Analyzing decision algorithms and problems

Outline

1. The Mathematical Method
2. Asymptotic notation and time complexity
3. Problem Complexity Classes: P, NP, PSPACE, EXPTIME

Methods of Learning

(This is not a well-established taxonomy, just Prof. Sunberg's thoughts)

Testimonial (?) Method

1. Read or hear a piece of knowledge
2. Decide whether to believe it
3. Learn by believing it

Scientific Method

1. Formulate a question
2. Formulate a hypothesis
3. Test the hypothesis with an experiment
4. Learn by analyzing the results of the experiment

Mathematical Method

1. Formulate a question
2. Create a conjecture that answers the question
   1. *Define* all concepts
   2. State as a theorem
3. Prove or disprove the conjecture
4. Learn by considering the theorem/false conjecture and the proof

Most RL research fits into the scientific method.

Mathematical Proofs

A mathematical proof *must* (according to Sunberg):

1. Be logically correct
2. Allow each step to be easily verified by a member of the intended audience

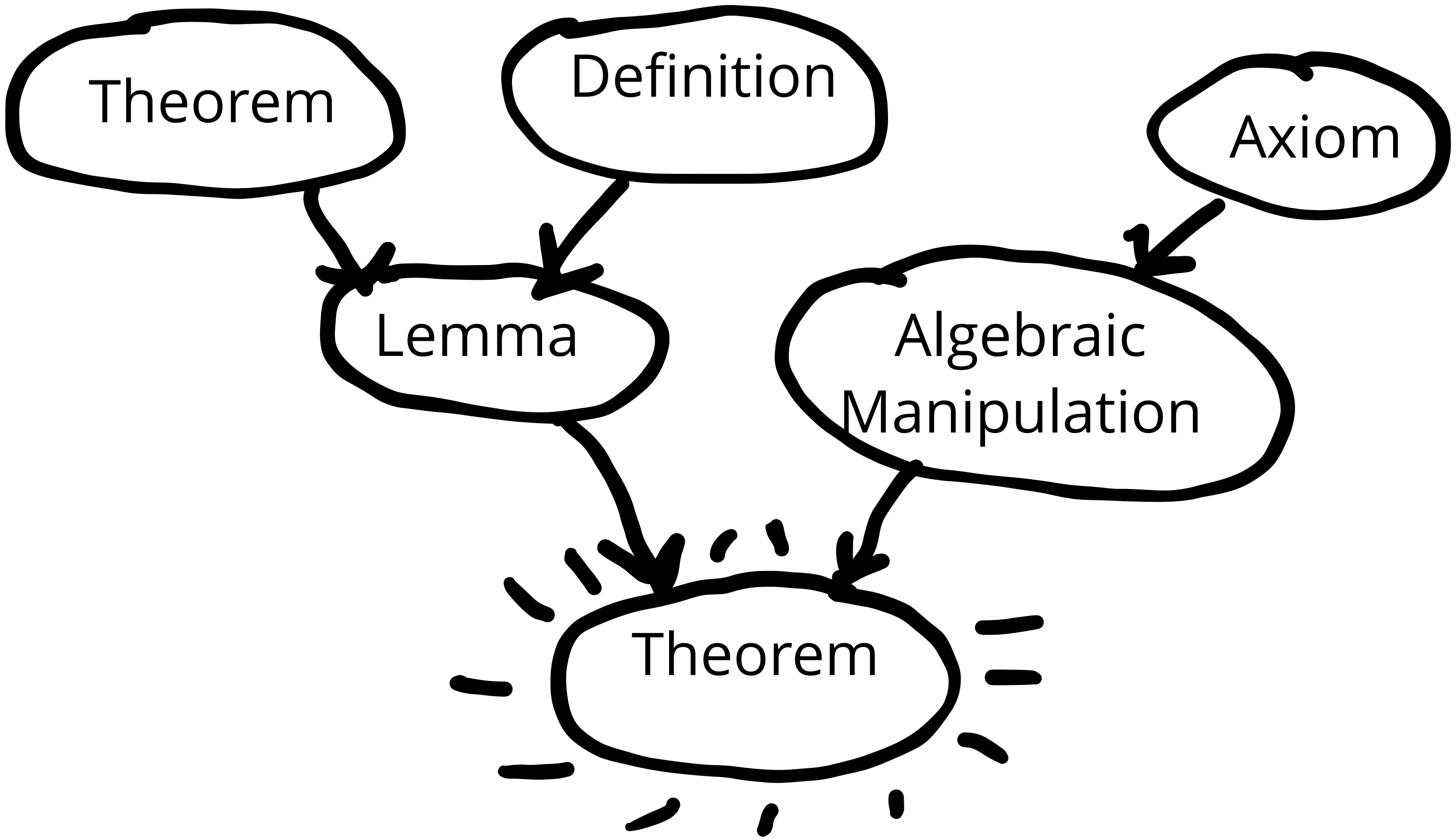
"Nice-to-have"s for theorems, definitions, proofs:

