**Chapter 17**

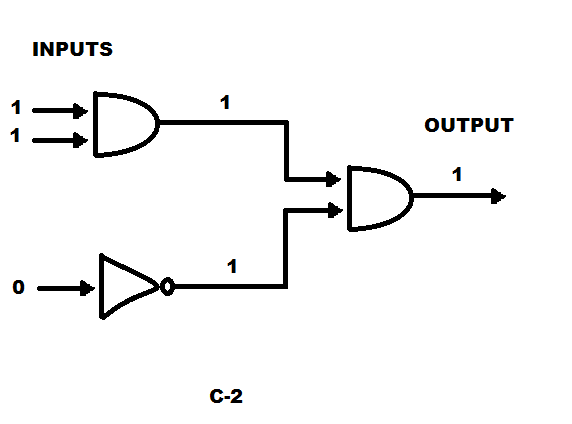
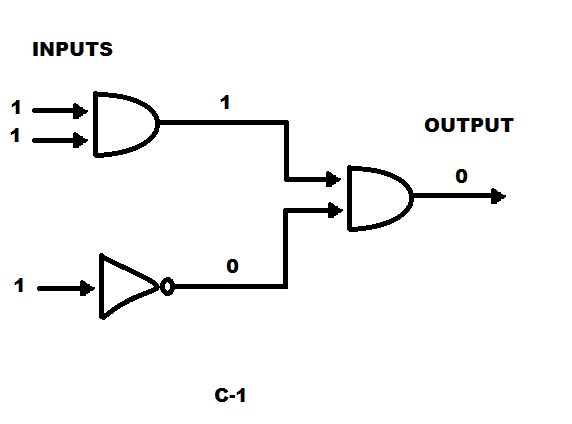
**R-17.3** Show that the problem SAT, which takes an arbitrary Boolean formula *S* as input and asks whether *S* is satisfiable, is *NP*-complete.

*Answer:*

It is not hard to show that CNF-SAT is in NP, for a given Boolean formula S, we can construct a simple nondeterministic algorithm that first attempts an assignment of Boolean values for the variable in S through guesses and then evaluates each clause of S in turn. If all clauses in formula 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 there is a Boolean circuit C, loss of generality it is assumed that each AND & OR gate has two inputs and each NOT gate has one input. TO start the construction of the formula S equivalent to C, a variable Xi is crated for each input of the entire circuit C.

Follows circuits are considered:



So, there are two cases:

Case1- If any check for a gate fails, or if the guessed value for the output is 0 then the output is “No.”

Case 2- If the checks for every gate succeeds and the output is 1 then the algorithm outputs “Yes”. Hence, as seen from the above circuits, circuit in c1 is not satisfiable, and if circuit in c2 is satisfiable output would be value 1 which 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 Cook- Levin Theorem Circuit-SAT is NP-Complete.

**R-17.7** Show that the CLIQUE problem is in *NP*.

*Answer:*

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

The problem is CLIQUE takes a graph G and an Integer K as input and asks whether there is clique in G of size at least K.

To prove a problem to be NP: we use verification algorithm which solves a problem in polynomial time and determines YES for given certificate. And If a situation is created where certificate is not given it 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 is 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 it would take exponential time to determine the resolution of such problem. As K vertices is in G the above conflict will not occur. Therefore, it can be verified in polynomial time that they form a complete graph.

Hence, CLIQUE problem in in NP.

**C-17.10** Define INDEPENDENT-SET as the problem that takes a graph *G* and an integer *k* and asks whether *G* contains an independent set of vertices of size *k*. That is, *G* contains a set *I* of vertices of size *k* such that, for any *v* and *w* in *I*, there is no edge (*v,w*) in *G*. Show that INDEPENDENT-SET is *NP*-complete.

*Answer:*

To show that Independent Set is NP-Complete, independent set of size K is guessed and checked in polynomial time. So, to prove that it is NP- complete Vertex-Cover is reduced. An instance of Vertex cover is graph G and a positive integer K. Here, if there is a set of K vertices it would be in such a way that every edge is incident to atleast one of the vertices in our set. An independent set is a set of k vertices that have no edges between them. 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 when S0 is an independent set, then V- S0 is a vertex cover. Therefore, it shows that G and K is used to produce same graph G, and the integer n- k. There is a vertex cover of size k if and only if there 1 is an independent set of size n – k and hence independent set is NP – complete.

