18-0: **Language Class P**

- A language \( L \) is polynomially decidable if there exists a polynomially bound deterministic Turing machine that decides it.

- A Turing Machine \( M \) is polynomially bound if:
  - There exists some polynomial function \( p(n) \)
  - For any input string \( w \), \( M \) always halts within \( p(|w|) \) steps

- The set of languages that are polynomially decidable is \( P \)

18-1: **Language Class NP**

- A language \( L \) is non-deterministically polynomially decidable if there exists a polynomially bound non-deterministic Turing machine that decides it.

- A Non-Deterministic Turing Machine \( M \) is polynomially bound if:
  - There exists some polynomial function \( p(n) \)
  - For any input string \( w \), \( M \) always halts within \( p(|w|) \) steps, for all computational paths

- The set of languages that are non-deterministically polynomially decidable is \( \text{NP} \)

18-2: **Language Class NP**

- If a Language \( L \) is in \( \text{NP} \):
  - There exists a non-deterministic Turing machine \( M \)
  - \( M \) halts within \( p(|w|) \) steps for all inputs \( w \), in all computational paths
  - If \( w \in L \), then there is at least one computational path for \( w \) that accepts (and potentially several that reject)
  - If \( w \notin L \), then all computational paths for \( w \) reject

18-3: **NP vs P**

- A problem is in \( P \) if we can generate a solution quickly (that is, in polynomial time)

- A problem is in \( \text{NP} \) if we can check to see if a potential solution is correct quickly
  - Non-deterministically create (guess) a potential solution
  - Check to see that the solution is correct

18-4: **NP vs P**

- All problems in \( P \) are also in \( \text{NP} \)
  - That is, \( P \subseteq \text{NP} \)
  - If you can generate correct solutions, you can check if a guessed solution is correct

18-5: **NP Problems**

- Finding Hamiltonian Cycles is \( \text{NP} \)
  - Non-deterministically pick a permutation of the nodes of the graph
• First, non-deterministically pick any node in the graph, and place it first in the permutation
• Then, non-deterministically pick any unchosen node in the graph, and place it second in the permutation
• ... 
• Check to see if that permutation forms a valid cycle

18-6: NP Problems

• Traveling Salesman decision problem is NP
  • Non-deterministically pick a permutation of the nodes of the graph
    • First, non-deterministically pick any node in the graph, and place it first in the permutation
    • Then, non-deterministically pick any unchosen node in the graph, and place it second in the permutation
    • ... 
    • Check to see if the cost of that cycle is within the cost bound.

18-7: Integer Partition

• Integer Partition is NP
  • Non-deterministically pick a subset $P \subset S$
  • Check to see if:
    $$\sum_{p \in P} p = \sum_{s \in S - P} s$$

18-8: NP Problems

• Satisfiability is NP
  • Count the number of variables in the formula
  • Non-deterministically write down True or False for each of the $n$ variables in the formula
  • Check to see if that truth assignment satisfies the formula

18-9: Reduction Redux

• Given a problem instance $P$, if we can
  • Create an instance of a different problem $P'$, in polynomial time, such that the solution to $P'$ is the same as the solution to $P$
  • Solve the instance $P'$ in polynomial time
  • Then we can solve $P$ in polynomial time

18-10: Reduction Example

• If we could solve the Traveling Salesman decision problem in polynomial time, we could solve the Hamiltonian Cycle problem in polynomial time
  • Given any graph $G$, we can create a new graph $G'$ and limit $k$, such that there is a Hamiltonian Circuit in $G$ if and only if there is a Traveling Salesman tour in $G'$ with cost less than $k$
  • Vertices in $G'$ are the same as the vertices in $G$
• For each pair of vertices $x_i$ and $x_j$ in $G$, if the edge $(x_i, x_j)$ is in $G$, add the edge $(x_i, x_j)$ to $G'$ with the cost 1. Otherwise, add the edge $(x_i, x_j)$ to $G'$ with the cost 2.

