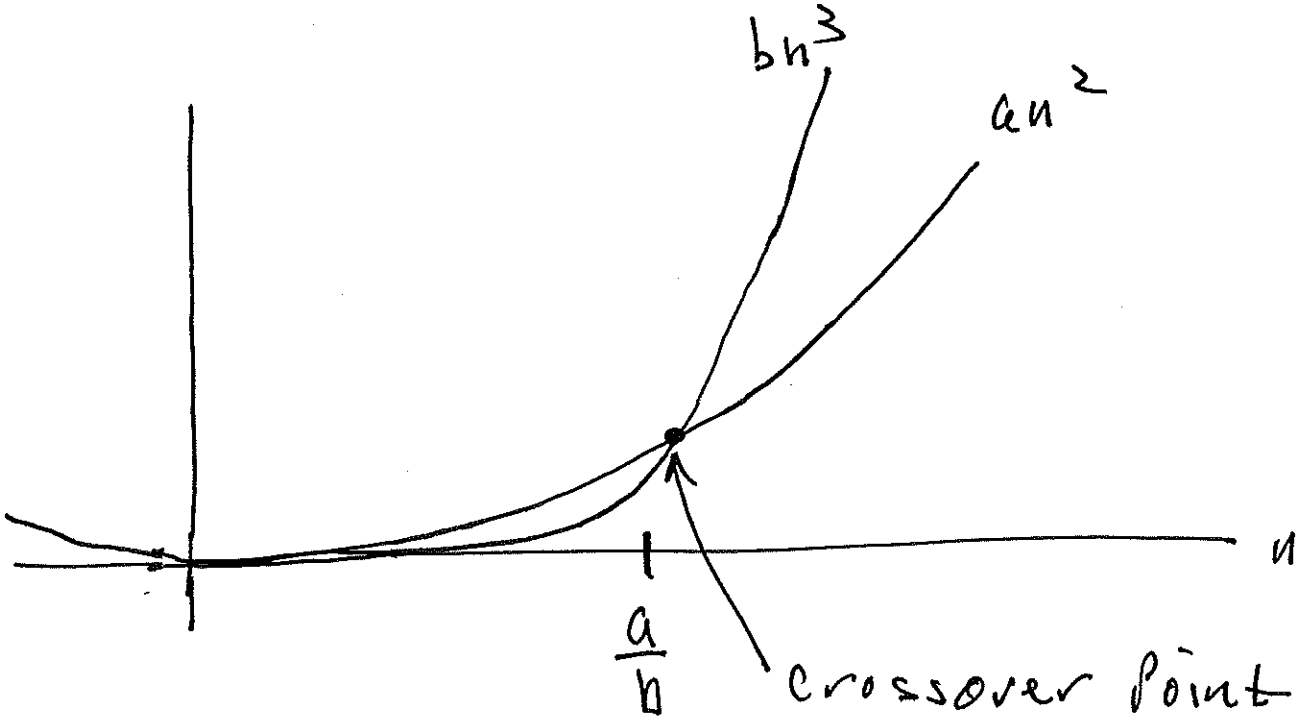
Ex.

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

runtime $2n^3$

we don't consider these to be different since they can be equalized by running 2nd one on 2x faster hardware

note: an^2 can not be equalized with one that runs in time bn^3.



Informal Procedure:

(1) Choose a basic operation appearing inside innermost loop

(2) Count the # of executions of or basic OP, determined as a function of input size = n.

(3) Simplify the fcn in (2) by
- dropping lower order terms
- replacing lead coefficient by 1.

How do we compare these functions?

