NEW CS 473: Theory II, Fall 2015

Some Hairy NP-Complete Problems

Lecture 5

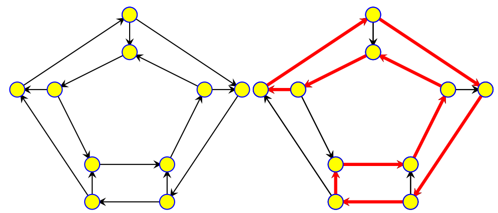
September 8, 2015

Directed Hamiltonian Cycle

Input Given a directed graph $G = (V, E)$ with n vertices

Goal Does G have a Hamiltonian cycle?

- A Hamiltonian cycle is a cycle in the graph that visits every vertex in G exactly once



Reduction construction

From 3SAT to Hamiltonian cycle in directed graph

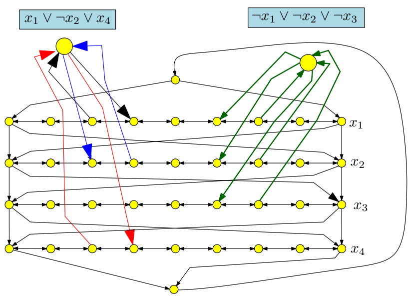
1. Given **3SAT** formula φ create a graph $\mathbf{G}_\varphi$ such that
   - G_φ has a Hamiltonian cycle if and only if φ is satisfiable
   - G_φ should be constructible from φ by a polynomial time algorithm $\mathcal{A}$
2. Notation: φ has n variables $x_1, x_2, \ldots, x_n$ and m clauses $C_1, C_2, \ldots, C_m$.

The Reduction: By figure

More details were given in the previous lecture

3SAT formula φ:

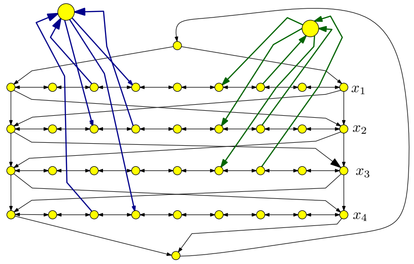
$$\varphi = \left(x_1 \lor \neg x_2 \lor x_4\right) \land \left(\neg x_1 \lor \neg x_2 \lor \neg x_3\right)$$



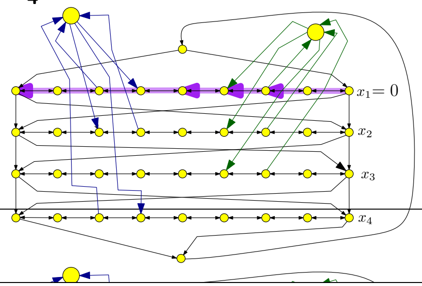
The Reduction: Assignment $\Rightarrow$ Hamiltonian cycle

3SAT formula φ:

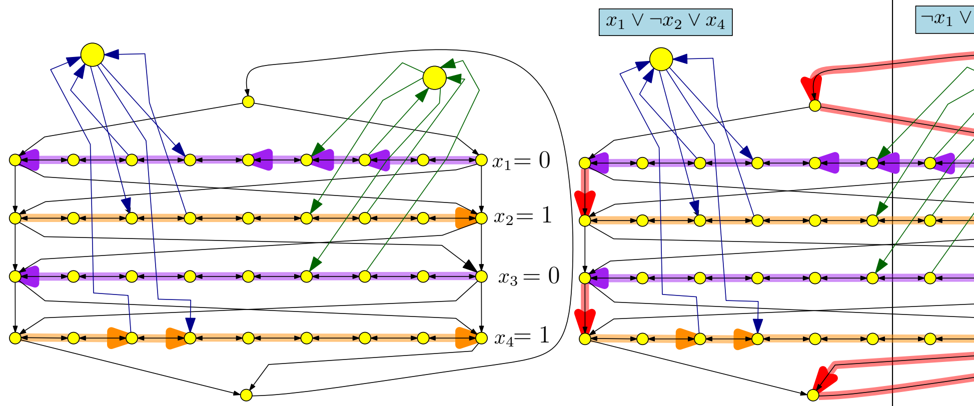
$$\varphi = \left(x_1 \vee \neg x_2 \vee x_4\right) \wedge \left(\neg x_1 \vee \neg x_2 \vee \neg x_3\right)$$



