More Reductions for **NP** Problems

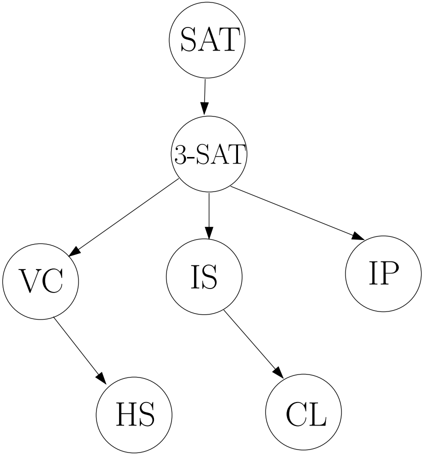
Nabil Mustafa

Computational Complexity

Reductions From Last Time

So far, we have shown the following problems **NP** complete:

- SAT , 3-CNF SAT, INDSET , CLIQUE , VERTEX-COVER , HITTING-SET , INTEGER-PROGRAMMING



HAMILTONIAN

Claim

HAMILTONIAN : Given a directed graph $G = (V, E)$, does G have a Hamiltonian path?
*The Hamiltonian path problem (HAMILTONIAN) is **NP** complete.*

HAMILTONIAN

Claim

HAMILTONIAN : Given a directed graph $G = (V, E)$, does G have a Hamiltonian path?
*The Hamiltonian path problem (HAMILTONIAN) is **NP** complete.*

- Review: Hamiltonian path visits each vertex in G exactly once

HAMILTONIAN

Claim

HAMILTONIAN : Given a directed graph $G = (V, E)$, does G have a Hamiltonian path?
*The Hamiltonian path problem (HAMILTONIAN) is **NP** complete.*

- Review: Hamiltonian path visits each vertex in G exactly once
- Computationally very different from Eulerian paths

HAMILTONIAN

Claim

HAMILTONIAN : Given a directed graph $G = (V, E)$, does G have a Hamiltonian path?
*The Hamiltonian path problem (HAMILTONIAN) is **NP** complete.*

- Review: Hamiltonian path visits each vertex in G exactly once
- Computationally very different from Eulerian paths
- Note that HAMILTONIAN is in **NP**

HAMILTONIAN

Claim

HAMILTONIAN : Given a directed graph $G = (V, E)$, does G have a Hamiltonian path?
*The Hamiltonian path problem (HAMILTONIAN) is **NP** complete.*

- Review: Hamiltonian path visits each vertex in G exactly once
- Computationally very different from Eulerian paths
- Note that HAMILTONIAN is in **NP**
- We reduce 3-CNF SAT to HAMILTONIAN
  - For a fixed 3-CNF SAT formula, show that it can be transformed into a graph whose Hamiltonian path will give us the assignments for SAT.

Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

- Given ϕ, have to construct a graph G such that G has a Hamiltonian path iff ϕ is satisfiable.

Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

- Given ϕ, have to construct a graph G such that G has a Hamiltonian path iff ϕ is satisfiable.
- We first define the mapping of variables, and then the clauses.

Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

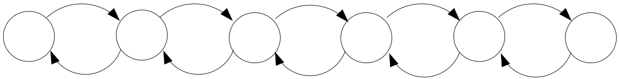
- Given ϕ, have to construct a graph G such that G has a Hamiltonian path iff ϕ is satisfiable.
- We first define the mapping of variables, and then the clauses.
- Each x_i will correspond to a path (chain) of $6m$ vertices

Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

- Given ϕ, have to construct a graph G such that G has a Hamiltonian path iff ϕ is satisfiable.
- We first define the mapping of variables, and then the clauses.
- Each x_i will correspond to a path (chain) of $6m$ vertices

x_i

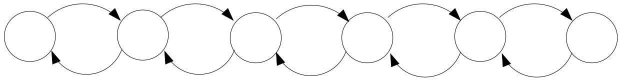


Reduction Sketch

$$\phi = \mathcal{C}_1 \wedge \mathcal{C}_2 \wedge \ldots \wedge \mathcal{C}_m, \quad n \text{ variables}, \quad \mathcal{C}_i = (x_1^i \vee x_2^i \vee x_3^i)$$

- Given ϕ, have to construct a graph G such that G has a Hamiltonian path iff ϕ is satisfiable.
- We first define the mapping of variables, and then the clauses.
- Each x_i will correspond to a path (chain) of $6m$ vertices
- If we are at the first (or end) vertex, only one path to follow

x_i



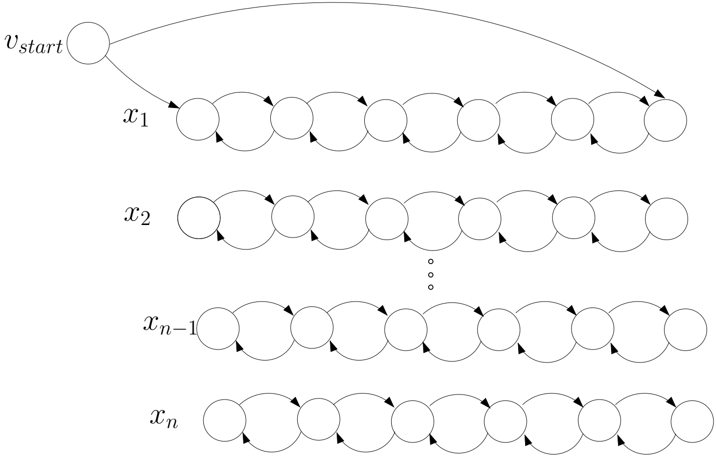
- Add a start vertex v_{start} which has

- Add a start vertex v_{start} which has
  - no incoming edges.

- Add a start vertex v_{start} which has
  - no incoming edges.
  - an outgoing edge to the first vertex of x_1 chain.

- Add a start vertex v_{start} which has
  - no incoming edges.
  - an outgoing edge to the first vertex of x_1 chain.
  - an outgoing edge to the last vertex of x_1 chain.

- Add a start vertex v_{start} which has
  - no incoming edges.
  - an outgoing edge to the first vertex of x_1 chain.
  - an outgoing edge to the last vertex of x_1 chain.



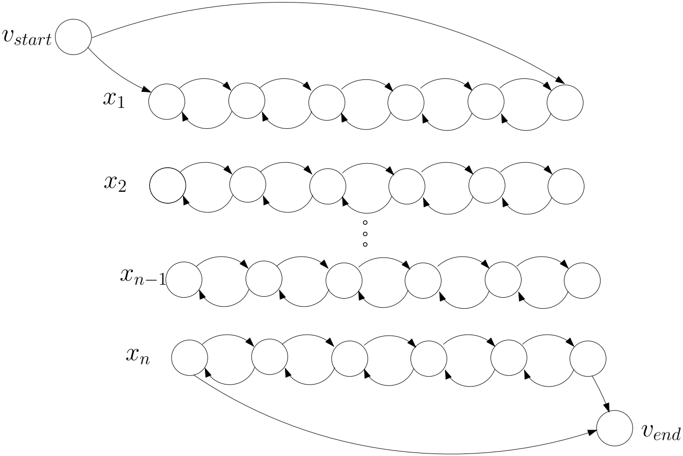
- Add an end vertex v_{end} which has

- Add an end vertex v_{end} which has
  - no outgoing edges.

