Algorithm design patterns and antipatterns

Algorithm design patterns.
• Greedy.
• Divide and conquer.
• Dynamic programming.
• Duality.
• Reductions.
• Local search.
• Randomization.

Algorithm design antipatterns.
• NP-completeness. $O(n^k)$ algorithm unlikely.
• PSPACE-completeness. $O(n^k)$ certification algorithm unlikely.
• Undecidability. No algorithm possible.

Classify problems according to computational requirements

Q. Which problems will we be able to solve in practice?

A working definition. Those with polynomial-time algorithms.


Theory. Definition is broad and robust.

Practice. Poly-time algorithms scale to huge problems.

constants $a$ and $b$ tend to be small, e.g., $3 N^2$
Classify problems according to computational requirements

Q. Which problems will we be able to solve in practice?

A working definition. Those with polynomial-time algorithms.

<table>
<thead>
<tr>
<th>yes</th>
<th>probably no</th>
</tr>
</thead>
<tbody>
<tr>
<td>shortest path</td>
<td>longest path</td>
</tr>
<tr>
<td>min cut</td>
<td>max cut</td>
</tr>
<tr>
<td>2-satisfiability</td>
<td>3-satisfiability</td>
</tr>
<tr>
<td>planar 4-colorability</td>
<td>planar 3-colorability</td>
</tr>
<tr>
<td>bipartite vertex cover</td>
<td>vertex cover</td>
</tr>
<tr>
<td>matching</td>
<td>3d-matching</td>
</tr>
<tr>
<td>primality testing</td>
<td>factoring</td>
</tr>
<tr>
<td>linear programming</td>
<td>integer linear programming</td>
</tr>
</tbody>
</table>

Classify problems

Desiderata. Classify problems according to those that can be solved in polynomial time and those that cannot.

Provably requires exponential time.
- Given a constant-size program, does it halt in at most \( k \) steps?
- Given a board position in an \( n \)-by-\( n \) generalization of checkers, can black guarantee a win?

Frustrating news. Huge number of fundamental problems have defied classification for decades.

Polynomial-time reductions

Desiderata’. Suppose we could solve \( X \) in polynomial-time. What else could we solve in polynomial time?

Reduction. Problem \( X \) polynomial-time (Cook) reduces to problem \( Y \) if arbitrary instances of problem \( X \) can be solved using:
- Polynomial number of standard computational steps, plus
- Polynomial number of calls to oracle that solves problem \( Y \).

Note. We pay for time to write down instances sent to oracle \( \Rightarrow \) instances of \( Y \) must be of polynomial size.

Caveat. Don’t mistake \( X \preceq_P Y \) with \( Y \preceq_P X \).
Polynomial-time reductions

Design algorithms. If $X \leq_p Y$ and $Y$ can be solved in polynomial time, then $X$ can be solved in polynomial time.

Establish intractability. If $X \leq_p Y$ and $X$ cannot be solved in polynomial time, then $Y$ cannot be solved in polynomial time.

Establish equivalence. If both $X \leq_p Y$ and $Y \leq_p X$, we use notation $X \equiv_p Y$. In this case, $X$ can be solved in polynomial time iff $Y$ can be.

Bottom line. Reductions classify problems according to relative difficulty.

---

Independent set

**INDEPENDENT-SET.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \geq k$, and for each edge at most one of its endpoints is in $S$?

**Ex.** Is there an independent set of size $\geq 6$?
**Ex.** Is there an independent set of size $\geq 7$?

---

Vertex cover

**VERTEX-COVER.** Given a graph $G = (V, E)$ and an integer $k$, is there a subset of vertices $S \subseteq V$ such that $|S| \leq k$, and for each edge, at least one of its endpoints is in $S$?

