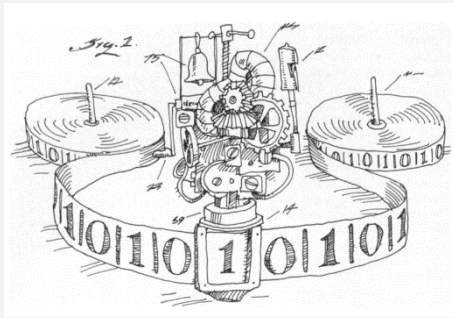
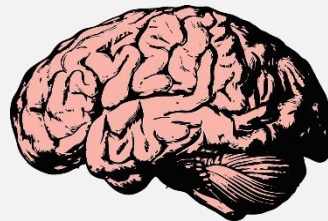
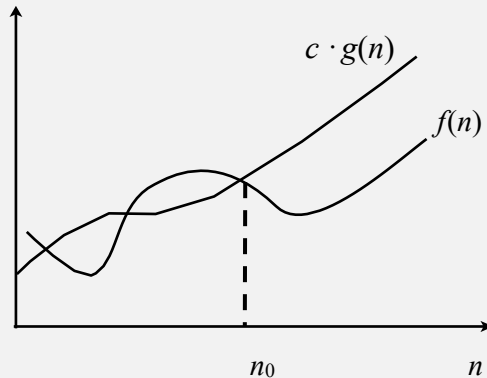


COMP 3602

Intro to Theory of Computation

Lecture 22



Announcements



- Complexity drill due Mar 20
- Two more tests left:
 - Oral exam on NP-completeness: Mon Mar 23
 - Practice problems are posted
 - Sign up by Friday before
 - Written exam on all the material we covered in the course.
 - Fri Mar 27th In class
- Presentations in class:
 - Mar 30 and Apr 1, please sign up for your preferred day
 - All slides due Sun Mar 29th

1. Show that NP is closed under Kleene star, by showing that if L is in NP, then L^* is in NP as well, where every string in L^* is a concatenation of 0 or more strings from L , that is, $L^* = \{ x \mid x = x_1 \dots x_k \text{ for some } k \geq 0 \text{ and strings } x_1, \dots, x_k \text{ such that } \forall i \leq k \ x_i \in L \}$. Use the quantifier/verifier definition of NP.

2. Prove that the problems below are NP-complete. You can assume that the following problems are NP-complete:

3SAT, SubsetSum, Partition, HamPath (LongestPath with $k=|V|$), IndSet and Clique.

- a. DoubleSAT = $\{ \langle \phi \rangle \mid \phi \text{ has at least two satisfying assignments} \}$.
- b. SixOccurrenceSAT = $\{ \langle \phi \rangle \mid \phi \text{ is a satisfiable 3CNF with at most 6 occurrences of every variable} \}$
- c. kClique-kIndSet = $\{ \langle G, k \rangle \mid G \text{ contains both a clique of size } k \text{ and an independent set of size } k \}$
- d. Half-Clique = $\{ \langle G \rangle \mid G \text{ has a clique of size } \lceil |V|/2 \rceil \}$
- e. HamCycle = $\{ \langle G \rangle \mid G \text{ contains a cycle which goes through all vertices exactly once} \}$
- f. FourPartition = $\{ \langle (a_1, \dots, a_n) \rangle \mid \forall i \leq n, a_i \geq 0, \text{ and } 1 \dots n \text{ can be partitioned into four disjoint sets } S_1, S_2, S_3 \text{ and } S_4 \text{ such that } \sum_{i \in S_1} a_i = \sum_{i \in S_2} a_i = \sum_{i \in S_3} a_i = \sum_{i \in S_4} a_i \text{ and } S_1 \cup S_2 \cup S_3 \cup S_4 = \{1, \dots, n\} \}$

3. Show that if $P=NP$, then every language in P other than $A = \emptyset$ and $A = \Sigma^*$ is NP-complete with respect to polytime reductions.
4. Show that Unary-SubsetSum, a variant of SubsetSum where all numbers are given in unary, is solvable in polynomial time.

Questions/comments?

Proving NP-completeness: template.

Goal: show that a language L' is NP-complete.

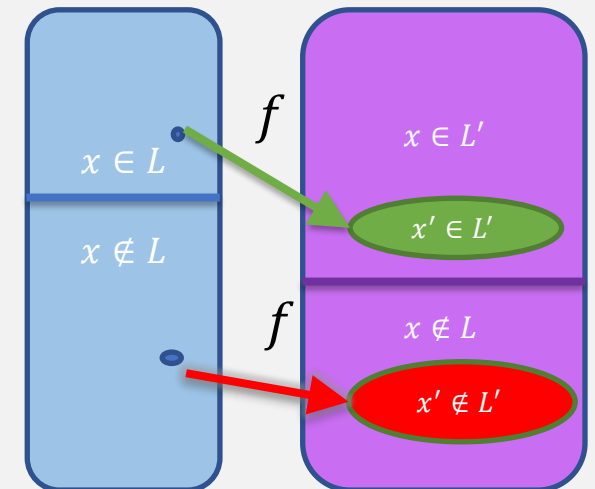
1. Show that $L' \in NP$

- Explain what potential solutions are and how to check them efficiently.
- That is, explain what y and R are in $L = \{x | \exists y |y| < poly(|x|) \wedge R(x, y)\}$,
- and why y is short and R is polytime computable.