- Add an end vertex v_{end} which has
  - no outgoing edges.
  - an incoming edge from the first vertex of x_n chain.

- Add an end vertex v_{end} which has
  - no outgoing edges.
  - an incoming edge from the first vertex of x_n chain.
  - an incoming edge from the last vertex of x_n chain.

- Add an end vertex v_{end} which has
  - no outgoing edges.
  - an incoming edge from the first vertex of x_n chain.
  - an incoming edge from the last vertex of x_n chain.

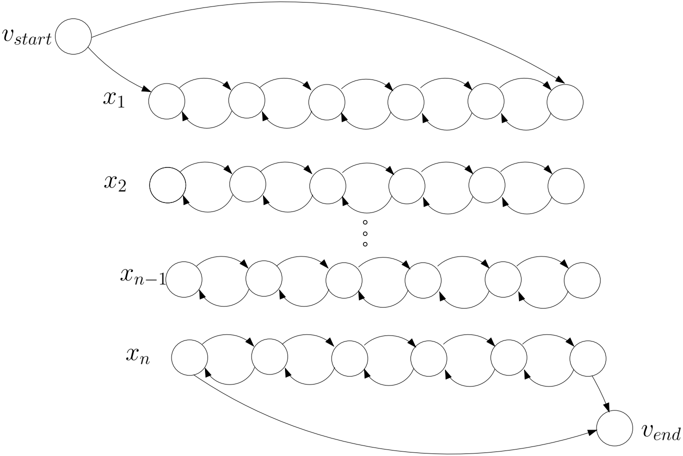


- From the first and last vertex of each chain x_i, add

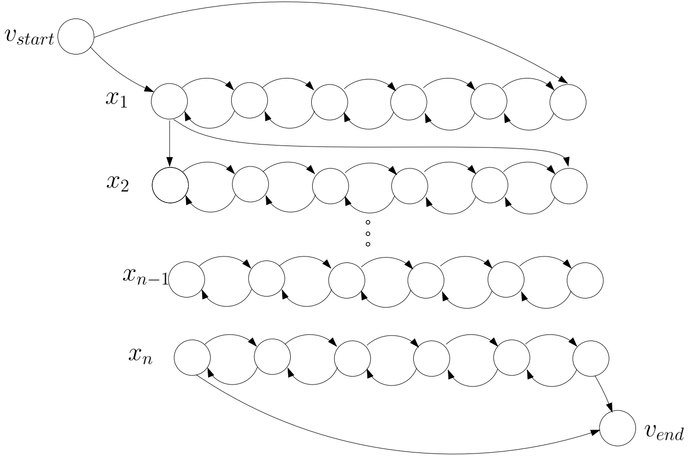
- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}

- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}
  - an outgoing edge to the last vertex of chain x_{i+1}

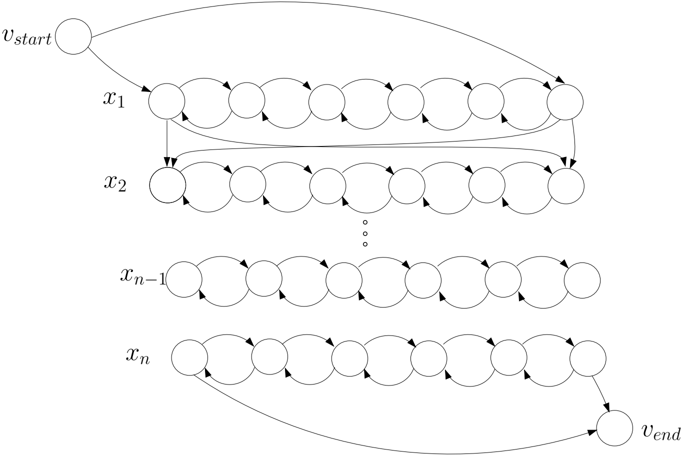
- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}
  - an outgoing edge to the last vertex of chain x_{i+1}



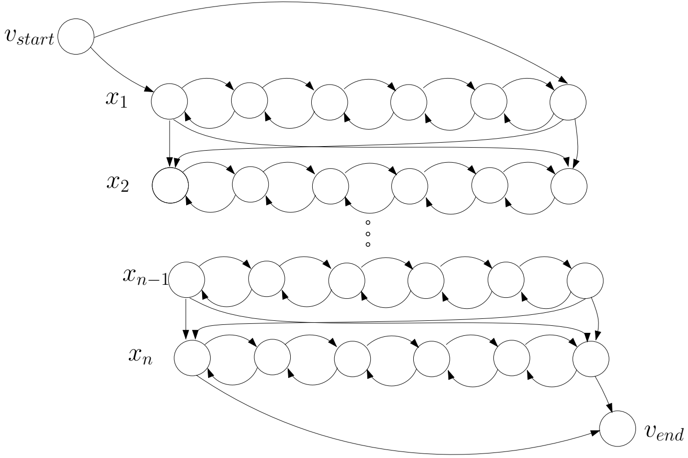
- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}
  - an outgoing edge to the last vertex of chain x_{i+1}



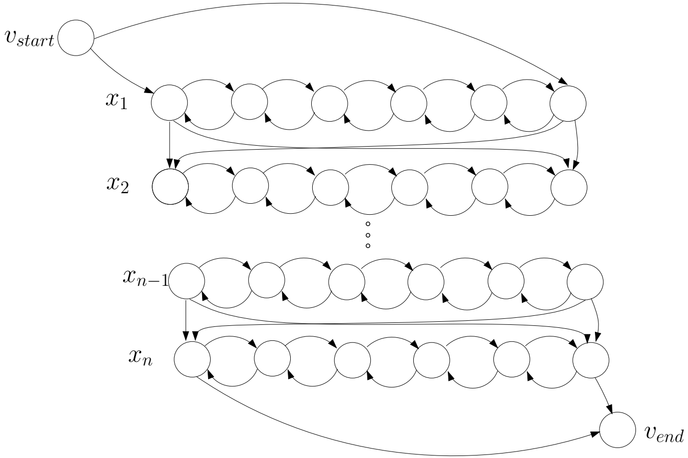
- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}
  - an outgoing edge to the last vertex of chain x_{i+1}



- From the first and last vertex of each chain x_i, add
  - an outgoing edge to the first vertex of chain x_{i+1}
  - an outgoing edge to the last vertex of chain x_{i+1}

