# Algorithm Sixth Homework

Zhang Yuan

Student ID: 2015E8013261189

## 1 Problem 1: Integer Programming

At first, given an answer of this problem, we can verify the instance in polynomial time to decide whether it is suitable: $Ax \geq b$. If A is a m*n matrix, then the time complexity to verify is O(mn).
Then we can reduce 3-SAT Problem to this problem, we can prove the inequality:

$$3 - SAT \leq pIntegerProgramming$$

For example:

$$(V_1 \cup V_2 \cup V_3) \cap (W_1 \cup W_2 \cup W_3)$$

We can reduce the problem to Integer Programming in polynomial time O(3n), n is the number of clauses:

$$\begin{cases} V_1 + V_2 + V_3 \geq 1 \\ W_1 + W_2 + W_3 \geq 1 \\ V_i \in \{0, 1\}, W_i \in \{0, 1\} \end{cases}$$

If we can solve the Integer Programming, then the answer would also make the 3-SAT Problem true, and the answer of 3-SAT Problem is suitable for Integer Programming too.
Because $3 - SAT \in NP - Complete$, Integer Programming is harder than 3-SAT Problem, and it is a NP Problem, so Integer Programming is a NP-Complete Problem.

## 2 Problem 2: Mine-Sweeper

Actually I have seen the proof of Mine-Sweeper's NP-Complete by Richard Kaye, but I think I can give another explanation to show NP-Complete.
At first, given an answer of this problem, we can verify the instance in polynomial time to decide whether the placement is suitable. If it is a m*n chessboard, then the time complexity to verify is O(mn).
Secondly, according to above problem, we know Integer Programming is a NP-Complete problem, we can move on, another Integer Programming s.t. Ax=b is also a NP-Complete problem. We can verify the answer in polynomial time O(mn).
For example:

$$Ax \geq b$$

We can reduce this Integer Programming to another Integer Programming in O(n).

$$Ax - \alpha = b, \alpha >= 0, integer$$

At last, a Mine-Sweeper Problem is an instance of the second Integer Programming, so we can prove the NP-Complete, we can also reduce the Integer Problem to Mine-Sweeper Problem to show the NP-Complete.

## 3 Problem 3: Half-3 SAT

At first, given an answer of this problem, we can verify the instance in polynomial time to decide whether it is suitable. If we have n clauses, then the time complexity to verify is O(3n).
Then we can reduce 3-SAT Problem to this problem, we can prove the inequality:

$$3 - SAT \leq pHalf - 3 - SAT$$

For example:

$$(V_1 \cup V_2 \cup V_3) \cap (W_1 \cup W_2 \cup W_3)$$

We can reduce the problem to Half-3 SAT in polynomial time O(n), n is the number of clauses:

$$(V_1 \cup V_2 \cup V_3) \cap (W_1 \cup W_2 \cup W_3) \cap (0 \cup 0 \cup 0) \cap (0 \cup 0 \cup 0)$$

If we can solve the Half-3 SAT Problem, then the answer would also make the 3-SAT Problem true, and the answer of 3-SAT Problem is suitable for Half-3 SAT Problem too.
Because $3 - SAT \in NP - Complete$, Half-3 SAT Problem is harder than 3-SAT Problem, and it is a NP Problem, so Half-3 SAT Problem is a NP-Complete Problem.

