9 July

Lies from yesterday:
- The transformations from HAMILTONIAN CIRCUIT to HAMILTONIAN PATH and LONGEST PATH given in the lecture were broken. These have been fixed in the lecture notes online.

The halting problem

All of this course has been a discussion of algorithms to solve problems. Today we look at a problem that cannot be solved by any algorithm.

Background: the Entscheidungsproblem (German for "decision problem").

David Hilbert, 1928: Construct a general "theorem checker": an algorithm that takes as input a set of axioms and a statement in first-order logic and answers "yes" or "no" depending on whether the statement can be proved from the axioms.

1936: Alonzo Church and Alan Turing independently showed that a general solution to this problem is impossible by showing that there exist easily stated statements for which no single algorithm can always correctly answer "yes" or "no".

Recall: Turing machines.

The Church-Turing thesis is the assertion that all "effective" models of computation can be modeled by a Turing machine (or, equivalently, by Church's model of "λ-calculus").

Therefore, Hilbert's problem asking for an algorithm can be viewed as asking for a Turing machine.

Previously, we were using Turing machines as models of polynomial-time verifiers for problems in NP, so we were able to assume that the length of the machine's tape was of polynomial length. In general we cannot make that assumption — the general Turing machine has a one-way infinite tape (it has a left end but no right end). The initial input to the tape appears in the leftmost cells of the tape, and the rest of the tape is initially filled with blanks.

The possible results of running such a Turing machine with an initial input are:

- the machine eventually reaches the accept instruction, and halts;
- the tape head eventually falls off the left end of the tape (a reject), and the machine halts;
- or the machine keeps running forever.

9 July

HALTING PROBLEM

Instance: A description M of a Turing machine (i.e., the list of instructions) and input I for that machine.

Question: If M is run with input I, will it eventually halt?

Theorem (Turing, 1936)
There does not exist an algorithm that solves the HALTING PROBLEM.

Proof. Assume for the sake of contradiction that such an algorithm does exist, i.e., there exists a Turing machine H such that the result of running H on the instance (M, I) [denoted $H(M, I)$] is *accept* if running M with input I will eventually halt, or *reject* otherwise.

Now, construct another Turing machine L that takes as input a description A of a Turing machine and does the following:

1. Run $H(A, A)$ as a subroutine.
2. If the result of $H(A, A)$ is *accept*, then loop forever; otherwise halt.

What happens when we run $L(L)$?

First, L runs $H(L,L)$.

- If the output of $H(L,L)$ is accept, then L will loop forever. So $L(L)$ does not halt. But then $H(L,L)$ was wrong — the result should have been reject.

- On the other hand, if $H(L,L)$ is reject, then L will halt. But then $H(L,L)$ is wrong here too — the result should have been accept.

Thus in either case $H(L,L)$ was wrong, which contradicts the assumption that H solves the HALTING PROBLEM. So no such algorithm can exist. □