6 July

Cook's theorem [P&S §15.5]

Recall:

- We say that a decision problem A_1 <u>polynomially transforms</u> to another decision problem A_2 if there is a polynomial-time algorithm to convert any instance x of A_1 to an instance y of A_2 such that the answer to x is "yes" if and only if the answer to y is "yes."

- A decision problem A is called <u>NP-complete</u> if
  - $A \in NP$ and
  - all other problems in NP polynomially transform to A.

Questions:

- Do NP-complete problems exist?
- If so, what is an example of one?

Cook's theorem answers both of these questions by showing that the Boolean satisfiability problem is NP-complete.
But first we need to develop some concepts to more precisely define the idea of a "verifier" in the definition of NP.

6 July

The Turing machine: A model of computation.

(This is a slightly different model of the Turing machine from the most common one, but it is polynomially equivalent in its computational power.)

The input to the machine is provided on a finite tape. The "goal" of the machine is to either accept or reject the input.
The tape is divided into cells. Each cell holds exactly one symbol from a finite set Σ of symbols called the alphabet. ["Blank" can be a symbol.]

The machine has a tape head that can move back and forth on the tape. It begins at the first cell of the tape.
The tape head can read symbols from the tape, and it can also erase a symbol and write a new one in its place.

The operation of the machine is specified by a program, which is a sequence of instructions. All but the last instruction are of the form

$$\ell: \text{ if } \sigma \text{ then } (\sigma', \Delta, \ell')$$

with the following meaning:

- ℓ is a line number (an instruction label): the first instruction has $\ell = 1$, the second has $\ell = 2$, etc.
- σ, σ' are symbols from the alphabet Σ.
- $\Delta \in \{-1, 0, 1\}$.
- ℓ' is the line number of another instruction in the program.

The meaning of the instruction

$$\ell: \text{if } \sigma \text{ then } (\sigma', \Delta, \ell')$$

is: If the symbol currently under the tape head is σ, then erase it and replace it with σ', move the tape head Δ cells to the right, and go to instruction ℓ'. Otherwise, continue to the next instruction (instruction $\ell + 1$). [Note: $\Delta = 0$ means don't move tape head; $\Delta = -1$ means move one cell to the left.]

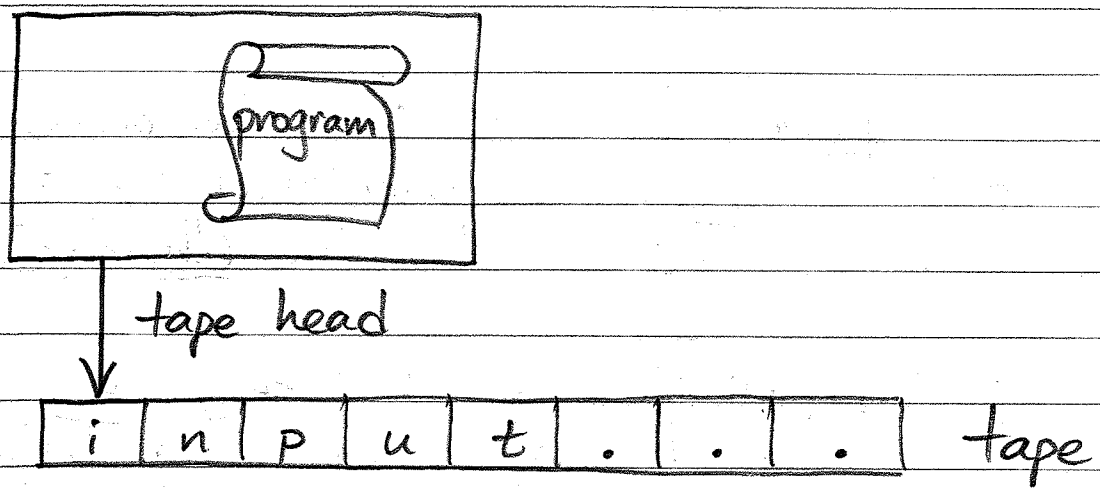
The last instruction in the program is

$$|\mathcal{A}| : \text{accept}$$

Its line number is $|\mathcal{A}|$, the length of the program $\mathcal{A}$. Once the program reaches this instruction, it halts. (It has accepted the input.)

6 July

Diagram of a Turing machine:



If the tape head ever attempts to move off the beginning or end of the tape, the program halts. (This is considered a rejection of the input.)

The machine begins with the tape head at the beginning of the tape and ready to execute instruction 1.

Advantages of the Turing machine model

- It is simple. There is really only one type of instruction.