**Ex.** Is there a vertex cover of size $\leq 4$?
**Ex.** Is there a vertex cover of size $\leq 3$?
Vertex cover and independent set reduce to one another

**Theorem.** \( \text{VERTEX-COVER} \equiv_p \text{INDEPENDENT-SET}. \)

**Pf.** We show \( S \) is an independent set of size \( k \) iff \( V - S \) is a vertex cover of size \( n - k \).

\[
\begin{align*}
\text{independent set of size 6} & \quad \text{vertex cover of size 4} \\
\end{align*}
\]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]

\[\text{ratsPtstTstTS} t T X \]
Theorem. \( \textsc{Vertex-Cover} \leq_p \textsc{Set-Cover} \).

Pf. Given a \textsc{Vertex-Cover} instance \( G = (V, E) \), we construct a \textsc{Set-Cover} instance \((U, S)\) that has a set cover of size \( k \) iff \( G \) has a vertex cover of size \( k \).

Construction.

- Universe \( U = E \).
- Include one set for each node \( v \in V : S_v = \{ e \in E : e \text{ incident to } v \} \).

\[
\begin{array}{c}
\text{vertex cover instance} \\
(k = 2)
\end{array}
\begin{array}{c}
\text{set cover instance} \\
(k = 2)
\end{array}
\]

Lemma. \( G = (V, E) \) contains a vertex cover of size \( k \) iff \((U, S)\) contains a set cover of size \( k \).

Pf. \( \Rightarrow \) Let \( X \subseteq V \) be a vertex cover of size \( k \) in \( G \).
- Then \( Y = \{ S_v : v \in X \} \) is a set cover of size \( k \). 

\[
\begin{array}{c}
\text{vertex cover instance} \\
(k = 2)
\end{array}
\begin{array}{c}
\text{set cover instance} \\
(k = 2)
\end{array}
\]

8. Intractability

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
3-satisfiability reduces to independent set

**Theorem.** 3-SAT \( \leq_p \) INDEPENDENT-SET.

**Pf.** Given an instance \( \Phi \) of 3-SAT, we construct an instance \((G, k)\) of \textsc{independent-set} that has an independent set of size \(k\) iff \( \Phi \) is satisfiable.

**Construction.**
- \( G \) contains 3 nodes for each clause, one for each literal.
- Connect 3 literals in a clause in a triangle.
- Connect literal to each of its negations.

```
\begin{align*}
G &= (x_1 \lor x_2 \lor x_3) \land (\overline{x_1} \lor x_2 \lor x_3) \land (\overline{x_1} \lor \overline{x_2} \lor x_4) \\
k &= 3
\end{align*}
```

**Review.**

**Basic reduction strategies.**
- Simple equivalence: \textsc{independent-set} \( =_p \) \textsc{vertex-cover}.
- Special case to general case: \textsc{vertex-cover} \( \leq_p \) \textsc{set-cover}.
- Encoding with gadgets: 3-SAT \( \leq_p \) INDEPENDENT-SET.

**Transitivity.** If \( X \leq_p Y \) and \( Y \leq_p Z \), then \( X \leq_p Z \).

**Pf idea.** Compose the two algorithms.

**Ex.** 3-SAT \( \leq_p \) \textsc{independent-set} \( \leq_p \) \textsc{vertex-cover} \( \leq_p \) \textsc{set-cover}.
**Search problems**

**Decision problem.** Does there exist a vertex cover of size \( \leq k \)?

**Search problem.** Find a vertex cover of size \( \leq k \).

**Ex.** To find a vertex cover of size \( \leq k \):
- Determine if there exists a vertex cover of size \( \leq k \).
- Find a vertex \( v \) such that \( G - \{ v \} \) has a vertex cover of size \( \leq k - 1 \).
  (any vertex in any vertex cover of size \( \leq k \) will have this property)
- Include \( v \) in the vertex cover.
- Recursively find a vertex cover of size \( \leq k - 1 \) in \( G - \{ v \} \).

Bottom line. \( \text{VERTEX-COVER} \equiv \text{P} \) \( \text{FIND-VERTEX-COVER} \).