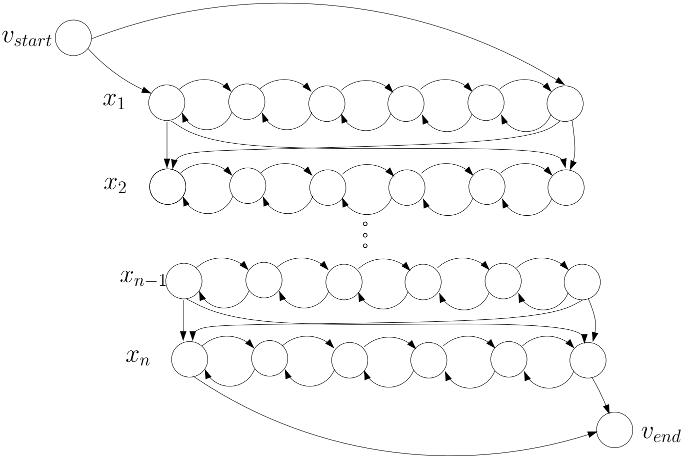


Construction Properties



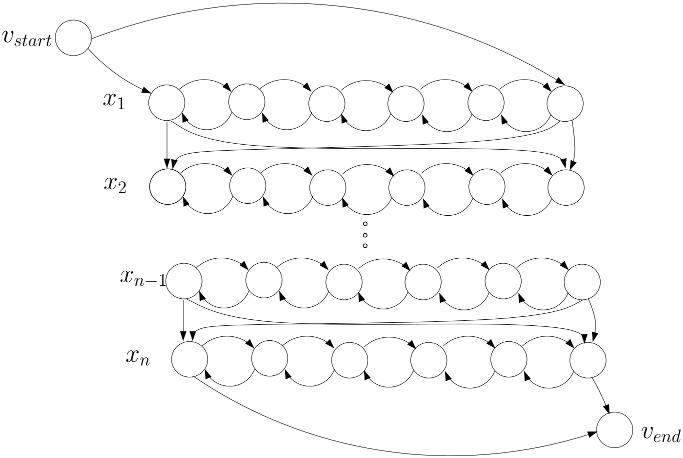
Construction Properties

- Any Hamiltonian path has to start at v_{start}



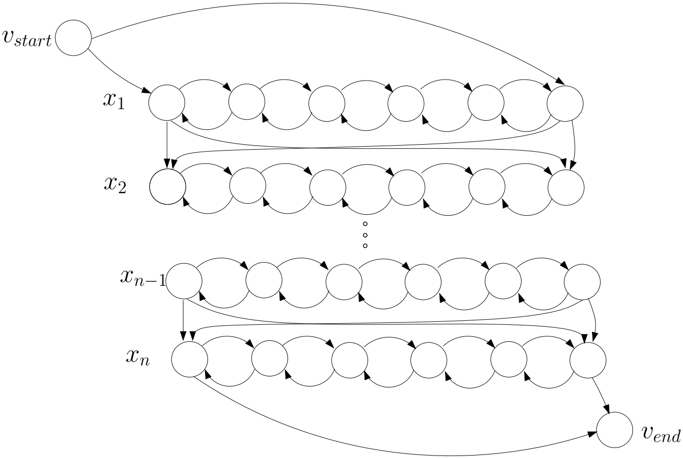
Construction Properties

- Any Hamiltonian path has to end at v_{end}



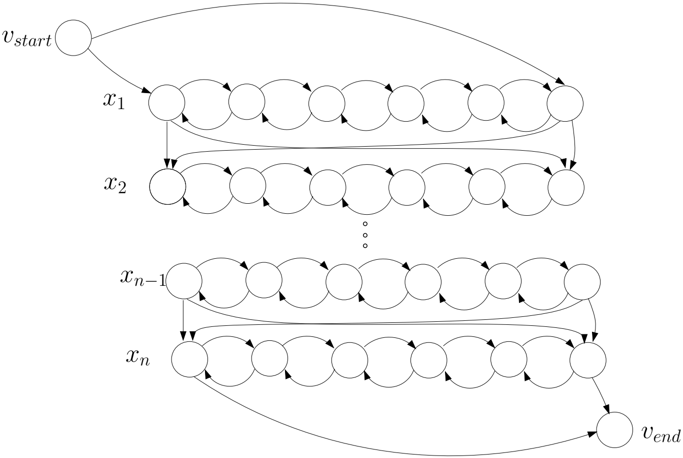
Construction Properties

- Any Hamiltonian path first traverses chain x_1, then x_2 etc.



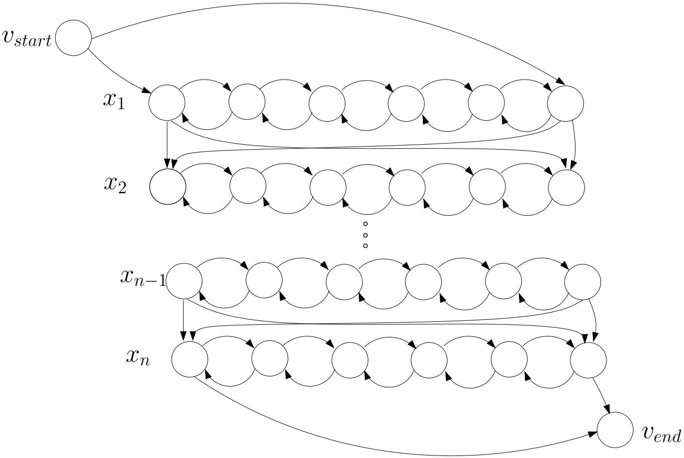
Construction Properties

- For each chain, only two ways of traversing it.
  - Left-to-right means $x_i = 1$, right-to-left means $x_i = 0$



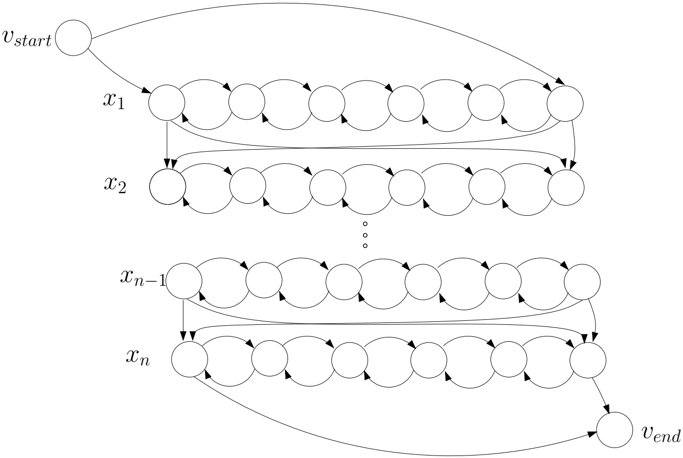
Construction Properties

- Each assignment of variables corresponds to a unique Hamiltonian path.