A satisfying assignment:
$x_1 = \mathbf{0}$, $x_2 = \mathbf{1}$, $x_3 = \mathbf{0}$, $x_4 = \mathbf{1}$



Reduction: Assignment $\Rightarrow$ Hamiltonian cycle

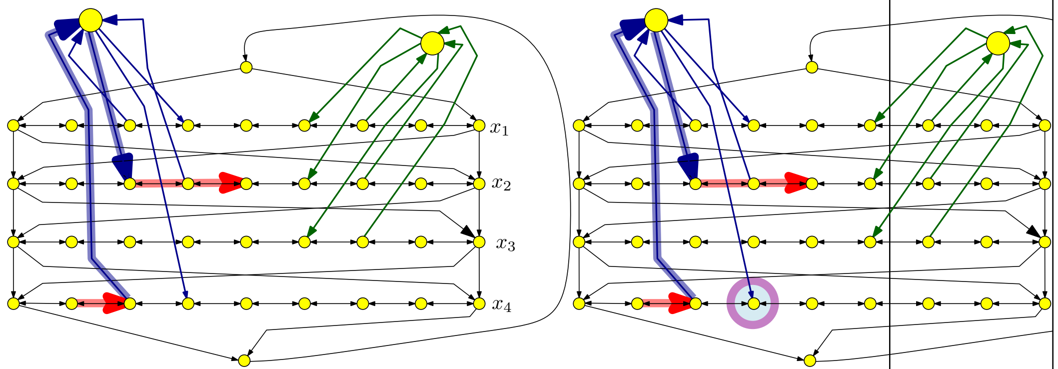


Satisfying assignment: $x_1 = \mathbf{0}$, $x_2 = \mathbf{1}$, $x_3 = \mathbf{0}$, $x_4 = \mathbf{1}$

Conclude: If φ has a satisfying assignment then there is an Hamiltonian cycle in $\mathbf{G}_\varphi$.

Reduction: Hamiltonian cycle $\Rightarrow$ Assignment

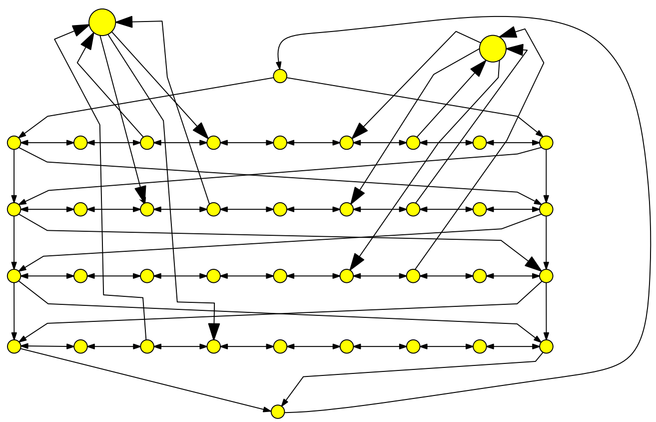
No shenanigan: Hamiltonian cycle can not leave a row in the middle



Conclude: Hamiltonian cycle must go through each row completely from left to right, or right to left. As such, can be interpreted as a valid assignment.

Reduction: Hamiltonian cycle $\Rightarrow$ Assignment

Drawing example



Correctness Proof

Proposition

φ has a satisfying assignment iff G_φ has a Hamiltonian cycle.

Proof.