• Set the limit $k = \#$ of vertices in $G$

18-11: Reduction Example

![Graph with edges and costs]

Limit = 4

18-12: Reduction Example

• If we could solve TSP in polynomial time, we could solve Hamiltonian Cycle problem in polynomial time
  • Start with an instance of Hamiltonian Cycle
  • Create instance of TSP
  • Feed instance of TSP into TSP solver
  • Use result to find solution to Hamiltonian Cycle

18-13: Reduction Example #2

• Given any instance of the Hamiltonian Cycle Problem:
  • We can (in polynomial time) create an instance of Satisfiability
  • That is, given any graph $G$, we can create a boolean formula $f$, such that $f$ is satisfiable if and only if there is a Hamiltonian Cycle in $G$
  • If we could solve Satisfiability in Polynomial Time, we could solve the Hamiltonian Cycle problem in Polynomial Time

18-14: Reduction Example #2

• Given a graph $G$ with $n$ vertices, we will create a formula with $n^2$ variables:
  - $x_{11}, x_{12}, x_{13}, \ldots, x_{1n}$
  - $x_{21}, x_{22}, x_{23}, \ldots, x_{2n}$
  - $\ldots$
  - $x_{n1}, x_{n2}, x_{n3}, \ldots, x_{nn}$

• Design our formula such that $x_{ij}$ will be true if and only if the $i$th element in a Hamiltonian Circuit of $G$ is vertex # $j$

18-15: Reduction Example #2

• For our set of $n^2$ variables $x_{ij}$, we need to write a formula that ensures that:
  • For each $i$, there is exactly one $j$ such that $x_{ij} = true$
• For each $j$, there is exactly one $i$ such that $x_{ij} = true$

• If $x_{ij}$ and $x_{(i+1)k}$ are both true, then there must be a link from $v_j$ to $v_k$ in the graph $G$

18-16: **Reduction Example #2**

• For each $i$, there is exactly one $j$ such that $x_{ij} = true$

  • For each $i$ in $1 \ldots n$, add the rules:
    • $(x_{i1} \lor x_{i2} \lor \ldots \lor x_{in})$

  • This ensures that for each $i$, there is at least one $j$ such that $x_{ij} = true$

  • (This adds $n$ clauses to the formula)

18-17: **Reduction Example #2**

• For each $i$, there is exactly one $j$ such that $x_{ij} = true$

  for each $i$ in $1 \ldots n$

  for each $j$ in $1 \ldots n$

  for each $k$ in $1 \ldots n$

  $j \neq k$

  Add rule $(x_{ij} \lor x_{ik})$

• This ensures that for each $i$, there is at most one $j$ such that $x_{ij} = true$

• (This adds a total of $n^3$ clauses to the formula)

18-18: **Reduction Example #2**

• For each $j$, there is exactly one $i$ such that $x_{ij} = true$

  • For each $j$ in $1 \ldots n$, add the rules:
    • $(x_{1j} \lor x_{2j} \lor \ldots \lor x_{nj})$

  • This ensures that for each $j$, there is at least one $i$ such that $x_{ij} = true$

  • (This adds $n$ clauses to the formula)

18-19: **Reduction Example #2**

• For each $j$, there is exactly one $i$ such that $x_{ij} = true$

  for each $j$ in $1 \ldots n$

  for each $i$ in $1 \ldots n$

  for each $k$ in $1 \ldots n$

  Add rule $(\overline{x_{ij}} \lor \overline{x_{kj}})$

• This ensures that for each $j$, there is at most one $i$ such that $x_{ij} = true$

• (This adds a total of $n^3$ clauses to the formula)

18-20: **Reduction Example #2**
• If \( x_{ij} \) and \( x_{(i+1)k} \) are both true, then there must be a link from \( v_i \) to \( v_k \) in the graph \( G \)

  for each \( i \) in \( 1 \ldots (n - 1) \)
  for each \( j \) in \( 1 \ldots n \)
  for each \( k \) in \( 1 \ldots n \)
  if edge \((v_j, v_k)\) is not in the graph:
    Add rule \( x_{ij} \lor x_{(i+1)k} \)

