# More NP-Complete and NP-hard Problems

Traveling Salesperson Path

Subset Sum

Partition

# NP-completeness Proofs

1. The first part of an NP-completeness proof is showing the problem is in **NP**.
2. The second part is giving a reduction **from** a known NP-complete problem.

- Sometimes, we can only show a problem ***NP-hard*** = "if the problem is in **P**, then **P** = **NP**," but the problem may not be in **NP**.

# Optimization Problems

- NP-complete problems are always **yes/no** questions.
- In practice, we tend to want to solve ***optimization problems***, where our task is to minimize (or maximize) a function, f(x), of the input, x.
- Optimization problems, strictly speaking, can't be NP-complete (only NP-hard).

# Turning an Optimization Problem into a Decision Problem

- **Optimization Problem**: Given an input, x, find the smallest (or, largest) optimization value, f(x), for x.
- **Corresponding Decision Problem**: Given an input, x, and integer k, is there an optimization value, f(x), for x, that is at most (or, at least) k?

# Optimization Problems – (2)

- Optimization problems are never, strictly speaking, in **NP**.
  - They are not yes/no.
- But there is always a simple polynomial-time reduction from the yes/no version to the optimization version. (How?)

# Example: TSP

- **Traveling Salesperson Problem**: Given an undirected complete graph, G, with integer weights on its edges, find the smallest-weight path from s to t in G that visits each other vertex in G.
- **Decision version**: Given G and an integer, K, is there a path from s to t of total weight at most K that visits each each vertex in G?

# TSP is in NP

- Guess a path, P, from s to t.
- Check whether it visits each vertex in G.
- Sum up the weights of the edges in P and accept if the total weight is at most K.

# Roadmap to show TSP is NP-hard

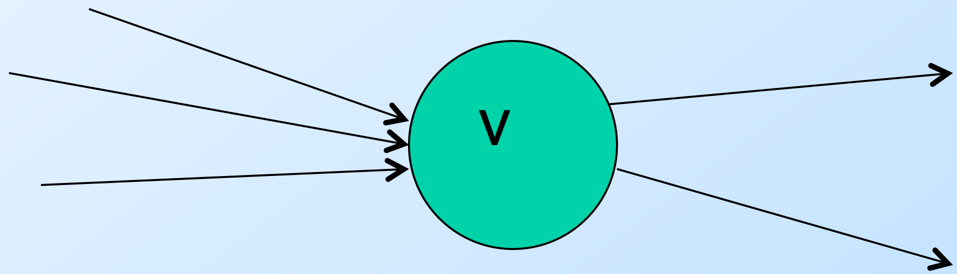
1. Provide a polytime reduction from Directed Hamiltonian Path (which we already know is NP-complete) to Undirected Hamiltonian Path
2. Provide a polytime reduction from Undirected Hamiltonian Path to TSP

# From Directed Hamiltonian Path

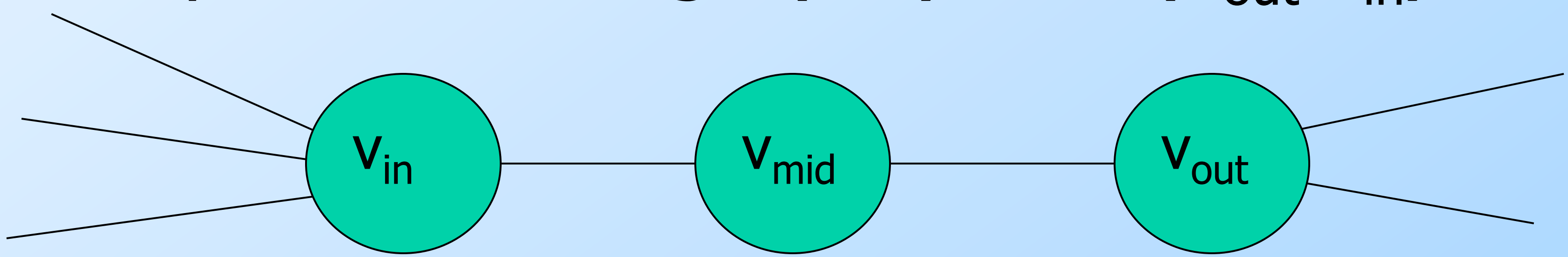
- DHP: Given a directed graph, G, and nodes s and t, is there a path from s to t in G that visits each other node exactly once?
- UHP: same question, but G is undirected.

# DHP to UHP

- Replace each vertex, v, in the original graph, with three vertices, $v_{in}$, $v_{mid}$, $v_{out}$.




- Replace each edge (u,v) with $(u_{out},v_{in})$



# UHP to TSP

- Given an undirected graph, G, and nodes s and t.
- Create an undirected complete graph, H:
  - If edge (u,v) is in G, then give (u,v) weight 1 in H.
  - If edge (u,v) is not in G, then give (u,v) weight 2 in H.
- Set K = n-1, where n is the number of nodes. H has a TSP of weight K iff G has an undirected Hamiltonian Path.

