Lec 15. # Recursion & iteration

A final word on proof by induction

- see page 329 (table of strategy).

- Prove by induction of inequalities

not informative example last class

1) prove $P(n) := (n < 2^n)$

Can prove directly:

take $\ln_2$ of both sides :

$$\log_2 n < n \log_2 2$$

$$\log_2 n < n$$

$$\frac{\log_2 n}{n} < 1$$

True for all $n \geq 1$.

2) More interesting is to prove; for instance:

Try Prooving $P(n) := (2^n < n!)$

3) From last time

$$n < 2^n$$

$$n+1 < 2^n + 1 < 2^n + 2^n = 2(2^n) = 2^{n+1}$$

since $2^n > 1$ for $n \geq 1$.

$$\therefore n+1 < 2^{(n+1)}$$

Example recursive fun'cs.

Explicit factorial func: $f(n) = n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1$

Recursive factorial func:

base case: $f(0) = 1$

$\forall n \geq 1 \quad f(n) = n \cdot f(n-1)$

4. A very important recursive sequence is the Fibonnaci numbers.

Bases cases: $f_0 = 0$

$f_1 = 1$

$\forall n > 1 \quad f_n = f_{n-1} + f_{n-2}$

$f = \{0, 1, 1, 2, 3, 5, 8, \ldots\}$

- $\lim_{n \to \infty} \dfrac{f_{n+1}}{f_n} = \left(\dfrac{1+\sqrt{5}}{2} \right) \equiv \phi$ "the golden ratio"

Recursion: start from biggest element and work down to base case.

"while" / "if" statements in code/algorithms.

Iteration: start from the base case and build up.

"for" statements in algorithms/code.

We looked at recursive vs iterative $f(n) = n!$

- Let's consider $f(a,n) = a^n$

Recursive algorithm.

```
rpow(a, n)
    if n=0
        return 1
    else
        return a·rpow(a, n-1)
```

Iterative algorithm.

```
rpow(a, n)
    x ← 1
    for j=1, 2, ... n
        x ← a·x
    return x.
```

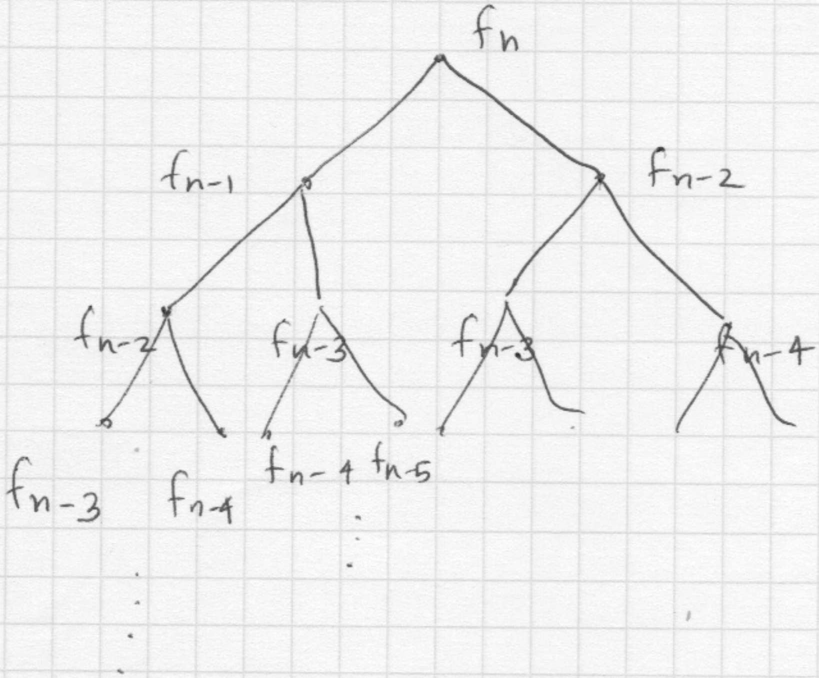
Let's look at Algorithms 1, 2, 3, 4 of handout 10.

→ Here is an algorithm for
<u>iterative</u> binary search with a check for $x = a_m$.

```
Binary search (a, x)
      i ← 1
      j ← n
      m ← ⌊(i+j)/2⌋

      while ((i<j) and (x ≠ a_m))
            if x > a_m
                  i ← m+1
            else
                  j ← m
            endif
            m ← ⌊(i+j)/2⌋

      end while

      if (x = a_m)
            location ← m

      else
            location ← 0
      endif

      return location.
```

Recursive Fibo(n)

```
if n = 0
    return 0

elseif n = 1
    return 1

else
    return RecurseFibo(n-1) + RecursiveFibo(n-2)
```



recursive is exponential run time

Next topic   Counting!

Identify the number of possible configurations / outcomes.
(for a discrete process).

Two basics — Product rule (outcomes multiple)
— Sum rule (outcomes add).

1) Product rule:

If there are A_1 choices for first task
there are A_2 choices for second task

$\therefore$ there are $A_1 \cdot A_2$ possible combinations.

Extended prod rule

A_1 choices for first
A_2 " " second
A_3 " " third
$\vdots$
A_n " " n'th

Total # of choices: $A_1 \cdot A_2 \cdot A_3 \cdots A_n$

a) How many #'s in the range 1000 – 9999?

$\_\ \ \_\ \ \_\ \ \_$ # $9 \cdot 10^3 = 9{,}000$.

# choices			
9 e.g. $\{1, 2, \ldots, 9\}$	10	10	10

b) How many don't repeat a digit

$\_\ \ \_\ \ \_\ \ \_$; # choices: $9 \cdot 9 \cdot 8 \cdot 7$.

#choices			
9	9	8	7

(unique a,b pairs)

c) How many functions are one-to-one from a set $|A| = m$ to $|B| = n$?

$\_\ \ \_\ \ \_\ \ \_\ \ \_\ \ \ldots\ \ \_$

$n \quad (n-1) \quad (n-2) \quad \ldots \quad (n-m+1)$

# of choices: $\dfrac{n!}{m!}$

e). # of telephones #'s of ten digits $Nxx - Nxx - xxx$

$N \in \{2, 3, \ldots 9\}$, $x \in \{0, 1, , \ldots 9\}$