**Optimization problems**

**Decision problem.** Does there exist a vertex cover of size \( \leq k \)?

**Search problem.** Find a vertex cover of size \( \leq k \).

**Optimization problem.** Find a vertex cover of minimum size.

**Ex.** To find vertex cover of minimum size:
- (Binary) search for size \( k^* \) of min vertex cover.
- Solve corresponding search problem.

Bottom line. \( \text{VERTEX-COVER} \equiv \text{P} \) \( \text{FIND-VERTEX-COVER} \equiv \text{P} \) \( \text{OPTIMAL-VERTEX-COVER} \).

**8. INTRACTABILITY I**

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems

**Hamilton cycle**

**HAM-CYCLE.** Given an undirected graph \( G = (V, E) \), does there exist a simple cycle \( \Gamma \) that contains every node in \( V \)?
Hamilton cycle

**HAM-CYCLE.** Given an undirected graph $G = (V, E)$, does there exist a simple cycle $\Gamma$ that contains every node in $V$?

Directed hamilton cycle reduces to hamilton cycle

**Lemma.** $G$ has a directed Hamilton cycle iff $G'$ has a Hamilton cycle.

**Pf.** ⇒
- Suppose $G$ has a directed Hamilton cycle $\Gamma$.
- Then $G'$ has an undirected Hamilton cycle (same order).

**Pf.** ⇐
- Suppose $G'$ has an undirected Hamilton cycle $\Gamma'$.
- $\Gamma'$ must visit nodes in $G'$ using one of following two orders:
  - ..., $B, G, R, B, G, R, B, G, R, B, ...$
- Blue nodes in $\Gamma'$ make up directed Hamilton cycle $\Gamma$ in $G$, or reverse of one.

3-satisfiability reduces to directed hamilton cycle

**Theorem.** 3-SAT $\leq_p$ DIR-HAM-CYCLE.

**Pf.** Given an instance $\Phi$ of 3-SAT, we construct an instance of DIR-HAM-CYCLE that has a Hamilton cycle iff $\Phi$ is satisfiable.

**Construction.** First, create graph that has $2^n$ Hamilton cycles which correspond in a natural way to $2^n$ possible truth assignments.
3-satisfiability reduces to directed hamilton cycle

**Construction.** Given 3-SAT instance $\Phi$ with $n$ variables $x_i$ and $k$ clauses.
- Construct $G$ to have $2^n$ Hamilton cycles.
- Intuition: traverse path $i$ from left to right $\iff$ set variable $x_i = \text{true}.$

**Lemma.** $\Phi$ is satisfiable iff $G$ has a Hamilton cycle.

**Pf.** $\Rightarrow$
- Suppose 3-SAT instance has satisfying assignment $x^\omega.$
- Then, define Hamilton cycle in $G$ as follows:
  - if $x_i^\omega = \text{true},$ traverse row $i$ from left to right
  - if $x_i^\omega = \text{false},$ traverse row $i$ from right to left
  - for each clause $C_j,$ there will be at least one row $i$ in which we are going in "correct" direction to splice clause node $C_j$ into cycle (and we splice in $C_j$ exactly once)

**Pf.** $\Leftarrow$
- Suppose $G$ has a Hamilton cycle $\Gamma.$
- If $\Gamma$ enters clause node $C_j,$ it must depart on mate edge.
  - nodes immediately before and after $C_j$ are connected by an edge $e \in E$
  - removing $C_j$ from cycle, and replacing it with edge $e$ yields Hamilton cycle on $G - \{C_j\}$
- Continuing in this way, we are left with a Hamilton cycle $\Gamma'$ in $G - \{C_1, C_2, \ldots, C_k\}.$
- Set $x_i^\omega = \text{true}$ iff $\Gamma'$ traverses row $i$ left to right.
- Since $\Gamma$ visits each clause node $C_j,$ at least one of the paths is traversed in "correct" direction, and each clause is satisfied. $\blacksquare$
3-satisfiability reduces to longest path

