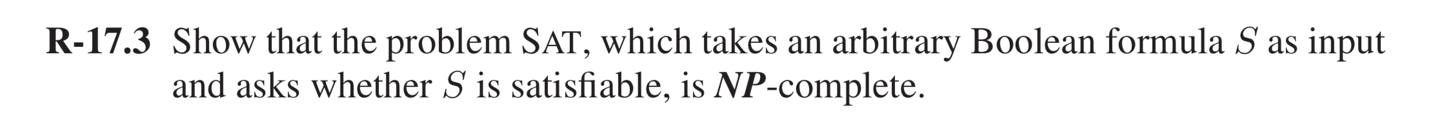
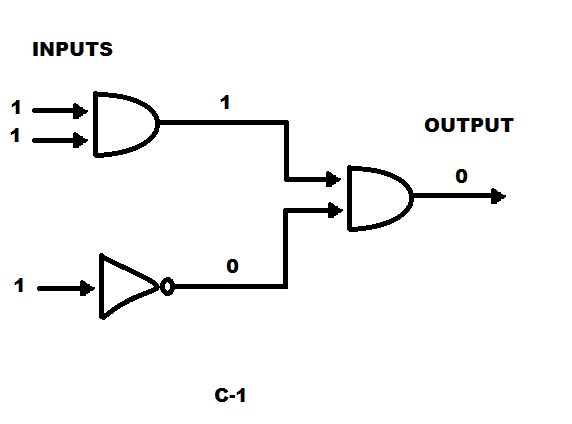
CS 600 Homework 10 | CWID 10430147 | Divyendra Patil | Username: dpatil3  
Date: 17/10/2017

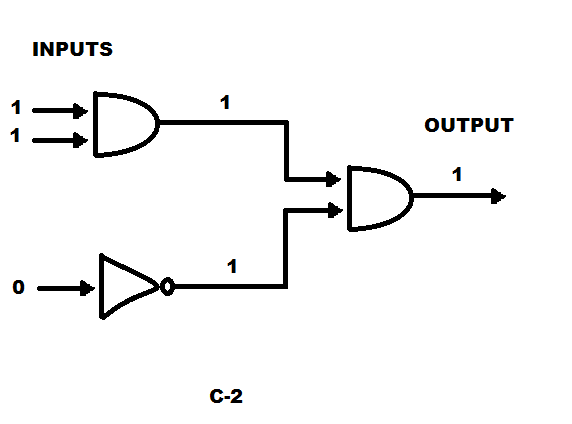


**Solution:** It is not difficult to show that CNF-SAT is in **NP**, for a given Boolean formula S, we can construct a simple nondeterministic algorithm that first “guesses” an assignment of Boolean values for the variables in S and then evaluates each clause of S in turn. If all the clauses of S evaluate to 1, then S is satisfied; otherwise, it is not.

So to show that CNF-SAT is **NP**-hard, we will reduce the CIRCUIT-SAT problem to it in polynomial time. So, suppose we are given a Boolean circuit, C. Without loss of generality, we assume that each AND and OR gate has two inputs and each NOT gate has one input. To begin the construction of a formula S equivalent to C, we create a variable xi for each input for the entire circuit C.

We consider the two Boolean circuits below





There are Two Cases-

Case 1: If any check for a gate fails, or if the guessed value for the output is 0 then we output “No”.

Case 2: If the checks for every gates succeeds and the output is 1 then the algorithm outputs “Yes”. Hence, as seen from the above circuits, circuit C1 is NOT SATISFIABLE, and if circuit C2 is SATISFIABLE with output value of 1 , that is “YES”.

If the function is true then, it is an NP complete problem. Thus, C1 is NOT SAT and hence it is NP complete problem.

By reference In the book, The Cook-Levin Theorem Theorem 17.5: CIRCUIT-SAT is NP-complete.



**Solution:** A clique in a graph G is a subset C of vertices such that, for each v and w in C, with v ≠ w, (v,w) is an edge, there is an edge between every pair of distinct vertices in C.