- It is powerful. Any other "reasonable" model of computation can be simulated by a Turing machine. This is called the Church-Turing thesis.

- It is implementable. It is easy to imagine how a working physical model of a Turing machine could be built. This means that the theory can be put into practice.
  - Of course, real computing devices are much more complex than the simple Turing machine model, but they are not more powerful as far as what they can compute.

6 July

Turing machine as a verifier.

A Turing machine can act as a verifier for a problem in NP:

- Both the instance and the corresponding certificate can be represented as strings of symbols from Σ.

- We reserve one symbol \$ in Σ to act as a separator and guarantee that it will never appear in the representation of the instance.

- So the instance and certificate together can be represented as a single string, separated by \$.

So we can define NP more formally:

Defn. NP is the class of decision problems for which there exists a Turing machine A and a polynomial $p(n)$, such that a string x represents a "yes" instance if and only if there exists a string $c(x)$, with $|c(x)| \leq p(|x|)$, such that A accepts the string $x\$c(x)$ in at most $p(|x|)$ steps.

Notes.

— An algorithm often needs "scratch space" to store temporary results as it works. We can provide this as part of the certificate $c(x)$ —— e.g., trailing blanks.

— Since we will run the algorithm for at most $p(|x|)$ steps, it can reach only the first $p(|x|)$ cells of the tape. So, WLOG we may assume that $x\$c(x)$ is always exactly $p(|x|)$ symbols long —— pad the certificate with blanks if needed.

6 July

Cook's Theorem [P&S Thm 15.1] — 1971.

The Boolean satisfiability problem (SAT) is NP-complete.

Proof. We need to establish:

- SAT is in NP;
- if A is a problem in NP, then A can be polynomially transformed to SAT.

It's easy to see that SAT is in NP: The certificate for an instance with n variables consists of n bits, one for each variable, giving a satisfying assignment. This has polynomial size. The verifier then just verifies that the truth value of the formula is "true" under these assignments, which can be done in $O(m)$ time for a formula of length m.

Now for the hard part.

Let $A \in NP$.

— We know, by definition, that there exists a Turing machine $\mathcal{A}$ and a polynomial $p(n)$ such that a string x represents a "yes" instance if and only if there exists a string $c(x)$, with $|c(x)| \leq p(|x|)$, such that $\mathcal{A}$ accepts the string $x\$c(x)$ in at most $p(|x|)$ steps.

Our goal: Show that A can be polynomially transformed to SAT.

This means: Given any instance x of A, construct an instance of SAT, i.e., a propositional formula $F(x)$, whose size is bounded by a polynomial in $|x|$, such that $F(x)$ is satisfiable if and only if x is a "yes" instance of A.

The idea: Simulate the operation of the verifier A within a propositional formula.

Variables in the propositional formula:
(these are all Boolean variables, of course)

- $t_{ij\sigma}$ for all $0 \le i \le p(|x|)$, $1 \le j \le p(|x|)$, $\sigma \in \Sigma$.

  Intended meaning: At time i (i.e., immediately following i steps of A), the jth cell of the tape contains the symbol σ.
  [Mnemonic: t for "tape." P&S use $X_{ij\sigma}$.]

- s_{ijl} for all $0 \le i \le p(|x|)$, $0 \le j \le p(|x|)+1$, $1 \le l \le |A|$. [$|A|$ = number of instructions in A]

  Intended meaning: At time i, the tape head is at cell j and the lth instruction is to be executed. $j=0$ or $j=p(|x|)+1$ means the tape head has fallen off the end of the tape.
  [Mnemonic: s for "status." P&S use Y_{ijl}.]

6 July

Cook's theorem — (2)

The formula $F(x)$ that we construct will be the conjunction of four subformulas:

$$\boxed{F(x) = U(x) \wedge S(x) \wedge W(x) \wedge E(x).}$$

$U(x)$: uniqueness.

The purpose of $U(x)$ is to ensure that at each time i, $0 \le i \le p(|x|)$:

- Each cell j of the tape contains a unique symbol:

$$U_1(x) = \bigwedge_{\substack{0 \le i \le p(|x|) \\ 1 \le j \le p(|x|)}} \bigvee_{\sigma \in \Sigma} t_{ij\sigma} \qquad \left[\begin{array}{c}\text{cell } j \text{ must} \\ \text{contain some} \\ \text{symbol}\end{array}\right]$$

$$U_2(x) = \bigwedge_{\substack{0 \le i \le p(|x|) \\ 1 \le j \le p(|x|) \\ \sigma \neq \sigma'}} \left(\overline{t}_{ij\sigma} \vee \overline{t}_{ij\sigma'}\right) \qquad \left[\begin{array}{c}\text{cell } j \text{ can't} \\ \text{contain two} \\ \text{different symbols}\end{array}\right]$$

