Qn.2-1 (34-5-2)

show that 3-CNF-SAT ≤P 01LP.
Proof:
Let φ be 3-CNF which has n input variables and m clauses.
We construct an instance of 01LP as follows.
Let A be a m + 2n by 2n matrix.
For $1 \leqslant i \leqslant m$,
if ($1 \leqslant j \leqslant n$ && clause C_i contains the literal x_j : set entry A(i, j) to -1
Otherwise set it to 0.
For $n+1 \leqslant j \leqslant 2n$,
if clause C_i contains the literal $\neg x_{j-n}$: set entry A(i, j) to -1
Otherwise set it to 0.

When $m + 1 \leqslant i \leqslant m + n$,
if i−m = j or i−m = j −n: set A(i, j) = 1
else: 0
When $m + n + 1 \leqslant i \leqslant m + 2n$,
if i − m− n = j or i − m − n = j − n: set A(i, j) = −1
else: 0
for example:
A :[-1 -1 -1 0]
…..[-1 0 1 1]
Let b be a m + 2n-vector.
Set the first k entries to -1, the next n entries to 1, and the last n entries to -1.
b [-1]
… [1]

So, we can conclude that the time of constructing A and b is polynomial time.

Reduction Algorithm
Algorithm F(φ)

Check whether φ is in 3-CNF format if it is the return (A =[1], b=[2]), let m be the number of clauses and n be the number of variables in φ

```
For i =1 to m:
NUM <- 0;
For j=1 to m: set A[i,j] =0
For each literal L in Ci:
        If L =xj, then
                A[i,j] =a [i,j] +1;
                NUM++;
        Else if 1 =7xj then:
                A[i,j] = a[i,j] -1;
                NUM++
                End if
        End for
b[i] =1 –NUM;
End for
Return (A,b)
```

Statement:

If φ is not in 3-CNF format then A=[1] and A =[2], and then system 1x1>=2 has no solution for x1 =1 or 0. It remains to look at the case that φ is in 3-CNF format. In that case, φ is satisfiable if and only if there exists a truth assignment to x1,x2..xn such that this is true if and only if there exists vale 0, 1 assgined to variable x1, x2… xn such that Ax>b, let us analyze each clause Ci. Since clause Ci is satisfied by x1,x2,xn iff at least one of those is assigned to be true

Qn.2-2 (34-5-3)

Since the 0-1 integer-programming problem is NP-hard, we need to show a reduction from the integer to 0-1 problem.

Proof:

If we take the A from 0-1 LP problem, and tack on a copy of the n x n identity matrix to its bottom, and track on n ones to the end of b from the 0-1 integer-programming problem. This has the effect of adding the restrictions that every entry of x must be at most 1. But, since for every i, we need x_i to be an integer rather than just 0 or 1, this only leaves the option that x_i =0 or x_i=1. This means that by adding these restrictions, we have that any solution to this system will be a solution to the 0-1 integer programming problem given by A and b. let y1y2y3 correspond to the truth values of each literals in Ci. So, Ci is satisfied iff at least one of y1y2y3 = 1, which is equaling to saying y1+y2+y3 >=1.

Now we relate be values of y1y2y3 with the values of x1x2 xn. Let l1 l2 and l3 be the literals in Ci. Then if li =xkj then yi= xkj, but if lj = 7xkj then yi = 1-xkj since y1 =1 if xkj =0 and y1 =1 if xkj =1. Substituting this into the equation y1+y2+y3 >=1 we get ak1*xk1 + ak2*xk2+ ak2*xk2 >=1-ni
Where a_{ki} = 1 if li =xkj or =0 if lj =7xki
And n_i is the number of variable in Ci that appear negated. That φ is satisfiable if and only if there exists a 0-1 assinment variables x1….xn such that Ax>=b.
Run in polunomial time:
Checking whether φ is in 3-CNF format can be done in linear time on the number of clauses. The loop on i runs in m steps, the loop om j runs in nsteps the loop on the literals run in constant number of steps, since there are only 3 literals per clause. So the running time of the function is O(nm), which is polynomial time.

Qn.2-2 (34-5-3)

The 0-1 integer programming problem will be proved as NP by using the face that 3-CNF-SAT <= 0-1 integer programming problem.

Let we set a CNF formula φ consists of n variables and m clauses. And build A and b as following:

Now, consider the following examples:

Here, the CNF formula φ is given as $\varphi = (x_1 \vee \sim x_3 \vee \sim x_4) \wedge (x_1 \vee x_2 \vee x_3)$. Then, the following will be exists:

$(x_1 \vee \sim x_3 \vee \sim x_4)$	$(x_1 + (1 - x_3) + (1 - x_4)) \geq 1 \rightarrow (-x_1 + x_3 + x_4) \leq -1$
$(x_1 \vee x_2 \vee x_3)$	$(x_1 + x_2 + x_3) \geq 1 \rightarrow (-x_1 - x_2 - x_3) \leq -1$

Then from the above given CNF formula, matrix corresponds to b and A are given as:

$b = \begin{bmatrix} -1 \\ 1 \end{bmatrix}$ and $A = \begin{bmatrix} -1 & -1 & -1 & 0 \\ -1 & 0 & 1 & 1 \end{bmatrix}$.

Use the algorithm in the next page to find the format of φ.

Now, consider if the format of φ is not in 3-CNF then (A=[1] b=[2]) and the system has no solution for xi ∈0,1.

In that case, if there exists a truth assignment to x1x2xn in such way that every clause is satisfiability then φ is said to be satisfiability.

Then it is true if there exist the values 0,1 assgined to variable x1x2xn in such way that Ax<=b.

Hence, form the above explanation it is proved that integer programming problem is NP-complete.

Consider the following reduction algorithm:

FREDUCTION ($\langle \varphi \rangle$)

1. First performed checking whether the format of $\langle \varphi \rangle$ is in 3-CNF or not.

2. If the format is not in 3-CNF then return $(\langle A=[1], b=[2] \rangle)$

//for loop is used to iterate through 1 to m, where m is the clauses.

3. **for k=1** to **m**

//Assign value zero to the variable NUM

4. NUM $\leftarrow$ 0;

5. **for i=1** to **m do**

6. $A[k,i]=0$;

7. **for** every literals **do**

8. **if** $x_i == 1$ **then**

9. $A[k,i]=-(A[k,i]+1)$;

10. **else if** $\neg x_i == 1$ **then**

11. $A[k,i]=-(A[k,i]-1)$;

12. NUM++;

13. **end if**

14. **end for**

15. $b[k]=-(1-\mathrm{NUM})$;

16. **end for**

17. **retrun** $(\langle A,b \rangle)$