• (This adds no more than \( n^3 \) clauses to the formula)

18-21: Reduction Example #2

• If \( x_{nj} \) and \( x_{0k} \) are both true, then there must be a link from \( v_j \) to \( v_k \) in the graph \( G \) (looping back to finish cycle)

  for each \( j \) in \( 1 \ldots n \)
  for each \( k \) in \( 1 \ldots n \)
  if edge \((v_j, v_k)\) is not in the graph:
    Add rule \( x_{nj} \lor x_{0k} \)

• (This adds no more than \( n^2 \) clauses to the formula)

18-22: Reduction Example #2

• In order for this formula to be satisfied:
  - For each \( i \), there is exactly one \( j \) such that \( x_{ij} \) is true
  - For each \( j \), there is exactly one \( i \) such that \( x_{ji} \) is true
  - if \( x_{ij} \) is true, and \( x_{(i+1)k} \) is true, then there is an arc from \( v_j \) to \( v_k \) in the graph \( G \)

• Thus, the formula can only be satisfied if there is a Hamiltonian Cycle of the graph

18-23: NP-Complete

• A language \( L \) is NP-Complete if:
  - \( L \) is in NP
  - If we could decide \( L \) in polynomial time, then all NP languages could be decided in polynomial time
  - That is, we could reduce any NP problem to \( L \) in polynomial time

18-24: NP-Complete

• How do you show a problem is NP-Complete?
  - Given any polynomially-bound non-deterministic Turing machine \( M \) and string \( w \):
    - Create an instance of the problem that has a solution if and only if \( M \) accepts \( w \)

18-25: NP-Complete

• First NP-Complete Problem: Satisfiability (SAT)
  - Given any (possibly non-deterministic) Turing Machine \( M \), string \( w \), and polynomial bound \( p(n) \)
• Create a boolean formula $f$, such that $f$ is satisfiable if and only of $M$ accepts $w$

18-26: **Cook’s Theorem**

• Satisfiability is NP-Complete
  • Given a Turing Machine $M$, string $w$, polynomial bound $p(n)$, we will create:
    • A set of variables
    • A set of clauses containing these variables
  • Such that the conjunction ($\land$) of the clauses is satisfiable if and only if $M$ accepts $w$ within $p(|w|)$ steps
  • WARNING: This explanation is somewhat simplified. Some subtleties have been eliminated for clarity.

18-27: **Cook’s Theorem**

• Variables
  • $Q[i, k]$ at time $i$, machine is in state $q_k$
  • $H[i, j]$ at time $i$, the machine is scanning tape square $j$
  • $S[i, j, k]$ at time $i$, the contents of tape location $j$ is the symbol $k$
  • How many of each of these variables are there?

18-28: **Cook’s Theorem**

• Variables
  • $Q[i, k]$ $|K| \cdot p(|w|)$
  • $H[i, j]$ $p(|w|) \cdot p(|w|)$
  • $S[i, j, k]$ $p(|w|) \cdot p(|w|) \cdot |\Sigma|$
  • How many of each of these variables are there?

18-29: **Cook’s Theorem**

$G_1$ At each time $i$, $M$ is in exactly one state

$G_2$ At each time $i$, the read-write head is scanning one tape square

$G_3$ At each time $i$, each tape square contains exactly one symbol

$G_4$ At time 0, the computation is in the initial configuration for input $w$

$G_5$ By time $p(|w|)$, $M$ has entered the final state and has hence accepted $w$

$G_6$ For each time $i$, the configuration of the $M$ at $i + 1$ follows by a single application of $\delta$

18-30: **Cook’s Theorem**

$G_1$ At each time $i$, $M$ is in exactly one state
\((Q[i, 0] \lor Q[i, 1] \lor \ldots \lor Q[i, |K|])\)

for each \(0 \leq i \leq p(|w|)\)

\((Q[i, j] \lor Q[i, j'])\)

for each \(0 \leq i \leq p(|w|), 0 \leq j < j' \leq |K|\)