# A Number Problem: The Subset Sum Problem

- We shall prove NP-complete a problem just involving integers:
  - Given a set S of integers and a budget K, is there a subset of S whose sum is exactly K?
- E.g., S = {5, 8, 9, 13, 17}, K = 27.
  - In this instance the answer is "Yes":
  - S′ = {5, 9, 13}

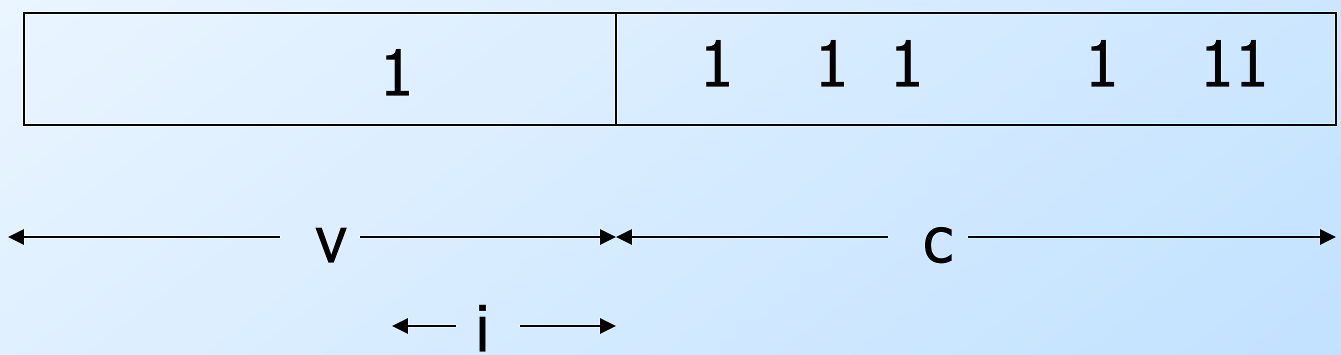
# Subset Sum is in **NP**

- Guess a subset of the set S.
- Add ‘em up.
- Accept if the sum is K.

# Polytime Reduction of 3SAT to Subset Sum

- Given 3SAT instance, F, we must construct a set S of integers and a budget K.
- Suppose F has c clauses and v variables.
- S will have base-32 integers of length c+v, and there will be 3c+2v of them.

# Picture of Integers for Literals

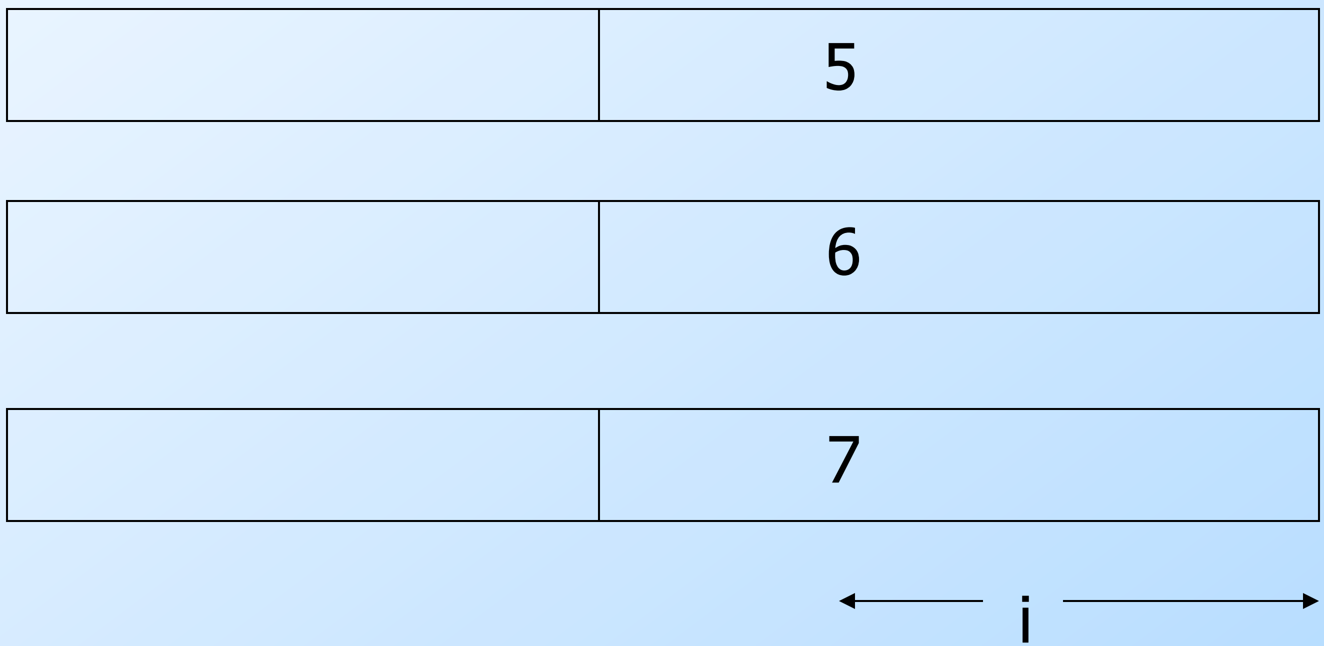



1 in i-th position if this integer is for $x_i$ or $-x_i$.