$\Rightarrow$ Let a be the satisfying assignment for φ. Define Hamiltonian cycle as follows

- If $a(x_i) = 1$ then traverse path i from left to right
- If $a(x_i) = 0$ then traverse path i from right to left
- For each clause, path of at least one variable is in the "right" direction to splice in the node corresponding to clause □

Hamiltonian Cycle $\Rightarrow$ Satisfying assignment

Suppose Π is a Hamiltonian cycle in G_φ

- If Π enters c_j (vertex for clause C_j) from vertex $3j$ on path i then it must leave the clause vertex on edge to $3j+1$ on the *same path* i
  - If not, then only unvisited neighbor of $3j+1$ on path i is $3j+2$
  - Thus, we don't have two unvisited neighbors (one to enter from, and the other to leave) to have a Hamiltonian Cycle
- Similarly, if Π enters c_j from vertex $3j+1$ on path i then it must leave the clause vertex c_j on edge to $3j$ on path i

Hamiltonian Cycle $\Longrightarrow$ Satisfying assignment (contd)

- Thus, vertices visited immediately before and after C_i are connected by an edge
- We can remove c_j from cycle, and get Hamiltonian cycle in $G - c_j$
- Consider Hamiltonian cycle in $G - \{c_1, \dots c_m\}$; it traverses each path in only one direction, which determines the truth assignment

Hamiltonian Cycle

Problem

Input *Given undirected graph $G = (V, E)$*

Goal *Does G have a Hamiltonian cycle? That is, is there a cycle that visits every vertex exactly one (except start and end vertex)?*

NP-Completeness

Theorem
Hamiltonian cycle *problem for undirected graphs is* **NP-Complete**.

Proof.

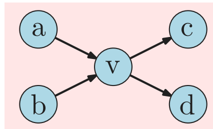
- The problem is in **NP**; proof left as exercise.
- Hardness proved by reducing Directed Hamiltonian Cycle to this problem □

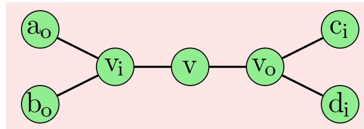
Reduction Sketch

Goal: Given directed graph G, need to construct undirected graph G' such that G has Hamiltonian Path iff G' has Hamiltonian path

Reduction

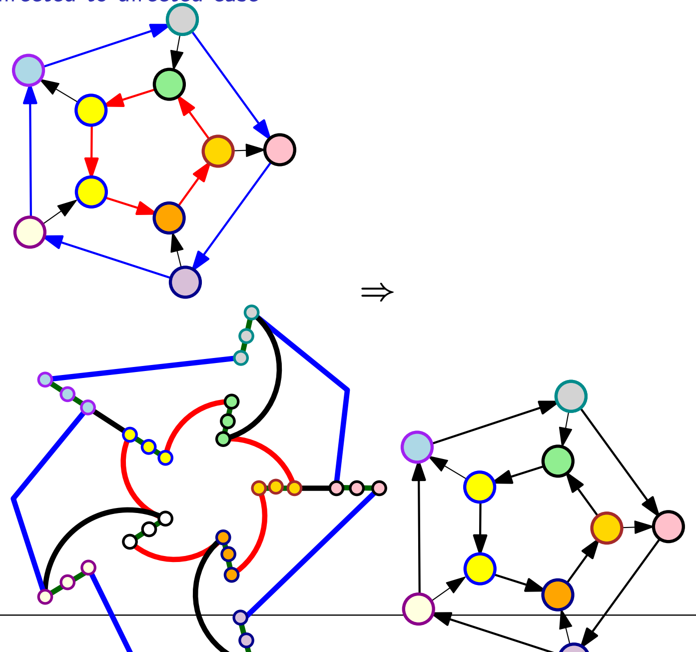
- Replace each vertex v by 3 vertices: v_{in}, v, and v_{out}
- A directed edge (a, b) is replaced by edge (a_{out}, b_{in})





Hamiltonian cycle reduction

Undirected to directed case



Reduction: Wrapup

- The reduction is polynomial time (exercise)
- The reduction is correct (exercise)