1. General (Definitions and Theorems)
2. Easy for audience to understand
3. Follows patterns the audience is familiar with
4. Composable: allows for construction of other definitions, lemmas, theorems, proofs
5. Answers the question "why" (Proofs)
6. Interesting, engaging, fun for humans

Proof Strategies

1. **Direct proof**
2. **Proof by contradiction**
3. **Proof by induction**
4. Proof by contrapositive
5. Proof by exhaustion

Direct Proof



Direct Proof Example

Definition: An integer x is *even* if there exists an integer a such that $x = 2a$

Theorem: The sum of two even integers is even.

Proof: Let x and y be even integers and z be their sum. By definition, there exist integers a and b such that

$$x = 2a \text{ and}$$

$$y = 2b.$$

Then $z = x + y = 2a + 2b = 2(a + b)$. Since $a + b$ is an integer, z is even.

Proof by Contradiction

To prove proposition P:

1. Assume P is false, i.e. assume $\neg P$
2. Show that $\neg P$ implies two mutually contradictory assertions, Q and $\neg Q$
3. Q and $\neg Q$ cannot both be true (a "contradiction"), so P must be true

Proof by Contradiction Example

Theorem: $\sqrt{2}$ is irrational.

Proof: Suppose $\sqrt{2}$ were rational. Then there exist integers a and b such that $\sqrt{2} = a/b$, **at least one of which is odd** (If a and b were both even, the fraction could be reduced).

Without loss of generality, suppose a is even. Then a^2 is a multiple of 4, and thus $2b^2$ is a multiple of 4. Therefore b^2 is even, and b is also even. **Thus, both a and b are even**. This is a contradiction.

Proof by Induction

To prove $P(n)$ holds for every natural number n:

1. Prove $P(1)$ (the base case)
2. Prove that $P(k) \implies P(k+1)$ (the induction step)

Proof by Induction Example

Theorem: For any natural number n, the sum of n and all natural numbers less than n is $\frac{n^2+n}{2}$

Proof: By induction:

- Base case: The sum of 1 and all natural numbers less than 1 is $1 = \frac{1^2+1}{2}$
- Induction step: If the sum of k and all natural numbers less than k is $\frac{k^2+k}{2}$, then the sum of $k+1$ and all natural numbers less than $k+1$ is $\frac{k^2+k}{2} + k + 1 = \frac{k^2+3k+2}{2} = \frac{(k+1)^2+k+1}{2}$

Algorithm Analysis

Runtime Analysis

$s \leftarrow 0$	c_1	1
for $i \in \{1...n\}$	c_2	n
$\quad s \leftarrow s + i$	c_3	n
end		

$c_1 + n c_2 + n c_3$

$s \leftarrow 0$	c_1	1
for $i \in \{1...n\}$	c_2	n
$\quad$ for $j \in \{1...n\}$	c_3	n^2
$\quad\quad s \leftarrow s + i$	c_4	n^2
$\quad$ end		
end		