- The tape head is at a unique position on the tape, and a unique instruction is run:

$$U_3(x) = \bigwedge_{0 \le i \le p(|x|)} \bigvee_{\substack{0 \le j \le p(|x|)+1 \\ 1 \le \ell \le |\mathcal{A}|}} S_{ij\ell} \qquad \left[\begin{array}{c}\text{tape head must} \\ \text{be somewhere,} \\ \text{some instr. must} \\ \text{be executed}\end{array}\right]$$

$$U_4(x) = \bigwedge_{\substack{0 \le i \le p(|x|) \\ j \neq j' \text{ or } \ell \neq \ell'}} \left(\overline{S}_{ij\ell} \vee \overline{S}_{ij'\ell'}\right) \qquad [\text{uniqueness}]$$

$$U_5(x) = \bigwedge_{\substack{0 \le i \le p(|x|) \\ 1 \le \ell \le |\mathcal{A}|}} \left(\overline{S}_{i0\ell} \wedge \overline{S}_{i[p(|x|)+1]\ell} \right)$$ [can't fall off tape]

Then $U(x) = U_1(x) \wedge U_2(x) \wedge U_3(x) \wedge U_4(x) \wedge U_5(x)$.

S(x): machine starts correctly.

- At time $i = 0$, the first $|x|+1$ cells of the tape contain the string $x\$$:

$$S_1(x) = \left(\bigwedge_{1 \le j \le |x|} t_{0j[x(j)]} \right) \wedge t_{0(|x|+1)\$}$$

- At time $i = 0$, the tape head is at cell 1 and the first instruction is to be executed:

$$S_2(x) = s_{011}$$

Then $S(x) = S_1(x) \wedge S_2(x)$.

[Note: Here we are not specifying the initial contents of cells $|x|+2, \ldots, p(|x|)$ — i.e., we are not specifying the certificate for x. So searching for an assignment to the variables $t_{ij\sigma}$, $s_{ij\ell}$ that satisfies $F(x)$ will be equivalent to searching for a valid certificate $c(x)$ for x that is accepted by $\mathcal{A}$ in at most $p(|x|)$ steps.]

6 June

Cook's theorem — ③

W(x): machine works correctly.

- For $0 \le i \le p(|x|)$, $1 \le j \le p(|x|)$, $\sigma \in \Sigma$, and $1 \le \ell \le |\mathcal{A}|$

  such that the ℓth instruction of $\mathcal{A}$ is

  ℓ: if σ then (σ', Δ, ℓ'):

  — At time i, if the symbol in cell j is σ (i.e., if $t_{ij\sigma}$ is true) and the tape head is at cell j and instruction ℓ is to be executed (i.e., if $s_{ij\ell}$ is true), then at time $i+1$ the symbol in cell j must be σ' (i.e., $t_{(i+1)j\sigma'}$ must be true) and the tape head must be at cell $j+\Delta$ and instruction ℓ' is to be executed (i.e., $s_{(i+1)(j+\Delta)\ell'}$ must be true).

$$W^{(1)}_{ij\sigma\ell}(x) = (\overline{t}_{ij\sigma} \vee \overline{s}_{ij\ell} \vee t_{(i+1)j\sigma'}) \wedge (\overline{t}_{ij\sigma} \vee \overline{s}_{ij\ell} \vee s_{(i+1)(j+\Delta)\ell'})$$

  — Otherwise, if at time i the tape head is at cell j and instruction ℓ is to be executed but the symbol in cell j is $\tau \neq \sigma$, then at time $i+1$ the symbol in cell j must be τ (unchanged),

the tape head must be at position j (unchanged), and instruction $\ell+1$ is to be executed:

$$W^{(2)}_{ij\sigma\ell}(x) = \bigwedge_{\tau\neq\sigma} \Big[(\bar{T}_{ij\tau} \vee \bar{S}_{ij\ell} \vee t_{(i+1)j\tau}) \wedge (\bar{T}_{ij\tau} \vee \bar{S}_{ij\ell} \vee S_{(i+1)j(\ell+1)}) \Big].$$

— For all i, j, σ, if the $|A|$th instruction (accept) is to be executed, then the program should stay there at time $i+1$:

$$W_{ij\sigma|A|} = \bar{T}_{ij\sigma} \vee \bar{S}_{ij|A|} \vee S_{(i+1)j|A|}$$

— Whenever the tape head is at the cell $j' \neq j$, the symbol in the jth cell should be unchanged:

$$W'(x) = \bigwedge_{\substack{0\leq i\leq p(|x|) \\ \sigma\in\Sigma \\ 1\leq\ell\leq|A| \\ j\neq j'}} (\bar{T}_{ij\sigma} \vee \bar{S}_{ij'\ell} \vee t_{(i+1)j\sigma})$$

Then $W(x)$ is the conjunction of all these subformulas.

