NP Problems
P -vs- NP

P = polynomial time

- There are many problems that can be solved correctly using algorithms that run in $O(n^c)$ time for some constant $c$.
- NOTE: We can say that an $n \log n$ algorithm is in $P$ since $n \log n \in O(n^2)$.

NP = non-deterministic polynomial time

- There are also many problems where you can look at a proposed solution to a problem and determine whether it is a valid solution in polynomial time.
- This “proposed solution” is typically called a certificate.
Non-Deterministic?

The “non-deterministic” part of the name “NP” comes from the idea that you could (in theory) run these verification checks on LOTS (all) potential solutions simultaneously on a non-deterministic machine.

We will just say that a problem is in NP if a “certificate” can be verified in polynomial time.
Satisfiability (SAT)

Given a Boolean expression over \( n \) variables in conjunctive normal form, is there a way to assign values to the \( n \) variables that will make the entire expression true?

For example, the following are each a simple example of that type of question on just two variables:

\[
(u \text{ OR } \sim v) \text{ AND } (\sim u \text{ OR } v) \\
\sim u \text{ AND } (u \text{ OR } v) \text{ AND } (u \text{ OR } \sim v)
\]

For the general problem on any type of Boolean expression there is no known polynomial time solution as of now but what about some general sub-cases?
Disjunction = “or’
Conjunction = “and”

We’ll define a clause as a disjunction of some number of Boolean variables.

When a Boolean expression is in “conjunctive normal form” that means that it contains an arbitrary number of clauses joined by conjunction.

Examples:
- \((A \lor C) \land (B \lor C)\)
- \((A \lor B \lor C) \land (A \land \neg C) \land (\neg B \lor C)\)
#-CNF

We’ll say that a Boolean expression is a #-CNF Boolean expression if every disjunctive clause is a disjunction of exactly # items (again, the clauses are joined by conjunction).

From our previous examples the first one of \((A \lor C) \land (B \lor C)\) is 2-CNF but the second one doesn’t have a consistent pattern so wouldn’t be any #-CNF as expressed.
2-SAT

In “2-SAT” you are given a 2-CNF Boolean expression and are asked whether there is a set of assignments to the variables that will cause the entire expression to evaluate to TRUE.

For example, given:

\[- (A ∨ C) ∧ (B ∨ C)\]

one valid solution is A=true, B=true.

Note that there MIGHT be more than one solution; we just care whether there IS a solution in this problem.
Is this satisfiable?

\[(A \lor C) \land \]
\[ (A \lor \sim D) \land \]
\[ (B \lor \sim D) \land \]
\[ (B \lor \sim E) \land \]
\[ (C \lor \sim E) \land \]
\[ (A \lor \sim F) \land \]
\[ (B \lor \sim F) \land \]
\[ (C \lor \sim F) \land \]
\[ (D \lor \sim G) \land \]
\[ (F \lor \sim G) \]

I have \textit{no} idea at a glance...
APT Algorithm

Aspvall, Plass, and Tarjan - 1979

Their idea was to construct a graph where each variable is a vertex and the negation of each variable is a vertex. Now, for each clause (X∨Y) add a pair of directed edges \( \overline{X} \rightarrow Y \) and \( \overline{Y} \rightarrow X \) to the graph.

– (think back to 250 for why this makes sense)

Then, run a strongly connected components algorithm (like the \(|V|+|E|\) one by Tarjan we saw earlier in the semester) on this graph.

We can now say that the 2-CNF expression that was used to construct the graph is satisfiable if and only if no variable and its negation appear in the same strongly connected component.
Let’s start with a small one…

\((\neg A \lor \neg B) \land (\neg A \lor \neg C) \land (A \lor \neg D) \land (\neg B \lor \neg C) \land (B \lor \neg D) \land (B \lor D)\)

Build the graph.
Identify the largest SCCs.
Check for a variable and its negation.

What’s the runtime in terms of the number of Boolean variables?
3-SAT

In “3-SAT” you are given a 3-CNF Boolean expression and are asked whether there is a set of assignments to the variables that will cause the entire expression to evaluate to TRUE.

Question: Do you think there is a similar technique for solving this variation in polynomial time?
3-SAT

In “3-SAT” you are given a 3-CNF Boolean expression and are asked whether there is a set of assignments to the variables that will cause the entire expression to evaluate to TRUE.

There is currently no known polynomial time solution for this problem.

– Why do we care that this is the case?

• In 1971 it was established as an NP-Complete problem by Cook.