① — Administrative
- switch.

② — Overview
- pseudocod

③ - Background
- algorithmic
  - sorting
- programming?
- math
  - calculus
  - probab
  - algebra

④ Fibonacci
A. Recursive Exp Time
B. Array Lin Time
C. Matrix mult Log(n)
D. Generate Function Constant Time

⑤ Order of growth

⑥ ~~Powersets~~

① Administrative

- switch
- website, Book
- ~~7-8~~ hws, Midterm, ~~Exam~~.
- Explain EC
- Explain 1-week extension
-

---

② Algorithms overview

algorithm = series of computational steps

input → output

- CORRECT
- ~~EFFICIENT~~ (time, space, other resources)
- ~~ELEGANT~~
- pseudocode → next example.

~~##~~ Max (Cormen example)

~~##~~ ~~[redacted]~~

(3) Prerequisites for CSG113

- **algebra** matrix, array manipulation
  equations, systems
  inequalities, series.
- **probabilities** Spaces, Events, distribution
  $\mathbb{E}[\,]$
- **calculus** limits, continuity, derivatives

- **programming** – imperative language

- **algorithms** – sort; basic graphs; GREEDY, ARRAYS etc

④ Fibonacii (4 methods - 1 alg)

---

⑤ ~~[illegible]~~ Strassen Alg
(if we have time)

---

6 orders of growth

Used Tom Cormen's notes for first lecture.

CS25-X94 Lecture 1 9/20/93 ~~6/24/94~~

Administrivia

Go over course outline & cause info handouts.
Next class: a week from today

Algorithm

A sequence of computational steps that, given an input, produces an output.

Desirable features:

- correct
- efficient — time, space, other resources
- elegant — simple, concise

How to specify?

English, pseudocode, or a mixture.
Use pseudocode to clarify control flow.

Max

Input: An array $A[1..n]$ of numbers, $n \geq 0$.
Output: The maximum number in $A[1..n]$.
More formally, x s.t. $\exists i, 1 \leq i \leq n$ s.t. $A[i] = x$ and $\forall i, 1 \leq i \leq n, A[i] \leq x$.
If $n = 0$, max is $-\infty$.

```
max ← −∞
for i ← 1 to n
    do if A[i] > max
        then max ← A[i]
```

Go over pseudocode - indentation, assignment

Need to
- show correctness
- prove running time

Correctness

If $n = 0$, for-loop iterates 0 times $\Rightarrow$ max $= -\infty$

If $n > 0$, max is either $-\infty$ or some $A[i]$.

Assume that x is real max, $x \neq$ max. Consider the point at which x is compared to max.

2 cases:

1. max wasn't set to x then. Must have had $x \leq$ max, contradiction.
2. max was set to x & later changed. Must have y s.t. $\text{max} \leq x \leq y$, contradiction

Note that max $= -\infty \Leftrightarrow A[i] \neq -\infty \; \forall i$.

Running Time

Here, figure each stmt takes constant time.
(Not necessarily true when stmts are English
Also, separate call of subroutine from its execution. Call is O(1).)

Best case: 1+2n stmts
Worst case: 1+3n stmts
Average case??
In any case, O(n) stmts ⇒ O(n) time

How many comparisons?
n (can get by with n-1)
Is this the best we can do?
Yes - think of a tournament. Everyone but the winner must lose ≥1 game ⇒ ≥ n-1 comparisons are necessary

Four Methods to Compute Fibonacci Numbers

Fibonacci Numbers:

	(F_0)	F_1	F_2	F_3	F_4	F_5	F_6	F_7	...
	(0)	1	1	2	3	5	8	13	...