Graph Coloring

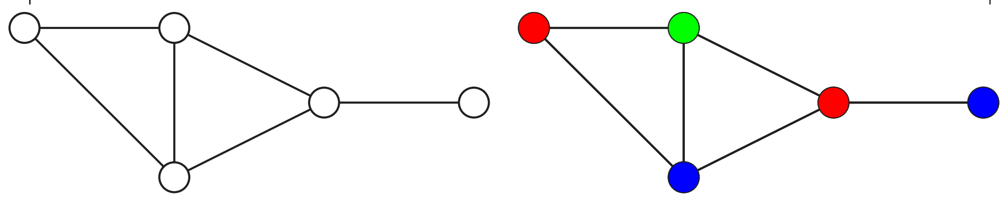
Graph Coloring

Instance: $G = (V, E)$: Undirected graph, integer k.
Question: Can the vertices of the graph be colored using k colors so that vertices connected by an edge do not get the same color?

Graph 3-Coloring

3 Coloring

Instance: $G = (V, E)$: Undirected graph.
Question: Can the vertices of the graph be colored using 3 colors so that vertices connected by an edge do not get the same color?



Graph Coloring

1. Observation: If G is colored with k colors then each color class (nodes of same color) form an independent set in G.
2. G can be partitioned into k independent sets iff G is k-colorable.
3. Graph 2-Coloring can be decided in polynomial time.
4. G is 2-colorable iff G is bipartite!
5. There is a linear time algorithm to check if G is bipartite using **BFS** (we saw this earlier).

Graph Coloring and Register Allocation

Register Allocation

Assign variables to (at most) k registers such that variables needed at the same time are not assigned to the same register

Interference Graph

Vertices are variables, and there is an edge between two vertices, if the two variables are "live" at the same time.

Observations

- [Chaitin] Register allocation problem is equivalent to coloring the interference graph with k colors
- Moreover, **3-COLOR** $\leq_P$ **k-Register Allocation**, for any $k \geq 3$

Class Room Scheduling

1. Given n classes and their meeting times, are k rooms sufficient?
2. Reduce to Graph k-Coloring problem
3. Create graph G
   - a node v_i for each class i
   - an edge between v_i and v_j if classes i and j *conflict*
4. Exercise: G is k-colorable iff k rooms are sufficient.

Frequency Assignments in Cellular Networks

1. Cellular telephone systems that use Frequency Division Multiple Access (FDMA) (example: GSM in Europe and Asia and AT&T in USA)
   - Breakup a frequency range $[a, b]$ into disjoint *bands* of frequencies $[a_0, b_0], [a_1, b_1], \ldots, [a_k, b_k]$
   - Each cell phone tower (simplifying) gets one band
   - Constraint: nearby towers cannot be assigned same band, otherwise signals will interference
2. Problem: given k bands and some region with n towers, is there a way to assign the bands to avoid interference?
3. Can reduce to k-coloring by creating intereference/conflict graph on towers.

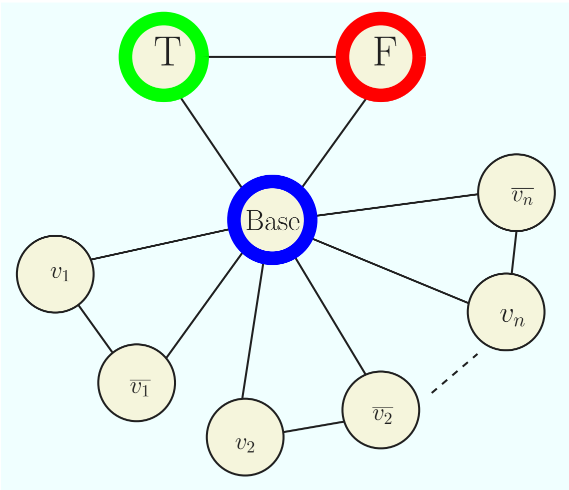
3-Coloring is NP-Complete

- **3-Coloring** is in **NP**.
  - Certificate: for each node a color from $\{1, 2, 3\}$.
  - Certifier: Check if for each edge (u, v), the color of u is different from that of v.
- Hardness: We will show **3-SAT** $\leq_P$ **3-Coloring**.

