CSCE 411: Design and Analysis of Algorithms

Lecture : SAT and reductions

Date: Week 13 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- CLRS Chapter 34
- Homework due Friday

1 NP-completeness

Definition: A problem Q is NP-complete if:

1. $Q \in$ NP
2. for every problem $B \in$ NP, $B \leq_p Q$.

In words:

This implies that NP-complete problems are the *hardest* problems in NP.

Why? Remember that $A \leq_p B$ means A is easier than B.

In fact, if you only take the second part of the definition, this defines the set of NP-hard problems.

2 Views of the computational problem universe

There are four different possibilities for the relationship between P, NP, and co-NP.

There are two possibilities for the relationship between P, NP, and NP-complete.

A few famous NP-complete problems

- Find if a graph has a clique of size at least k
- Find if a graph has a Hamiltonian cycle
- Find if a graph has a simple path with at least k edges
- **Subset sum**: given a set of numbers, find if any subset sums to zero
- **SAT**: Logic problems we will look at in more depth later

3 Summary on Reduction and NP-completeness

If A can be reduced in polynomial time to B, we write $A \leq_p B$. This means that A is no harder than B; a solver for B can be used to solve A.

A problem Q is NP-hard if every problem in NP can be reduced to Q in polynomial time.

A problem Q is NP-complete if

- It is in NP
- It is NP-hard

Question 1. *Assume that A and B are both in NP and $A \leq_p B$. Which of the following statements is false?*

A *If $A \in P$, then $B \in P$*

B *If A is NP-complete, then B is NP-complete*

C *If $B \in P$, then $A \in P$*

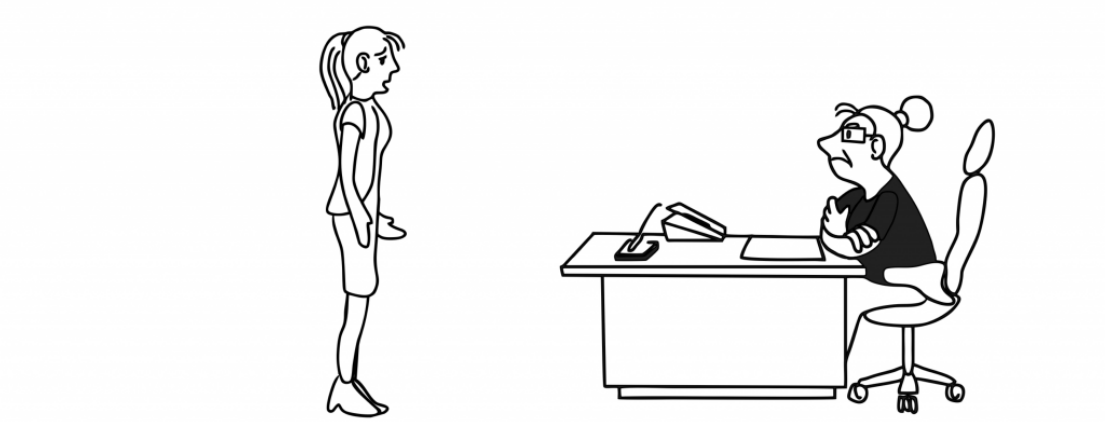
D *If A is NP-complete, then $B \leq_p A$*

If two problems are NP-complete, then they are both in NP, and:

4 Why does it matter whether a problem is in P, NP, or is NP-complete?

A famous book by Garey and Johnson, *Computers and Intractability*, includes some famous cartoons to help us answer this question.[1]

Let's say your boss tells you that because you are a CS major or minor you should be able to find an efficient algorithm for a certain problem at work. You work for weeks without progress. You don't want to tell your boss:



"I can't find an efficient algorithm, I guess I'm just too dumb."

