Plan

Search vs. Decision (3-coloring)

Many-one Reduction

Circuit-SAT and 3-SAT

Clique and ind. set

Efficient Reductions

Reductions

Comparison between a mathematician and an engineer:

Put an empty kettle in the middle of the kitchen floor and tell your subjects to boil some water.

The engineer will fill the kettle with water, put it on the stove, and turn the flame on. The mathematician will do the same thing.

Next, put the kettle already filled with water on the stove, and ask the subjects to boil the water.

The engineer will turn the flame on.

The mathematician will empty the kettle and put it in the middle of the kitchen floor... thereby reducing the problem to one that has already been solved!

K-Coloring

We define a k-coloring of a graph:

Each node gets colored with one color
At most k different colors are used
If two nodes have an edge between them they must have different colors

A graph is called k-colorable if and only if it has a k-coloring.

A 2-CRAYOLA Question!

Is Gadget 2-colorable?
No, it contains a triangle

A 2-CRAYOLA Question!

Given a graph G, how can we decide if it is 2-colorable?

Answer: Enumerate all 2^n possible colorings to look for a valid 2-color

How can we efficiently decide if G is 2-colorable (aka bipartite)?

Theorem: G contains an odd cycle if and only if G is not 2-colorable.
Efficient 2-coloring algorithm:

To 2-color a connected graph G, pick an arbitrary node v, and color it white
Color all v’s neighbors black
Color all their uncolored neighbors white, and so on
If the algorithm terminates without a color conflict, output the 2-coloring
Else, output graph is not 2-colorable (the conflict proves no 2-coloring is possible, and there is an odd cycle)

A 3-CRAYOLA Question!
Is this graph 3-colorable?

A 3-CRAYOLA Question!
Is the "wheel" 3-colorable?

3-Coloring Is Decidable by Brute Force

Try out all 3^n colorings until you determine if G has a 3-coloring

A 3-CRAYOLA Oracle

NO, or YES here is how: gives 3-coloring of the nodes

Better 3-CRAYOLA Oracle

3-Colorability Decision Oracle

3-Colorability Search Oracle
How do I turn a decision oracle into a search oracle?

If a new graph is 3-colorable, we know the color of one vertex.

If a new graph is not 3-colorable, we try another color.

A 3-colorability search oracle can be simulated using a polynomial number of calls to a decision oracle!

But how efficient the decision oracle?

Solving search problem efficiently means that decision problem can be solved efficiently.

However, if decision problem is difficult then search version is definitely difficult.

In some case we can use an algorithm for decision problem to solve the search problem.

Let’s now look at two other problems:

1. k-Clique
2. Independent Set
K-Cliques

A K-clique is a set of K nodes with all K(K-1)/2 possible edges between them

This graph contains a 4-clique

Given: (G, k)
Question: Does G contain a k-clique?

BRUTE FORCE: Try out all \( n \choose k \) possible locations for the k clique

Independent Set

An independent set is a set of nodes with no edges between them

This graph contains an independent set of size 3

Given: (G, k)
Question: Does G contain an independent set of size k?

BRUTE FORCE: Try out all \( n \choose k \) possible locations for the k independent set

Clique / Independent Set

Two problems that are cosmetically different, but substantially the same

Complement of G

Given a graph G, let \( G^c \), the complement of G, be the graph obtained by the rule that two nodes in \( G^c \) are connected if and only if the corresponding nodes of G are not connected
Thus, we can quickly reduce a clique problem to an independent set problem and vice versa. There is a fast method for one if and only if there is a fast method for the other.

Many-one reduction

To reduce problem A to problem B (we write $A \leq_p B$) we want a function $f$ that maps $A$ to $B$ such that:
1) $f$ is a polynomial time computable
2) $x \in A$ if and only if $f(x) \in B$.

This also called Karp's reduction and mapping reduction.

Let's now look:
1. Circuit Satisfiability
2. Graph 3-Colorability

Combinatorial Circuits

AND, OR, NOT, 0, 1 gates wired together with no feedback allowed

Circuit-Satisfiability

Given a circuit with $n$-inputs and one output, is there a way to assign 0-1 values to the input wires so that the output value is 1 (true)?

Yes, this circuit is satisfiable: 110
**Circuit-Satisfiability**

Given: A circuit with \( n \)-inputs and one output, is there a way to assign 0-1 values to the input so that the output value is 1 (true)?

BRUTE FORCE: Try out all \( 2^n \) assignments

**3-Colorability**

Given an oracle for 3-colorability, how can you quickly solve circuit SAT?

**Circuit Satisfiability**

These two problems are fundamentally the same!
3-colorability vs. circuit-Sat

\[ f(x, y) \]

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>( f(x, y) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

AND Gate from OR and NOT

\[ x \quad \text{OR} \quad \text{NOT} \quad \text{NOT} \quad y \]

3-colorability vs. circuit-Sat

\[ f(x, y) \]

<table>
<thead>
<tr>
<th>x</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

How do we force the graph to be 3 colorable exactly when the circuit is satisfiable?

