- What's NP-completeness mean?
- What's P=NP or P≠NP mean?
- How do we show that a problem is intractable?

*intractable = unlikely to be solved efficiently*

---

\[ P = \text{class of all search problems that are solvable in polynomial time} \]

\[ \text{NP} = \text{class of all search problems (regardless of time required to solve them)} \]

**What's a search problem?**

Roughly, a problem where we can efficiently verify solutions.

So, P=NP or P≠NP addresses whether or not:

\[ \text{Solving a Problem (i.e., constructing a solution)} \quad \text{is as easy as verifying a solution} \]

\[ P \quad ? \quad \text{NP} \]
Formally what is a search problem?

**Search problem:**

Given instance \( I \) (for example, graph \( G \)) we are asked to find a solution if one exists and if no solution exists we output NO.

Further to be a search problem if we are given a solution \( S \) we can verify (check) in time polynomial in \( |I| \) that \( S \) is a solution to \( I \).

In other words, when there is a solution, given such a solution we can check it is a solution efficiently.

If there are NO solutions then we don't need to do anything.
Efficiently means $\text{Poly}(\text{I})$

Thus we need that the solution $|S| \leq \text{Poly}(\text{I})$

& we need to show an algorithm $A$ that takes $I \& S$ as input and in polynomial-time verifies that $S$ is a solution to $I$.

Examples of search Problems:

$k$-coloring: undirected

input: $G = (V, E)$ & integer $k > 0$

output: Assign each vertex a color in $\{1, \ldots, k\}$ so that adjacent vertices get different colors.

and NO if no such $k$-coloring exists for $G$.

Given $G$ & a $k$-coloring in $O(|V| + |E|)$ time we can verify that it is a valid coloring.

Hence, coloring $\in \text{NP}$.
SAT:

input: Boolean formula $f$ in CNF with $n$ variables & $m$ clauses

output: Satisfying assignment if one exists
No otherwise

What's this mean?

Variables $x_1, x_2, \ldots, x_n$ which take values TRUE or FALSE

Literals $x_1, \overline{x_1}, x_2, \overline{x_2}, \ldots, x_n, \overline{x_n}$

$\land = \text{and}$

$\lor = \text{or}$

Clause is the OR of some literals

Example: $(x_2 \lor \overline{x_4} \lor x_1)$

So either $x_2 = T$ or $x_4 = F$ or $x_1 = T$

$f$ in CNF is the AND of $m$ clauses:

Example: $(\overline{x_2}) \land (x_2 \lor x_4 \lor x_1) \land (x_3 \lor x_2 \lor x_1)$

SAT CNF

Why? Given $f$ and an assignment,
in $O(n)$ time per clause we can verify that each clause is satisfied

& hence in $O(nm)$ total time we can verify that the assignment satisfies $f$. ✅
Knapsack:

Input: n objects with
integer weights \( w_1, \ldots, w_n \)
& integer values \( v_1, \ldots, v_n \)
capacity B

Output: subset \( S \) of objects with
total weight \( \leq B \)
& maximum total value.

Given instance \( \Sigma w_i, \Sigma v_i, B \)
& solution \( S \),
we can verify in \( O(n) \) time that the total weight is \( \leq B \)
but how do we verify that it has maximum value?
(need to do it in \( \text{Poly}(n, \log B) \))

Not clear how to do it.

This is an optimization problem.
Look at search version:

\[ \text{input: } w_1, \ldots, w_n, v_1, \ldots, v_n, B \text{ & goal } g. \]

as before

\[ \text{output: } \text{subset } S \text{ with } \]
\[ \text{total weight } \leq B \quad \left( \sum_{i \in S} w_i \leq B \right) \]
\[ \text{& total value } \geq g \quad \left( \sum_{i \in S} v_i \geq g \right) \]
\[ \text{& } \text{NO if no such } S \text{ exists.} \]

Given \( S \), in \( O(n) \) time can check that it
has total weight \( \leq B \)
\& total value \( \geq g. \)

\[ \Rightarrow \text{Knapsack-search } \text{NP?} \]

**Note:** if we can solve the search version in poly-time then we can solve
the optimization version in poly-time
by doing binary search for
\[ \max g \text{ s.t. } \sum_{i=1}^{n} v_i \geq g \]

where \( V = \sum_{i=1}^{n} v_i \).
MST:

- input: G = (V, E) with positive edge lengths
- output: tree T with min weight

MST is a search problem.
Hence, MST \in NP.

Why?
Given G & T we can run Kruskal’s or Prim’s
to verify that T has min weight
& then run BFS or DFS to verify that
T is a tree.

NP stands for non-deterministic polynomial time.
= Problems that can be solved in
poly-time on a non-deterministic machine,
allowed to guess at each step
(there exists a path to the accepting state)
$NP = \text{all search problems}$

$P = \text{search problems that can be solved in poly-time}$

Hence $P \leq NP$

If $P = NP$: $\bigcirc$

means that if we can verify solutions efficiently,

then we can construct solutions efficiently

(e.g., verifying a proof for a theorem

is as hard as constructing a proof)

If $P \neq NP$: then there are some

search problems that can't be

solved in poly-time.

What are these problems?

$NP$-complete problems: hardest problems in $NP$. 
Colorings is NP-complete.

This means:

1. Colorings \(\in\) NP
2. If we can solve colorings in poly-time, then we can solve every problem in NP in poly-time.

Thus if \(P \neq NP\) then there is no poly-time algorithm for colorings.

How to show (b)?

**Reductions:**

Problems A & B (Example: A = MST, B = Colorings)

\[ A \rightarrow B \text{ or } A \leq B \]

Means we can reduce A to B.

\[ A \rightarrow B \text{ means that:} \]

If we can solve B in poly-time, then we can use that algorithm to solve A in poly-time.
So we suppose there is an efficient algorithm for B and we use that algorithm as a black-box to solve A.

In other words:

- take input I for A (that we want to solve)
  1) create input f(I) for B
  2) Run blackbox algorithm for B
  3) Given solution S for f(I) we convert it to get solution h(S) for I
     & given NO for f(I) we output NO for I.
We need to define $f$ & $h$

A prove that if $S$ is a solution to $f(I)$
then $h(S)$ is a solution to $I$

& if NO solution for $f(I)$
then NO solution for $I$

To show: Colorings is NP-complete
we need to show:

a) Colorings $\in$ NP

b) for all $A \in$ NP, $A \Rightarrow$ Colorings

How to do (b) for all $A \in$ NP?
Suppose we know SAT is NP-complete. Hence we know that for every A ∈ NP
A → SAT.

If we show SAT → Colorings
Then A → SAT → Colorings
So A → Colorings.

To show Colorings is NP-complete we need to show:
a) Colorings ∈ NP
b) for a known