2. Show that L' is NP-hard

- Pick a problem L we have showed to be NP-complete already.
- Show $L \leq_p L'$

- Construct a polytime reduction f from L to L' ,
 - that is, describe how to get $x' = f(x)$ for each x
 - » **Important: f cannot determine whether $x \in L$!**
- Show correctness: two directions.
 - $\Rightarrow$ Given a solution y to x , construct a solution y' to $f(x)$
 - $\Leftarrow$ Given a solution y' to $f(x)$, reconstruct a solution y to x
 - » **This applies only to $x' = f(x)$, not arbitrary element of L' !**
- Explain briefly why f is polynomial-time computable





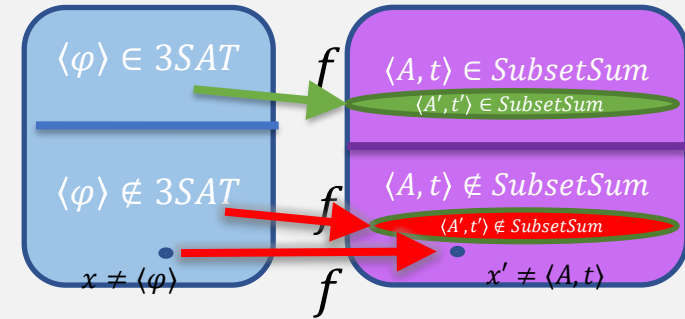
SubsetSum and friends

- Assume $\mathbb{N}$ starts with 0. All numbers in the following are written in binary.
- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $\text{Partition} = \{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$
- Claim: SubsetSum and Partition in NP.
 - Candidate solutions are the respective sets S , which are of polynomial size.
 - Taking sums and checking equalities and inequalities can be done in polynomial time.
- To prove NP-hardness, we will show $3SAT \leq_p \text{SubsetSum} \leq_p \text{Partition}$
 - We will do the easier reduction $\text{SubsetSum} \leq_p \text{Partition}$ first

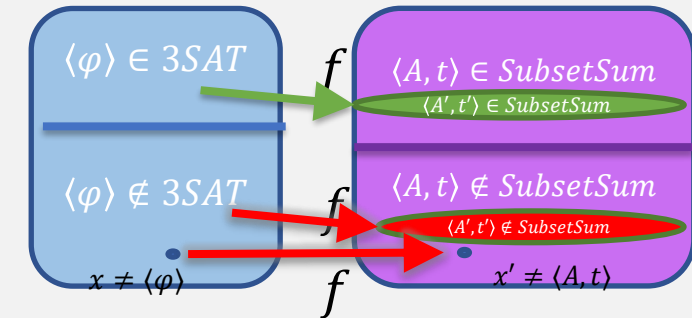
- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $3\text{SAT} = \{\varphi \mid \varphi \text{ is a satisfiable 3CNF formula}\}$
 - n : number of variables $x_1 \dots x_n$, m : number of clauses $c_1 \dots c_m$

• Claim: $3\text{SAT} \leq_p \text{SubsetSum}$

- Want: $f(\varphi) = \langle A', t' \rangle$ such that the 3CNF φ is satisfiable iff $\exists S' \subseteq \{1, \dots, n'\} \sum_{i \in S'} a'_i = t'$
- Idea: use (decimal) digits of a' 's to code a formula, t to enforce satisfiability condition.
- Example: $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee x_4)$



- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $3\text{SAT} = \{\varphi \mid \varphi \text{ is a satisfiable 3CNF formula}\}$
 - n : number of variables $x_1 \dots x_n$, m : number of clauses $c_1 \dots c_m$



- Claim: $3\text{SAT} \leq_p \text{SubsetSum}$

- Want: $f(\varphi) = \langle A', t' \rangle$ such that the 3CNF φ is satisfiable iff $\exists S' \subseteq \{1, \dots, n'\} \sum_{i \in S'} a'_i = t'$
- Idea: use (decimal) digits of a' 's to code a formula, t to enforce satisfiability condition.
- Use two numbers for each variable (one for positive, one for negative):
 - Let y_i, z_i code positive and negative occurrence of variable x_i
 - Split their digits into “code variable” and “code clauses” parts
 - Make $y_i[i] = z_i[i] = 1$ in “code variable” position i , and other “code variable” positions 0
 - » Set $t = 1$ at position i to force exactly one of them to be in S' .
 - If a literal has 1 in a “code clause” position j , it will encode that it occurs in clause j
 - selecting this literal will “contribute” a 1 to the clause j total
 - A small issue is that a clause can be satisfied by 1, 2 or 3 literals.
 - Solution: for every clause, add two “slack variables” g_j, h_j with 1s only in the position j
 - Make the corresponding digit of t exactly 3