The Problem CLIQUE takes a graph G and an integer k as input and asks whether there is a clique in G of size at least k.

To prove a problem to be NP:

a. We use a verification algorithm which solves a problem in polynomial time and determines yes for given certificate.

b. If a situation is created where “certificate is not given” takes exponential time, then resolution is a problem.

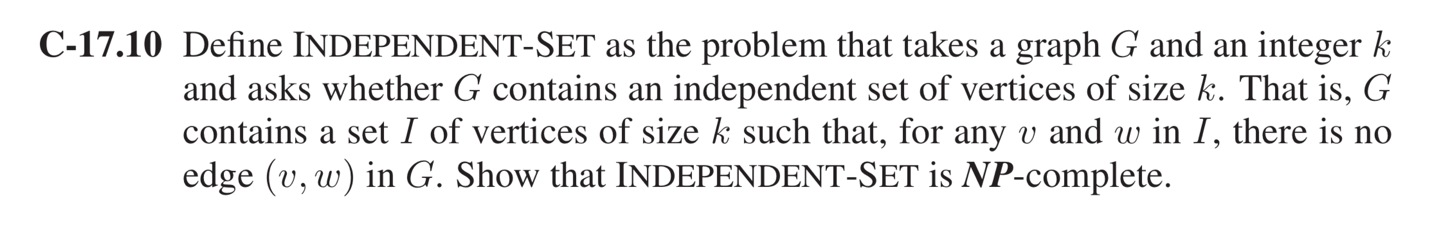
We give a certificate as the k vertices then we can find if a graph has k clique in polynomial time i.e. algorithm can check if there is edge between 2 vertices.

Hence, the verification algorithm can say yes in polynomial time when we provide it a certificate.

But for any graph G if we are supposed to find the max clique and if there are too many vertices, there is no certificate or verification algorithm to say yes then it will take exponential time to determine the resolution of such problem. Since we know there are k vertices in G, the above situation will not occur.

Hence, we can verify in polynomial time that they form a complete graph.

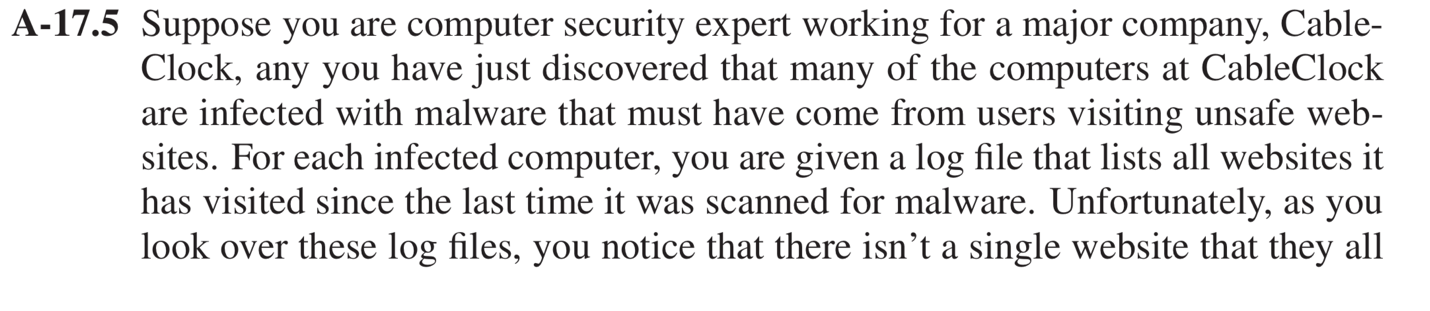
Thus, CLIQUE ∈ NP.

Solution: To prove this is a NP problem. We can do this by guessing an independent set of size k and check it in polynomial time.