**LONGEST-PATH.** Given a directed graph $G = (V, E)$, does there exist a simple path consisting of at least $k$ edges?

**Theorem.** $3$-$\text{Sat} \leq_p \text{LONGEST-PATH}.$

**Pf 1.** Redo proof for $\text{DIR-HAM-CYCLE}$, ignoring back-edge from $i$ to $s$.

**Pf 2.** Show $\text{HAM-CYCLE} \leq_p \text{LONGEST-PATH}.$

Traveling salesperson problem

**TSP.** Given a set of $n$ cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

13,509 cities in the United States
http://www.tsp.gatech.edu

Traveling salesperson problem

**TSP.** Given a set of $n$ cities and a pairwise distance function $d(u, v)$, is there a tour of length $\leq D$?

11,849 holes to drill in a programmed logic array
http://www.tsp.gatech.edu
Traveling salesperson problem

**TSP.** Given a set of \( n \) cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \) ?

Hamilton cycle reduces to traveling salesperson problem

**TSP.** Given a set of \( n \) cities and a pairwise distance function \( d(u, v) \), is there a tour of length \( \leq D \) ?

**HAM-CYCLE.** Given an undirected graph \( G = (V, E) \), does there exist a simple cycle \( \Gamma \) that contains every node in \( V \)?

**Theorem.** HAM-CYCLE \( \leq_p \) TSP.

**Pf.**
- Given instance \( G = (V, E) \) of HAM-CYCLE, create \( n \) cities with distance function
  \[ d(u, v) = \begin{cases} 
  1 & \text{if } (u, v) \in E \\
  2 & \text{if } (u, v) \notin E 
\end{cases} \]
- TSP instance has tour of length \( \leq n \) iff \( G \) has a Hamilton cycle.

**Remark.** TSP instance satisfies triangle inequality: \( d(u, w) \leq d(u, v) + d(v, w) \).

Polynomial-time reductions

8. **Intractability I**

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems
3-dimensional matching

**3D-MATCHING.** Given \( n \) instructors, \( n \) courses, and \( n \) times, and a list of the possible courses and times each instructor is willing to teach, is it possible to make an assignment so that all courses are taught at different times?

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Course</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wayne</td>
<td>COS 226</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Wayne</td>
<td>COS 423</td>
<td>TTh 11-12:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 423</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Tardos</td>
<td>COS 523</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>TTh 3-4:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 226</td>
<td>MW 11-12:20</td>
</tr>
<tr>
<td>Kleinberg</td>
<td>COS 423</td>
<td>MW 11-12:20</td>
</tr>
</tbody>
</table>

**Theorem.** \( 3\text{-SAT} \leq_p \text{3D-MATCHING}. \)

**Pf.** Given an instance \( \Phi \) of 3-SAT, we construct an instance of 3D-MATCHING that has a perfect matching if and only if \( \Phi \) is satisfiable.

**Remark.** Generalization of bipartite matching.

**3-satisfiability reduces to 3-dimensional matching**

**Construction.** (part 1)
- Create gadget for each variable \( x_i \) with \( 2k \) core elements and \( 2k \) tip ones.
3-satisfiability reduces to 3-dimensional matching

**Construction.** (part 1)
- Create gadget for each variable $x_i$ with $2k$ core elements and $2k$ tip ones.
- No other triples will use core elements.
- In gadget for $x_i$, any perfect matching must use either all gray triples (corresponding to $x_i = \text{true}$) or all blue ones (corresponding to $x_i = \text{false}$).