Reduction Idea

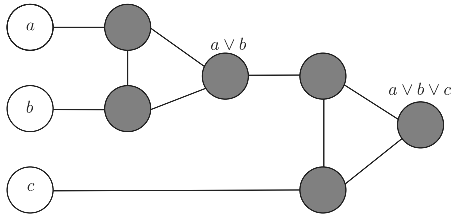
1. φ: Given **3SAT** formula (i.e., **3**CNF formula).
2. φ: variables $x_1, \ldots, x_n$ and clauses $C_1, \ldots, C_m$.
3. Create graph G_φ s.t. G_φ 3-colorable $\iff$ φ satisfiable.
   - encode assignment $x_1, \ldots, x_n$ in colors assigned nodes of G_φ.
   - create triangle with node True, False, Base
   - for each variable x_i two nodes v_i and $\bar{v}_i$ connected in a triangle with common Base
   - If graph is 3-colored, either v_i or $\bar{v}_i$ gets the same color as True. Interpret this as a truth assignment to v_i
   - Need to add constraints to ensure clauses are satisfied (next phase)

Figure



Clause Satisfiability Gadget

1. For each clause $C_j = (a \vee b \vee c)$, create a small gadget graph
   - gadget graph connects to nodes corresponding to a, b, c
   - needs to implement OR
2. OR-gadget-graph:



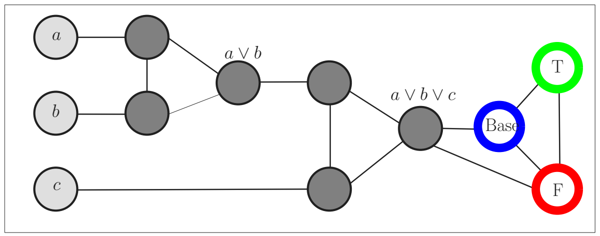
OR-Gadget Graph

Property: if a, b, c are colored False in a 3-coloring then output node of OR-gadget has to be colored False.

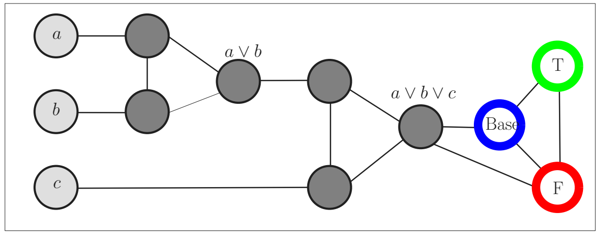
Property: if one of a, b, c is colored True then OR-gadget can be 3-colored such that output node of OR-gadget is colored True.

Reduction

- create triangle with nodes True, False, Base
- for each variable x_i two nodes v_i and $\bar{v}_i$ connected in a triangle with common Base
- for each clause $C_j = (a \vee b \vee c)$, add OR-gadget graph with input nodes a, b, c and connect output node of gadget to both False and Base