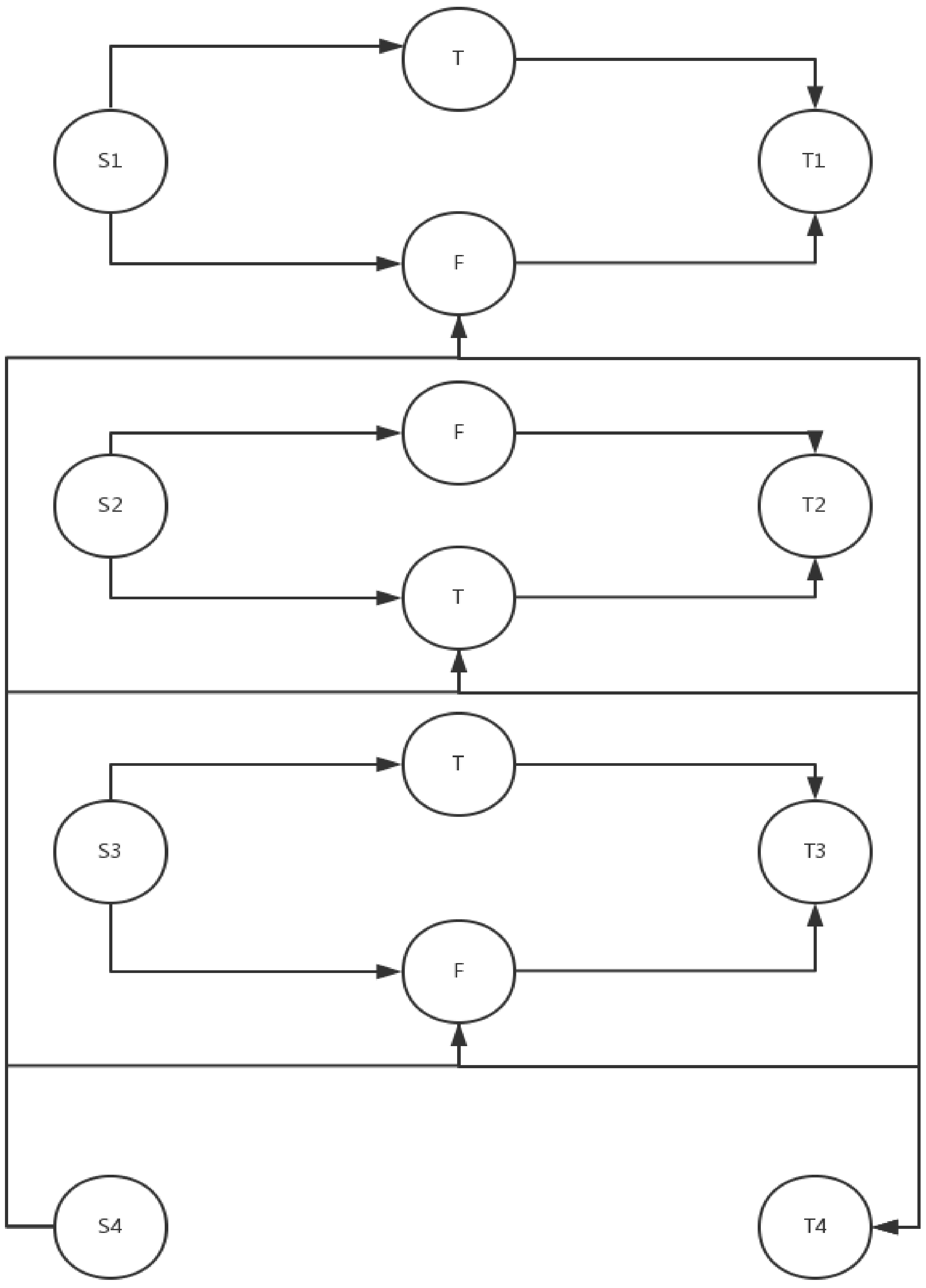
## 4 Problem 5: Directed Disjoint Paths Problem

At first, given an answer of this problem, we can verify the instance in polynomial time to decide whether it is suitable. If we have a graph which has k pairs, then the time complexity to verify is O(2k).
Then we can reduce 3-SAT Problem to this problem, we can prove the inequality:

$$3 - SAT \leq pDirectedDisjointPaths$$

For example:

$$U_1 \cup U_2 \cup U_3$$



We can build more complex 3-SAT clauses, and reduce it into a graph.If m clauses can be true, and the number of variables is n, then there is directed disjoint paths, and the number of pairs would be n+m. If m clauses can not be true, then there is directed disjoint paths too, but the number would no more be n+m.
Because $3 - SAT \in NP - Complete$, Directed Disjoint Path is harder than 3-SAT Problem, and it is a NP Problem, so Directed Disjoint Path is a NP-Complete Problem.

## 5 Problem 6: Longest Common Sub-Sequence Problem

At first, given an answer of this problem, we can verify the instance in polynomial time to decide whether it is suitable, whether it is a common sub-sequence of all sequences. If we have a common sub-sequence, and the number of listing of sequence is n, we can decide the correctness in O(n), ignoring the length of sequence.
Then we can reduce Independent-Set Problem to this problem, we can prove the inequality:

$$IndependtSet \leq pLCSProblem$$

Suppose we have a LCS problem over alphabet $\Sigma = \{0, 1\}$, given a graph $G =< V, E >$, suppose we have n vertexes and m edges, we want to find a maximal independent set of this graph, so we can reduce the problem to a LCS Problem, for each vertex, we construct m+1 0-1 sequence $s_0, ..., s_m$, $s_0 = (01)^n$, for each edge i $< u, v >\in E$, we construct $s_1 = (01)^{u-1}0(01)^{v-u}0(01)^{n-v}$. We will see why doing so soon.
If vertex $j \in IndependentSet$, we will append 01, if not, we append 0, and we construct a (n+k) sequence. We should prove some lemma to complete the problem.
Lemma:If we have k Independent Set at most, then exists (n+k) LCS, and (n+k) at most.
Becase $s_0$'s length is 2n, and 0 was all inserted into T, since we append 01 or 0 only, so if we add another 1, the corresponding edge will no longer have the LCS.
Lemma: If there is (n+k) LCS, then k vertexes in Independent Set at most.
Find such LCS, it must contains n '0', if not, then we can improve the length of LCS by adding '0'. Then we can decide which vertex belong to set by corresponding to $s_0$. If we find ith and i+1th position in T is '01', then remark i and i+1 as a part of set, if ith and (i+1)th position in T is '00', then remark i as a part of set. It will not appear '1', because after reading '01', if it is still '1', we can improve LCS by appending '0'.
Getting remarked sub-sequence, we know the max Independent Set has k vertexes, if two vertexes have been connected, then the corresponding s will no longer have the LCS.
After proof above, we can reduce Independent Set Problem to Longest Common Sub-Sequence Problem, and as is known to us, Independent Set Problem is NP-Complete, LCS Problem is harder than it, and it is NP, so Longest Common Sub-Sequence Problem is a NP-Complete Problem.