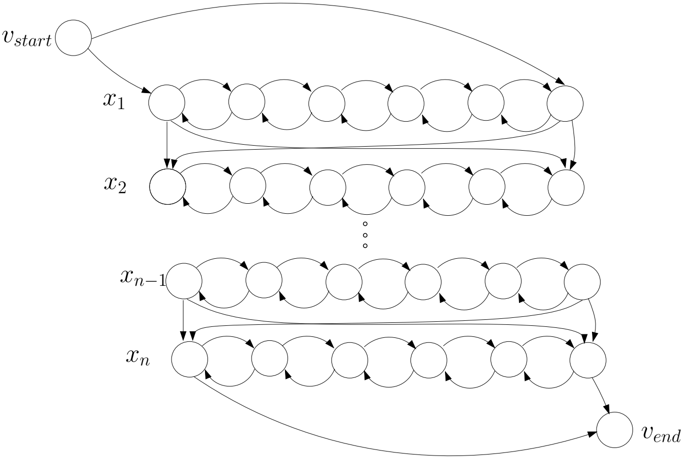
Construction Properties

- Each Hamiltonian path corresponds to a unique variable assignment.



Construction Properties

- So far, no constraints – they will come from the clauses now.



Clause Constraints

- Each clause $\mathcal{C}_j$ corresponds to a new vertex u_j.

Clause Constraints

- Each clause $\mathcal{C}_j$ corresponds to a new vertex u_j.
- If $\mathcal{C}_j$ contains a non-negated literal x_i, add edges to u_j:

Clause Constraints

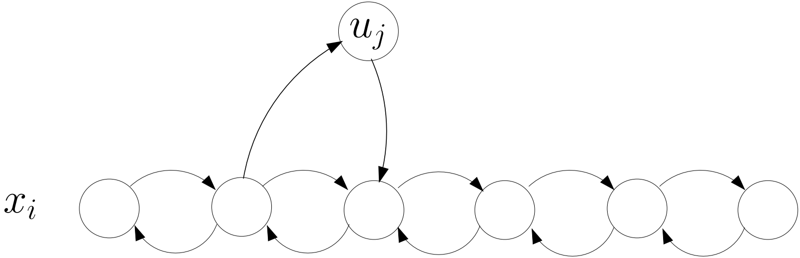
- Each clause $\mathcal{C}_j$ corresponds to a new vertex u_j.
- If $\mathcal{C}_j$ contains a non-negated literal x_i, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain

Clause Constraints

- Each clause $\mathcal{C}_j$ corresponds to a new vertex u_j.
- If $\mathcal{C}_j$ contains a non-negated literal x_i, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain
  - Add an outgoing edge to v_{k+1} in the x_i chain

Clause Constraints

- Each clause $\mathcal{C}_j$ corresponds to a new vertex u_j.
- If $\mathcal{C}_j$ contains a non-negated literal x_i, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain
  - Add an outgoing edge to v_{k+1} in the x_i chain



Clause Constraints

- If $\mathcal{C}_j$ contains a negated literal, reverse the edge directions

Clause Constraints

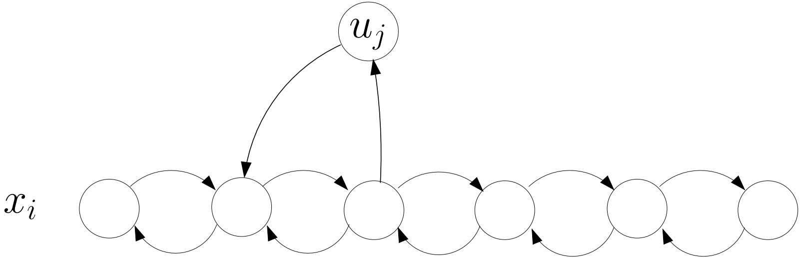
- If $\mathcal{C}_j$ contains a negated literal, reverse the edge directions
- If $\mathcal{C}_j$ contains a negated literal $\overline{x}_i$, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain

Clause Constraints

- If $\mathcal{C}_j$ contains a negated literal, reverse the edge directions
- If $\mathcal{C}_j$ contains a negated literal $\overline{x}_i$, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain
  - Add an outgoing edge to v_{k-1} in the x_i chain

Clause Constraints

- If $\mathcal{C}_j$ contains a negated literal, reverse the edge directions
- If $\mathcal{C}_j$ contains a negated literal $\overline{x}_i$, add edges to u_j:
  - Add an incoming edge from a vertex, say v_k, in the x_i chain
  - Add an outgoing edge to v_{k-1} in the x_i chain



An Example

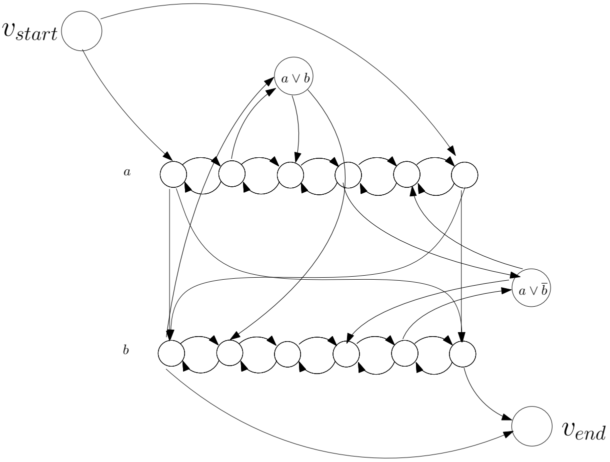
Construction of Hamiltonian path for

$$(a \vee b) \wedge (a \vee \overline{b})$$

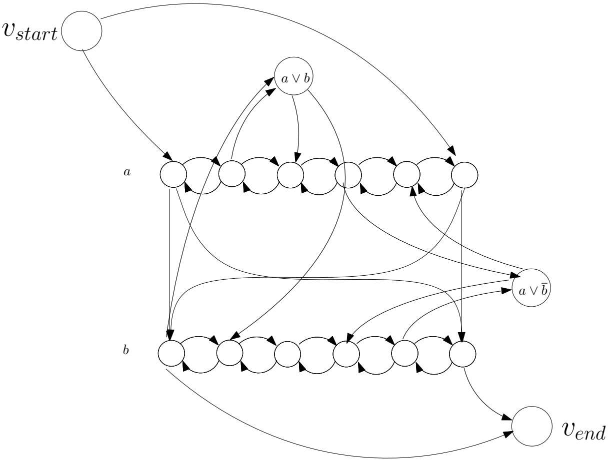
Here:

- $\mathcal{C}_1 = (a \vee b)$
- $\mathcal{C}_2 = (a \vee \overline{b})$

The Graph Construction



The Graph Construction

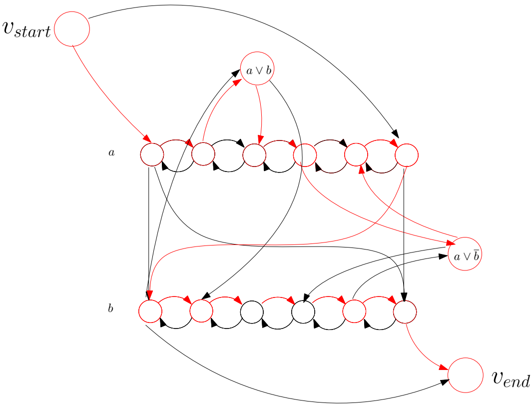


Claim

Hamiltonian path exists ONLY if you go from left to right in *a* and chose any one of the two directions for *b*.

