An universe of optimization problems

- Minimum spanning tree
- Steiner tree
- Longest common superstring
- Shortest common superstring
- Linear programming
- Integer programming
- Graph coloring
- Travelling salesman
- Steiner tree
- Minimum spanning tree
- Longest common substring
- Linear programming
- Shortest paths
- Integer programming
- Travelling salesman
- Steiner tree
- Minimum spanning tree
- Longest common substring
- Linear programming
- Shortest paths
- Integer programming

Problems for which we have a polynomial algorithm:
- Longest path
- Graph coloring
- Traveling salesman
- Shortest common superstring
- Steiner tree
- Minimum spanning tree
- Longest common substring
- Linear programming
- Shortest paths
- Integer programming

Problems for which we don't know a polynomial algorithm:
- 2SAT
- 3SAT

Define:
- EASY = polynomial
- HARD = not polynomial

Are these easy or hard?
- We aren't sure. But either they are ALL easy or they are ALL hard!

Reductions

"Problem A reduces to Problem B" means:
- If you are given the ability to solve B in polynomial time, then you can also solve A in polynomial time

eg. Bipartite Matching reduces to Max Flow
- Since Max Flow can be solved efficiently, this reduction tells us that Bipartite Matching can also be solved efficiently.

For simplicity define:
- EASY = polynomial time
- HARD = not polynomial time

Suppose A reduces to B.

What can we conclude?

- If B is easy:
  - If A is easy:
    - If A is hard:
  - If B is hard:

Reductions at work

Independent set

An independent set of an undirected graph is a subset of vertices such that there are no edges between them.

Does IS reduce to MIS?

Does MIS reduce to IS?

Conclusion: MIS and IS are either BOTH EASY or they are BOTH HARD.

Boolean formulas

Example:
- f(x, y, z) = (x = y = z)
- What is f(0, 1, 1)? [0 = false, 1 = true]

Variable: {x, y, z}
Literal: variable or its negation: x, ¬x, y, ¬y, z, ¬z.

Conjunctive normal form (CNF):
- Any Boolean formula can be written as an AND-of-ORs:
  \( C_1 \land (C_2 \lor C_3) \land \ldots \land (C_m \lor \ldots) \)
  where each clause \( C_i \) is an OR of literals.
- 3CNF: Each clause \( C_i \) has \( \leq 3 \) literals
- 2CNF: Each clause \( C_i \) has \( \leq 2 \) literals

f(x, y, z) = (x = y) \land (x = z) \land (¬x = y = ¬z)

Does 2SAT reduce to 3SAT?

Does 3SAT reduce to 3CNF?

Reductions at work
3SAT reduces to Independent Set

Independent set: nodes with no edges between them.

Given formula:

\( (x_1 \lor \neg x_2 \lor y_3) \land (x_4 \lor y_2 \lor \neg y_3) \land \ldots \land C_m \)

we'll create an undirected graph \( G \) with 3m nodes, which has the following property:

- \( f \) is satisfiable if and only if \( G \) has an independent set of size \( \geq m \).
- \( G \) can be constructed in polynomial time. Here's the reduction:

procedure 3SAT(\( f \))
- create \( 3m \) nodes in a triangle:
  - eg: 1 2 3

Then, put an edge between any two nodes that correspond to literals of opposite polarity.
- eg: 1 2 3 = (\( x \lor y \lor z \lor w \)) \land (\( \neg x \lor \neg y \lor \neg z \lor w \)) \land (\( x \lor \neg y \lor z \lor \neg w \))

Conclusion: If 3SAT is hard, so is Independent Set...

Vertex cover

A vertex cover of an undirected graph is a subset of vertices \( S \) such that every edge touches \( S \).

\( G \) has an independent set of size \( \geq k \) if and only if \( G \) has a vertex cover of size \( \leq |V| - k \).

Thus IS and VC reduce to each other.

Clique

Complement of a graph \( G' \): the same nodes as \( G \), and has exactly the edges that \( G \) doesn't have.

Ex:

\[ (x \lor y \lor z \lor w) \land (\neg x \lor \neg y \lor \neg z \lor w) \land (x \lor \neg y \lor z \lor \neg w) \]

Claim: \( f \) is satisfiable if and only if \( G \) has an independent set of size \( m \).

- Suppose \( f \) is satisfiable.
  - Pick one node from each triangle: therefore \( |S| = m \).
  - Moreover \( S \) cannot contain nodes which correspond to literals of opposite polarity.
  - \( S \) specifies a satisfying assignment of \( f \).

Claim: \( f \) is satisfiable if and only if \( G \) has an independent set of size \( m \).

- Suppose \( G \) has an independent set of size \( m \). Call this set \( S \).
- \( S \) can only have one node from each triangle: therefore \( |S| = m \).

Thus IS, Clique reduce to each other.

P, NP, NP-hard, NP-complete

- A problem is in \( P \) if there is a polynomial time algorithm for solving it.
- A problem is in \( NP \) if there is a polynomial time algorithm to check if a solution is correct. (In particular, the solution should be short).
- A problem is \( \text{NP-hard} \) if all problems in \( NP \) can be reduced to it. (i.e. it is harder than all of the problems in \( NP \)).
- A problem is \( \text{NP-complete} \) if it is \( \text{NP-hard} \) and is in \( NP \).
- If there is a polynomial time algorithm for any \( \text{NP-complete} \) problem then \( P = NP \).
- \( \text{NP-complete} \) problems: 3SAT, IS, Clique, VC, Traveling Salesman, Integer Programming...
Epilogue: undecidability

The intractable problems we’ve studied (3SAT, Clique, ...) can all be solved in exponential time. Are there problems that take even longer?

Some problems cannot be solved, no matter how much time the computer is allowed to take!

Example:

Input: Program P, input value x

Question: When P is run on x, does it eventually halt?

Claim: No computer program can solve term.

Proof by contradiction.

Suppose you could solve it. Then consider the following computer program:

\[
\text{program diag(P)}
\]

\[
\text{if term(P,P) then loop forever}
\]

Does diag(diag) terminate?

Contradiction, \text{term} is impossible to write!

This is the halting problem.