6 July

Cook's theorem — ④

E(x): machine ends correctly.

At time $p(|x|)$, the instruction to be executed must be the $|\mathcal{A}|$th (accept):

$$E(x) = \bigvee_{1 \le j \le p(|x|)} S_{[p(|x|)]\, j\, |\mathcal{A}|}.$$

Note:

— The total length of the formula $F(x)$ is $O(p^3(|x|) \log p(|x|))$, when you count all of the $O(p^3(|x|))$ literals in the formula multiplied by the length $O(\log p(|x|))$ of the subscripts needed to encode them. So $F(x)$ is of polynomial length in $|x|$, and the construction of $F(x)$ can be done in polynomial time.

Claim. $F(x)$ is satisfiable if and only if x is a "yes" instance of A.

Proof. ($\Rightarrow$) Suppose $F(x)$ is satisfiable, so there exists an assignment of values to the variables $t_{ij\sigma}$, s_{ijl} that satisfies all of $U(x)$, $S(x)$, $W(x)$, and $E(x)$. Because $U(x)$ is satisfied, for every i,j exactly one $t_{ij\sigma}$ is true — take this to mean that at time i cell j contains the symbol σ — and for every i exactly one s_{ijl} is true — take this to mean that at time i the tape head is at cell j and instruction l is to be executed; also no s_{i0l} or $s_{i[p(x)+1]l}$ can be true, so the tape head does not fall off the end of the tape. So the values of the variables $t_{ij\sigma}$, s_{ijl} describe a sequence of tape contents, head positions, and instructions to be executed. Since $S(x)$ is satisfied, the first portion of the tape at time 0 contains the string $x\$$, and at time 0 the tape head is at cell 1 and instruction 1 is to be executed. Since $W(x)$ is satisfied, the sequence changes in accordance with the rules of the program for A. And since $E(x)$ is satisfied, the sequence ends in an accepting state. Therefore, if $F(x)$ is satisfied, there exists a certificate $c(x)$, namely, the contents of cells $|x|+2, \ldots, p(|x|)$ at time $i=0$, such that A accepts $x\$c(x)$ in at most $p(|x|)$ steps. So x is a "yes" instance. ✓

6 July

($\Leftarrow$) Suppose x is a "yes" instance. Then there exists a certificate $c(x)$ such that A accepts $x\$c(x)$ in at most $p(|x|)$ steps. WLOG, as justified earlier, $x\$c(x)$ has length exactly $p(|x|)$. The execution of A on the input $x\$c(x)$ yields a sequence of tape contents [beginning with $x\$c(x)$ at time $i=0$], head positions, and instructions to be executed that gives an assignment of truth values to the variables $t_{ijσ}$, $s_{ijℓ}$ such that $F(x)$ will be satisfied. ✓ ■

By this claim, $F(x)$ is a "yes" instance of SAT iff x is a "yes" instance of A. So this gives a polynomial transformation of A to SAT.

Because $A \in NP$ was arbitrary, this proves the theorem. □

Corollary. CNF-SAT is NP-complete.

(CNF-SAT is the Boolean satisfiability problem in which the propositional formula is restricted to be in conjunctive normal form, i.e., a conjunction of disjunctions of literals.)

Proof. If we carefully examine the structure of the formula $F(x)$ constructed in the proof of Cook's theorem, we see that it is in CNF. So in fact this proof gives a polynomial transformation from any $A \in NP$ to CNF-SAT.
— And as CNF-SAT is a special case of SAT, it is also in NP. □

Corollary. (The decision version of) integer linear programming is NP-complete.

Proof. The IP formulation of CNF-SAT presented in lecture on June 24 is a polynomial transformation of CNF-SAT to ILP.
— And ILP $\in$ NP: Certificate for a "yes" instance is a feasible solution; verifier verifies that the solution satisfies all constraints. (See P&S Example 15.8, §15.3, for a more careful justification that ILP $\in$ NP.)
Since every problem in NP can be polynomially transformed to CNF-SAT, and CNF-SAT can be poly transformed to ILP, this shows that ILP is NP-complete. □