Math 135 - Undecidable Problems

Note Title 3/1/2010

Announcements

- HW due today
- New HW out tomorrow or Wed., & due after break.

Last time: Algorithm Complexity

We use big-O. + worst case $\underbrace{\text{running time}}_{\text{\# of operations}}$.

Why?

- Actually timing a program leads to a lot of variance:
  - computer type
  - programming language
  - input

The Halting Problem

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.

We will prove that writing such a program is impossible.

(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 my program simulates the other one, & it contains an infinite loop, then I never halt either.

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

pf: by contradiction

Assume we have a program to solve the halting problem → $H(P, I)$.

my program

input program

input to P we want to simulate

$H(P, I)$ outputs either "halts" or "loops forever".

Any program is written as a string of characters (or 0's + 1's).

So any P can be written down!

So we could "feed" a program to H along with itself as the input: H(P,P)

To show H can't exist:

Design an algorithm K which accepts a program P as input, + runs H(P,P). If the output of H(P,P) is "loops forever", then design K so that it halts. If the output of H(P,P) is "halts", K(P) will loop forever.

Well, K is a program too!

So run $K(K)$.

If $H(K,K)$ says "loops forever", then (by dfn of K), $K(K)$ halts.

If $H(K, K)$ says "halts", then by by dfn of K, $K(K)$ loops forever.

Either way, $K(K)$ isn't correct,

So H cannot exist. ∎