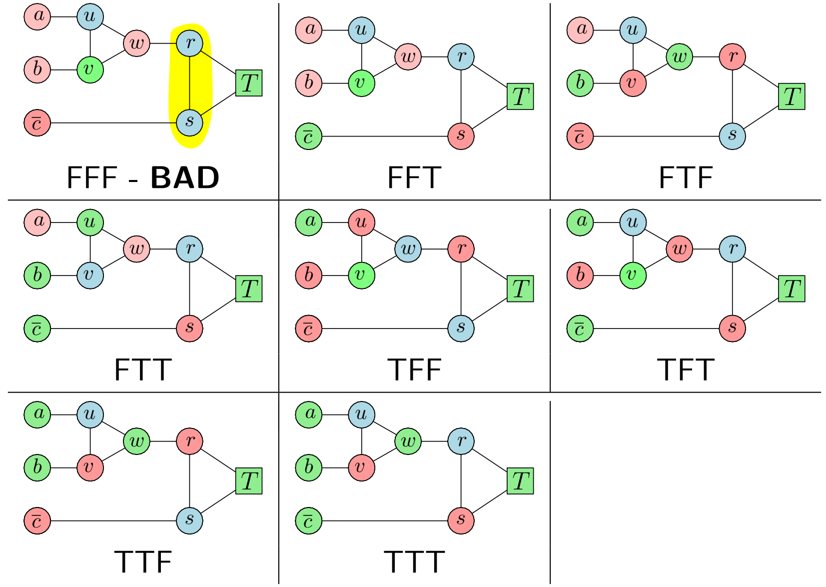
Reduction



Claim

*No legal **3**-coloring of above graph (with coloring of nodes T, F, B fixed) in which a, b, c are colored False. If any of a, b, c are colored True then there is a legal **3**-coloring of above graph.*

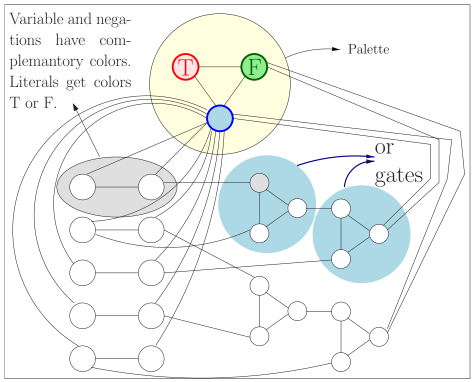
3 coloring of the clause gadget



Reduction Outline

Example

$\varphi = (u \vee \neg v \vee w) \wedge (v \vee x \vee \neg y)$



Correctness of Reduction

φ is satisfiable implies G_φ is 3-colorable

- if x_i is assigned True, color v_i True and $\bar{v}_i$ False
- for each clause $C_j = (a \vee b \vee c)$ at least one of a, b, c is colored True. OR-gadget for C_j can be 3-colored such that output is True.

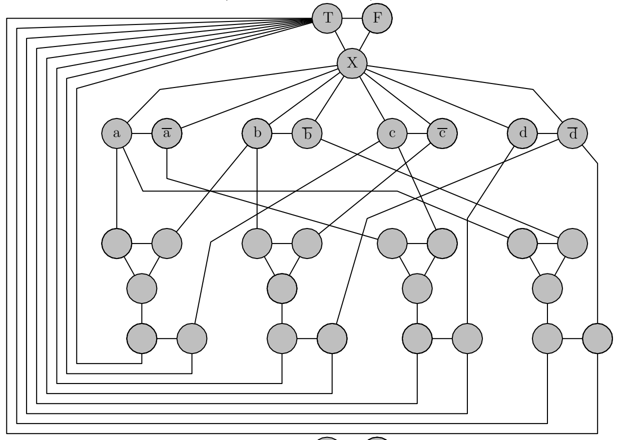
G_φ is 3-colorable implies φ is satisfiable

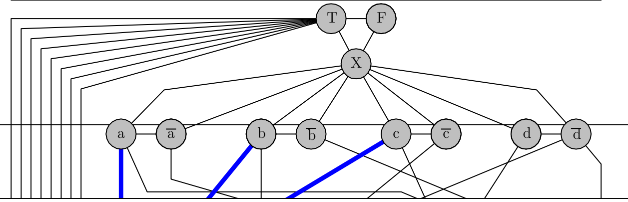
- if v_i is colored True then set x_i to be True, this is a legal truth assignment
- consider any clause $C_j = (a \vee b \vee c)$. it cannot be that all a, b, c are False. If so, output of OR-gadget for C_j has to be colored False but output is connected to Base and False!

Graph generated in reduction...

... from 3SAT to 3COLOR

$$(a \vee b \vee c) \wedge \left(b \vee \overline{c} \vee \overline{d}\right) \wedge (\overline{a} \vee c \vee d) \wedge \left(a \vee \overline{b} \vee \overline{d}\right)$$





Subset Sum

Subset Sum

Instance: S - set of positive integers, t: - an integer number (Target)
Question: Is there a subset $X \subseteq S$ such that $\sum_{x \in X} x = t$?

