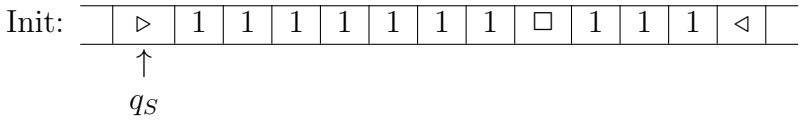
Lab08-Computational Complexity

CS214-Algorithm and Complexity, Xiaofeng Gao, Spring 2019.

∗ If there is any problem, please contact TA Jiahao Fan or TA Mingran Peng.
∗ Name: KylinChen Student ID: 517030910155 Email: k1017856853@icloud.com

1. Design a one-tape TM M that computes the function $f(x, y) = x - y$, where x and y are positive integers ($x > y$). The alphabet is $\{1, 0, \square, \triangleright, \triangleleft\}$, and the inputs are x 1's, $\square$ and y 1's. Below is the initial configuration for input $x = 7$ and $y = 3$. The result $z = f(x, y)$ should also be represented in the form of z 1's on the tape with the pattern of $\triangleright 111 \cdots 111 \triangleleft$.



   (a) Please describe your design and then write the specifications of M in the form like $\langle q_S, \triangleright\rangle \to \langle q_1, \triangleright, R\rangle$. Explain the transition functions in detail.

   (b) Please draw the state transition diagram using Microsoft Visio.

   (c) Show briefly and clearly the whole process from initial to final configurations for input $x = 7$ and $y = 3$.

Solution.

(a)
- For one-tape TM M, the reading head q may write a symbol, move left, move right. The we give our state-transfer prototype as:

$$\langle q_i, s_w\rangle \to \langle q_j, s_k, L/R/S\rangle$$

  which means reading head state transfer: $q_i \to q_j$, tape value replace s_w with s_k, reading head movement is L(Left) / R(Right) / S(Stand Still).
- Define state q_i:
  q_S: start/right move state; q_E: end/return front state; q_H: Halt state(refering to Powpoint, we set q_H at the first letter rather than $\triangleright$); q_1: delete y 1's state; q_2: left move state; q_3: delete x 1's state;
- Given a Turing machine M state-transfer table:

$$\begin{aligned}
&\langle q_s, \triangleright\rangle \to \langle q_s, \triangleright, R\rangle, && \langle q_s, 1\rangle \to \langle q_s, 1, R\rangle\\
&\langle q_s, \square\rangle \to \langle q_s, \square, R\rangle, && \langle q_s, \triangleleft\rangle \to \langle q_1, \square, L\rangle\\
&\langle q_1, 1\rangle \to \langle q_2, \triangleleft, S\rangle, && \langle q_1, \square\rangle \to \langle q_E, \triangleleft, S\rangle\\
&\langle q_2, \triangleleft\rangle \to \langle q_2, \triangleleft, L\rangle, && \langle q_2, 1\rangle \to \langle q_2, 1, L\rangle\\
&\langle q_2, \square\rangle \to \langle q_2, \square, L\rangle, && \langle q_2, \triangleright\rangle \to \langle q_3, \square, R\rangle\\
&\langle q_3, 1\rangle \to \langle q_s, \triangleright, S\rangle, && \langle q_E, \triangleleft\rangle \to \langle q_E, \triangleleft, L\rangle\\
&\langle q_E, 1\rangle \to \langle q_E, 1, L\rangle, && \langle q_E, \triangleright\rangle \to \langle q_H, \triangleright, R\rangle
\end{aligned}$$

(b) We can use *Microsoft Visio* 2013 to draw the state transition diagram below, the *visio* file *TM.vsdx* is attach in .zip file folder. Its export TM.pdf can be as Figure. 1