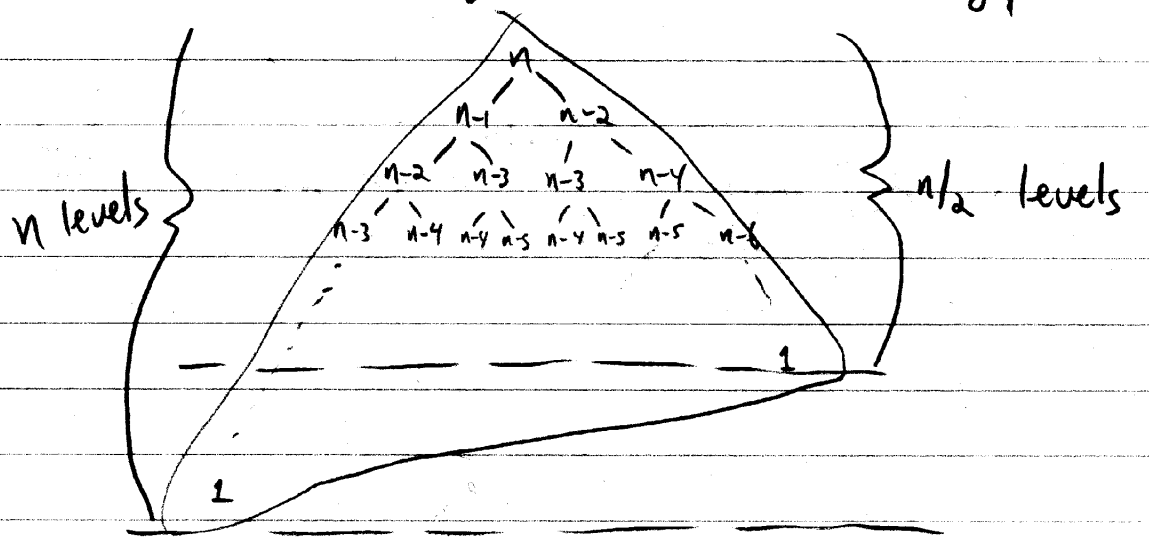
$$F_n = \begin{cases} 1 & \text{if } n = 1 \text{ or } 2 \\ F_{n-1} + F_{n-2} & \text{otherwise} \end{cases}$$

1) Recursively (recursive recursive analysis)

```
fib(n) {
    if (n==1 || n==2)
        return 1;
    else
        return fib(n-1) + fib(n-2);
}
```

Analysis: Constant time for each call; total cost is proportional to total number of recursive calls.

⇒ tree of recursive calls indexed by parameter argument:



Full for $n/2$ levels $\Rightarrow$ $T(n) = \Omega(2^{n/2}) = \Omega(\sqrt{2}^n)$

At most n levels $\Rightarrow$ $T(n) = O(2^n)$

So, somewhere between 1.414^n and 2^n

(can prove that $T(n) = \Theta(\phi^n)$ where ϕ is golden ratio ≈ 1.618)

$\therefore$ Exponential time

c) With an array (previous dynamic programming)

```
fib(n) {
    f[1] = 1;
    f[2] = 1;
    for (i = 3; i <= n; i++)
        f[i] = f[i-1] + f[i-2];
    return f[n];
}
```

Analysis: single loop $\Rightarrow T(n) = \Theta(n)$ $\therefore$ Linear Time

3) Matrix Multiplication (reviews induction)

Consider matrix $M = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$

Claim: $M^i = \begin{pmatrix} F_{i+1} & F_i \\ F_i & F_{i-1} \end{pmatrix}$

Proof: Base Case ($i=1$): ✓

Inductive step Assume $M^k = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}$

$$M^{k+1} = M^k \cdot M = \begin{pmatrix} F_{k+1} & F_k \\ F_k & F_{k-1} \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} = \begin{pmatrix} F_{k+1}+F_k & F_{k+1} \\ F_k+F_{k-1} & F_k \end{pmatrix}$$

$$= \begin{pmatrix} F_{k+2} & F_{k+1} \\ F_{k+1} & F_k \end{pmatrix}.$$

✓

Now use repeated squaring to get answer in $\lg n$ time.—

E.g. $n = 11$ $\frac{8+2+1}{1011}$ in binary

(4)

```
fib(n) {
    M = (1 1 / 1 0);
    A = (1 0 / 0 1);        // "answer" matrix; initially identity matrix
    for(i=0; i <= lg n; i++) {      // i is bit column
        if (n%2 == 1)               // i-th bit is a 1
            A = A * M;              // matrix multiplication
        M = M * M;                  // matrix multiplication
        n = n/2;                    // integer division
    }
    return A[0][1];                 // A = M^n at this point.
```

2 Analysis

simple loop — $T(n) = \Theta(\log n)$

∴ Log time

4) Closed form solution (via generating functions)

$$F_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}} \quad \text{where} \quad \phi = \frac{1+\sqrt{5}}{2} \quad \text{(golden ratio)}$$

$$\hat{\phi} = \frac{1-\sqrt{5}}{2}$$

- essentially constant time w/ math chip.