18-31: **Cook’s Theorem**

**G_2**  At each time \(i\), the read-write head is scanning one tape square

\((H[i, 0] \lor H[i, 1] \lor \ldots \lor H[i, p(|w|)])\)

for each \(0 \leq i \leq p(|w|)\)

\((H[i, j] \lor H[i, j'])\)

for each \(0 \leq i \leq p(|w|), 0 \leq j < j' \leq p(|w|)\)

18-32: **Cook’s Theorem**

**G_3**  At each time \(i\), each tape square contains exactly one symbol

\((S[i, j, 0] \lor S[i, j, 1] \lor \ldots \lor S[i, j, |\Sigma|])\)

for each \(0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|)\)

\((S[i, j, k] \lor S[i, j, k'])\)

for each \(0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|), 0 \leq k < k' \leq |\Sigma|\)

18-33: **Cook’s Theorem**

**G_4**  At time 0, the computation is in the initial configuration for input \(w\)

\(Q[0, 0]\)

\(H[0, 1]\)

\(S[0, 0, 0]\)

\(S[0, 1, w_1]\)

\(S[0, 2, w_2]\)

\(\ldots\)

\(S[0, |w|, w_{|w|}]\)

\(S[0, |w| + 1, 0]\)

\(S[0, |w| + 2, 0]\)

\(\ldots\)

\(S[0, p(|w|), 0]\)

18-34: **Cook’s Theorem**

**G_5**  By time \(p(|w|)\), \(M\) has entered the final state and has hence accepted \(w\)
Where \( q_r \) is the accept state

18-35: **Cook’s Theorem**

\( G_6 \) For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \)

For each deterministic transition \( ((q_k, \Sigma_a), (q_l, \rightarrow)) \)

For all \( 0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|) \)

Add:
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow H[i + 1, j + 1]
\]
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow Q[i + 1, l]
\]

18-36: **Cook’s Theorem**

\( G_6 \) For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \)

For each deterministic transition \( ((q_k, \Sigma_a), (q_l, \leftarrow)) \)

For all \( 0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|) \)

Add:
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow H[i + 1, j - 1]
\]
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow Q[i + 1, l]
\]

18-37: **Cook’s Theorem**

\( G_6 \) For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \)

For each deterministic transition \( ((q_k, \Sigma_a), (q_l, \Sigma_b)) \)

For all \( 0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|) \)

Add:
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow H[i + 1, j]
\]
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow Q[i + 1, l]
\]
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow S[i, j, b]
\]

18-38: **Cook’s Theorem**

\( G_6 \) For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \)

For each non-deterministic transition \( ((q_k, \Sigma_a), (q_l, \rightarrow)) \) and \( ((q_k, \Sigma_a), (q_m, \rightarrow)) \)

For all \( 0 \leq i \leq p(|w|), 0 \leq j \leq p(|w|) \)

Add:
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow H[i + 1, j + 1]
\]
\[
Q[i, k] \land H[i, j] \land S[i, j, a] \Rightarrow Q[i + 1, l] \lor Q[i + 1, m]
\]

18-39: **Cook’s Theorem**

\( G_6 \) For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \)

- ... similar rules for other non-deterministic cases
Cook’s Theorem

For each time \( i \), the configuration of the \( M \) at \( i + 1 \) follows by a single application of \( \delta \):

\[
H[i, j] \land S[i, k, a] \Rightarrow S[i + 1, k, a]
\]

for all values of \( k, j \) between 0 and \( p(|w|) \) where \( k \neq j \), and all values \( 0 \leq a < |\Sigma| \)

More NP-Complete Problems

So, if we could solve Satisfiability in Polynomial Time, we could solve any \( \text{NP} \) problem in polynomial time

- Including factoring large numbers ...
- Satisfiability is \( \text{NP} \)-Complete
- There are many \( \text{NP} \)-Complete problems
  - Prove \( \text{NP} \)-Completeness using a reduction

More NP-Complete Problems

- Exact Cover Problem
  - Set of elements \( A \)
  - \( F \subset 2^A \), family of subsets
  - Is there a subset of \( F \) such that each element of \( A \) appears exactly once?