Claim

Subset Sum *is* **NP-Complete**.

Vec Subset Sum

We will prove following problem is **NP-Complete**...

Vec Subset Sum

Instance: S - set of n vectors of dimension k, each vector has non-negative numbers for its coordinates, and a target vector $\overrightarrow{t}$.
Question: Is there a subset $X \subseteq S$ such that $\sum_{\overrightarrow{x} \in X} \overrightarrow{x} = \overrightarrow{t}$?

Reduction from **3SAT**.

Vec Subset Sum

Handling a single clause

Think about vectors as being lines in a table.

First gadget

Selecting between two lines.

Target	??	??	01	???
a_1	??	??	01	??
a_2	??	??	01	??

Two rows for every variable x: selecting either $x = 0$ or $x = 1$.

Handling a clause...

We will have a column for every clause...

numbers	...	$C \equiv a \vee b \vee \overline{c}$	...
a	...	01	...
$\overline{a}$	...	00	...
b	...	01	...
$\overline{b}$	...	00	...
c	...	00	...
$\overline{c}$	...	01	...
C fix-up 1	000	07	000
C fix-up 2	000	08	000
C fix-up 3	000	09	000
TARGET		10	

3SAT to Vec Subset Sum

numbers	$a \vee \overline{a}$	$b \vee \overline{b}$	$c \vee \overline{c}$	$d \vee \overline{d}$	$D \equiv \overline{b} \vee c \vee \overline{d}$	$C \equiv a \vee b \vee \overline{c}$
a	1	0	0	0	00	01
$\overline{a}$	1	0	0	0	00	00
b	0	1	0	0	00	01
$\overline{b}$	0	1	0	0	01	00
c	0	0	1	0	01	00
$\overline{c}$	0	0	1	0	00	01
d	0	0	0	1	00	00
$\overline{d}$	0	0	0	1	01	01
C fix-up 1	0	0	0	0	00	07
C fix-up 2	0	0	0	0	00	08
C fix-up 3	0	0	0	0	00	09
D fix-up 1	0	0	0	0	07	00
D fix-up 2	0	0	0	0	08	00
D fix-up 3	0	0	0	0	09	00
TARGET	1	1	1	1	10	10

Vec Subset Sum to Subset Sum

numbers
010000000001
010000000000
000100000001
000100000100
000001000100
000001000001
000000010000
000000010101
000000000007
000000000008
000000000009
000000000700
000000000800
000000000900
010101011010

Other NP-Complete Problems

- 3-Dimensional Matching
- Subset Sum

Read book.

Need to Know NP-Complete Problems

- **3SAT**.
- **Circuit-SAT**.
- **Independent Set**.
- **Vertex Cover**.
- **Clique**.
- **Set Cover** / **Hitting Set**.
- **Hamiltonian Cycle** (in Directed/Undirected Graphs).
- **3Coloring**.
- **3-D Matching**.
- **Subset Sum** / **Partition**.

Subset Sum and Knapsack

1. Subset Sum Problem: Given n integers $a_1, a_2, \ldots, a_n$ and a target B, is there a subset of S of $\{a_1, \ldots, a_n\}$ such that the numbers in S add up *precisely* to B?
2. Subset Sum is **NP-Complete**— see book.
3. Knapsack: Given n items with item i having size s_i and profit p_i, a knapsack of capacity B, and a target profit P, is there a subset S of items that can be packed in the knapsack and the profit of S is at least P?
4. Show Knapsack problem is **NP-Complete** via reduction from Subset Sum (exercise).

Subset Sum and Knapsack

1. Subset Sum can be solved in $O(nB)$ time using dynamic programming (exercise).
2. Implies that problem is hard only when numbers $a_1, a_2, \ldots, a_n$ are exponentially large compared to n. That is, each a_i requires polynomial in n bits.
3. *Number problems* of the above type are said to be **weakly NPComplete**.