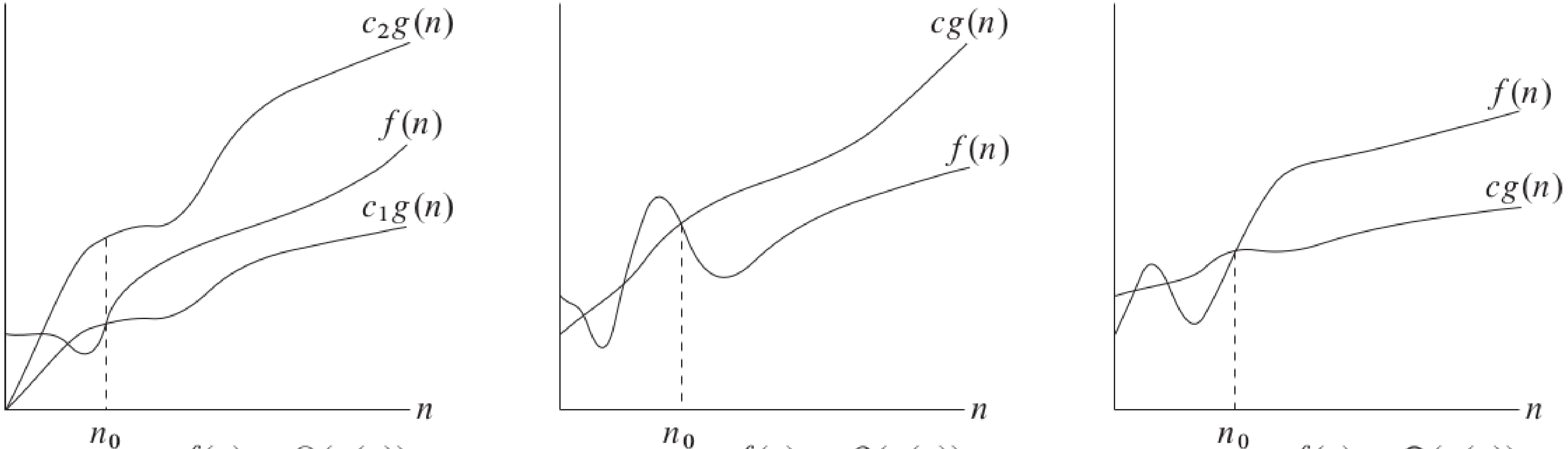
$c_1 + c_2 n + c_3 n^2 + c_4 n^2$

Asymptotic Notation

$$O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le f(n) \le cg(n) \text{ for all } n \ge n_0 \} .$$

$$\Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \le cg(n) \le f(n) \text{ for all } n \ge n_0 \} .$$

$$\Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } 0 \le c_1 g(n) \le f(n) \le c_2 g(n) \text{ for all } n \ge n_0 \} .$$ [1]



