Remaining Topics

- Taking a step back
- Problem Classes: Classifying problems based on the complexity of algorithms to solve them
- Who wants to be a billionaire?
- What you can’t make computers do…

A Broad Look at Complexity

Levels of algorithmic complexity

	if n=8 and k=2
constant time: $O(1)$	1
less-than-linear: $O(\log\log n)$	approx. 1
less-than-linear: $O(\log n)$	3
linear: $O(n)$	8
more-than-linear: $O(n\log n)$	24
polynomial: $O(n^k)$	64
exponential: $O(k^n)$	256
combinatorial: $O(n!)$	40,320
super-exponential: $O(n^n)$	16777216
...	
undecidable:	unsolvable by algorithms

efficient

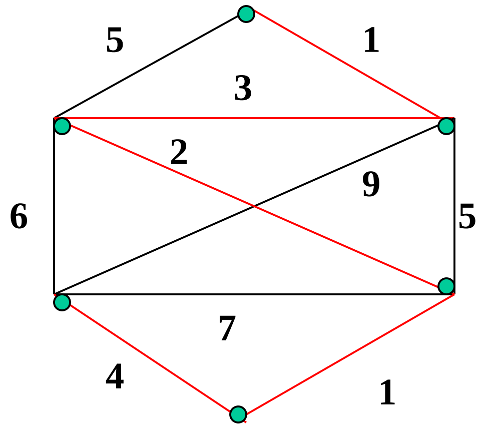
$\updownarrow$

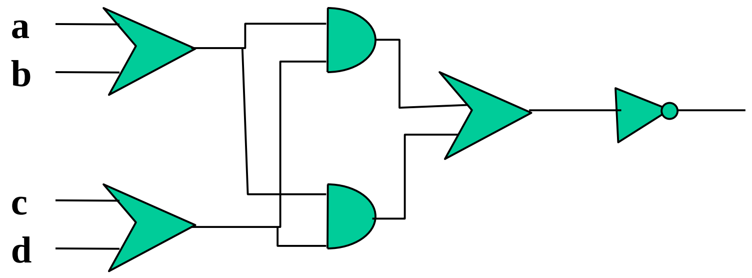
inefficient

Problem Classes

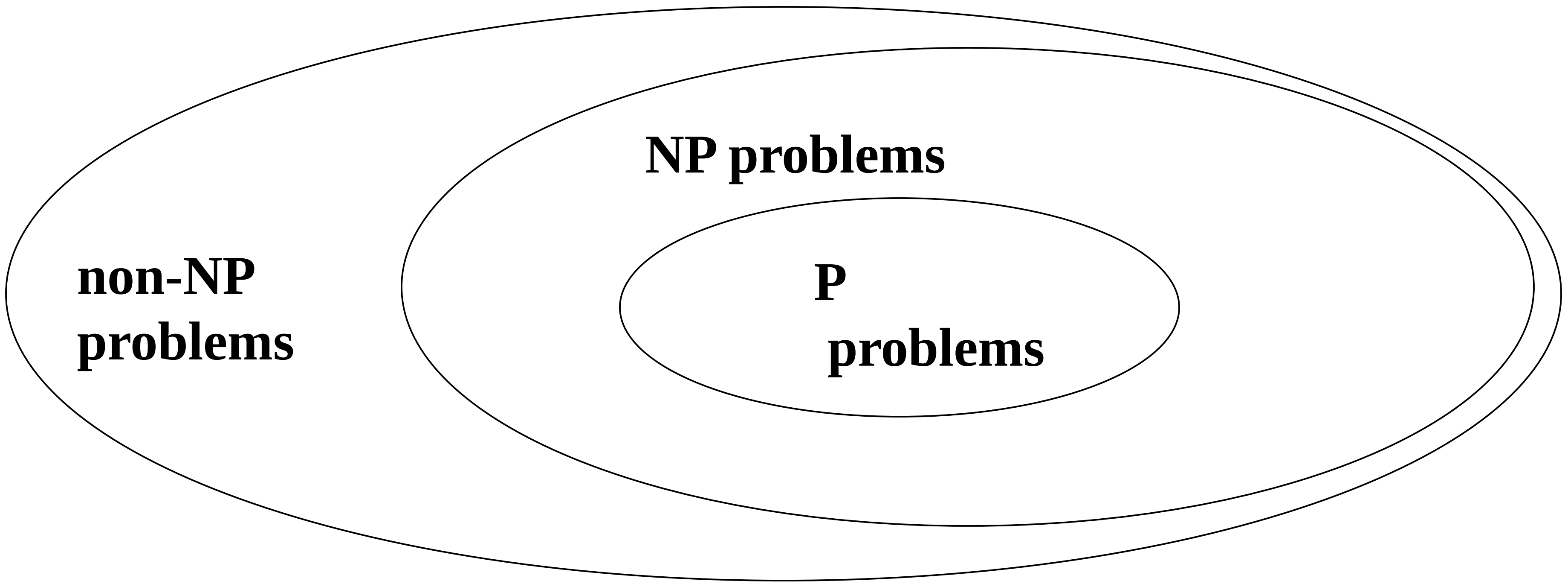
- P (Polynomial)
  - all problems that can be solved by algorithms of polynomial complexity
  - all problems we discussed in this course are in P

