**1. Prove TSP-Decision is NP-complete**

1) Show that TSP belongs to NP.

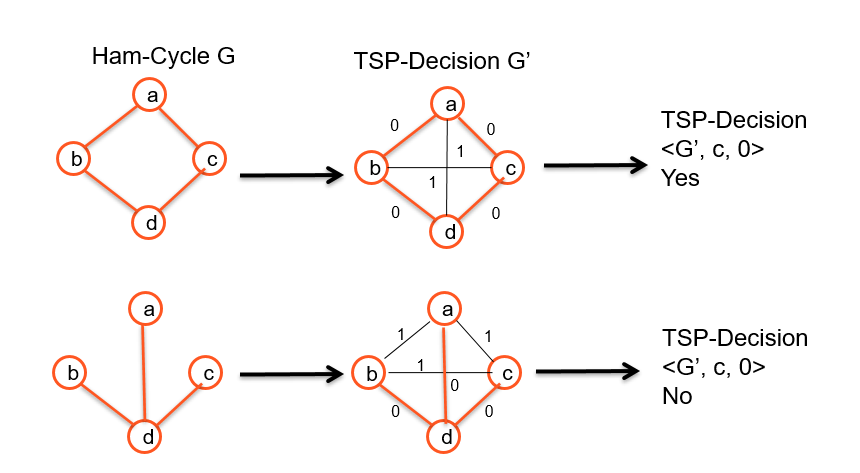
Given an instance of the problem the certificate is the sequence of *n* vertices (cities) in the tour.

The certifier (verification algorithm) checks that

* this sequence contains each vertex exactly once,
* sums up the edge costs and checks whether the sum is at most ***k***.

This process can be done in polynomial time.

Therefore TSP-Decision is in NP



2) Prove that TSP is NP-hard. We can show that

Ham-cycle ≤ p TSP.

where Ham-cycle ∈ NP-Complete

Let *G=(V,E)* be an instance of Ham-cycle. We construct an instance of TSP as follows

* + Form the complete graph *G' = (V,E')* where *E' = { (i,j) : i, j ∈ V and i≠j}* and
  + Define the cost function *c* by *c(i,j) = { 0 if (i,j) ∈ E, 1 if (i,j) ∉ E }*

The instance of TSP is then *<G',c,0>* which is easily formed in polynomial time.

By proving 2) TSP-Decision is NP-Hard. Since 1) held too then we have shown that TSP-Decision is NP-Complete

We now show that graph ***G*** has a Hamiltonian cycle if and only if graph ***G'*** has a tour of cost at most 0.

Suppose the graph ***G*** has a Hamiltonian cycle ***h***.

Each edge in ***h*** belongs to ***E*** and thus has a cost 0 in ***G‘***

Thus ***h*** is a tour in ***G'*** with cost 0

Conversely suppose that graph ***G'*** has a tour ***h'*** of cost at most 0.

Since the cost of edges in ***E'*** are 0 and 1, the cost of tour ***h'*** is exactly 0 and each edge on the tour must have cost 0.

Thus ***h'*** contains only edges in ***E***.

Hence we conclude that ***h'*** is a Hamiltonian cycle in graph ***G***.

By proving 2) TSP-Decision is NP-Hard. Since 1) held too then we have shown that TSP-Decision is NP-Complete

SUBSET-SUM

Instance: A set of numbers denoted S and a target number t.

Problem: To decide if there exists a subset S’ ⊆ S, s.t Σy∈S’ y=t.

Prove : SUBSET-SUM is NP-Complete

Proof:

1. Show SUBSET-SUM is in NP.
2. Show 3SAT≤pSUBSET-SUM.

1.SUBSET-SUM is in NP

Given a set S and target t:

* Verify that S’⊆S is a solution
* The answer is YES iff Σy∈S’y=t.

The length of the certificate: O(n) (n=|S|)

Time complexity: Is the time to add the numbers in S’ which is O(n).