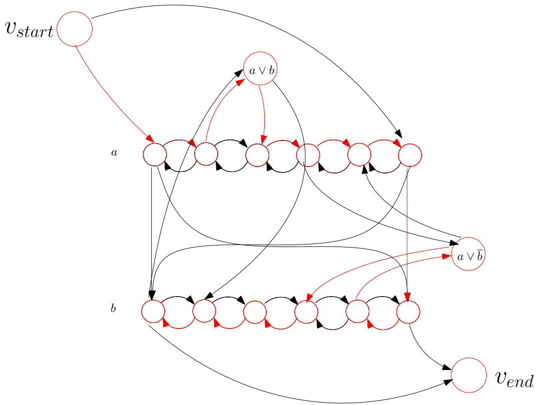
Path Corresponding to Assignment 1

$a = 1; b = 1$



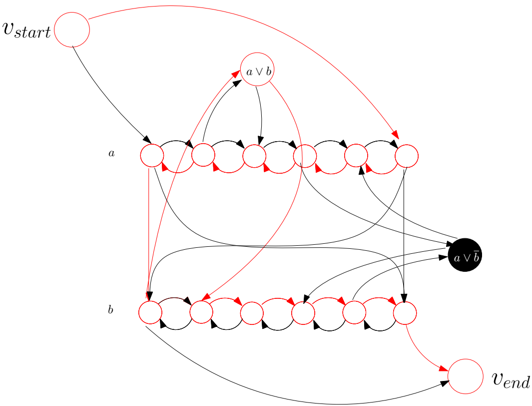
Path Corresponding to Assignment 2

$a = 1; b = 0$



Path Corresponding to Assignment 3

$a = 0; b = 1$

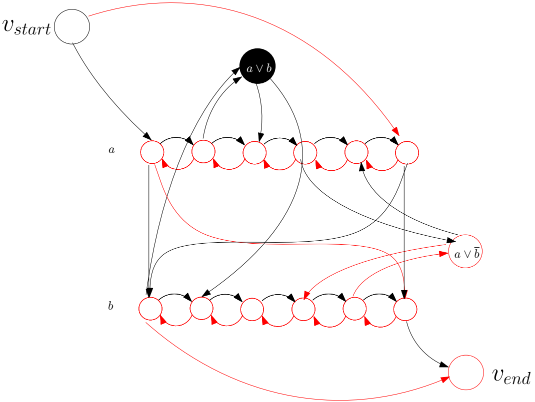


Error in finding HAMILTONIAN

No HAMILTONIAN PATH as $(a \vee \overline{b})$ is not accessible

Path Corresponding to Assignment 4

$a = 0; b = 0$



Error in finding HAMILTONIAN

No HAMILTONIAN PATH as $(a \vee b)$ is not accessible

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}

Construction Claim

Claim

*The constructed graph G has a Hamiltonian path **iff** ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}

Construction Claim

Claim

*The constructed graph G has a Hamiltonian path **iff** ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.
- Each assignment of variables corresponds to a path.

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.
- Each assignment of variables corresponds to a path.
- Each path corresponds to a unique variable assignment.

Construction Claim

Claim

The constructed graph G has a Hamiltonian path ***iff*** *ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.
- Each assignment of variables corresponds to a path.
- Each path corresponds to a unique variable assignment.
- If $x_i \in \mathcal{C}_j$ and $x_i = 1$, can visit u_j along the way.

Construction Claim

Claim

*The constructed graph G has a Hamiltonian path **iff** ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.
- Each assignment of variables corresponds to a path.
- Each path corresponds to a unique variable assignment.
- If $x_i \in \mathcal{C}_j$ and $x_i = 1$, can visit u_j along the way.
- If $\overline{x}_i \in C_j$ and $x_i = 0$, can visit u_j along the way.

Construction Claim

Claim

*The constructed graph G has a Hamiltonian path **iff** ϕ satisfiable.*

Things to note about the final construction:

- Any Hamiltonian path has to start at v_{start}
- Any Hamiltonian path has to end at v_{end}
- Any Hamiltonian path first traverses chain x_1, then x_2 etc.
- For each chain, only two ways of traversing it.
- Each assignment of variables corresponds to a path.
- Each path corresponds to a unique variable assignment.
- If $x_i \in C_j$ and $x_i = 1$, can visit u_j along the way.
- If $\overline{x}_i \in C_j$ and $x_i = 0$, can visit u_j along the way.
- The above two *only* ways to visit u_j without getting stuck.

SET-COVER

Claim

SET-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of a base set X, $|X| = n$, and parameter k, find a set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size k
*The set cover problem (SET-COVER) is **NP** complete.*

SET-COVER

Claim

SET-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of a base set X, $|X| = n$, and parameter k, find a set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size k
The set cover problem (SET-COVER) is ***NP*** *complete.*

- $\mathcal{C}' = \{S_{i_1}, \ldots, S_{i_k}\}$ is a set-cover iff $\bigcup_j S_{i_j} = X$.

SET-COVER

Claim

SET-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of a base set X, $|X| = n$, and parameter k, find a set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size k
The set cover problem (SET-COVER) is ***NP*** *complete.*

- $\mathcal{C}' = \{S_{i_1}, \ldots, S_{i_k}\}$ is a set-cover iff $\bigcup_j S_{i_j} = X$.
- Note that SET-COVER is in **NP**

SET-COVER

Claim

SET-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of a base set X, $|X| = n$, and parameter k, find a set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size k
*The set cover problem (SET-COVER) is **NP** complete.*

- $\mathcal{C}' = \{S_{i_1}, \ldots, S_{i_k}\}$ is a set-cover iff $\bigcup_j S_{i_j} = X$.
- Note that SET-COVER is in **NP**
- How to prove **NP** hardness?

SET-COVER

Claim

SET-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_m\}$ of subsets of a base set X, $|X| = n$, and parameter k, find a set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size k
*The set cover problem (SET-COVER) is **NP** complete.*

- $\mathcal{C}' = \{S_{i_1}, \ldots, S_{i_k}\}$ is a set-cover iff $\bigcup_j S_{i_j} = X$.
- Note that SET-COVER is in **NP**
- How to prove **NP** hardness?
- Reduce from VERTEX-COVER

Reduction from `VERTEX-COVER`

- Given a graph $G = (V, E)$, the idea is that:

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \dots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \dots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.
  - Each vertex $v_i \in V$ maps to set $C_i = \{a_j \ s.t. \ e_j$ incident to $v_i\}$

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.
  - Each vertex $v_i \in V$ maps to set $C_i = \{a_j \; s.t. \; e_j$ incident to $v_i\}$

Claim

$V' = \{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_i}, \ldots, C_{i_k}\}$ is a set cover.