More NP-Complete Problems

- Exact Cover Problem
  - \( A = \{a, b, c, d, e, f, g\} \)
  - \( F = \{\{a, b, c\}, \{d, e, f\}, \{b, f, g\}, \{g\}\} \)
  - Exact cover exists:
    \[\{a, b, c\}, \{d, e, f\}, \{g\}\]

More NP-Complete Problems

- Exact Cover Problem
  - \( A = \{a, b, c, d, e, f, g\} \)
  - \( F = \{\{a, b, c\}, \{c, d, e, f\}, \{a, f, g\}, \{c\}\} \)
  - No exact cover exists

More NP-Complete Problems

- Exact Cover is in \( \text{NP} \)
  - Guess a cover
  - Check that each element appears exactly once
- Exact Cover is \( \text{NP} \)-Complete
• Reduction from Satisfiability
• Given any instance of Satisfiability, create (in polynomial time) an instance of Exact Cover

18-46: Exact Cover is NP-Complete

• Given an instance of SAT:
  • \( C_1 = (x_1, \overline{x_2}) \)
  • \( C_2 = (\overline{x_1} \lor x_2 \lor x_3) \)
  • \( C_3 = (x_2) \)
  • \( C_4 = (\overline{x_2}, \overline{x_3}) \)
• Formula: \( C_1 \land C_2 \land C_3 \land C_4 \)
• Create an instance of Exact Cover
  • Define a set \( A \) and family of subsets \( F \) such that there is an exact cover of \( A \) in \( F \) if and only if the formula is satisfiable

18-47: Exact Cover is NP-Complete

\[
C_1 = (x_1 \lor \overline{x_2}) \quad C_2 = (\overline{x_1} \lor x_2 \lor x_3) \quad C_3 = (x_2) \quad C_4 = (\overline{x_2}, \overline{x_3})
\]
\[
A = \{x_1, x_2, x_3, C_1, C_2, C_3, C_4, p_{11}, p_{12}, p_{21}, p_{22}, p_{23}, p_{31}, p_{41}, p_{42}\}
\]
\[
F = \{\{p_{11}\}, \{p_{12}\}, \{p_{21}\}, \{p_{22}\}, \{p_{23}\}, \{p_{31}\}, \{p_{41}\}, \{p_{42}\}\},
\]
\[
X_1, f = \{x_1, p_{11}\}
\]
\[
X_1, t = \{x_1, p_{21}\}
\]
\[
X_2, f = \{x_2, p_{22}, p_{31}\}
\]
\[
X_2, t = \{x_2, p_{12}, p_{41}\}
\]
\[
X_3, f = \{x_3, p_{23}\}
\]
\[
X_3, t = \{x_3, p_{42}\}
\]
\[
\{C_1, p_{11}\}, \{C_1, p_{12}\}, \{C_2, p_{21}\}, \{C_2, p_{22}\}, \{C_2, p_{23}\}, \{C_3, p_{31}\}, \{C_4, p_{41}\}, \{C_4, p_{42}\}\} \quad 18-48: Knapsack
\]
• Given a set of integers \( S \) and a limit \( k \):
  • Is there some subset of \( S \) that sums to \( k \)?
• \{3, 5, 11, 15, 20, 25\} Limit: 36
  • \{5, 11, 20\}
• \{2, 5, 10, 12, 20, 27\} Limit: 43
  • No solution
• Generalized version of Integer Partition problem

18-49: Knapsack

• Knapsack is NP-Complete
• By reduction from Exact Cover
  • Given any Exact Cover problem (set \( A \), family of subsets \( F \)), we will create a Knapsack problem (set \( S \), limit \( k \)), such that there is a subset of \( S \) that sums to \( k \) if and only if there is an exact cover of \( A \) in \( F \)
• Each set will be represented by a number – bit-vector representation of the set
  \[ A = \{a_1, a_2, a_3, a_4\} \]

<table>
<thead>
<tr>
<th>Set</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 = {a_1, a_2, a_3} )</td>
<td>1110</td>
</tr>
<tr>
<td>( F_2 = {a_2, a_4} )</td>
<td>0101</td>
</tr>
<tr>
<td>( F_3 = {a_1, a_3} )</td>
<td>1010</td>
</tr>
<tr>
<td>( F_4 = {a_2, a_3, a_4} )</td>
<td>0111</td>
</tr>
</tbody>
</table>

There is an exact cover if some subset of the numbers sum to ...