- NP (Non-deterministic Polynomial)
  - this includes the problem class P
  - also includes problems to solve which polynomial time or faster algorithms are not known (i.e., all known algorithms to solve these problems have worse than polynomial complexity)
  - but not all problems with worse than polynomial complexity are in NP
  - to be in NP problems must be such that their solutions can be verified by algorithms of polynomial complexity or less
  - e.g. Boolean Circuit Satisfiability Problem





Every P problem is also an NP problem



- Any problem in P is also a problem in NP; so P is a subset of NP

Is every NP problem a P problem?

- i.e., if a problem is in NP but it is not in P, is it possible that it can be solved by a polynomial time or faster algorithm?
- i.e., is NP a subset of P?
- If it was, then the two problem classes will be the same, i.e., P=NP
- Is P=NP? Nobody knows!

P=NP?

- **This is one of the most important open problem in Algortihms**
- If P=NP this means that ALL problems in NP can be solved by efficient algorithms! There are thousands of scientifically and commercially important problems in NP.

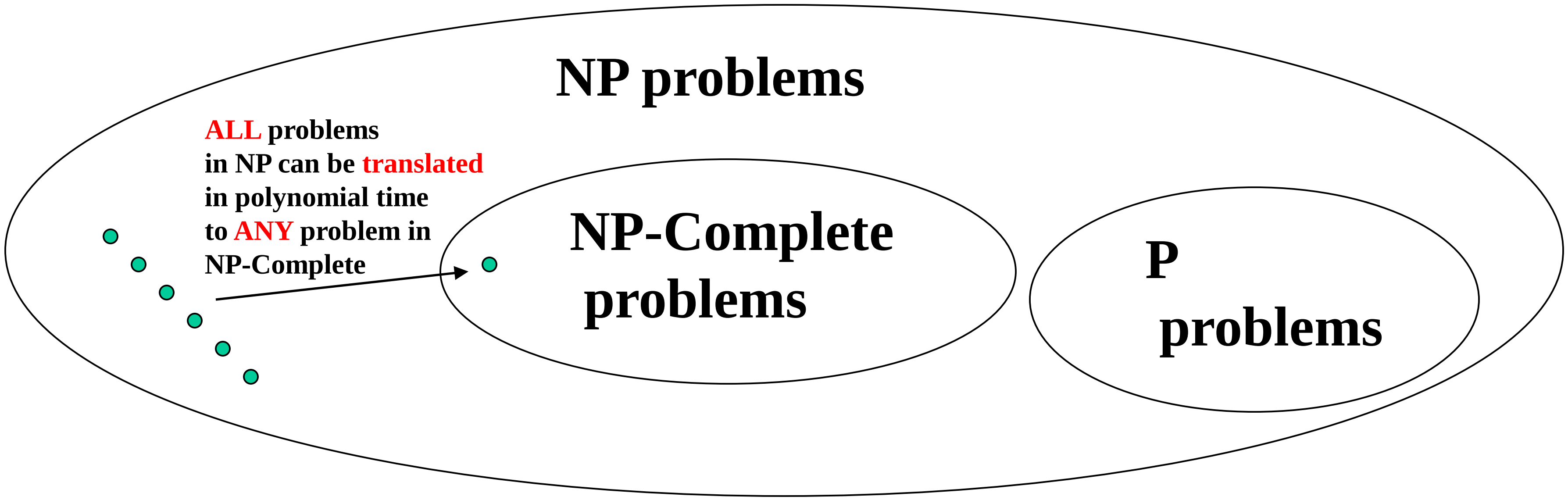
P=NP?