$f(n) = \Theta(g(n))$ $f(n) = O(g(n))$ $f(n) = \Omega(g(n))$

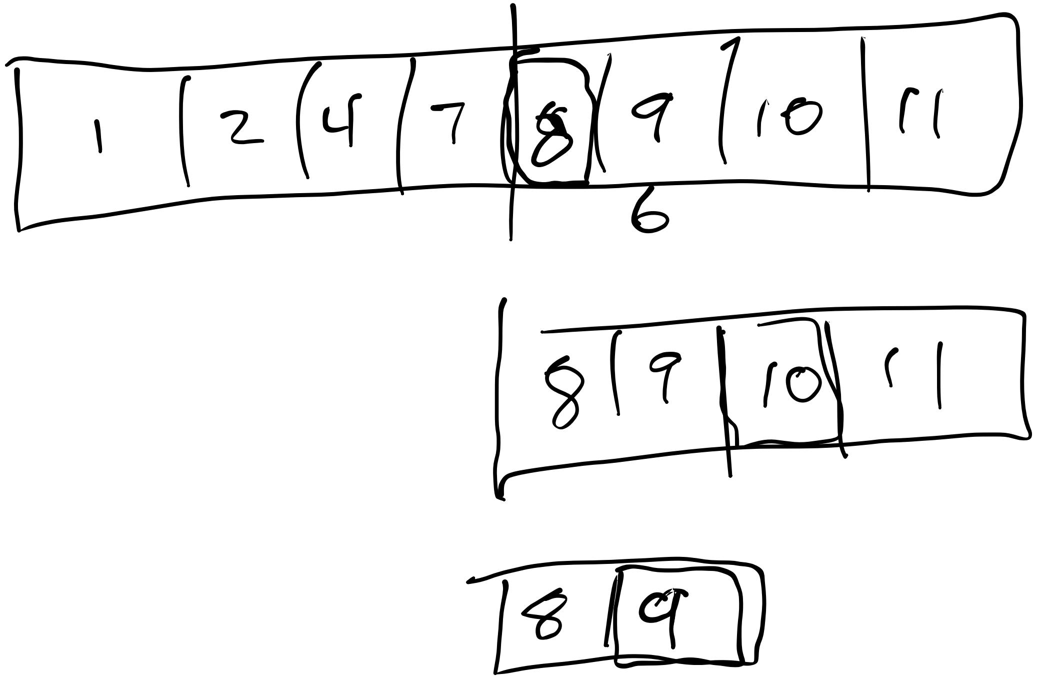
Runtime Analysis

function search(A, x)	$T(n)$
mid $\leftarrow \lceil$ |A|/2 $\rceil$	c_1
if x = A[mid]	
return mid	c_2
elseif x < A[mid]	
return search(A[1:mid-1], x)	$T(\lfloor n/2 \rfloor)$
else	
return search(A[mid+1:|A|], x)	$T(\lfloor n/2 \rfloor)$
end	
end	

$$T(n) = \begin{cases} c_1 + c_2 & \text{if x} = \text{A[mid]} \\ c_1 + T(\lfloor n/2 \rfloor) & \text{otherwise} \end{cases}$$

Assume $T(n) = O(\log(n))$ and prove via induction.

$x = 9$



$O(1)$

$O(\log n)$

$O(n)$

$O(n^2)$

$O(n^3)$

$O(n^k)$

larger

$O(k^n)$

Tractability

What definition should we use to separate tractable from intractable problems?

$O(1)$

$O(\log(n))$

$O(n)$

$O(n^2)$

$O(n^k)$

$O(k^n)$

Table 2.1 The running times (rounded up) of different algorithms on inputs of increasing size, for a processor performing a million high-level instructions per second. In cases where the running time exceeds 10^{25} years, we simply record the algorithm as taking a very long time.

	n	$n \log_2 n$	n^2	n^3	1.5^n	2^n	$n!$
$n = 10$	<1 sec	<1 sec	<1 sec	<1 sec	<1 sec	<1 sec	4 sec
$n = 30$	<1 sec	<1 sec	<1 sec	<1 sec	<1 sec	18 min	10^{25} years
$n = 50$	<1 sec	<1 sec	<1 sec	<1 sec	11 min	36 years	very long
$n = 100$	<1 sec	<1 sec	<1 sec	1 sec	12,892 years	10^{17} years	very long
$n = 1{,}000$	<1 sec	<1 sec	1 sec	18 min	very long	very long	very long
$n = 10{,}000$	<1 sec	<1 sec	2 min	12 days	very long	very long	very long
$n = 100{,}000$	<1 sec	2 sec	3 hours	32 years	very long	very long	very long
$n = 1{,}000{,}000$	1 sec	20 sec	12 days	31,710 years	very long	very long	very long

We say that a problem is *tractable* if it can be solved in $O(n^k)$ for some k.

Exercise: Complexity of Value Iteration for fixed horizon

Fixed Horizon Case:

for $t \in \{T, T-1, ..., 0\}$ $|T|$

for $s \in \mathcal{S}$ $|S|$

$V_t(s) = \max_{a \in \mathcal{A}}(R(s,a) + \gamma \sum_{s' \in \mathcal{S}} T(s' \mid s,a) V_{t+1}(s'))$ $|A||S|$

end

end

$O(T|S|^2|A|)$

Complexity Theory

Which problems are hard?