18-51: **Knapsack**

• Each set will be represented by a number – bit-vector representation of the set
  \[ A = \{a_1, a_2, a_3, a_4\} \]

<table>
<thead>
<tr>
<th>Set</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 = {a_1, a_2, a_3} )</td>
<td>1110</td>
</tr>
<tr>
<td>( F_2 = {a_2, a_4} )</td>
<td>0101</td>
</tr>
<tr>
<td>( F_3 = {a_1, a_3} )</td>
<td>1010</td>
</tr>
<tr>
<td>( F_4 = {a_2, a_3, a_4} )</td>
<td>0111</td>
</tr>
</tbody>
</table>

There is an exact cover if some subset of the numbers sum to 1111

18-52: **Knapsack**

• Bug in our reduction:
  \[ A = \{a_1, a_2, a_3, a_4\} \]

<table>
<thead>
<tr>
<th>Set</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 = {a_2, a_3, a_4} )</td>
<td>0111</td>
</tr>
<tr>
<td>( F_2 = {a_2, a_4} )</td>
<td>0101</td>
</tr>
<tr>
<td>( F_3 = {a_3} )</td>
<td>0010</td>
</tr>
<tr>
<td>( F_3 = {a_4} )</td>
<td>0001</td>
</tr>
<tr>
<td>( F_4 = {a_1, a_3, a_4} )</td>
<td>1011</td>
</tr>
</tbody>
</table>

• \( 0111 + 0101 + 0001 + 0010 = 1111 \)

• What can we do?

18-53: **Knapsack**

• Construct the numbers just as before

• Do addition in base \( m \), where \( m \) is the number of element in \( A \). \( A = \{a_1, a_2, a_3, a_4\} \)

<table>
<thead>
<tr>
<th>Set</th>
<th>Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>( F_1 = {a_2, a_3, a_4} )</td>
<td>0111</td>
</tr>
<tr>
<td>( F_2 = {a_2, a_4} )</td>
<td>0101</td>
</tr>
<tr>
<td>( F_3 = {a_3} )</td>
<td>0010</td>
</tr>
<tr>
<td>( F_3 = {a_4} )</td>
<td>0001</td>
</tr>
<tr>
<td>( F_4 = {a_1, a_3, a_4} )</td>
<td>1011</td>
</tr>
</tbody>
</table>

• \( 0111 + 0101 + 0001 + 0010 = 0223 \)
• No subset of numbers sums to 1111

18-54: Integer Partition

• Integer Partition
  • Special Case of the Knapsack problem
  • “Half sum” \( H \) (sum of all elements in the set / 2) is an integer
  • Limit \( k = H \)

• Integer Partition is \( \text{NP} \)-Complete
  • Reduce Knapsack to Integer Partition

18-55: Integer Partition

• Given any instance of the Knapsack problem
  • Set of integers \( S = \{a_1, a_2, \ldots, a_n\} \) limit \( k \)
  • Is there a subset of \( S \) that sums to \( k \)?