So to show that it is NP-hard, we reduce from VERTEX-COVER. An instance of VERTEXCOVER is a graph G and a positive integer k. We accept if there is a set of k vertices such that every edge is incident to at least one of the vertices in our set. An independent set is a set of k vertices that have no edges between them.

Also note that, if S is a vertex cover of G, then V − S is an independent set, since at least one vertex of every edge is contained in S. Similarly, if S 0 is an independent set, then V − S 0 is a vertex cover.

Hence our reduction will take G and k and produce the same graph G, and the integer n − k. (This obviously takes polynomial time.) There is a vertex cover of size k if and only if there 1 is an independent set of size n − k. Therefore, INDEPENDENT-SET is NP-complete.



To prove that problem is NP-Complete, we need to show that it is NP-Hard.

We will use the SET-COVER approach to solve this problem.

SET-COVER consists of collection of m set from S1 to Sm and an integer parameter k as input and we find whether there is a sub-collection of k set Si1 to Sik.

Now we consider that the m set in set-cover problem is the m computers and k number of sets are needed to be checked to find the malware.

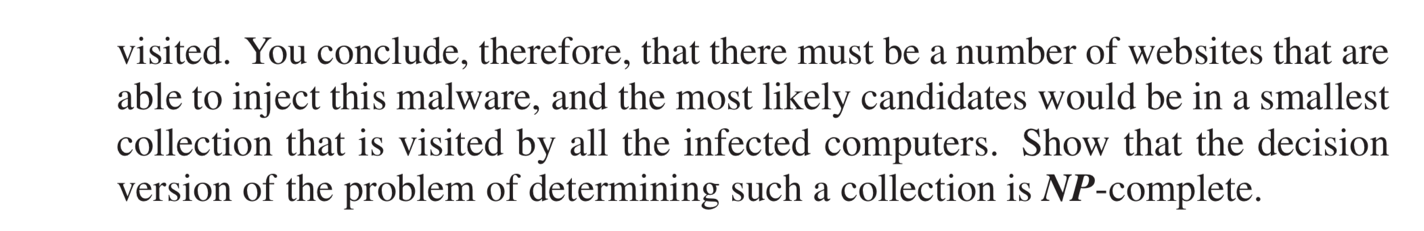
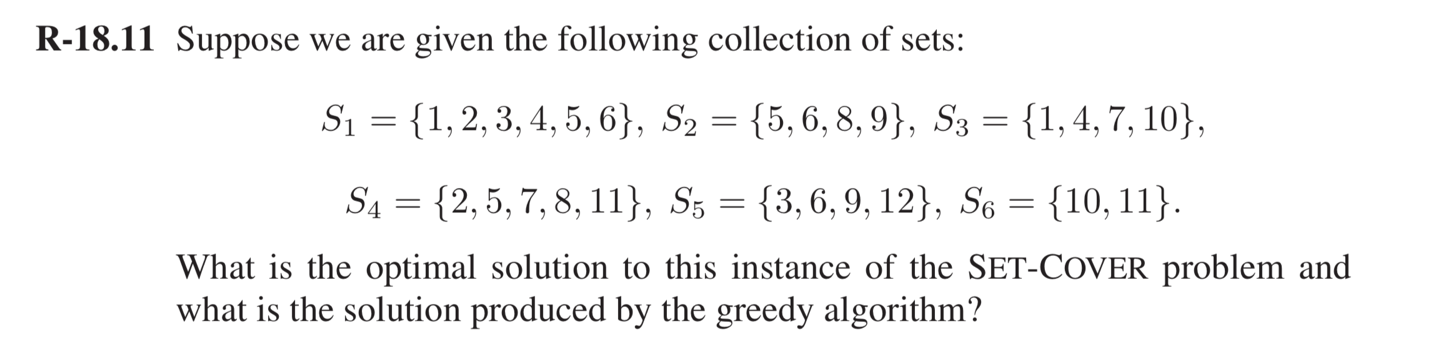
Hence, we are finding here the smallest collection that is visited by all the m computers.

