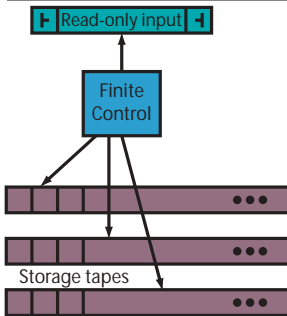


CS20a: Complexity (Nov 19, 2002)

- Complexity definitions
 - Space and time bounded TMs
 - Asymptotic complexity
 - Complexity classes



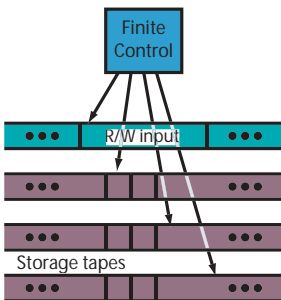
Space-bounded TMs



- Input tape is read-only
- The number of storage tapes is an arbitrary constant k
- M is $DSPACE(T(n))$ if:
 - M is deterministic
 - For any input of length n , M scans at most $T(n)$ cells on any storage tape
- $NSPACE(T(n))$ is the nondeterministic bound



Time-bounded TMs



- All tapes are 2-way infinite
- M is $DTIME(T(n))$ if
 - M is deterministic
 - For any input of length n , M takes at most $T(n)$ steps
- M is $NTIME(T(n))$
 - Nondeterministic case



Asymptotic complexity: big-Oh notation

Let $f, g : \mathbb{N} \rightarrow \mathbb{N}$

- f is $O(g)$ if

$$\exists c \in \mathbb{N}. \forall n \in \mathbb{N}. f(n) \leq c \cdot g(n)$$

\forall means "for all but finitely many." Asymptotically, f grows no faster than g within a constant multiple.



Asymptotic complexity: little-oh notation

- f is $o(g)$ if

$$\forall c \in \mathbb{N}. \exists n \in \mathbb{N}. f(n) \leq g(n)/c$$

This means that f grows strictly more slowly than any constant multiple of g . For example n^{10751} is $o(2^n)$.



Asymptotic complexity: upper bounds

- f is $\Theta(g)$ if g is $O(f)$

$$\exists c \in \mathbb{N}. \forall n \in \mathbb{N}. f(n) \geq g(n)/c$$

- f is $\Theta(g)$ is f is both $O(g)$ and $\Omega(g)$
- Always use O and o for upper bounds, and Ω for lower bounds. Never use O for lower bounds!



Some examples

- $21x^2 + 17x$ is $O(x^2)$
- $x^2 + \log(x)$ is $O(x^2)$
- For any constant c , c is $O(1)$
- 2017^x is $O(2^x)$
- 2017^x is $\Theta(x^{31})$
- 2^x is $o(2^{x^2})$



Space-bounded classes

Suppose M is space-bounded with bound $T(n)$

Complexity of $T(n)$	D/ND	Name of class
$O(\log(n))$	*	LOGSPACE
$O(n^c)$	*	PSPACE
$O(2^n)$	*	EXSPACE



Time-bounded classes

Suppose M is time-bounded with bound $T(n)$

Complexity of $T(n)$	D/ND	Name of class
$O(1)$	D/ND	constant time
$O(\log(n))$	D	DLOGTIME
$O(n^c)$	D	P
$O(n^c)$	ND	NP
$O(2^n)$	D	EXPTIME
$O(2^n)$	ND	NEXPTIME



Complexity hierarchy

- There are many more classes...

$$\text{LOGSPACE} \subset P \stackrel{?}{\subseteq} NP \stackrel{?}{\subseteq} \text{PSPACE} \subset \text{EXPTIME}$$

- Why choose this hierarchy?
 - It is remarkably robust across machine models
 - Turing machines
 - RAM models (normal machines)
 - Lambda-calculus



Outline

- Next few classes
 - NP problems
 - Graph theory
 - NP-completeness and Cook's theorem
- Later
 - P and P-completeness
 - P vs. NP
 - PSPACE
 - Alternating Turing Machines