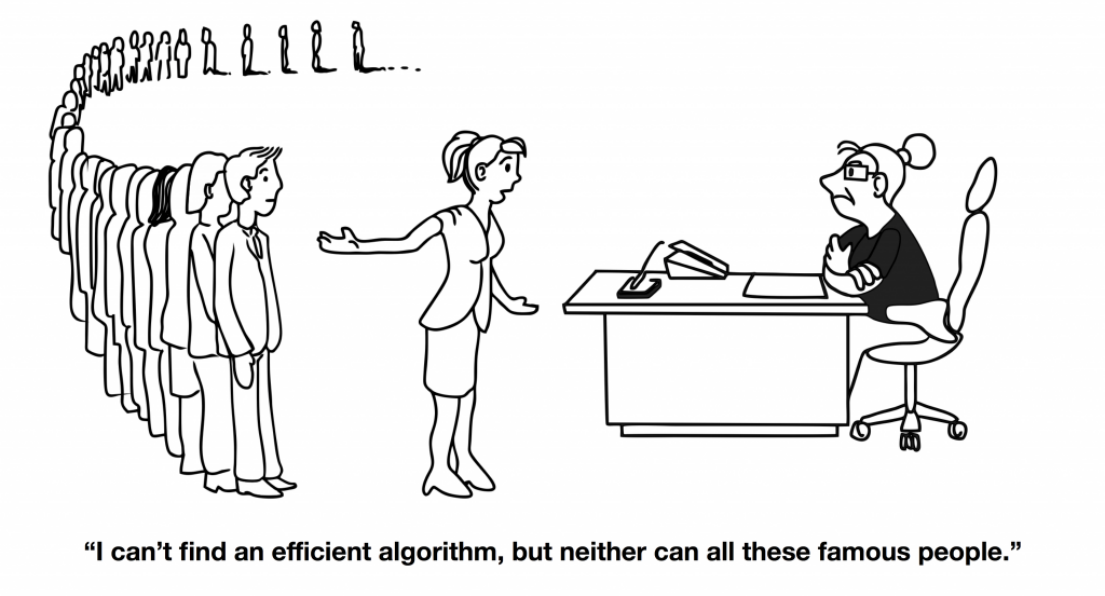
Ideally you'd like to be able to say:

"I can't find an efficient algorithm, because no such algorithm is possible!"

But in most situations, proving definitively that this is impossible is unlikely.

[1] Garey, Michael R., and David S. Johnson. Computers and intractability. Vol. 174. San Francisco: freeman, 1979. High-resolution versions of the cartoon are available online.

If you fail to find an efficient solution to a problem, ask yourself: am I struggling to find a solution simply because this is NP-complete? If you can prove it is, you have a great answer for your boss.



"I can't find an efficient algorithm, but neither can all these famous people."

5 A brief and incomplete history of NP-completeness

- 1971: Stephen Cook proves that every problem in NP can be reduced to a problem called *Boolean Satisfiability* (SAT).

  In modern terminology: SAT is NP-complete.

  The same results were developed independently by someone named Leonid Levin, so this is now called the *Cook-Levin* Theorem.

- 1972: Richard Karp publishes a paper *Reducibility Among Combinatorial Problems*, showing that SAT can be reduced to 21 other problems.

- 1979: Garey and Johnson publish the go-to reference on NP-completeness: *Computers and Intractability.*

6 Boolean Satisfiability Problems

A *boolean variable* is a variable x that can take on the values True or False.

A *boolean expression* or *boolean formula* is built from

- Variables: $x_1, x_2, \ldots, x_n$, all in $\{\text{True}, \text{False}\}$
- Operators:
  - AND ($\wedge$): e.g., True $\wedge$ False = False
  - OR ($\vee$): e.g., True $\vee$ False = True
  - NOT: $\neg$True = False; $\neg$False = True

    This is often also denoted with an overline:
- Parentheses: telling you the order in which to evaluate statements

Here are some examples of Boolean expressions:

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

Question 2. *Does the following statement evaluate to True or False?*

$$(T \wedge F) \vee (T \wedge F \wedge T) \vee (F \vee T)$$

A *True*

B *False*

Question 3. *Is the following Boolean expression satisfiable or not?*

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

A *Yes, it's satisfiable*

B *No, it's not satisfiable*

Theorem 6.1. *Checking whether a boolean formula is satisfiable is* _____

We will not prove this, but we'll use it as a starting point for proving other problems are NP-complete.

7 CNF Satisfiability and 3SAT

Some more terminology

- Literal: a boolean variable or its negation: e.g.,

- Clause: the OR of multiple literals: e.g.,

- A boolean formula is in conjunctive normal form if it is the AND of clauses.