1's in all positions such that this literal makes the clause true.

All other positions are 0.

# Pictures of Integers for Clauses




For the i-th clause

# Example: Base-32 Integers

(x + y + z)(x + -y + -z)

- c = 2; v = 3.
- Assume x, y, z are variables 1, 2, 3, respectively.
- Clauses are 1, 2 in order given.

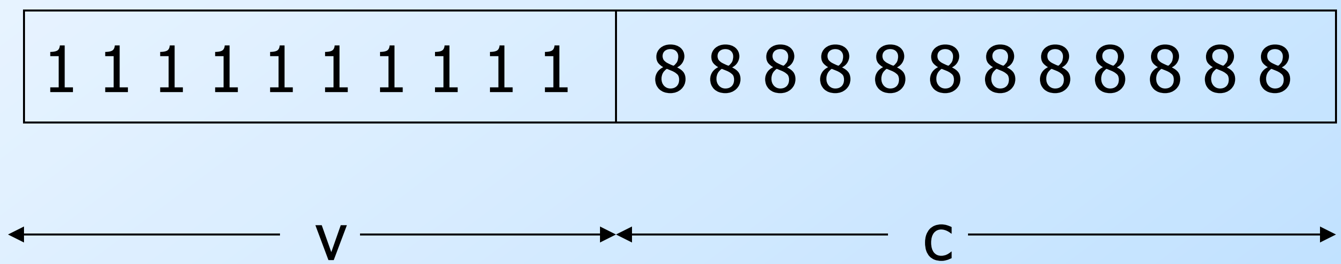
# Example: (x + y + z)(x + -y + -z)

- For x: 00111
- For -x: 00100
- For y: 01001
- For -y: 01010
- For z: 10001
- For -z: 10010

- For first clause: 00005, 00006, 00007
- For second clause: 00050, 00060, 00070

# The Budget

- $K = 8(1+32+32^2+\ldots+32^{c-1}) + 32^c(1+32+32^2+\ldots+32^{v-1})$




- That is, 8 for the position of each clause and 1 for the position of each variable.
- Key Point: there can be no carries between positions.

# Key Point: Details

- Among all the integers, the sum of digits in the position for a variable is 2.
- And for a clause, it is 1+1+1+5+6+7 = 21.
  - 1's for the three literals in the clause; 5, 6, and 7 for the integers for that clause.
- Thus, the budget must be satisfied on a digit-by-digit basis.

# Key Point: Concluded

- Thus, if a set of integers matches the budget, it must include exactly one of the integers for x and -x.
- For each clause, at least one of the integers for literals must have a 1 there, so we can choose either 5, 6, or 7 to make 8 in that position.

# Proof the Reduction Works

- Each integer can be constructed from the 3SAT instance F in time proportional to its length.
  - Thus, reduction is $O(n^2)$.
- If F is satisfiable, take a satisfying assignment A.
- Pick integers for those literals that A makes true.

# Proof the Reduction Works – (2)

- The selected integers sum to between 1 and 3 in the digit for each clause.
- For each clause, choose the integer with 5, 6, or 7 in that digit to make a sum of 8.
- These selected integers sum to exactly the budget.

# Proof: Converse

- We must also show that a sum of integers equal to the budget k implies F is satisfiable.
- In each digit for a variable x, either the integer for x or the digit for -x, but not both is selected.
  - let truth assignment A make this literal true.

# Proof: Converse – (2)

- In the digits for the clauses, a sum of 8 can only be achieved if among the integers for the variables, there is at least one 1 in that digit.
- Thus, truth assignment A makes each clause true, so it satisfies F.

# The *Partition* Problem

- Given a list of integers L, can we partition it into two disjoint sets whose sums are equal?
  - E.g., L = (3, 4, 5, 6).
  - Yes: 3 + 6 = 4 + 5.
- Partition is NP-complete; reduction from Subset Sum.

# Reduction of Subset Sum to Partition

- Given instance (S, K) of Subset Sum, compute the sum total, T, of all the integers in S.
  - Linear in input size.
- Output is S followed by two integers: 2K and T.
- Example: S = {3, 4, 5, 6}; K = 7.
  - Partition instance = (3, 4, 5, 6, 14, 18).

# Proof That Reduction Works

- The sum of all integers in the output instance is 2(T+K).
  - Thus, the two partitions must each sum to exactly T + K.
- If the input instance has a subset, S', of S that sums to K, then pick it plus the integer T to solve the output Partition instance:
  - T + S' = T + K = (T - K) + 2K = (T - S') + 2K

# Proof: Converse

- Suppose the output instance of Partition has a solution.
- The integers T and 2K cannot be in the same partition.
  - Because their sum is more than half 2(T+K).
- Thus, the subset, S', of S that is in the partition with T sums to K:
  - T + S' = (T - S') + 2K; Hence, 2S' = 2K.
  - Thus, S' = K, i.e., it solves Subset Sum.