- In the early seventies a UC Berkeley computer scientist called Cook discovered a peculiar property of a problem, the Circuit Satisfiability Problem, known to be in NP
- He discovered that it was possible to translate any instance of ALL problems in NP to instances of the Circuit Satisfiability Problem (so that you can construct a solution to the original NP problem from the solution to this instance of the Satisfiability problem and vice versa), and that this translation will only take polynomial time or less for ANY NP problem.
- Why is this significant?

P=NP?

- An NP-Complete problem is one such that **ALL** problems in NP can be transformed to particular instances of it (so that you can construct a solution to the original NP problem from the solution to this instance of the NP-Complete problem and vice versa), where this translation can be done in polynomial time or less for any NP problem.
- This means that the invention of an efficient (polynomial time or faster) algorithm to solve an NP-Complete problem will provide the means to solve **ALL** NP problems efficiently!
- Satisfiability was the first NP-Complete problem to be discovered.

P=NP?



- Now NP-Complete class of problems contain hundreds of problems (see http://en.wikipedia.org/wiki/List_of_NP-complete_problems)

Two seemingly simple problems

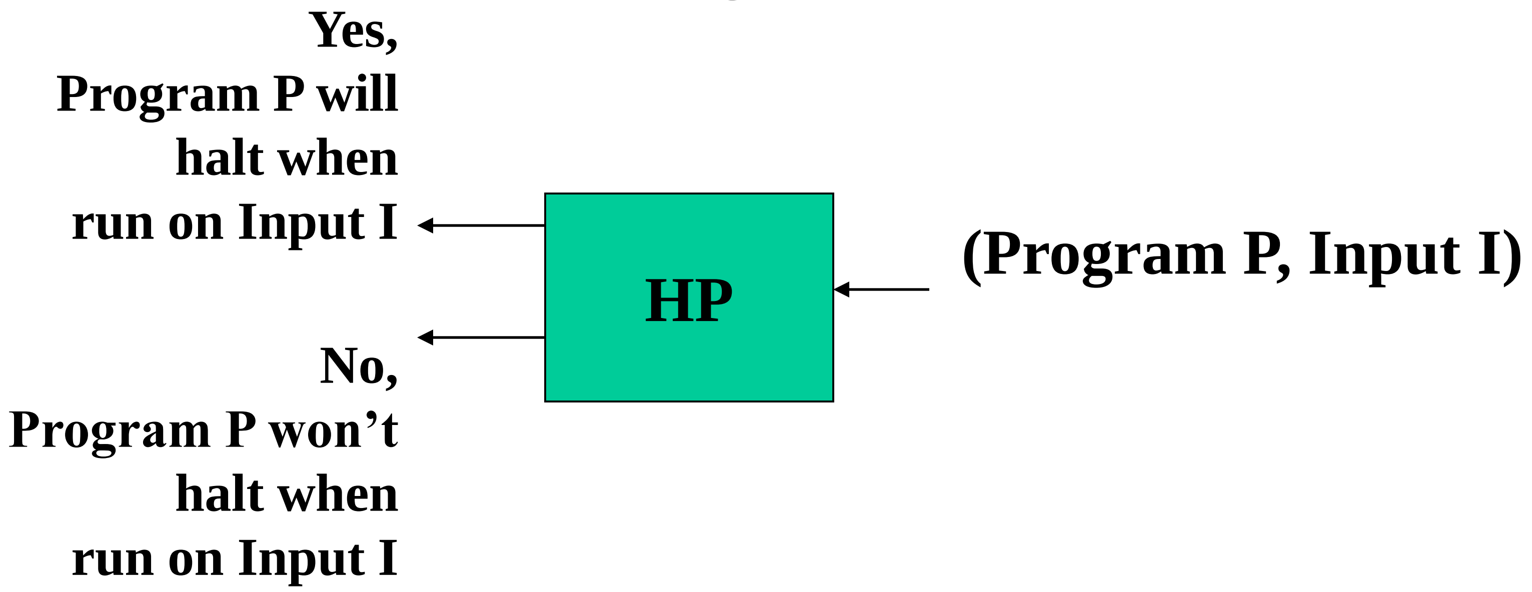
- The Hamiltonian Circuit Problem (HCP) is to find a cycle in a graph that visits every node **exactly once**.
- The Euler Circuit Problem (ECP) is to find a cycle in a graph that visits every edge **exactly once**.
- ECP can be solved in **linear** time using an algorithm that employs DFS, but HCP is **NP-Complete** and all known algorithms to solve it are exponential or worse!