A boolean formula is in 3-CNF if _____

Theorem 7.1. *Checking whether a boolean formula in 3-CNF is satisfiable is NP-complete. This is called 3SAT.*

Again: we'll use it as a starting point for proving other problems are NP-complete.

8 The clique problem is NP-complete

The clique problem: Check if an unweighted and undirected graph $G = (V, E)$ has a clique of size k. Formally we write this as:

$$\textsc{Clique} = \{\langle G, k\rangle : G \text{ has a clique size } k\}$$

Theorem 8.1. *The clique problem is* _____

Proof structure:

- We have already shown the problem _____
- We will show how to reduce every 3SAT problem to a clique problem.
- We will show that a YES instance of 3SAT is a YES instance of clique
- We will show that a YES instance of clique maps to a YES instance of 3SAT

The reduction. Consider an arbitrary instance of 3SAT, and let k be the number of 3-clauses it contains, denoted by

$$C_1 \wedge C_2 \wedge \cdots \wedge C_k \tag{1}$$

For $r \in \{1, 2, \ldots k\}$, let the literals in the rth clause be denoted as l_1^r, l_2^r, l_3^r so that:

$$C_r = (l_1^r \vee l_2^r \vee l_3^r) \tag{2}$$

In other words, l_i^r is the ith literal in the rth clause.

For each literal l_i^r, we introduce a node v_i^r.

For each pair of literals $\{l_i^r, l_j^s\}$, we add an edge (v_i^r, v_j^s) if:

- $r \neq s$
- v_i^r is not the negation of v_j^s; this means they are *consistent*

Let's consider an example $(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee x_2 \vee x_3) \wedge (x_1 \vee x_2 \vee x_3)$

The rest of the proof: We need to prove that YES for the 3SAT instance maps to YES for the clique instance, and that NO for the 3SAT instance maps to NO for the clique instance.

This is equivalent to proving that the 3SAT instance is satisfiable if

It turns out this is true but we will not prove it. The outline of the proof is:

- ($\Longrightarrow$) Assume there is a satisfying assignment, and prove that the literals set to TRUE to satisfy the k clauses map to

- ($\Longleftarrow$) Assume that there is a k-clique in the reduced graph and prove that

9 The 2SAT Problem

Similar to 3SAT, the problem 2SAT is the task of finding a satisfying assignment for a boolean satisfiability formula in conjunctive normal form (an AND of ORs) when all clauses involve exactly two literals.

Question 4. *Consider the following two true facts.*

Fact 1: *Every instance of 2SAT is an instance of SAT.*

Fact 2: *You can reduce 2SAT to* Clique*, similar to the way we reduced 3SAT to* Clique

Which of the following statements is true?

A *Fact 1 implies that 2SAT is NP-hard, but Fact 2 does not*

B *Fact 2 implies that 2SAT is NP-hard, but Fact 1 does not*

C *Both facts imply that 2SAT is NP-hard*

D *Neither fact is sufficient to conclude that 2SAT is NP-hard*

10 Another reduction for 2SAT

Given an instance of 2SAT, construct the *implication graph* as follows:

- Create a node for each variable and its negation
- For a clause $(\ell_1 \vee \ell_2)$ (which may be positive or negative literals), add:
  - A directed edge $(\neg\ell_1, \ell_2)$
  - A directed edge $(\neg\ell_2, \ell_1)$

An example.

$$(x_1 \vee x_2) \wedge (x_1 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3) \wedge (x_4 \vee \overline{x}_3) \wedge (\overline{x}_1 \vee \overline{x}_3)$$

Theorem 10.1. *In the implication graph for the 2SAT instance, a variable and its negation will be in separate strongly connected components if and only if*

11 Vertex Cover Problem

A vertex cover of an undirected graph $G = (V, E)$ is a set of nodes $S \subseteq V$ such that:

We say an edge is _____ if at least one of its vertices are in the vertex cover.