• Create an instance of Integer Partition
  • Set of integers \( S = \{a_1, a_2, \ldots, a_m\} \)
  • Can we divide \( S \) into two subsets that have the same sum?
  • Equivalently, is there a subset if \( S \) that sums to \( H = (\sum_{i=1}^m a_i)/2 \)

18-56: Integer Partition

• Given any instance of the Knapsack problem
  • Set of integers \( S = \{a_1, a_2, \ldots, a_n\} \) limit \( k \)

• We create the following instance of Integer Partition:
  • \( S' = S \cup \{2H + 2k, 4H\} \) (\( H \) is the half sum of \( S \))

18-57: Integer Partition

• \( S' = S \cup \{2H + 2k, 4H\} \) (\( H \) is the half sum of \( S \))
  • If there is a partition for \( S' \), \( 2H + 2k \) and \( 4H \) must be in separate partitions (why)?

\[
4H + \sum_{a_i \in P} a_i = 2H + 2k + \sum_{a_j \in S - P} a_j
\]

18-58: Integer Partition

\[
4H + \sum_{a_i \in P} a_i = 2H + 2k + \sum_{a_j \in S - P} a_j
\]
• Adding $\sum_{a_i \in P} a_i$ to both sides:

\[
4H + 2 \sum_{a_i \in P} a_i = 2H + 2k + \sum_{a_j \in S} a_j \\
4H + 2 \sum_{a_i \in P} a_i = 4H + 2k \\
\sum_{a_i \in P} a_i = k
\]

• Thus, if $S'$ has a partition, then there must be some subset of $S$ that sums to $k$

18-59: **Directed Hamiltonian Cycle**

• Given any directed graph $G$, determine if $G$ has a Hamiltonian Cycle
  - Cycle that includes every node in the graph exactly once, following the direction of the arrows

![Directed Hamiltonian Cycle](image1.png)

18-60: **Directed Hamiltonian Cycle**

• Given any directed graph $G$, determine if $G$ has a Hamiltonian Cycle
  - Cycle that includes every node in the graph exactly once, following the direction of the arrows

![Directed Hamiltonian Cycle](image2.png)

18-61: **Directed Hamiltonian Cycle**

• The Directed Hamiltonian Cycle problem is NP-Complete
• Reduce Exact Cover to Directed Hamiltonian Cycle
  - Given any set $A$, and family of subsets $F$:
    - Create a graph $G$ that has a hamiltonian cycle if and only if there is an exact cover of $A$ in $F$

18-62: **Directed Hamiltonian Cycle**

• Widgets:
Consider the following graph segment:

If a graph containing this subgraph has a Hamiltonian cycle, then the cycle must contain either $a \rightarrow u \rightarrow v \rightarrow w \rightarrow b$ or $c \rightarrow w \rightarrow v \rightarrow u \rightarrow d$ – but not both (why)?

18-63: Directed Hamiltonian Cycle

- Widgets:
  - XOR edges: Exactly one of the edges must be used in a Hamiltonian Cycle

18-64: Directed Hamiltonian Cycle

- Widgets:
  - XOR edges: Exactly one of the edges must be used in a Hamiltonian Cycle

18-65: Directed Hamiltonian Cycle

- Add a vertex for every variable in $A$ (+ 1 extra)
18-66: Directed Hamiltonian Cycle

- Add a vertex for every subset $F$ (+ 1 extra)

18-67: Directed Hamiltonian Cycle

- Add an edge from the last variable to the 0th subset, and from the last subset to the 0th variable
18-68: Directed Hamiltonian Cycle

- Add 2 edges from $F_i$ to $F_{i+1}$. One edge will be a “short edge”, and one will be a “long edge”. 

18-69: Directed Hamiltonian Cycle

- Add an edge from $a_{i-1}$ to $a_i$ for each subset $a_i$ appears in.
18-70: Directed Hamiltonian Cycle

- Each edge \((a_{i-1}, a_i)\) corresponds to some subset that contains \(a_i\). Add an XOR link between this edge and the long edge of the corresponding subset.

18-71: Directed Hamiltonian Cycle

\[
F_0 = \{ a_1, a_2 \} \\
F_1 = \{ a_2, a_3 \} \\
F_2 = \{ a_3 \} \\
F_3 = \{ a_1 \}
\]
Directed Hamiltonian Cycle

$F_1 = \{ a_2, a_4 \}$
$F_2 = \{ a_2, a_1 \}$
$F_3 = \{ a_1, a_3 \}$
$F_4 = \{ a_2 \}$

XOR edge