**Lemma.** Instance $(X, Y, Z)$ has a perfect matching iff $\Phi$ is satisfiable.

**Q.** What are $X, Y,$ and $Z$?
3-satisfiability reduces to 3-dimensional matching

**Lemma.** Instance \((X, Y, Z)\) has a perfect matching iff \(\Phi\) is satisfiable.

**Q.** What are \(X\), \(Y\), and \(Z\)?

**A.** \(X = \text{red}, Y = \text{green}, \) and \(Z = \text{blue}\).

**Pf.** \(\Rightarrow\) If 3d-matching, then assign \(x_i\) according to gadget \(x_i\).

**Pf.** \(\Leftarrow\) If \(\Phi\) is satisfiable, use any true literal in \(C_j\) to select gadget \(C_j\) triple.

---

3-colorability

**3-COLOR.** Given an undirected graph \(G\), can the nodes be colored red, green, and blue so that no adjacent nodes have the same color?
Application: register allocation

Register allocation. Assign program variables to machine register so that no more than \( k \) registers are used and no two program variables that are needed at the same time are assigned to the same register.

Interference graph. Nodes are program variables names; edge between \( u \) and \( v \) if there exists an operation where both \( u \) and \( v \) are "live" at the same time.

Observation. [Chaitin 1982] Can solve register allocation problem iff interference graph is \( k \)-colorable.

Fact. 3-COLOR \( \leq_p \) K-REGISTER-ALLOCATION for any constant \( k \geq 3 \).

3-satisfiability reduces to 3-colorability

Theorem. 3-SAT \( \leq_p \) 3-COLOR.

Pf. Given 3-SAT instance \( \Phi \), we construct an instance of 3-COLOR that is 3-colorable iff \( \Phi \) is satisfiable.

Construction.
(i) Create a graph \( G \) with a node for each literal.
(ii) Connect each literal to its negation.
(iii) Create 3 new nodes \( T, F, \) and \( B \); connect them in a triangle.
(iv) Connect each literal to \( B \).
(v) For each clause \( C_j \), add a gadget of 6 nodes and 13 edges.

Lemma. Graph \( G \) is 3-colorable iff \( \Phi \) is satisfiable.

Pf. \( \Rightarrow \) Suppose graph \( G \) is 3-colorable.
   - Consider assignment that sets all \( T \) literals to true.
   - (iv) ensures each literal is \( T \) or \( F \).
   - (ii) ensures a literal and its negation are opposites.
3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Rightarrow$ Suppose graph $G$ is 3-colorable.
- Consider assignment that sets all $T$ literals to true.
- (iv) ensures each literal is $T$ or $F$.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is $T$.

$\neg$ Consider assignment that sets all $T$ literals to true.
- (i) ensures a literal and its negation are opposites.
- (ii) ensures a literal and its negation are opposites.
- (v) ensures at least one literal in each clause is $T$.

3-satisfiability reduces to 3-colorability

**Lemma.** Graph $G$ is 3-colorable iff $\Phi$ is satisfiable.

**Pf.** $\Leftarrow$ Suppose $3$-SAT instance $\Phi$ is satisfiable.
- Color all true literals $T$.
- Color node below green node $F$, and node below that $B$.
- Color remaining middle row nodes $B$.
- Color remaining bottom nodes $T$ or $F$ as forced.

**Polynomial-time reductions**

3-SAT polytime reduces to Independent-Set

- Constraint satisfaction
- Independent-Set
- Dir-Ham-Cycle
- Graph-3-Color
- Subset-Sum
- Vertex-Cover
- Ham-Cycle
- Planar-3-Color
- Scheduling

packing and covering

partitioning

numerical
8. **Intractability I**

- poly-time reductions
- packing and covering problems
- constraint satisfaction problems
- sequencing problems
- partitioning problems
- graph coloring
- numerical problems

### Subset sum

**Theorem.** \(3\text{-SAT} \leq_p \text{SUBSET-SUM} \).  

**Pf.** Given an instance \(\Phi\) of 3-SAT, we construct an instance of \(\text{SUBSET-SUM}\) that has solution iff \(\Phi\) is satisfiable.