The **vertex-cover problem** is the task of finding a vertex cover of minimum size. The decision version of the problem is written as follows:

VertexCover =

Theorem 11.1. *The vertex-cover problem is* _____

Question 5. *Assume we have proven the above theorem. Which of the following statements is true?*

A *Vertex-cover is NP-complete, because any problem that is NP-hard is also NP-complete*

B *Vertex-cover is NP-complete, because of a different reason from the above reason*

C *It is possible that vertex-cover is not NP-complete, this is still an unsettled open question*

D *The fact that vertex-cover is NP-hard means that we can for sure rule out the possibility of having a polynomial-time algorithm for solving it.*

The complement of a graph

Given an undirected graph $G = (V, E)$, the *complement* graph $\bar{G} = (V, \bar{E})$ is the graph with the same set of nodes where:

Formally, this means that:

$$\bar{E} =$$

Observation. The complement of $\bar{G}$ is G. In other words, taking the complement of a complement returns the original graph.

Proof that vertex-cover is NP-hard

We will reduce the Clique Problem to vertex cover.

The reduction

- Let $G = (V, E)$ be the input for $\textsc{Clique}(G, k)$
- Construct the complement graph $\bar{G}$
- The output is $\langle \bar{G}, |V| - k \rangle$; i.e.,

What must hold for the reduction

- ($\implies$) Assume G has a k-clique and prove this means $\bar{G}$ has a vertex cover of size $|V| - k$
- ($\impliedby$) Assume $\bar{G}$ has a vertex cover of size $|V| - k$ and prove that G has a k-clique

We could prove each piece separately, but in some cases there is an easier way to show a reduction all at once.

Theorem 11.2. *A graph G has a size k clique $\iff$ $\bar{G}$ has a size $|V| - k$ vertex cover.*

12 Other NP-complete problems

We have already talked about the following NP-complete problems

- SAT and 3SAT
- The CLIQUE problem
- The VERTEX COVER problem

Although we will not provide explicit reductions to prove the following are NP-complete, here are some more well-studied NP-complete problems you should be aware of:

- Subset sum
- Maximum Cut
- Hamitonian Path
- Traveling Salesman
- Graph Coloring
- Clique Cover

Warm-up question before we move on

Question 6. *It is NP-hard to find a vertex cover for a graph G*

A *True*

B *False*

13 Summary of a few NP-complete graph problems

Subset sum The input to a subset sum problem is a list of integers L (could be positive or negative) and a target sum T. The goal is to find a subset of integers in L that sum to T. Several special variants are also NP-complete:

- The special case where $T = 0$
- The special case where all integers are positive
- The special case where all integers are positive and $T = \frac{1}{2}\text{sum}(L)$.

All graph problem in the rest of the lecture take as input an undirected graph $G = (V, E)$ (which may or may not have weights) and an arbitrary nonnegative integer k.

Hamiltonian Cycle A Hamiltonian cycle is a cycle in the graph that visits every node exactly once.

Traveling salesman problem The traveling salesman problem seeks the smallest weight Hamiltonian cycle. The decision version asks: is there a Hamiltonian cycle with weight at most W?

Graph Coloring Check whether we can assign one of k colors to each node $v \in V$ in such a way that if $(u, v) \in E$, then u and v have been assigned different colors.

This problem is NP-hard even if $k = 3$, but becomes polynomial time for $k = 2$.

Clique Cover A clique cover of size k is set of nodes $\mathcal{S} = \{S_1, S_2, \ldots S_k\}$ such that:

- $\mathcal{S}$ is a partition: $S_i \cap S_j = \emptyset$ if $i \neq j$, and $\bigcup_{i=1}^{k} S_i = V$
- S_i is a clique for each $i = 1, 2, \ldots, k$

Checking whether a graph G has a size k clique cover is NP-complete

14 Relationship between clique cover and graph coloring

Recall the following theorem regarding the relationship between the Vertex Cover problem and the Clique problem.

Theorem 14.1. *A graph $G = (V, E)$ has a clique of size k if and only if the complement graph $\bar{G}$ has a vertex cover of size $|V| - k$.*

Proof.

There is a relationship between graph coloring and clique cover that is very similar and can be proven using similar arguments. This will be a good practice problem for the final exam!

Maximum cut A *cut set* of an undirected graph G is a partition of the nodes $\{S, V - S\}$ and the cut value is the weight of edges across the cutset:

$$\mathbf{cut}(S) = \sum_{(u,v)\in E} w_{uv}. \tag{3}$$

Finding a set S that maximizes $\mathbf{cut}(S)$ is the *maximum cut* problem.

The decision version asks: given a value k, is there a set S with $\mathbf{cut}(S) \geq k$?