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \dots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \dots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.
  - Each vertex $v_i \in V$ maps to set $C_i = \{a_j \ s.t. \ e_j \text{ incident to } v_i\}$

Claim

$V' = \{v_{i_1}, \dots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_i}, \dots, C_{i_k}\}$ is a set cover.

Proof.

- If e_j incident to v_i, then set C_i contains a_j

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.
  - Each vertex $v_i \in V$ maps to set $C_i = \{a_j \; s.t. \; e_j$ incident to $v_i\}$

Claim

$V' = \{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_i}, \ldots, C_{i_k}\}$ is a set cover.

Proof.

- If e_j incident to v_i, then set C_i contains a_j
- Picking v_i covers all edges incident to $v_i \iff$ picking C_i covers all corresponding elements a_j

Reduction from VERTEX-COVER

- Given a graph $G = (V, E)$, the idea is that:
  - Each vertex v_i maps to a set $\mathcal{C}_i$
  - $\{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_1}, \ldots, C_{i_k}\}$ is a set-cover.
- The exact construction is as follows:
  - Each edge $e_j \in E$ maps to an element $a_j \in X$.
  - Each vertex $v_i \in V$ maps to set $C_i = \{a_j \ s.t. \ e_j \text{ incident to } v_i\}$

Claim

$V' = \{v_{i_1}, \ldots, v_{i_k}\}$ is a vertex-cover iff $\{C_{i_i}, \ldots, C_{i_k}\}$ is a set cover.

Proof.

- If e_j incident to v_i, then set C_i contains a_j
- Picking v_i covers all edges incident to $v_i \iff$ picking C_i covers all corresponding elements a_j
- V' covers all edges $\iff \mathcal{C}'$ cover all elements

$\square$

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:

Another Hardness Proof for `INTEGER-PROGRAMMING`

Another reduction for `INTEGER-PROGRAMMING`

`SET-COVER` can be reduced to `INTEGER-PROGRAMMING`

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets:

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets: $$\sum_j x_j \leq k$$

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets: $$\sum_j x_j \leq k$$

All elements covered:

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets: $$\sum_j x_j \leq k$$

All elements covered: $$\sum_{j \mid a_i \in S_j} x_j \geq 1 \quad \forall i$$

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets: $$\sum_j x_j \leq k$$

All elements covered: $$\sum_{j \mid a_i \in S_j} x_j \geq 1 \quad \forall i$$

Variable 0 or 1:

Another Hardness Proof for INTEGER-PROGRAMMING

Another reduction for INTEGER-PROGRAMMING

SET-COVER can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the set-cover problem as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff S_j picked in set-cover

At most k sets: $$\sum_j x_j \leq k$$

All elements covered: $$\sum_{j | a_i \in S_j} x_j \geq 1 \quad \forall i$$

Variable 0 or 1: $$x_i \in \{0, 1\}$$

□

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

At most k elements:

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

At most k elements: $$\sum_j x_j \leq k$$

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \dots, S_m\}$ over $X = \{a_1, \dots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

At most k elements: $$\sum_j x_j \leq k$$

All sets covered:

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

At most k elements: $$\sum_j x_j \leq k$$

All sets covered: $$\sum_{j | a_j \in S_i} x_j \geq 1 \quad \forall i$$

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

At most k elements: $$\sum_j x_j \leq k$$

All sets covered: $$\sum_{j | a_j \in S_i} x_j \geq 1 \quad \forall i$$

Variable 0 or 1:

Yet Another Hardness Proof for INTEGER-PROGRAMMING

Yet another reduction for INTEGER-PROGRAMMING

HITTING-SET can be reduced to INTEGER-PROGRAMMING

Proof.

- Formulate the hitting-set as an integer program
- Given $\mathcal{C} = \{S_1, \ldots, S_m\}$ over $X = \{a_1, \ldots, a_n\}$ and k:
- Integer program variables: $x_j = 1$ iff x_j picked in hitting-set

$$\begin{aligned}
\text{At most k elements:} \quad & \sum_j x_j \leq k \\
\text{All sets covered:} \quad & \sum_{j | a_j \in S_i} x_j \geq 1 \quad \forall i \\
\text{Variable 0 or 1:} \quad & x_i \in \{0, 1\}
\end{aligned}$$

□

EXACT-COVER

Claim

EXACT-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set X, $|X| = 3m$, find a disjoint set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size m.
*The exact cover by 3-sets problem (EXACT-COVER) is **NP** complete.*

EXACT-COVER

Claim

EXACT-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set X, $|X| = 3m$, find a disjoint set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size m.
*The exact cover by 3-sets problem (EXACT-COVER) is **NP** complete.*

- Note that EXACT-COVER is in **NP**

EXACT-COVER

Claim

EXACT-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set X, $|X| = 3m$, find a disjoint set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size m.
*The exact cover by 3-sets problem (EXACT-COVER) is **NP** complete.*

- Note that EXACT-COVER is in **NP**
- How to prove **NP** hardness?

EXACT-COVER

Claim

EXACT-COVER : A collection $\mathcal{C} = \{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set X, $|X| = 3m$, find a disjoint set cover $\mathcal{C}' \subseteq \mathcal{C}$ of size m.
*The exact cover by 3-sets problem (EXACT-COVER) is **NP** complete.*

- Note that EXACT-COVER is in **NP**
- How to prove **NP** hardness?
- Reduce from SAT . Read from the Papadimitriou book.

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ *of* n *elements, where each element* a_i *has a weight* w_i *and a value* v_i*, both positive integers. Find a subset* $A' \subseteq A$ *such that*

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

KNAPSACK

Claim

KNAPSACK : *Given a set* $A = \{a_1, \ldots, a_n\}$ *of* n *elements, where each element* a_i *has a weight* w_i *and a value* v_i*, both positive integers. Find a subset* $A' \subseteq A$ *such that*

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

The Knapsack problem (KNAPSACK) is **NP** *complete.*

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

*The Knapsack problem (KNAPSACK) is **NP** complete.*

- Note that KNAPSACK is in **NP**

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

The Knapsack problem (KNAPSACK) is **NP** *complete.*

- Note that KNAPSACK is in **NP**
- How to prove **NP** hardness?

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

*The Knapsack problem (KNAPSACK) is **NP** complete.*

- Note that KNAPSACK is in **NP**
- How to prove **NP** hardness?
- Special case: $K = W$ and $w_i = v_i$ for all i.

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

*The Knapsack problem (KNAPSACK) is **NP** complete.*

- Note that KNAPSACK is in **NP**
- How to prove **NP** hardness?
- Special case: $K = W$ and $w_i = v_i$ for all i. Then problem?

KNAPSACK

Claim

KNAPSACK : Given a set $A = \{a_1, \ldots, a_n\}$ of n elements, where each element a_i has a weight w_i and a value v_i, both positive integers. Find a subset $A' \subseteq A$ such that

1. $\sum_{a_j \in A'} w_i \leq W$,
2. $\sum_{a_j \in A'} v_i \geq K$.

