Math 135 – Undecidable Problems

Note Title 3/1/2010

Announcements

- Midterm grades will be posted ASAP
- Next HW will be posted later today (due next Wed. or Fri.)
- Second exam will be in ~3 weeks

Last time: Algorithm Complexity

We use big-O. (& count primitive operations)

Why?

- processer independent
- worst case
- language independent

We saw $O(1)$ to $O(2^n)$

$O(n)$, $O(n^2)$

The Halting Problem

Undecidable problems

(end of 3.1)

Q: Can we write a program which accepts as input another program & input, then decides if the program will run forever or halt on that input.

(So if it contains infinite loop, will run forever, for example, & our program will say that.)

Note: Our program can't just run the input program.

Why?

If I simulate the other program & it runs forever, then so do I.

Thm: The halting problem is undecidable.
(that is, no program to solve it can exist!)

pf: by contradiction

Assume $H(P, I)$ takes P (a program) & I (an input) & outputs "halts" or "runs forever".

Any program is just a string of bits.

So H is also a program

$\to$ $H(H, I)$ a program $H(P, P)$

(continued) Design a program that uses H, called K.

K(P) calls H(P,P).

If H(P,P) says "loop forever", K will halt.

If H(P,P) says "halt", K will loop forever.

What does K(K) do?

depends on H(K,K).

if it says "loop forever", K(K) will halt

if it says "halt", then K(K) loops forever.

either way, reached a contradiction. ☺

Recurrence Relations (4.3-4.4)

- Use them to model counting problems
- Useful for runtime analysis of recursive algorithms (more next time)
- To solve:
  - unrolling
  - → induction
  - more advanced techniques such as Master thm & characteristic eqn method (after break)

Example:

Consider a sequence of numbers:

$$\begin{array}{cccccc} a_0 & a_1 & a_2 & a_3 & a_4 & \\ \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \cdots \\ 0 & 2 & 4 & 6 & 8 & \end{array}$$

Closed form

$a_n = 2 \cdot n$

Recursive

~~Inductive~~ Dfn:

$a_0 = 0$

$a_n = a_{n-1} + 2$

Another example: Fibonacci numbers

	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	...
f_n:	0	1	1	2	3	5	8	13	21	34	...

$$f_0 = 0$$
$$f_1 = 1$$

Recursive Dfn? $f_n = f_{n-1} + f_{n-2}$

Closed form? $f_n = \dfrac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}$

Give recursive dfn of:

① $F(n) = n!$

$F(1) = 1$

$F(n) = n \cdot F(n-1)$

$n! = n\,(n-1)!$

② $A(n) = a^n$

$A(n) = a \cdot A(n-1)$

$A(n-1) = a^{n-1}$

Compound interest

- $P_0 = \$1{,}000$ (initial investment)
- make 6% interest per year

How much will you have after n years?

Recursive Dfn: $P_0 = 1{,}000$

$$P_n = (1.06)P_{n-1}$$

$$P_n = (1.06)^n(1{,}000)$$

Claim: $P_n = (1.06)^n P_0 = (1.06)^n (1{,}000)$

Pf: (induction on n)

base case: $n=0$ $\quad P_0 = 1000$

$(1.06)^0 \cdot 1000 = 1 \cdot 1000 = 1000$ ✓

ind hyp: Assume $P_{n-1} = (1.06)^{n-1} \cdot (1000)$

ind step: Consider P_n.

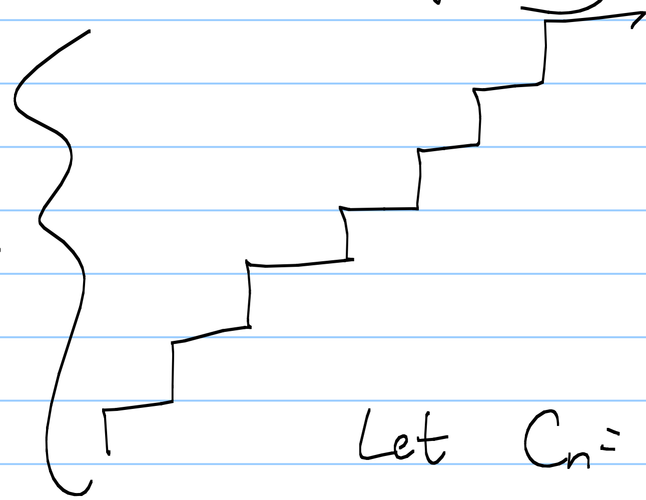
Know $P_n = (1.06)\, P_{n-1}$ by recursive dfn.

By IH $\quad = (1.06)(1.06)^{n-1}(1000)$

$= (1.06)^n (1000)$ ✓ $\boxed{}$

Stair stepping

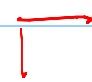
n steps



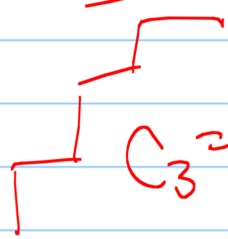
If I can take stairs 1 or 2 at a time, how many different ways are there to climb the stairs?

Let $C_n :=$ # ways to climb n stairs

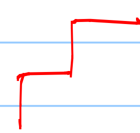
Base cases:



$C_1 = 1$



$C_3 = 3$



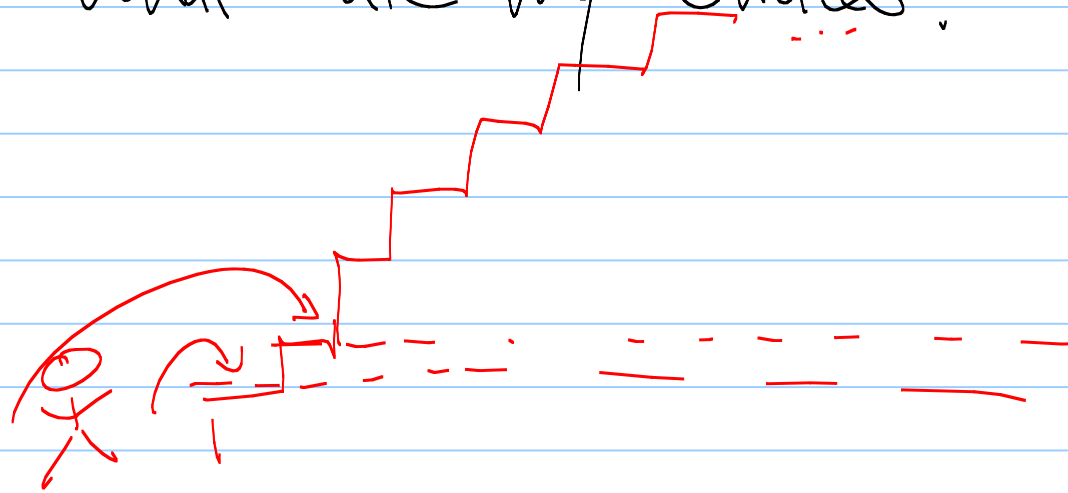
$C_2 = 2$

$1 + 1$ or 2

Think recursively!

First step: What are my choices?

C_n



$$C_n = C_{n-1} + C_{n-2} = (C_{n-2} + C_{n-3}) + (C_{n-3} + C_{n-4})$$

Fibonacci numbers!