P-time problems

- (Reminder) A *decision problem* is a problem with a yes/no answer
- A problem is *polynomial time* (the problem is in P) if it can be decided by a TM taking $O(n^c)$ steps
 - n is the length of the input
 - c is a constant
- How long does it take to decide a problem in P?
 - It takes $O(n^c)$ steps (run the program; simulate the TM)



NP problems

- A problem is *in NP* if it can be decided yes/no by a nondeterministic TM in $O(n^c)$ steps
 - n is the length of the input
 - c is a constant
- How long does it take to decide a problem in NP?
 - It takes $O(n^c)$ time to explore each possible nondeterministic choice
 - There are an exponential number of choices
 - So it takes $O(2^n)$ time worst case
- How long does it take to verify an NP execution?
 - There are $O(n^c)$ steps of size $O(n^c)$
 - So it takes $O(n^{2c})$ time

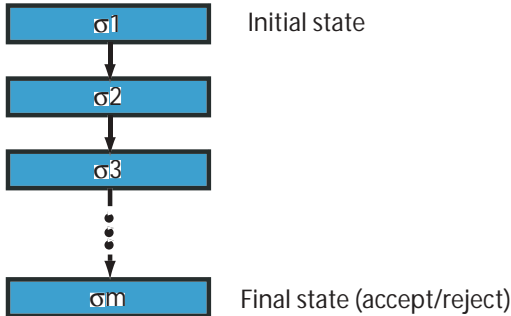


Executions

- An instantaneous description $ID(\sigma)$ is $\alpha_1 q \alpha_2$, where
 - q is the current state of the TM,
 - $\alpha_1 \alpha_2 \in \Gamma^*$ is the contents of the tape (to the last non-blank symbol)
 - The current symbol is the first symbol of α_2
- A PTIME execution is
 - A sequence $\sigma_1 \sigma_2 \dots \sigma_m$,
 - where σ_1 has the form $q_0 \alpha$ and $|\alpha| = n$
 - and $\sigma_i \rightarrow \sigma_{i+1}$
 - and m is $O(n^c)$



PTIME executions



CNF satisfiability

- A propositional formula is in *conjunctive normal form* (CNF) if it is a conjunction of disjunctions of literals.
- A *literal* is a propositional letter, or the negation of a propositional letter.
- Every propositional formula can be represented in CNF.

$$\begin{aligned}
 & (A \vee \neg B \vee C) \\
 \wedge & (D \vee \neg B \vee \neg C) \\
 \wedge & (\neg A \vee B \vee E \vee \neg F) \\
 & \vdots \\
 \wedge & (E \vee F \vee \neg D \vee A \vee \neg B)
 \end{aligned}$$



Satisfiability algorithm

- CNF should make the decision problem easier, since there is only conjunction, disjunction, negation
- An algorithm in NP
 - *Guess a truth assignment*
 - *Verify the assignment*
- An algorithm in P
 - *Must be deterministic (no "guessing")*
 - *Just figure out if the formula is true*

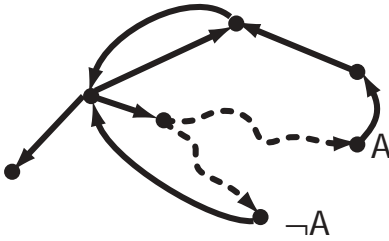


Building an algorithm for 2SAT

- 2SAT: each clause has at most two literals $A_1, \neg A_2, (A_3 \vee A_4), (A_5 \vee \neg A_6), \dots$
- **Algorithm**
 - Consider a clause $(l \vee l')$
 - Rewrite the clauses as implications $(\neg l \Rightarrow l') \wedge (l \Rightarrow l')$
 - Draw a graph
 - The formula is unsatisfiable iff there are two paths
 - * $l_1 \Rightarrow \dots \Rightarrow A$
 - * $l_1 \Rightarrow \dots \Rightarrow \neg A$



2SAT graph





Building a general algorithm for kSAT

Theorem (Resolution) For any formulas C, D, E and any variable x not in $C, D,$ or E , the formula

$$1. (x \vee C) \wedge (\neg x \vee D) \wedge E$$

is satisfiable iff the formula

$$2. (C \vee D) \wedge E$$

is satisfiable.