We know that a Set-Cover is a NP problem. As to the reduction, we note that we can define an instance of SET-COVER from an instance G and k of VERTEX-COVER.

Namely, for each vertex v of G, there is set Sv, which contains the edges of G incident on v.

Clearly, there is a set cover among these sets Sv of size k if and only if there is a vertex cover of size k in G SET-COVER is NP -complete problem.

Hence, above problem is NP-Complete problem.



**Look at the greedy approach:**

Greedy approach algorithm selects sets one at a time, each time selecting the set that has the most uncovered elements. When every element in U is covered, we are done. We give a simple pseudocode description as follows:

**Algorithm** SetCoverApprox(S):

**Input**: A collection S of sets S1, S2, . . . , Sm whose union is U

**Output**: A small set cover C for S

C ← ∅ // The set cover built so far

E ← ∅ // The elements from U currently covered by C

**while** E = U **do**

select a set Si that has the maximum number of uncovered elements

add Si to C

E ← E ∪ Si

Return C.

**Algorithm 18.7**: An approximation algorithm for SET-COVER.

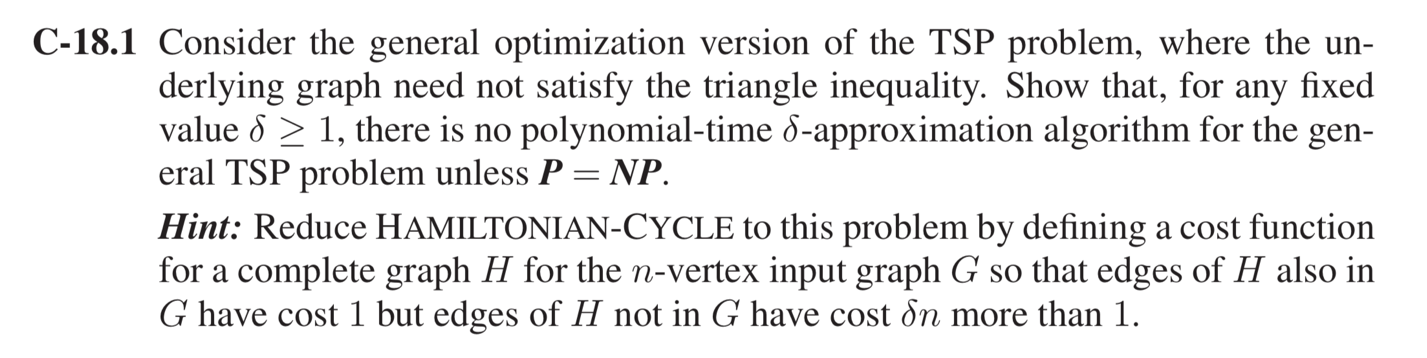
The algorithm runs in polynomial time.

The greedy approach will always select those sets who have maximum number of uncovered Elements.

So, iterating through set S1 to S6,

The Solution with Greedy Algorithm would be C → {{S1 S4 S5 S3}}

The Optimal Solution is C → {S3,S4,S5}



**Solution**: In the TSP (Traveling Salesperson Problem),

we are given an undirected graph G = (V, E) and cost c(e) > 0 for each edge e ∈ E.

Our goal is to find a Hamiltonian cycle with minimum cost.

A cycle is said to be Hamiltonian if it visits every vertex in V exactly once. TSP is known to be NP-Hard. Moreover, we cannot hope to find a good approximation algorithm for it unless P = NP. This is because if one can give a good approximation solution to TSP in polynomial time, then we can exactly solve the NP-Complete Hamiltonian cycle problem (HAM) in polynomial time, which is not possible unless P = NP.

Assume that there is an Approximation Algorithm **A** having a **δ** factor as an integer.   
This can be solved using A on **HAMILTON-CYCLE** problem.

