// some code:

int positive = 0

for (i = 0; i < n; i++)
    positive = 1;

Mar 10, 09

+ Thursday 3/12
  - Go over problems
  - Quiz 1 Mini Test

+ Friday 3/13

Spring break.

T: 3/24

R: 3/26

Project progress report

Luc Langre

Problem: We need to separate feasible algorithms from non-feasible ones.

Examples:
+ $n^2$ — feasible
+ $2^n$ — not feasible

Definition:

$$t^w(n) = \max t_k(x)$$

$x: \text{len}(x) = n$

An algorithm $u$ is called feasible if there exists a polynomial $p(n) \leq t^w(n)$.
\[ \forall n \quad t_w(n) \leq \mathcal{O}(n) \]

- Important fact:

This definition does not fully reflect the notion of feasibility.

- Ex 1: \[ t_w(n) = 10^{1000} \cdot n \]
  - commonsense viewpoint: not feasible
  - definition: still polynomial: feasible!

- Ex 2: \[ t_w(n) = \exp(\#0,000,000,01 \cdot n) \]
  - commonsense viewpoint: feasible
  - definition: NOT feasible

Problem: No one knows how to produce a better definition.

We can use the knowingly imperfect definition of feasibility.

What is a problem:

\[
\begin{array}{c}
\text{input} \\
\rightarrow \\
\rightarrow \\
\text{output} \\
\text{condition} \\
\text{x} \\
\leftarrow \\
\text{y} \\
\end{array}
\]

Mathematics: \( x \) - statement, \( y \) - proof of \( x \) or \( \neg x \).
Comments:
Checking where \( y \) is the correct proof of \( x \) is relatively easy.

The length of the proof should be feasible.

- Condition \( R(x, y) \) is feasible:
  \[ \text{len}(y) \leq p(x) \text{len}(x) \]

We have:
- feasible predicate \( R(x, y) \) returns \( T/F \).
- polynomial \( p(x) \).

Given \( x \): statement
we want: find \( y \) such that
\[ R(x, y) \text{ and } \text{len}(y) \leq p(x) \text{len}(x) \]

Physics:
- \( x \): observation
- \( y \): theory that explains observation.

\( R(x,y) \): check whether the observations are consistent with the data.
\[ \text{len}(y) \leq \text{len}(x) \]

Engineers
- \( x \): specification (cat, how much it weighs, wind)
- \( y \): design

\( R(x, y) \): feasible.
In principle every practical problem can be algorithmically solved by trying all binary strings $y$ of length $\leq \text{len}(x)$.

Problem: We need $2^n$ time, not feasible.

Good news: if we guess an answer then we can check its correctness in polynomial time.

A pr. problem = can be solved on a non-deterministic TM in a polynomial time.

$NP = \text{Non-deterministic Polynomial}$

$= \text{class of all practical problems}$

- CS problems $x$ - original list
  $y$ - sorted list
  $r(x, y), \text{len}(y) = \text{len}(x)$

$P$ = class of all problems which can be solved by a feasible algorithm

Is it possible to solve any practical problem in reasonable time?

$\rightarrow$ Open problem / Don't know.

Reduction: if there is an algorithm to solve a problem in class $A$ => also can be used to solve other problems in the same class $A$. 

Humanity, $x$ - emotions
$y$ - a form
\[ N^2 = P \]

\[ ax^2 + bx + c = 0 \]

\[ ax + b \]