Reducing 3SAT to SubSet Sum

**Proof idea:**

* Choosing the subset numbers from the set S corresponds to choosing the assignments of   
  the variables in the 3SAT formula.
* The different digits of the sum correspond   
  to the different clauses of the formula.
* If the target t is reached, a valid and satisfying assignment is found.
* Let φ∈3CNF with k clauses and variables x1,…,x.
* Create a Subset-Sum instance <Sφ,t> by:   
  2+2k elements of   
  Sφ = {y1,z1,…,y,z,g1,h1,…,gk,hk}
  + yj indicates positive xj literals in clauses
  + zj indicates negated xj literals in clauses
  + gj and hj are dummies
  + and

Proof 3SAT ≤P Subset Sum

* For every 3CNF φ, take target t=1…13…3 and the corresponding set Sφ.
* If φ∈3SAT, then the satisfying assignment defines a subset that reaches the target.
* Also, the target can only be obtained via a set that gives a satisfying assignment for φ.

Phi belongs to 3 SAT if and only if <S phi, 1, ..13..3> belongs to SubsetSum

* 1. Knapsack

1. Prove the following knapsack problem to be NP complete
2. Decision version
3. n objects, each with a weight wi > 0 and a benefit bi > 0 capacity of knapsack : W. Can you fill the knapsack so that the sum of benefits is at least K?
4. For all item i in the solution set S.
5. ∑bi ≥ K and ∑wi ≤ W

Prove - KNAPSACK is NP-Complete

1) Show NP – Verify a solution in polynomial time

2) Show NP-Hard. Reduce SUBSET-SUM to KNAPSACK

1) Show NP – Verify a solution in polynomial time

Given a certificate solution X = { x1, …, xn } can we verify in poly time.

* Sum the weights of x∈X must be ≤ W. Time O(n)
* Sum the benefits of x∈X must be ≥ K. Time O(n)

2) Show NP-Hard. SUBSET-SUM ≤P KNAPSACK. How can you use Knapsack to solve Subset-Sum

Yes Y

b

a

f

KNAPSACK

No

Subset Sum

2) Show NP-Hard. SUBSET-SUM ≤P KNAPSACK. How can you use Knapsack to solve Subset-Sum

Reduction from *SUBSET-SUM* to Decision-*KNAPSACK*:

*SUBSET-SUM* = { <*S*, *t*> | *S = {y1, …, yk} and for some*

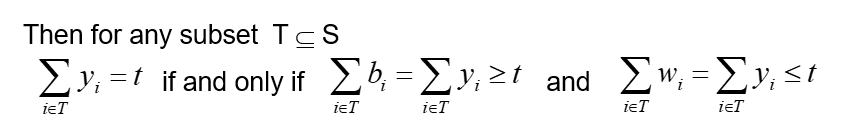
Subset T = {yj, …, yl } ⊆ S, Σyi = t }

Set bi = yi and wi= yi

Set W = t and K = t

Then for any subset T belonging to S,

Summation of yi = t if and only if Summation of



Example: Reduce SUBSET-SUM to KNAPSACK.

*SUBSET-SUM* = < *{12, 6, 8, 13, 20}*, *26* >

Set bi = wi = xi

Set W=k =26

Knapsack has capacity 26. Is there a subset of items that will fit in the knapsack and have a total benefit of at least 26?

|  |  |  |
| --- | --- | --- |
| **item** | **weight** | **benefit** |
| 1 | 12 | 12 |
| 2 | 6 | 6 |
| 3 | 8 | 8 |
| 4 | 13 | 13 |
| 5 | 20 | 20 |
|  |

Yes T = { item2, item 5}. This corresponds to a subset sum T = {6, 20} sum 26

3-SAT (CNF)

**A special of CNF problem:** Each clause contains three boolean literals

*Φ* = (x1 ∨ ¬x1 ∨ ¬x2) ∧ (x3 ∨ x2 ∨ x4) ∧ (¬x1 ∨ ¬x3 ∨ ¬ x4)

**3-SAT** is NP-Complete

* + 3-SAT is in NP
  + SAT ≤p 3-SAT

Is 4-SAT NP-Complete?

**Clique**

**Clique Problem:**

* + Undirected graph G = (V, E)
  + **Clique:** a subset of vertices in V all connected to each other by edges in E (i.e., forming a complete graph)
  + **Size of a clique:** number of vertices it contains

**Optimization problem:**

* + Find a clique of maximum size

**Decision problem:**

* + Does G have a clique of size k?