Since, HAMILTON-CYCLE is a NP-complete problem, we can solve it only if P = NP. HAMILTONIAN-CYCLE is the problem that takes a graph G and asks whether there is a cycle in G that visits each vertex in G exactly once, returning to its starting vertex

Let there be a Hamilton cycle **problem G(V, E).** With this we check if A contains Hamilton cycle.

Assume H = (V, E′) to be the complete graph on V

So, for a set of vertices V we have a Hamilton Cycle Graph and a Complete Graph

A Complete graph is a graph having an edge between each distinct vertex.

We assign an integer to each Edge E’

**c(u, v) = 1 if (u, v) ∈ E**

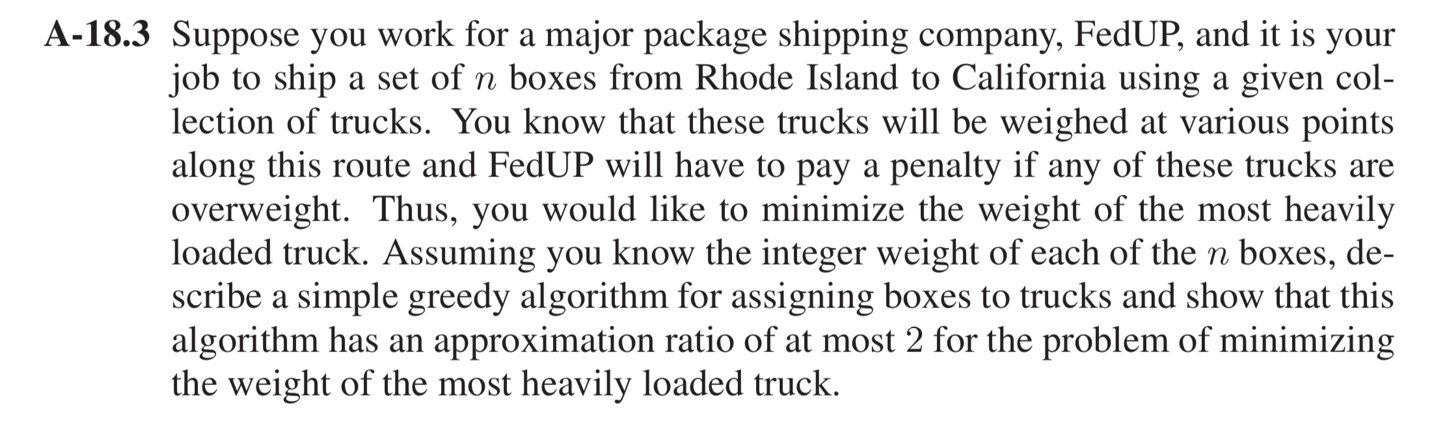
**δn + 1 else**

Now, we have a TSP problem (H, c).

Suppose that G(V, E) contains a Hamilton cycle G′. So, each edge in E has a value 1 that we get from above function c. Hence, (H,t) has a tour with a cost |V|.

If we had considered that there is no Hamilton Cycle for graph G, then a tour in H would contain edges that are not present in E. In such case the cost would be greater than δn + 1.

It is implied that we can use A to solve Hamiltonian Cycle with polynomial Cost. Hence, we can say that for any fixed value δ ≥1, there is no polynomial-time δ-approximation algorithm for the general TSP problem unless P =NP



**Solution**: Suppose there are n items to unload and the weights are w1, w2, . . . , wn, then let N\* be the optimal number of trucks needed. Then, since each truck cannot carry more then K units of load, we get that

i ≤ KN\*

and so N\* ≥ 1/K i …..(1)

Now let N be the number of trucks that the greedy algorithm finds. We prove that it is within a factor two of the minimum possible number, for any set of weights and any value of K.

**Proof**

Let Ij denote the set of items that truck j loads and let Wj be the total weight of the items in Ij, that is Wj := w(a).

By analyzing the greedy algorithm we can conclude that the following holds for any j > 1,

Wj + Wj-1 > K

On the other hand, we have that

Suppose N = 2m, for some m, then