- $f(\varphi) = \langle (y_1, \dots, y_n, z_1, \dots, z_n, g_1, \dots, g_m, h_1, \dots, h_m), \overbrace{1 \dots 1}^n \overbrace{3 \dots 3}^m \rangle$
 - Where $y_1, \dots, y_n, z_1, \dots, z_n, g_1, \dots, g_m, h_1, \dots, h_m$ are decimal numbers, defined as follows
 - For $i, i' \leq n$: $y_i[i] = z_i[i] = 1$, Else, for $i' \neq i$, $y_i[i'] = z_i[i'] = 0$
 - For $1 \leq j \leq m$:
 - if c_j contains x_i , $y_i[n + j] = 1$, else 0.
 - If c_j contains $\neg x_i$, $z_i[n + j] = 1$, else 0
 - $g_j[j] = h_j[j] = 1$, else 0
- Example: $\varphi = (x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4)$

- Construction: $f(\varphi) = \langle (y_1, \dots, y_n, z_1, \dots, z_n, g_1, \dots, g_m, h_1, \dots, h_m), \overbrace{1 \dots 1}^n \overbrace{3 \dots 3}^m \rangle$
 - Where $y_1, \dots, y_n, z_1, \dots, z_n, g_1, \dots, g_m, h_1, \dots, h_m$ are decimal numbers, defined as follows
 - For $i, i' \leq n$: $y_i[i] = z_i[i] = 1$, Else, for $i' \neq i$, $y_i[i'] = z_i[i'] = 0$
 - For $1 \leq j \leq m$:
 - if c_j contains x_i , $y_i[n+j] = 1$, else 0.
 - If c_j contains $\neg x_i$, $z_i[n+j] = 1$, else 0
 - $g_j[j] = h_j[j] = 1$, else 0

Correctness proof:

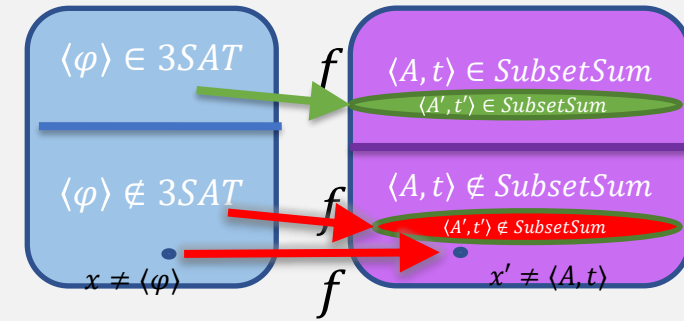
$\Rightarrow$ Suppose that $\exists S$ satisfying assignment for φ

- Then for each $i \leq n$, if x_i is set to true by S add i to S' and if to false, add $n+i$ to S'
- For every clause, if it is satisfied by less than 3 literals, add $2n+j$ to S'
- If it is satisfied by one literal, also add $2n+m+j$ to S'
- Now, can check that each of the first n digits of the sum of S' is 1, and of the last m digits is 3

$\Leftarrow$ Suppose that $\exists S' \subseteq \{1, \dots, 2n+2m\} \sum_{i \in S'} a_i = t$.

- Then for $i \leq n$ if $i \in S'$ then set x_i to true, else to false.
- The sum in every digit is at most 2 for the “code variable” positions, at most 5 for “code clause” (because 3CNF)
 - Because numbers are decimal, there cannot be a carry into another digit. So values in each digit sum independently.
- By construction, exactly one of y_i, z_i contributes to the sum for each i
- To make digits $n+1 \dots n+m$ sum up to 3, at least one of the elements in the sum other than g_j, h_j has to be 1 in the position j for every j . That element is the literal satisfying clause j , so the assignment S we constructed from S' is a satisfying assignment.

- Note: though the numbers are huge, they still contain only polynomially many digits each. So the construction is polytime computable.





SubsetSum and friends

- $\text{SubsetSum} = \{\langle A, t \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } t \in \mathbb{N} \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = t\}$
- $\text{Partition} = \{\langle A \rangle \mid A = (a_1, \dots, a_n) \in \mathbb{N}^n \text{ and } \exists S \subseteq \{1, \dots, n\} \sum_{i \in S} a_i = \sum_{i \notin S} a_i\}$

- We showed that SubsetSum and Partition are in NP.
- And $3SAT \leq_p \text{SubsetSum} \leq_p \text{Partition}$
- Therefore, SubsetSum and Partition are NP-complete
 - They are “weakly” NP-complete, since the reduction relied on numbers treated as strings.
 - As opposed to 3SAT and CLIQUE that are strongly NP-complete