Given: an undirected graph G = (V, E)

* **Problem**: Does G have a clique of size k?
* **Certificate**:
  1. A set of k nodes
* **Verifier**:
  1. Verify that for all pairs of vertices in this set there exists an edge in E
* CLIQUE = { <G,k> | G is a graph with a clique of size k }
* A clique is a subset of vertices that are all connected

3-CNF ≤p Clique

We can prove this byConstructing a graph G such that *Φ* is satisfiable only if G has a clique of size k

* Pick an instance of 3-SAT, Φ, and transform into <G,k> an instance of Clique
* If Φ has m clauses, we create a graph with m clusters of 3 nodes each and set k = m.
* Each cluster corresponds to a clause.
* Each node in a cluster is labeled with a literal from the clause.
* We do not connect any nodes in the same cluster
* We connect nodes in different clusters whenever they are not contradictory
* Any k-clique in this graph corresponds to a satisfying assignment

**Prove that SET-PARTITION is in NP-complete**

Given a Set S, can we partition S into S and S-X of equal sum?

To show that SET-PARTITION is NP-Complete, we need to show:

1. That SET-PARTITION ∈ NP. Given a partition of set S we can verify in polynomial time that the two subsets X and S-X have equal sums by adding the values in each set. This clearly takes time O(n) where n is the number of elements in S.
2. That some NP-Complete problem A can be reduced to SET-PARTITION in polynomial time and the original problem A has a yes solution if and only if SET-PARTITION has a yes solution.

We will select A to be SUBSET-SUM which has been proven to be in NP-Complete. SUBSET-SUM is defined as follows: Given a set S of integers and a target number t, find a subset Y ⊆ S such that the members of Y add up to exactly t.

We will need to show a polynomial time reduction from SUBSET-SUM to SET-PARTITION, SUBSET-SUM SET-PARTITION

Let s be the sum of numbers in S. Feed S’ = S ∪ {s − 2t} into SET-PARTITION and answer YES if and only if SET-PARTITION answers YES.

This reduction takes polynomial time since all we did was add a single element, s − 2t to S. To calculate s-2t we must computer the sum of all numbers in S which takes O(n) time.

*Note: sum(S’) = sum(S) + (s-2t) = s + (s-2t) = 2s − 2t = 2(s – t) .*

We must show that <S, t> ∈ SUBSET-SUM iff <S’> ∈ SET-PARTITION. In other words There exists a subset of S that sums to t if and only if there exists a set partition of S’ with equal sums.

1) If Y is a solution to <S,t> SUBSET-SUM then S’ has a SET-PARTITION.

If there exists a subset Y of numbers in S that sum to t then the remaining numbers in S-Y sum to s − t. Now S’ = (S-Y) ∪ Y ∪ {s-2t} such that we can partition S’ into two sets (S-Y) and Y ∪ {s-2t} with each partition summing to s − t. Therefore there is a solution to SET-PARTITION.

2) If there exists a partition of S’ then there exists a solution to <S, t> ∈ SUBSET-SUM.

Recall sum(S’) = 2(s – t) and S’ = S ∪{s-2t} .

If there exists a partition of S’ into two sets X and S’-X such that the sum over each set is s−t then one of these sets say X must contain the number s−2t. By removing it, we get a set X’ = X- {s-2t} with sum(X’) = (s – t) - (s-2t) = t, and since S’ = S ∪ {s − 2t} all of the elements in X’ are in S. Therefore there exists a subset X’ of S that sums to t.

**Since, SET-PARTITION ∈ NP and SUBSET-SUM ≤p SET-PARTITION, SET-PARTITION is in NP-Complete.**

**Bin Packing–Dec. is NP-complete**

Bin Packing problem: Given n items of sizes a1, a2,…, an (0 < ai ≤ 1), pack these items in at most k bins of size 1.

1. Bin packing in in NP
   * To verify a solution
     + Add the weights of the items in each bin.
     + Each bin must contain < 1unit.
     + Check that each item is in a bin
     + There are at most k bins used.
   * This can be done in O(n).
2. SET-PARTITION reduces to Bin Packing

SET-PARTITION ≤p Bin Packing

SET-PARTITION: Given a set of numbers X = { *x1,x2,… xk* }. Is there a subset of X, B, such that the sum of the elements in B is equal to the sum of the elements in S-B.

Bin Packing : Given n items of sizes a1, a2,…, an (0 < ai ≤ 1), pack these items in at most k bins of size 1.

It can be done in the following way:

Let sum = . Define S = { *s1,s2,… sk* }

where for i = 1, ..k. Then if { *s1,s2,… sk* } can be

packed into 2 bins, X can be partitioned into 2 sets.

Thus Bin Packing is NP-Complete