### 3-satisfiability reduces to subset sum

**Construction.** Given 3-SAT instance \(\Phi\) with \(n\) variables and \(k\) clauses, form \(2n + 2k\) decimal integers, each of \(n + k\) digits:

- Include one digit for each variable \(x_i\) and for each clause \(C_j\).
- Include two numbers for each variable \(x_i\).
- Include two numbers for each clause \(C_j\).
- Sum of each \(x_i\) digit is 1; sum of each \(C_j\) digit is 4.

**Key property.** No carries possible \(\Rightarrow\) each digit yields one equation.

\[
\begin{align*}
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]

**Example.** Given an instance of 3-SAT with variables \(x_1, x_2, x_3\) and clauses \(C_1 = \neg x_1 \lor x_2 \lor x_3, C_2 = x_1 \lor \neg x_2 \lor x_3, C_3 = \neg x_1 \lor \neg x_2 \lor \neg x_3\), we construct the corresponding subset sum instance as follows:

- For variable \(x_1\): \(1, 00,001, 100,010, 011,000\)
- For variable \(x_2\): \(0, 100,101, 100,111, 011,001\)
- For variable \(x_3\): \(0, 011,010, 110,001, 101,100\)
- For clause \(C_1\): \(100,000, 000,001, 001,000\)
- For clause \(C_2\): \(010,000, 000,011, 011,000\)
- For clause \(C_3\): \(100,100, 000,000, 000,100\)

The subset sum instance is:

- \(\Phi = \{1, 00,001, 100,010, 011,000, 0, 100,101, 100,111, 011,001, 0, 011,010, 110,001, 101,100, 100,000, 000,001, 001,000, 010,000, 000,011, 011,000, 100,100, 000,000, 000,100\}\)

The sum of \(\Phi\) is 3754, which is the sum of the subset sum instance.

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.

---

**Subset sum**

**SUBSET-SUM.** Given natural numbers \(w_1, \ldots, w_n\) and an integer \(W\), is there a subset that adds up to exactly \(W\)?

**Ex.** \(\{1, 4, 16, 64, 256, 1040, 1041, 1093, 1284, 1344\}\), \(W = 3754\).

**Yes.** \(1 + 16 + 64 + 256 + 1040 + 1093 + 1284 = 3754\).

---

**Remark.** With arithmetic problems, input integers are encoded in binary. Poly-time reduction must be polynomial in binary encoding.
Lemma. \( \Phi \) is satisfiable iff there exists a subset that sums to \( W \).

Pf. \( \Rightarrow \) Suppose \( \Phi \) is satisfiable.
   - Choose integers corresponding to each true literal.
   - Since \( \Phi \) is satisfiable, each \( C_j \) digit sums to at least 1 from \( x_i \) rows.
   - Choose dummy integers to make clause digits sum to 4.

\[
\begin{align*}
C_1 &= \neg x_1 \lor x_2 \lor x_3 \\
C_2 &= x_1 \lor \neg x_2 \lor x_3 \\
C_3 &= \neg x_1 \lor \neg x_2 \lor \neg x_3
\end{align*}
\]

\( 3 \)-Sat instance

\[
\begin{array}{cccccc}
\text{W} & 1 & 1 & 1 & 4 & 4 & 4 & 4 & 111,444 \\
\text{SUBSET-SUM instance}
\end{array}
\]

My hobby

**My Hobby:**

**EMBEDDING NP-COMPLETE PROBLEMS IN RESTAURANT ORDERS**

**COTTONS RESTAURANT**

**APPETIZERS**
- FRENCH FRIES 2.75
- SIDES SALAD 3.35
- HOT WINGS 5.55
- MOZZARELLA STICKS 4.20
- SAMPLER PLATE 5.80

**SANDWICHES**
- CRISPY CHICKEN 6.55

RANDALL MUNRO

http://xkcd.com/c287.html

3-satisfiability reduces to subset sum

**Lemma.** \( \Phi \) is satisfiable iff there exists a subset that sums to \( W \).

**Pf.** \( \Leftrightarrow \) Suppose there is a subset that sums to \( W \).
   - Digit \( x_i \) forces subset to select either row \( x_i \) or \( \neg x_i \) (but not both).
   - Digit \( C_j \) forces subset to select at least one literal in clause.
   - Assign \( x_i = \text{true} \) iff row \( x_i \) selected.