Proof of resolution

Proof (of resolution)

- Since one of x or $\neg x$ is false, one of C or D is true, so

$$((x \vee C) \wedge (\neg x \vee D) \wedge E) \Rightarrow ((C \vee D) \wedge E)$$

- If 2, then C or D is true. If C , let $x = \perp$; otherwise $x = \top$. In either case, 1 is true, so

$$((C \vee D) \wedge E) \Rightarrow ((x \vee C) \wedge (\neg x \vee D) \wedge E)$$



Simplifying clauses

- Let \mathcal{B} be the formula in CNF
- Suppose it has a clause of the form

$$(l_1 \vee l_2 \vee \dots \vee l_{m-1} \vee l_m)$$

- Let x_1, x_2, \dots, x_{m-3} be new variables
- Replace the clause with

$$\begin{aligned} & (l_1 \vee l_2 \vee x_1) \\ \wedge & (\neg x_1 \vee l_2 \vee x_2) \\ & \vdots \\ \wedge & (\neg x_{m-4} \vee l_{m-2} \vee x_{m-3}) \\ \wedge & (\neg x_{m-3} \vee l_{m-1} \vee l_m) \end{aligned}$$



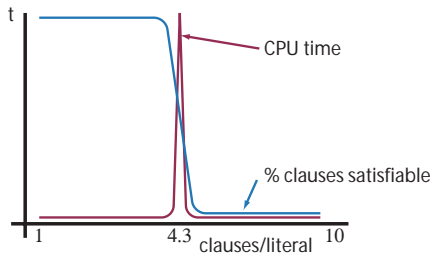
kSAT → 3SAT

- Satisfiability follows from several applications of resolution
- So, we can reduce kSAT to 3SAT
 - The formula rewriting takes polynomial time
- Now, how do we solve 3SAT?
 - As far as we know, guess+verify is the only algorithm that is known to work in all cases



Satisfiability phase transition (Selman)

- Using modern algorithms, plot how long it takes to solve a SAT formula as ratio of clauses/literals



Some additional NP problems

Knapsack problem Given a finite set S , integer weight functions $w : S \rightarrow \mathbb{N}$, benefit function $b : S \rightarrow \mathbb{N}$, weight limit $W \in \mathbb{N}$, and desired benefit $B \in \mathbb{N}$, determine whether there is a subset $S' \subseteq S$ such that

$$\begin{aligned} \sum_{a \in S'} w(a) &\leq W \\ \sum_{a \in S'} b(a) &\geq B \end{aligned}$$



Subset Sum

Subset sum Given a finite set S , integer weight function $w : S \rightarrow \mathbb{N}$, and target $B \in \mathbb{N}$, does there exist a subset $S' \subseteq S$ such that

$$\sum_{a \in S'} w(a) = B$$

Partition Given a finite set S and integer weight function $w : S \rightarrow \mathbb{N}$, does there exist a subset $S' \subseteq S$ such that

$$\sum_{a \in S'} w(a) = \sum_{a \in S - S'} w(a)$$



Reductions

- **Partition** \rightarrow **SubsetSum** by taking $B = \frac{1}{2} \sum_{a \in S} w(a)$
- **SubsetSum** \rightarrow **Partition**

- Introduce two new elements of weight $N - B$ and weight $N - (\Sigma - B)$, where N is large

$$\Sigma = \sum_{a \in S} w(a)$$

- Ask whether the new set can be partitioned into two sets of equal weight.

- **Partition** \rightarrow **Knapsack** take $b = w$ and $W = B = \frac{1}{2} \Sigma$



NP reductions

- Reductions are a fundamental tool
 - If A reduces to B , then if B can be solved efficiently, so can A
- Next up
 - Graph theory
 - NP problems in graph theory
 - Ptime reductions
 - NP-completeness