The Knapsack problem (KNAPSACK) is **NP** *complete.*

- Note that KNAPSACK is in **NP**
- How to prove **NP** hardness?
- Special case: $K = W$ and $w_i = v_i$ for all i. Then problem?
- Subset sum! Find a subset with value and weight equal to W.

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

SUBSET-SUM

Claim

The subset sum problem (SUBSET-SUM) is ***NP*** *complete.*

- Reduce EXACT-COVER to SUBSET-SUM

SUBSET-SUM

Claim

The subset sum problem (SUBSET-SUM) is ***NP*** *complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \dots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \dots, a_{3m}\}$
- Map each set S_j to an integer t_j

SUBSET-SUM

Claim

The subset sum problem (SUBSET-SUM) is ***NP*** *complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector
    - Represent set S as a vector $v(S)$ where $v_i(S) = 1$ iff $a_i \in S$

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector
    - Represent set S as a vector $v(S)$ where $v_i(S) = 1$ iff $a_i \in S$
    - Example: $S = \{a_1, a_4, a_5\}$ out of 6 elements

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector
  - Represent set S as a vector $v(S)$ where $v_i(S) = 1$ iff $a_i \in S$
  - Example: $S = \{a_1, a_4, a_5\}$ out of 6 elements
  - $v(S) =$

SUBSET-SUM

Claim

The subset sum problem (SUBSET-SUM) is ***NP*** *complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector
  - Represent set S as a vector $v(S)$ where $v_i(S) = 1$ iff $a_i \in S$
  - Example: $S = \{a_1, a_4, a_5\}$ out of 6 elements
  - $v(S) = 1\ 0\ 0\ 1\ 1\ 0$

SUBSET-SUM

Claim

*The subset sum problem (SUBSET-SUM) is **NP** complete.*

- Reduce EXACT-COVER to SUBSET-SUM
- $\{S_1, \ldots, S_n\}, |S_j| = 3$ over a base set $X = \{a_1, \ldots, a_{3m}\}$
- Map each set S_j to an integer t_j
- If S_j's are disjoint and cover X, then t_j's add up to some T
- For mapping sets into integers, use the characteristic vector
  - Represent set S as a vector $v(S)$ where $v_i(S) = 1$ iff $a_i \in S$
  - Example: $S = \{a_1, a_4, a_5\}$ out of 6 elements
  - $v(S) = 1\ 0\ 0\ 1\ 1\ 0$
  - Easy to see that its a one-to-one mapping

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

1 2 3 4 5 6 7 8 9

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	0	0	0

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	0	0	0
0	0	0	0	0	1	0	1	1

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	0	0	0
0	0	0	0	0	1	0	1	1
0	1	0	1	0	0	1	0	0

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

1	2	3	4	5	6	7	8	9
1	0	0	1	0	1	0	0	0
0	0	0	0	0	1	0	1	1
0	1	0	1	0	0	1	0	0
1	1	0	0	0	0	0	1	0
0	1	0	1	0	1	0	0	0
1	0	1	0	1	0	0	0	0

An Example

- Lets take the two sets $\{6, 8, 9\}, \{2, 4, 7\}$.
- The corresponding vectors are:

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1$$

An Example

- Lets take the two sets $\{6, 8, 9\}, \{2, 4, 7\}$.
- The corresponding vectors are:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

An Example

- Lets take the two sets $\{6,8,9\},\{2,4,7\}$.
- The corresponding vectors are:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

- The union of the two sets is: $\{2,4,6,7,8,9\}$

An Example

- Lets take the two sets $\{6, 8, 9\}, \{2, 4, 7\}$.
- The corresponding vectors are:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

- The union of the two sets is: $\{2, 4, 6, 7, 8, 9\}$
- The characteristic vector of the union is:

$$\begin{array}{ccccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

An Example

- Lets take the two sets $\{6,8,9\},\{2,4,7\}$.
- The corresponding vectors are:

$$\begin{array}{ccccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

- The union of the two sets is: $\{2,4,6,7,8,9\}$
- The characteristic vector of the union is:

$$\begin{array}{ccccccccc} 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{array}$$

- Any observations?

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccc l} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336 \end{array}$$

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

$$\begin{array}{ccccccccc l}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & : 2^9 - 1 = 511
\end{array}$$

An Example

$$\{1,4,6\},\{6,8,9\},\{2,4,7\},\{1,2,8\},\{2,4,6\},\{1,3,5\}$$

$$\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3+2^1+2^0=011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7+2^5+2^2=164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8+2^6+2^4=336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

$$\begin{array}{ccccccccc l}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & : 2^9-1=511
\end{array}$$

- $11+164+336=511$.

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

$$\begin{array}{ccccccccc l}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & : 2^9 - 1 = 511
\end{array}$$

- $11 + 164 + 336 = 511$. So set $W = 2^n - 1$, and check.

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$
\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}
$$

- The union covers X and the sets are pair-wise disjoint:

$$
\begin{array}{ccccccccc l}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & : 2^9 - 1 = 511
\end{array}
$$

- $11 + 164 + 336 = 511$. So set $W = 2^n - 1$, and check.
- There exists subset equal to $W \implies$ exact-cover possible

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccccl}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

$$\begin{array}{ccccccccccl}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & : 2^9 - 1 = 511
\end{array}$$

- $11 + 164 + 336 = 511$. So set $W = 2^n - 1$, and check.
- There exists subset equal to $W \implies$ exact-cover possible
- Problem:

An Example

$$\{1,4,6\}, \{6,8,9\}, \{2,4,7\}, \{1,2,8\}, \{2,4,6\}, \{1,3,5\}$$

$$\begin{array}{ccccccccc l}
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & : 2^3 + 2^1 + 2^0 = 011 \\
0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & : 2^7 + 2^5 + 2^2 = 164 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & : 2^8 + 2^6 + 2^4 = 336
\end{array}$$

- The union covers X and the sets are pair-wise disjoint:

$$1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1 \quad : 2^9 - 1 = 511$$

- $11 + 164 + 336 = 511$. So set $W = 2^n - 1$, and check.
- There exists subset equal to $W \implies$ exact-cover possible
- Problem: False positives.

A Problem

$$\{3,4\},\{2,4\},\{2,3,4\}$$

A Problem

$$\{3, 4\}, \{2, 4\}, \{2, 3, 4\}$$

$0 \quad 0 \quad 1 \quad 1 \quad : 3$

A Problem

$$\{3,4\}, \{2,4\}, \{2,3,4\}$$

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & : 3 \\ 0 & 1 & 0 & 1 & : 5 \end{array}$$

A Problem

$$\{3,4\},\{2,4\},\{2,3,4\}$$

$$\begin{array}{ccccl} 0 & 0 & 1 & 1 & :3 \\ 0 & 1 & 0 & 1 & :5 \\ 0 & 1 & 1 & 1 & :7 \end{array}$$

A Problem

$$\{3, 4\}, \{2, 4\}, \{2, 3, 4\}$$

$$\begin{array}{cccccl} 0 & 0 & 1 & 1 & & : 3 \\ 0 & 1 & 0 & 1 & & : 5 \\ 0 & 1 & 1 & 1 & & : 7 \end{array}$$

- The union does not cover X, and not disjoint.