How to be a billionaire?

- Invent a polynomial time deterministic algorithm to solve any one NP-Complete problem (and there are hundreds to choose from)
- Instantly claim the million dollar prize at www.claymath.org/prizeproblems/
- Patent it
- Live on the royalties
- Even if you are this smart, there are problems for which you won’t be able to develop an algorithm.

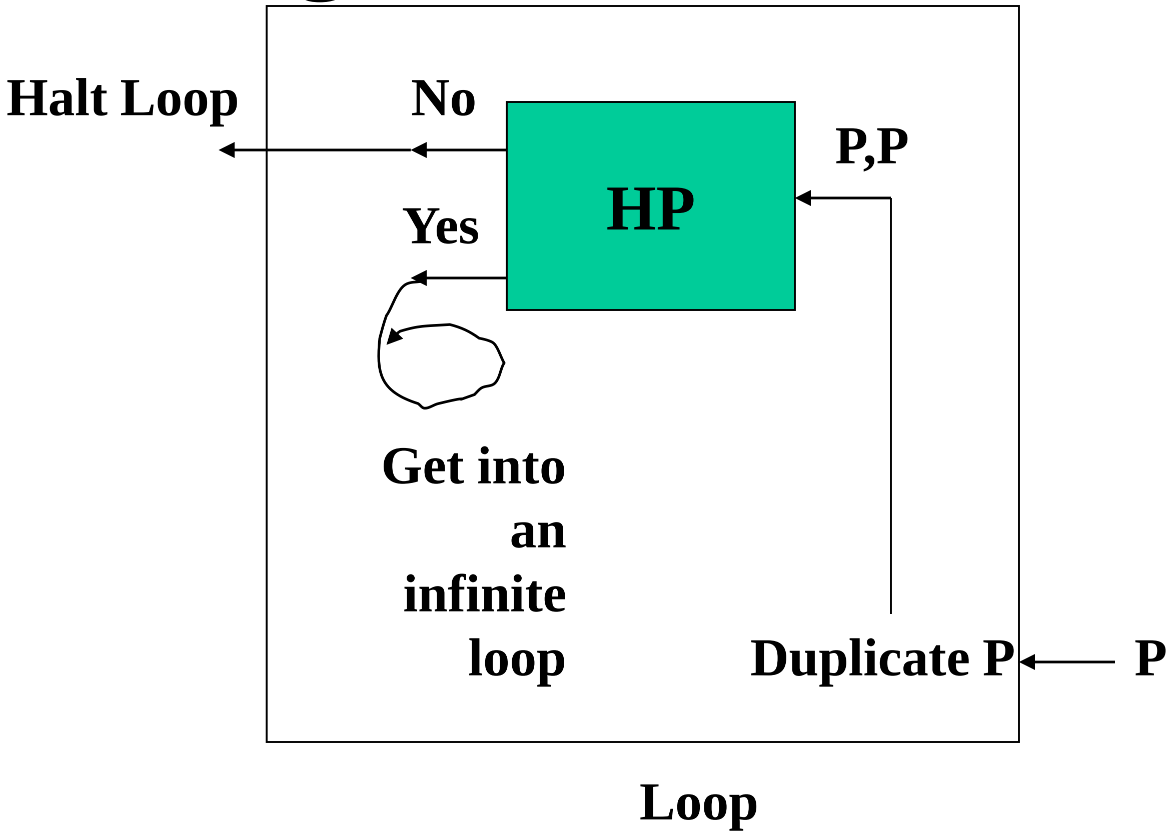
Undecidable Problems

- Example: The Halting Problem



The Halting Problem

```
Loop(P)
output=HP(P,P)
if output=yes then
    infinite loop
else
    exit
end Loop
```



- **What will happen if Loop is called with the code of Loop as the input?**

The Halting Problem

- What we just went through is an elegant proof by contradiction, showing that no algorithm (program) to solve the Halting Problem can exist!
- Such problems unsolvable by computers, for which no algorithm or program can ever be written, are called Undecidable Problems

The End

- This is an appropriate place to end a course on algorithms!