\[
\begin{array}{cccccc}
\text{x}_1 & 1 & 0 & 0 & 0 & 1 & 0 & 100,010 \\
\text{x}_2 & 1 & 0 & 0 & 1 & 0 & 1 & 100,101 \\
\neg x_2 & 0 & 1 & 0 & 0 & 1 & 0 & 10,100 \\
\neg x_3 & 0 & 1 & 0 & 0 & 1 & 1 & 10,011 \\
x_3 & 0 & 0 & 1 & 1 & 1 & 1 & 1,110 \\
\neg x_3 & 0 & 0 & 1 & 1 & 0 & 1 & 1,001 \\
\end{array}
\]

\( 3 \)-Sat instance

\[
\begin{array}{cccccc}
\text{W} & 1 & 1 & 1 & 4 & 4 & 4 & 4 & 111,444 \\
\text{SUBSET-SUM instance}
\end{array}
\]

Partition

**SUBSET-SUM.** Given natural numbers \( w_1, \ldots, w_n \) and an integer \( W \), is there a subset that adds up to exactly \( W \)?

**PARTITION.** Given natural numbers \( v_1, \ldots, v_m \), can they be partitioned into two subsets that add up to the same value \( \frac{1}{2} \sum v_i \)?

**Theorem.** \( \text{SUBSET-SUM} \leq_p \text{PARTITION} \).

**Pf.** Let \( W, w_1, \ldots, w_n \) be an instance of \text{SUBSET-SUM}.
   - Create instance of \text{PARTITION} with \( m = n + 2 \) elements.
     - \( v_1 = w_1, v_2 = w_2, \ldots, v_n = w_n, v_{n+1} = 2 \sum w_j - W, v_{n+2} = \sum w_j + W \)
   - Lemma: there exists a subset that sums to \( W \) iff there exists a partition since elements \( v_{n+1} \) and \( v_{n+2} \) cannot be in the same partition.

\[
\begin{array}{cc}
W & \text{subset A} \\
\sum w_j - W & \text{subset B}
\end{array}
\]
Scheduling with release times

**SCHEDULE.** Given a set of $n$ jobs with processing time $t_j$, release time $r_j$, and deadline $d_j$, is it possible to schedule all jobs on a single machine such that job $j$ is processed with a contiguous slot of $t_j$ time units in the interval $[r_j, d_j]$?

Ex.

---

**Theorem.** $\text{SUBSET-SUM} \leq_p \text{SCHEDULE}$. 

**Pf.** Given $\text{SUBSET-SUM}$ instance $w_1, \ldots, w_n$ and target $W$, construct an instance of $\text{SCHEDULE}$ that is feasible iff there exists a subset that sums to exactly $W$.

**Construction.**

- Create $n$ jobs with processing time $t_j = w_j$, release time $r_j = 0$, and no deadline ($d_j = 1 + \sum w_j$).
- Create job 0 with $t_0 = 1$, release time $r_0 = W$, and deadline $d_0 = W + 1$.
- Lemma: subset that sums to $W$ iff there exists a feasible schedule.

---

**Polynomial-time reductions**

- 3-SAT
- INDEPENDENT-SET
- VERTEX-COVER
- SET-COVER
- TSP
- HAM-CYCLE
- PLANAR-3-COLOR
- SCHEDULING
- GRAPH-3-COLOR

---

**Karp's 21 NP-complete problems**

Dick Karp (1972) 1985 Turing Award