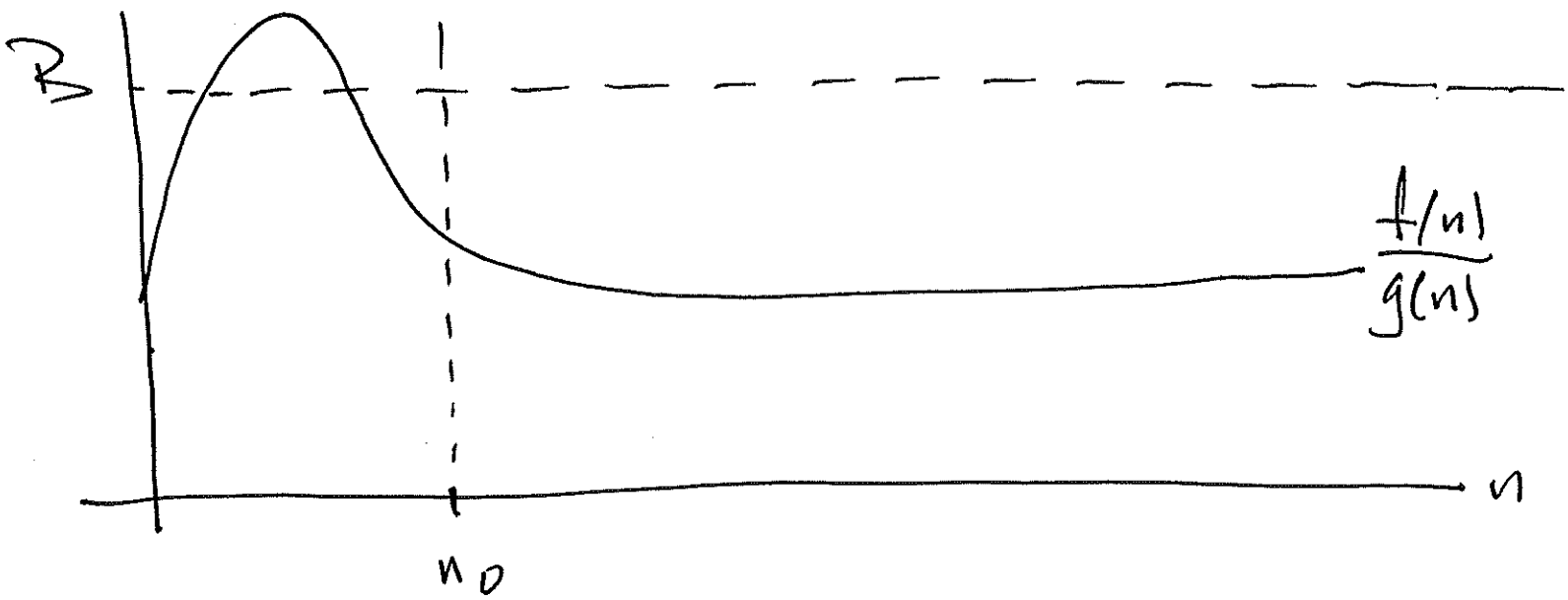
Asymptotic growth of functions

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

Defn

Write $f(n) = O(g(n))$ iff there exist $B > 0$ and $n_0 > 0$ such that

$$\frac{f(n)}{g(n)} \leq B \quad \text{for all } n \geq n_0$$



note: B and n_0 are not unique

Means: $f(n)$ grows no faster than $g(n)$

Say: $g(n)$ is an asymptotic upper bound for $f(n)$.

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

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

(3 ↑ B; 5 ↑ n_0)

Therefore $\boxed{2n+5 = O(n)}$.

note: nothing special about 2 and 5:

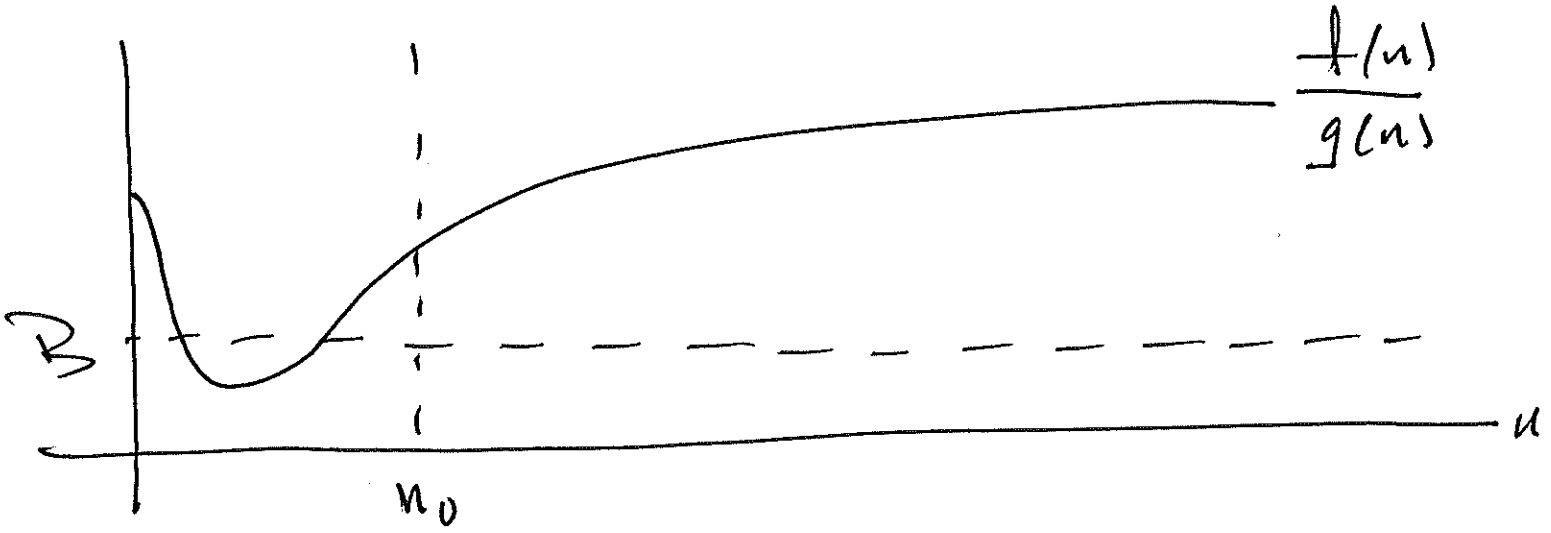
$$an+b = O(n)$$

for any a, b $(a>0)$

Defn

$f(n) = \Omega(g(n))$ iff there exist $B > 0$ and $n_0 > 0$ such that

$$\frac{f(n)}{g(n)} \geq B \quad \text{for all } n \geq n_0$$



mean: $f(n)$ grows at least as fast as $g(n)$

say: $g(n)$ is an asymptotic lower bound for $f(n)$.

note:

since $\frac{f(n)}{g(n)} \leq B_1$ iff $\frac{g(n)}{f(n)} \geq \frac{1}{B_1} = B_2$

We have;

$f(n) = O(g(n))$ iff $g(n) = \Omega(f(n))$.