**A-17.5** Suppose you are computer security expert working for a major company, Cable-Clock, any you have just discovered that many of the computers at CableClock are infected with malware that must have come from users visiting unsafe websites. For each infected computer, you are given a log file that lists all websites it has visited since the last time it was scanned for malware. Unfortunately, as you look over these log files, you notice that there isn’t a single website that they all visited. You conclude, therefore, that there must be a number of websites that are able to inject this malware, and the most likely candidates would be in a smallest collection that is visited by all the infected computers. Show that the decision version of the problem of determining such a collection is *NP*-complete.

*Answer:*

The decision version of the problem determining such a collection is NP-complete is as following:

Set cover approach is used to solve this problem. Set cover consists of collection of m set from S1 to Sn and an integer parameter k as input and we find whether there is a sub-collection of k set Si1 to Sik. The N set in set-cover problem is the n computers and k number of sets are needed to be checked to find the malware. Through this the smallest collection that is visited by all the m computers are found. The Ste-Cover is a Np Problem because of the reduction that is defined as an instance of Set-Cover from an instance G and K of Vertex-Cover. Here for each vertex v og G, there is a set Sv which consists the edges of G incident on V. This shows that 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 in Np- complete problem. So, the above problem is NP-Complete problem.

**Chapter 18**

**R-18.11** Suppose we are given the following collection of sets:

*S*1 = *{*1*,* 2*,* 3*,* 4*,* 5*,* 6*}, S*2 = *{*5*,* 6*,* 8*,* 9*}, S*3 = *{*1*,* 4*,* 7*,* 10*},*

*S*4 = *{*2*,* 5*,* 7*,* 8*,* 11*}, S*5 = *{*3*,* 6*,* 9*,* 12*}, S*6 = *{*10*,* 11*}.*

What is the optimal solution to this instance of the SET-COVER problem and

what is the solution produced by the greedy algorithm?

*Answer:*

The greedy algorithm approach:

It selects sets one at a time, each time sleecting the set that has the most uncovered elements. When every element in the Universal set is covered the algorithm ends.

Algorithm is SetcoverApprox(S) where the input is a collection S of sets S1, S2, S3, …. Sm, whose union is U and the output would be a smallest set cover C for S

Here C ← ∅ which is the set cover built so far and E ← ∅ which is 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 U Si which would return C.

Also look at Algorithm 18.7

This algorithm runs in polynomial time.

To analyze the approximation factor of the above greedy SET-COVER algorithm,

we will use an amortization argument based on a charging scheme (Section 1.4).

Namely, each time our approximation algorithm selects a set Sj, we will charge the

elements of Sj for its selection.

**C-18.1** Consider the general optimization version of the TSP problem, where the underlying graph need not satisfy the triangle inequality. Show that, for any fixed value *δ ≥* 1, there is no polynomial-time *δ*-approximation algorithm for the general TSP problem unless *P* = *NP*.

*Hint:* Reduce HAMILTONIAN-CYCLE to this problem by defining a cost function for a complete graph *H* for the *n*-vertex input graph *G* so that edges of *H* also in *G* have cost 1 but edges of *H* not in *G* have cost *δn* more than 1.

*Answer:*

In this Traveling Salesperson Problem(TSP), there is an undirected graph G = (V, E) and cost c(e) > 0 for each edge e ∈ E. The 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. Also, it is hard 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 the NP-Complete Hamiltonian cycle problem (HAM) in polynomial time in the exact time, which is not possible unless P = NP.

If there is an Approximation Algorithm A having a δ factor as an integer.   
It can be solved using A on HAMILTON-CYCLE problem.

Since, HAMILTON-CYCLE is a NP-complete problem, it can be solved 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 if A contains Hamilton cycle is checked.

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.

Assign an integer to each Edge E’ c(u, v) = 1; if (u, v) ∈ E; δn + 1 else

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

**A-18.3** Suppose you work for a major package shipping company, FedUP, and it is your job to ship a set of *n* boxes from Rhode Island to California using a given collection of trucks. You know that these trucks will be weighed at various points along this route and FedUP will have to pay a penalty if any of these trucks are overweight. Thus, you would like to minimize the weight of the most heavily loaded truck. Assuming you know the integer weight of each of the *n* boxes, describe a simple greedy algorithm for assigning boxes to trucks and show that this algorithm has an approximation ratio of at most 2 for the problem of minimizing the weight of the most heavily loaded truck.

*Answer:*

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 than K units of load, result is

i ≤ KN\*

and so N\* ≥ 1/K i …. (1)

Now let N be the number of trucks that the greedy algorithm finds. Let’ss 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, It can be concluded that the following holds for any j > 1,

Wj + Wj-1 > K

On the other hand, there is another equation that says

Suppose N = 2m, for some m, then