Complexity Classes

Optimization Problem

Example: Find an optimal policy for the Tiger POMDP

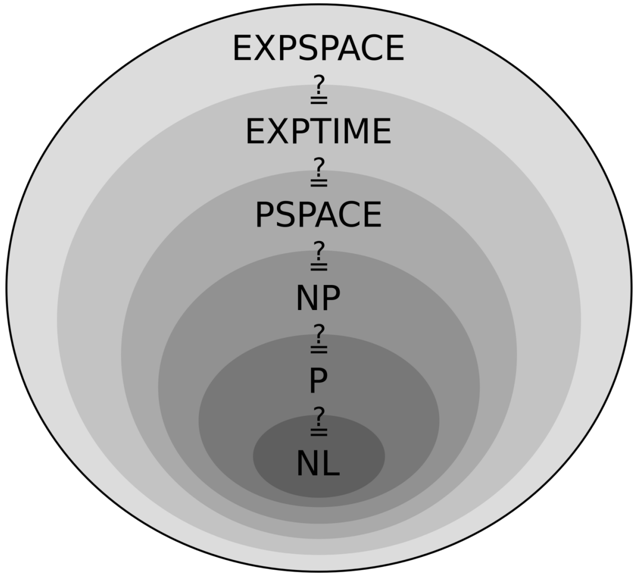
Decision Problem: answer in $\{0, 1\}$

Example: Is "listen" an optimal first action for the Tiger POMDP

- Abstract problem: A mapping from every *instance* in a set to a solution
  - Example abstract problem: Is a an optimal first action in POMDP $(S, A, O, R, T, Z, \gamma, b_0)$?
  - Example instance: Is "listen" an optimal first action for the Tiger POMDP?

Complexity Classes

- P: Any instance can be solved in polynomial time in the size of the input
  - Example: Finite Horizon MDPs
- EXPTIME: Any instance can be solved in exponential time in the size of the input



The NP Complexity Class

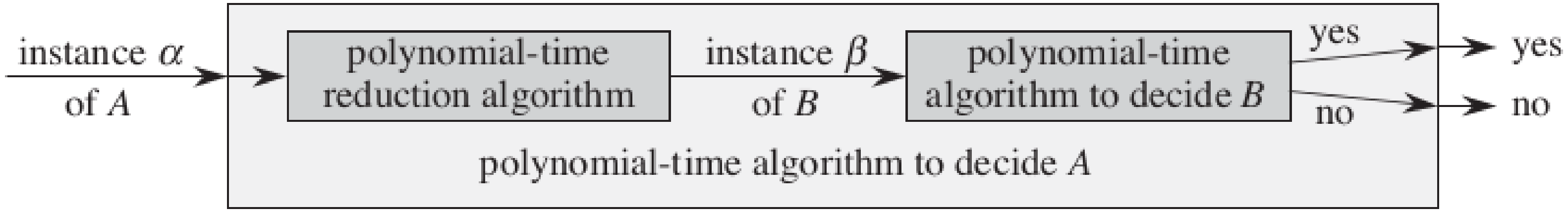
- "Nondeterministic Polynomial"
- Solution can be *verified* in polynomial time
- Is P $\subseteq$ NP?
- Example of a problem that is in NP but not P?

$$P \subset NP$$

(We think - this is one of the biggest unsolved problems in CS)

Polynomial Reduction

How can we prove that problem B is at least as hard as problem A?



A$\leq_P$B

A is no harder than B

NP Completeness

Problem A is said to be *NP-complete* if A$\in$ NP and

$$B \leq_P A \quad \forall B \in NP.$$

Example: 3-SAT:

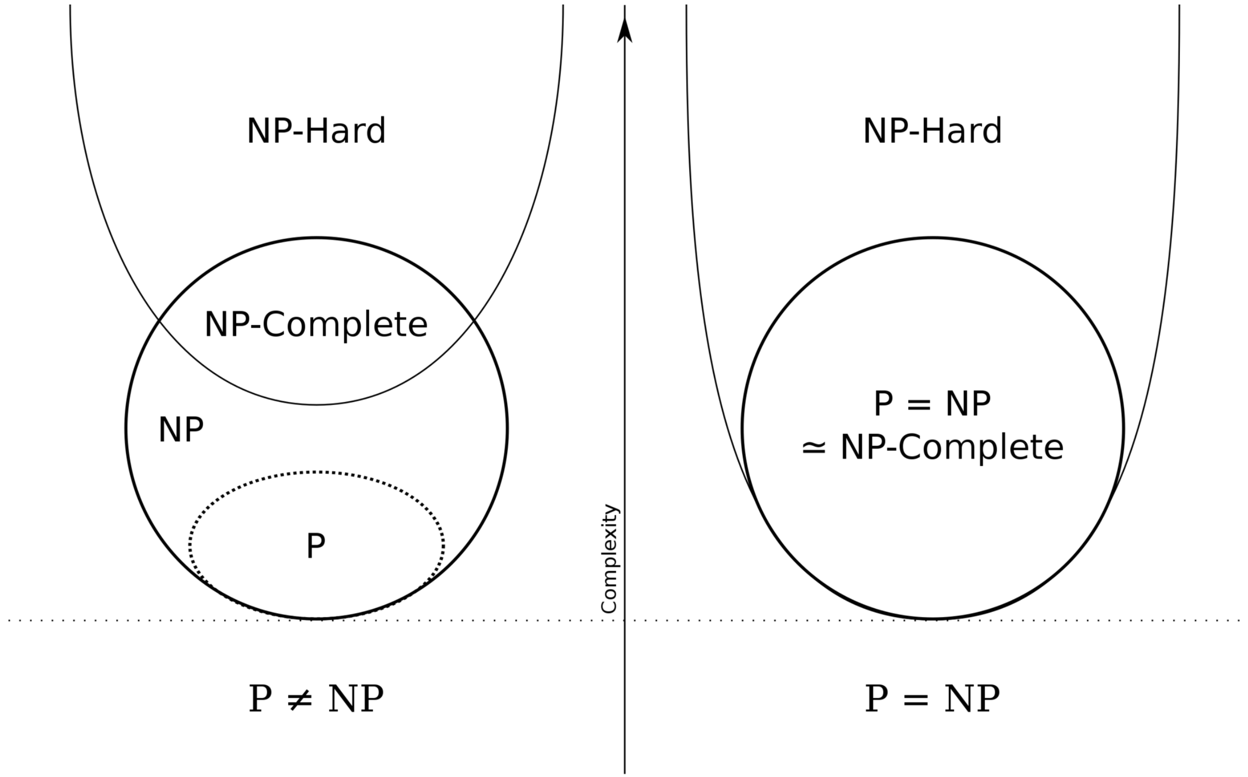
Given a set of clauses, $C_1, \ldots, C_k$, each of length 3, over a set of binary variables $X = \{x_1, \ldots, x_n\}$, does there exist a satisfying truth assignment?

