HornSAT:

(Special Case of SAT)

SAT (satisfiability):

Input: Boolean variables $x_1, \ldots, x_n \in \{0, 1\}$

Constraints:

$x_1 \land x_3 \Rightarrow x_5$

$x_1 \land \neg x_2$

Goal: Find an assignment $x_i \in \{0, 1\}$

that satisfies all the constraints.

(if 3 one)
HORNSAT:

Input: Boolean variables $X_1, \ldots, X_n$.

(Hosting a party $X_i = 1/0$ invite person $i' \lor \lnot$)

- Singletons: $X_i = 1$ for some $i$

  $X_5 = 1$, $X_7 = 1$, $X_9 = 1$

- Implications: $X_i \land X_j \Rightarrow X_k$

  $X_1 \land X_7 \land X_8 \land X_9 \Rightarrow X_{11}$

( $X_1 \land \lnot X_5 \Rightarrow X_7$)

- Negative Clauses (Complications) $X_i \land X_j \land X_k \land X_8$

Goal: Find a solution/an assignment satisfying all constraints.
Greedy: 1) (Invite as many people as possible) "as many variables to be TRUE"

2) (Invite as few people as possible) "as few variables to be TRUE" as possible

"Set a variable \( X_i = T \) only if it's FORCED"
- Set all $X_i = 0$.
- For each singleton $X_i = 1$
  \[\Rightarrow \text{Set } X_i = 1\]
- While some implication say $X_i \land X_j \Rightarrow X_k$ is not satisfied
  Set the right hand variable to be 1
  \(\text{(net } X_k = 1)\)
- Check if any negative clause is violated
  if violated then output \(\exists \) no solution
  otherwise output the assignment.
Algorithm:

Singletons: \( \{ X_1, X_7, X_9, X_{10} \} \)

Implication:
- \( X_8 \land X_9 \Rightarrow X_4 \)
- \( X_2 \land X_7 \Rightarrow X_8 \)
- \( X_1 \land X_6 \Rightarrow X_5 \)
- \( X_3 \land X_6 \land X_9 \Rightarrow X_{10} \)
- \( X_1 \Rightarrow X_2 \)

Complications:

\( X_4 \land X_{10} \), \( X_4 \land X_6 \), \( X_4 \land X_5 \)

"No solution"
Thm: The set of variables set to TRUE by algorithm are TRUE in every assignment satisfying.

- Only modification to the algo's assignment allowed are changing $X_i: 0 \rightarrow 1$

- If a negative clause is violated, it will continue to be violated after any modification.
\[
\text{Input: } 0, 12, 2, 16, 9, 3, 10
\]

\[
\max \left( L(1), L(2), L(3), L(4), L(5), L(6), L(7) \right)
\]

\[
L(1) = 1
\]

\[
L(2) = \max \text{ over } \left\{ L[j] + 1 \right\} = \max \left( \left( L[1] + 1 \right) \right)
\]

\[
L(3) = \max \text{ over } \left\{ L[j] + 1 \right\} = \max \left\{ L[1] + 1, L[2] + 1, L[3] + 1 \right\}
\]

\[
L(4) = \max \text{ over } \left\{ L[j] + 1 \right\} = \max \left\{ L[1] + 1, L[2] + 1, L[3] + 1, L[4] + 1 \right\}
\]

\[
L(5) = \max \left\{ L(1) + 1, L(3) + 1 \right\}
\]

\[
\text{Note: } a[j] < a[5] = 9
\]
Dynamic Programming

**Longest Increasing Subsequence**

**Input:** n numbers $a[1], a[2], \ldots, a[n]$.

**Goal:** Find the longest increasing subsequence
**Step 1:** Define “subproblems” whose solution would yield the solution we want.

\[ L[i] = \text{length of longest increasing subsequence ending at } a[i]. \]

longest increasing subsequence = \( \max \{ L[i] \} \)

= \max \{ L[1], L[2], \ldots, L[n] \}
**Step 2:** Write a recurrence relation for subproblems.

Let longest increasing sequence ending at $i$. Let $a[j]$ previous number in sequence $j = \{1, \ldots, i-1\}$ such that $a[j] < a[i]$.

Sequence up to $j = \text{longest sequence ending at } j$.

$L[i] = \max_{j < i} \left\{ L[j] + 1 \right\}$
Steps
Order the subproblems (as specified by the recurrence relation)

Alg:

\[ L[i] = 1 \quad \forall \ i \]
for \( i = 1 \) to \( n \)

\[ L[i] = \max \text{ maximum over } \left\{ L[j] + 1 \right\} \]
\( j < i \) \& \( a[j] < a[i] \)

Return \( (\max \{ L[1], \ldots, L[n] \}) \)

\[ j \rightarrow i \]