Circuit is satisfiable

Graph is 3-colorable

3-colorability vs. circuit-Sat

\[ x \quad \text{OR} \quad \text{NOT} \quad \text{OR} \]

3-colorability vs. circuit-Sat

\[ x \quad \text{OR} \quad \text{NOT} \quad \text{OR} \]

3-colorability vs. circuit-Sat

\[ x \quad \text{OR} \quad \text{NOT} \quad \text{OR} \]

3-colorability vs. circuit-Sat

\[ x \quad \text{OR} \quad \text{NOT} \quad \text{OR} \]
3-colorability vs. circuit-Sat

There is a linear-time function that reduces instances of circuit-Sat to instances of 3-colorability.

Fact: There are efficient ways to reduce an instance of any of the four problems we discussed to an instance of any other.

But nobody knows how to efficiently solve any of these four problems in the worst case!

Polynomial Time Complexity

Is there a fixed constant c and an algorithm A such that A solves the decision problem in time O(n^c)?

Verifying solutions

In some problems (like coloring), verifying the solution can be done efficiently.

NP = Decision problems whose solutions can be verified in polynomial time in their input size.

The N in NP stands for "nondeterministically."

Here's how P vs. NP is usually (informally) stated:

Let L be an algorithmic task.

Suppose there is an efficient algorithm for verifying solutions to L. "L ∈ NP"

Is there always also an efficient algorithm for finding solutions to L? "L ∈ P"

Definition of P

An input is encoded as a binary string.

P = {L ⊂ {0, 1}^* | ∃ polynomial time algorithm for deciding L}

Definition of NP

NP = {L ⊂ {0, 1}^* | ∃ polynomial time verifier R(x, y) = true, where x ∈ L and |y| ≤ O(|x|)}
Definition of NP-hard

NP-hard = \{ L \subseteq \{0, 1\}^* \mid \forall X \in \text{NP and } X \leq_p L \}

To reduce problem X to problem L (we write X \leq_p L) we want a function f that maps X to L such that:
1) f is a polynomial time computable
2) x \in X if and only if f(x) \in L.

In short. We need to convert X into L, and L to X.

Lemma. If A \leq_p B and B \in P then A \in P.

Venn Diagram (P \neq NP)

NP-complete Reduction

A recipe for proving any L \in NP-complete:
1) Prove L \in NP
2) Choose A \in NPC and reduce it to/from L
2.1) Describe mapping f:A \to L
2.2) Prove x \in A iff f(x) \in L
2.3) Prove f is polynomial

Definition of NP-complete

L is NP-complete iff
1) L \subseteq \text{NP}
2) L \subseteq \text{NP-hard}
2) For all Y \subseteq \text{NP}, Y \leq_p L

Conjunctive Normal Form

Let X_k denote variables.
We define literals as either X_k or \neg X_k.

The conjunctive normal form (CNF) is an AND of OR clauses. For example,

\((X_1 \lor \neg X_3) \land (X_1 \lor \neg X_2 \lor X_4 \lor \neg X_3) \land \ldots\)

SAT Problem: is there exist a set of variables that satisfy a given CNF?
Cook-Levin Theorem (1971)

SAT is NP-complete

No proof...

Cook received a Turing Award for his work.

3-CNF problem (or 3-SAT)

Each clause has a most 3 literals.

Question: Is there such a set of input variables that 3-CNF is true?

Theorem. 3-CNF is NP-complete

Proof. 3-CNF ⊆ NP
We need to show CNF ₃ ⊆ 3-CNF.

CNF ₃ 3-CNF

We need to convert any CNF into 3-CNF...

Claim:
(a ∨ b ∨ c ∨ d) is true iff
(a ∨ b ∨ x) ∧ (x ∨ c ∨ d) is true

(a ∨ b ∨ c ∨ d ∨ e) converts to
(a ∨ b ∨ x) ∧ (x ∨ c ∨ y) ∧ (y ∨ d ∨ e)

The rest of the proof is left as an exercise to a reader.

Clique is NP-complete

1) k-clique is in NP
2) We will show that SAT ₃ Clique

Given a Boolean formula in CNF, we will show how to construct a graph.

(X₁ ∨ X₂) ∧ (X₁ ∨ X₂) ∧ (X₁ ∨ X₂)

Create a vertex for each variable in a clause
Two vertices from different clauses are connected if one is NOT negation of other

Clique is NP-complete

For example,

(X₁ ∨ X₂) ∧ (X₁ ∨ X₂) ∧ (X₁ ∨ X₂)

A CNF is satisfiable if at least one literal in each clause is true. Thus those literals create a k-clique.

Independent Set

The Ind. Set problem is to determine, given G=(V,E) and k>0, whether G an independent set of size at least k.

The problem is reduced to/from the clique problem, using a complementary graph, so the ind. set problem is NP-complete.
A vertex cover problem is to determine if there is a cover of the size at most $k$.

A vertex cover in $G=(V,E)$ is a set of vertices $U$ s.t. every edge is adjacent some vertex from $U$.

A vertex cover is NP-complete
1) it is in NP
2) We will show that Ind. Set $\leq_p$ Vertex Cover

$U$ is a vertex cover iff $V-U$ is an independent set.

All of these problems poly-reduce to one another!

Karp received a Turing Award for his work.