Intractability

Q: What is a general purpose computer?

- Simple model of computation: DFA
- Universal model of computation: Turing machines

Q Which algorithms are useful in practice?

- useful in practice ("efficient") = polynomial time for all inputs

Definition A problem is intractable if it can't be solved in polynomial time

Search Problems

Search Problem = given an instance I of the problem, find a solution S (or report none exists)

Requirement: must be able to efficiently check that S is a solution

P vs NP

Def NP is the class of all search problems

NP = nondeterministic polynomial time

Def P = the class of search problems solvable in polynomial time

Nondeterminism

- Nondeterministic machine can guess the solution

NP = search problems solvable in polynomial time on a nondeterministic Turing Machine

P = search problems solvable in polynomial time in the natural world

Does **P = NP**? Can you always avoid brute force searching and do better?

* overwhelming consensus $P \neq NP$

Classifying Problems

A key problem: satisfiability

SAT: Given a system of boolean equations find a solution

Q: Is it NP?

Conjecture: No polynomial-time algorithm for SAT

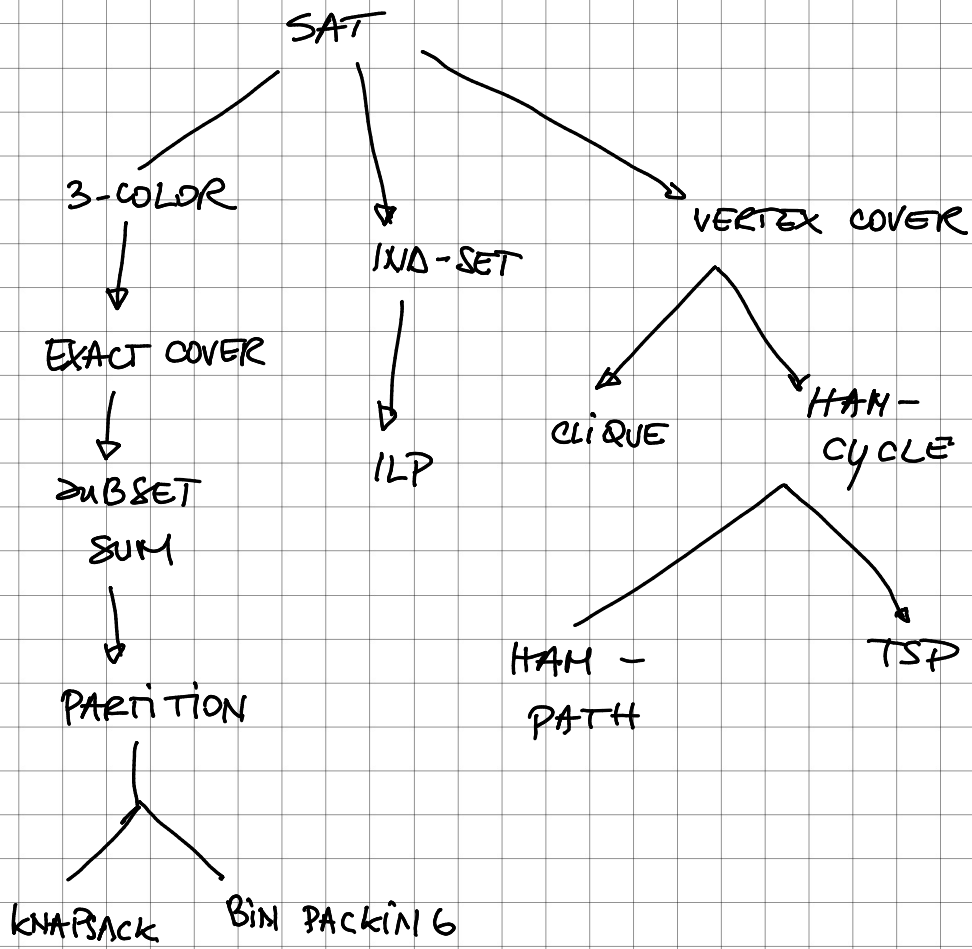
worst case 2^n

! Problem X poly-time reduces to problem Y if X can be solved with:

- polynomial number of computational steps
- polynomial number of calls to Y

=> Consequence: if we can poly-time reduce SAT to problem Y, then we can conclude

that Y is (probably) intractable.



NP - Completeness

Def An NP problem is NP-Complete if all problems in NP poly-time reduce to it

SAT is NP-complete (every NP problem is a SAT problem in disguise)

* All the problems in the diagram are NP-complete; they are manifestations of the same really hard problems

Coping with Intractability

- Exploiting it
  - modern cryptography
- Factor - given an n-bit integer x, find a non-trivial factor
- Relax one of desired features
- Special cases may be tractable
- approximation algorithm

- solve the problem in poly-time: chaff solves real-world SAT instances with ~10k variable

* Most famous NP-complete problem:

  HAMILTON PATH

Goal: Find a simple path that visits every vertex exactly once