$$(x_1 \vee x_3 \vee x_2) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3)$$

- 3-SAT is NP-complete (Cook, 1971)
- If 3-SAT can be reduced to A $\in$ NP, A is also NP-complete!

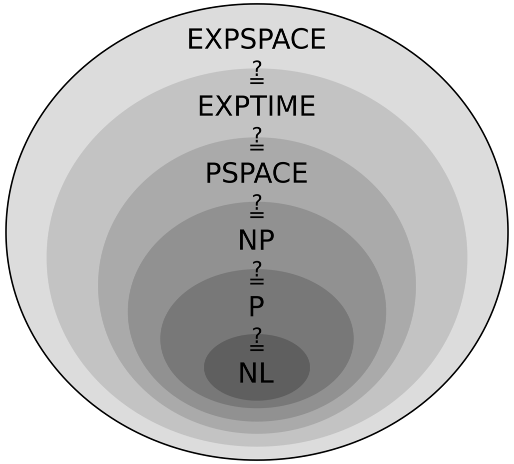
NP-Hardness

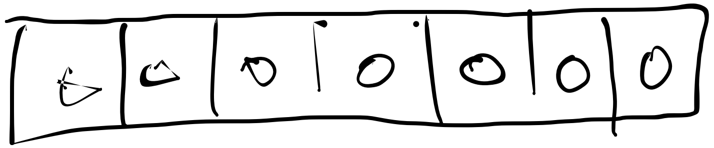
- NP-hard: at least as hard as all of the problems in NP
- NP-complete: NP-hard and in NP



PSPACE Complexity Class

- PSPACE: Set of all problems that can be solved using polynomial *space*
- Observation 1: P $\subseteq$ PSPACE
- Observation 2: There are algorithms that use exponential time, but only polynomial space
- Observation 3: There is an algorithm that solves 3-SAT using polynomial space, therefore NP $\subseteq$ PSPACE
- NP $\subset$ PSPACE ???





0 0 0 0 0 0 1

0 0 0 0 0 1 0

PSPACE-Complete Problems

QSAT: Let $\Phi(x_1, \ldots, x_n)$ be a 3-SAT expression with odd n, then

$$\exists x_1 \forall x_2 \cdots \exists x_{n-2} \forall x_n \Phi(x_1, \ldots, x_n)?$$

(for comparison 3-SAT asked $\exists x_1 \exists x_2 \cdots \exists x_{n-2} \exists x_n \Phi(x_1, \ldots, x_n)?$)

- QSAT is PSPACE-complete (Stockmeyer and Meyer, 1973)
- If QSAT can be reduced to A, A is PSPACE-hard