A Problem

$$\{3, 4\}, \{2, 4\}, \{2, 3, 4\}$$

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & : 3 \\ 0 & 1 & 0 & 1 & : 5 \\ 0 & 1 & 1 & 1 & : 7 \end{array}$$

- The union does not cover X, and not disjoint.

$$0\,0\,1\,1 \;+\; 0\,1\,0\,1 \;+\; 0\,1\,1\,1$$

A Problem

$$\{3,4\},\{2,4\},\{2,3,4\}$$

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & & :3 \\ 0 & 1 & 0 & 1 & & :5 \\ 0 & 1 & 1 & 1 & & :7 \end{array}$$

- The union does not cover X, and not disjoint.

$$0\,0\,1\,1 + 0\,1\,0\,1 + 0\,1\,1\,1 = 1\,1\,1\,1$$

A Problem

$$\{3,4\}, \{2,4\}, \{2,3,4\}$$

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & :3 \\ 0 & 1 & 0 & 1 & :5 \\ 0 & 1 & 1 & 1 & :7 \end{array}$$

- The union does not cover X, and not disjoint.

$$0\,0\,1\,1 \ + \ 0\,1\,0\,1 \ + \ 0\,1\,1\,1 \ = \ 1\,1\,1\,1 = 15$$

A Problem

$$\{3,4\}, \{2,4\}, \{2,3,4\}$$

$$\begin{array}{cccc} 0 & 0 & 1 & 1 & :3 \\ 0 & 1 & 0 & 1 & :5 \\ 0 & 1 & 1 & 1 & :7 \end{array}$$

- The union does not cover X, and not disjoint.

$$0\,0\,1\,1 \;+\; 0\,1\,0\,1 \;+\; 0\,1\,1\,1 \;=\; 1\,1\,1\,1 = 15$$

- So, where is the problem coming from?

A Problem

$$\{3,4\},\{2,4\},\{2,3,4\}$$

$$\begin{array}{cccccc} 0 & 0 & 1 & 1 & : 3 \\ 0 & 1 & 0 & 1 & : 5 \\ 0 & 1 & 1 & 1 & : 7 \end{array}$$

- The union does not cover X, and not disjoint.

$$0\,0\,1\,1 \;+\; 0\,1\,0\,1 \;+\; 0\,1\,1\,1 \;=\; 1\,1\,1\,1 = 15$$

- So, where is the problem coming from?
- Carrying! It messes up the addition.

A Solution

- There are actually two different problems due to carrying:

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution:

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2, 3, 4\} \ = \ 0\ 1\ 1\ 1 \ =$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2, 3, 4\} \ = \ 0\ 1\ 1\ 1 \ = 2^2 + 2^1 + 2^0,$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2, 3, 4\} \ = \ 0\ 1\ 1\ 1 \ = 2^2 + 2^1 + 2^0, \ \ 5^2 + 5^1 + 5^0 = 31$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\}$ = 0 1 1 1 $= 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\}$ = 0 0 1 1 =

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2, 3, 4\} \;=\; 0\ 1\ 1\ 1 \;= 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3, 4\} \;=\; 0\ 0\ 1\ 1 \;= 2^1 + 2^0,$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} \;=\; 0\;1\;1\;1 \;= 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} \;=\; 0\;0\;1\;1 \;= 2^1 + 2^0, \quad 5^1 + 5^0 = 6$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 =$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} \ = \ 0\ 1\ 1\ 1 \ = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} \ = \ 0\ 0\ 1\ 1 \ = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} \ = \ 0\ 1\ 0\ 1 \ = 2^2 + 2^0,$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1 \ +\ 0\ 1\ 0\ 1 \ +\ 0\ 1\ 1\ 1$$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1\ +\ 0\ 1\ 0\ 1\ +\ 0\ 1\ 1\ 1\ =\ 0\ 2\ 2\ 3$$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1\ +\ 0\ 1\ 0\ 1\ +\ 0\ 1\ 1\ 1\ =\ 0\ 2\ 2\ 3 = 63 \neq\ 1\ 1\ 1\ 1$$

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} \ = \ 0\ 1\ 1\ 1 \ = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} \ = \ 0\ 0\ 1\ 1 \ = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} \ = \ 0\ 1\ 0\ 1 \ = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1 \ + \ 0\ 1\ 0\ 1 \ + \ 0\ 1\ 1\ 1 \ = \ 0\ 2\ 2\ 3 = 63 \neq \ 1\ 1\ 1\ 1$$

- Adding at most n 1's $\implies$ no carry-over at any digit

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1 \ + \ 0\ 1\ 0\ 1 \ + \ 0\ 1\ 1\ 1 \ = \ 0\ 2\ 2\ 3 = 63 \neq \ 1\ 1\ 1\ 1$$

- Adding at most n 1's $\implies$ no carry-over at any digit
- This solves both the problems since:

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2, 3, 4\} = 0\ 1\ 1\ 1 = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3, 4\} = 0\ 0\ 1\ 1 = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2, 4\} = 0\ 1\ 0\ 1 = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1\ +\ 0\ 1\ 0\ 1\ +\ 0\ 1\ 1\ 1\ =\ 0\ 2\ 2\ 3 = 63 \neq\ 1\ 1\ 1\ 1$$

- Adding at most n 1's $\implies$ no carry-over at any digit
- This solves both the problems since:
  - If a column has all 0's, no carry-over ensures it gets a 0.

A Solution

- There are actually two different problems due to carrying:
  - If a column has all 0's, could get 1 from carry-over
  - If a column has more than one 1, could still get one 1
- Solution: Interpret vector over base $3m$, not base 2!
  - $\{2,3,4\} \; = \; 0\ 1\ 1\ 1 \; = 2^2 + 2^1 + 2^0, \quad 5^2 + 5^1 + 5^0 = 31$
  - $\{3,4\} \; = \; 0\ 0\ 1\ 1 \; = 2^1 + 2^0, \quad 5^1 + 5^0 = 6$
  - $\{2,4\} \; = \; 0\ 1\ 0\ 1 \; = 2^2 + 2^0, \quad 5^2 + 5^0 = 26$

$$0\ 0\ 1\ 1 \; + \; 0\ 1\ 0\ 1 \; + \; 0\ 1\ 1\ 1 \; = \; 0\ 2\ 2\ 3 = 63 \neq \; 1\ 1\ 1\ 1$$

- Adding at most n 1's $\implies$ no carry-over at any digit
- This solves both the problems since:
  - If a column has all 0's, no carry-over ensures it gets a 0.
  - If a column has more than one 1, it will get the sum, not 1.

Reductions So Far