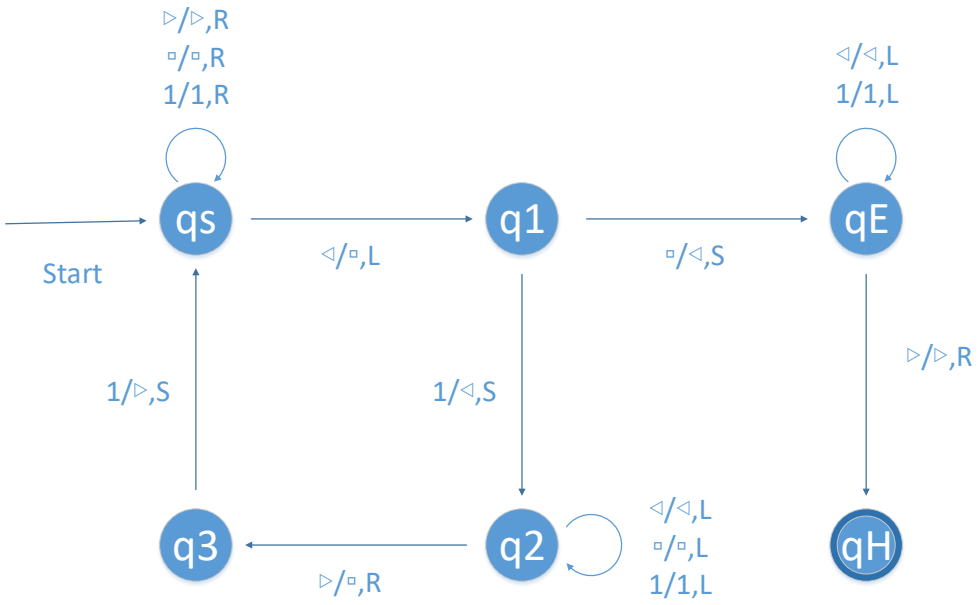


Figure 1: **State Transition Diagram by Microsoft Visio 2013**

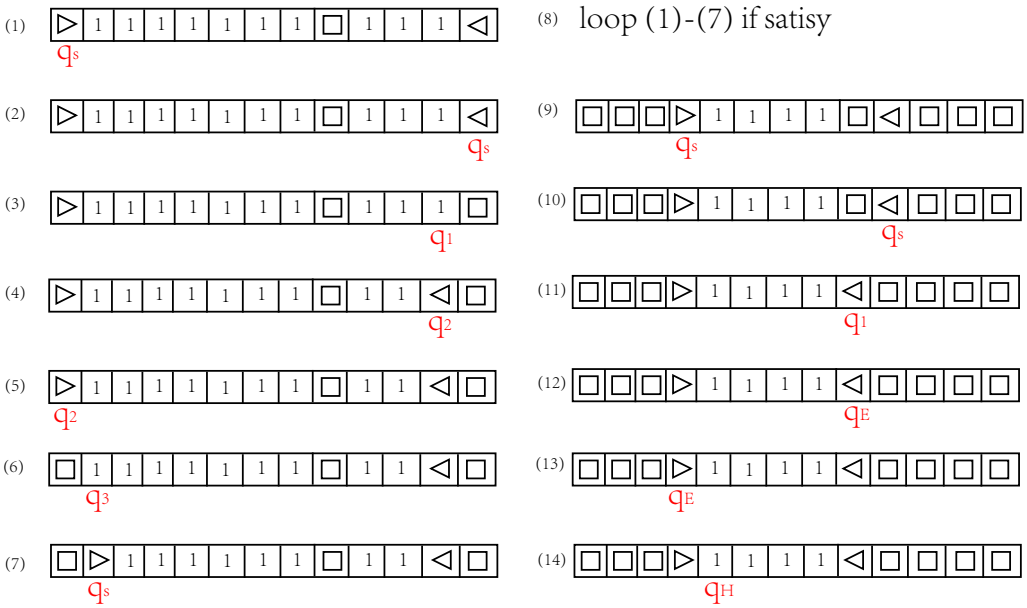


Figure 2: **TM Computation Process**

(c) For the given example, whose initial configuration is set for input x = 7 and y = 3, and put reading head q_S at front $\triangleright$, we can give the TM M compute process as Figure.2
(In the computation process, we leave out some loop to list the key steps instead)

2. What is the "certificate" and "certifier" for the following problems?

(a) *PARTITION*: Given a finite set A and a size $s(a) \in \mathbb{Z}$ for each $a \in A$, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = \sum_{a \in A - A'} s(a)$?

(b) *CLIQUE*: Given a graph $G = (V, E)$ and a positive integer $K \leq |V|$, is there a subset $V' \subseteq V$ with $|V'| \geq K$ such that every two vertices in V' are joined by an edge in E ?

(c) *ZERO-ONE INTEGER PROGRAMMING*: Given an integer $m \times n$ matrix A and an integer m-vector b, is there an integer n-vector x with elements in the set $\{0, 1\}$ such that $Ax \leq b$?

Solution.

(a) **certificate:** A certificate is a subset $A' \in A$, which satisfys the problem requirements, note that such a certificate exists iff problem requirements can be satified.
certifier:

Algorithm 1: *certifier* 1

```
Input: set A; set A';
Output: bool res;
count ← 0;
for item a ∈ A do
    if a ∈ A' then
        count ← count + s(a);
    else
        count ← count − s(a);
return (count == 0);
```

(b) **certificate:** A certificate is a k-node set $N' \in N$, which satisfys the problem requirements, note that such a certificate exists iff all node pair in N' have connected by at least one edge e ($e \in E$).
certifier:

Algorithm 2: *certifier* 2

```
Input: set A; set A';
Output: bool res;
for node i ∈ N' do
    for node j ∈ N'/{j} do
        if Connect(i, j) == False then
            return False;
return True;
```

(c) **certificate:** A certificate is a n-vector x, which satisfys the problem requirements, note that such a certificate exists iff $Ax \leq b$.

certifier:

Algorithm 3: *certifier* 3

```
Input: set A; set A';
Output: bool res;
for i ← 1 to m do
    A[i] ← A's i-th row;
    if A[i] · x > b[i] then
        return False;
return True;
```

3. *SUBSET SUM*: Given a finite set A, a size $s(a) \in \mathbb{Z}$ for each $a \in A$ and an integer B, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) = B$?

   KNAPSACK: Given a finite set A, a size $s(a) \in \mathbb{Z}$ and a value $v(a) \in \mathbb{Z}$ for each $a \in A$ and integers B and K, is there a subset $A' \subseteq A$ such that $\sum_{a \in A'} s(a) \leq B$ and $\sum_{a \in A'} v(a) \geq K$?

   (a) Prove *PARTITION* $\leq_p$ *SUBSET SUM*.

   (b) Prove *SUBSET SUM* $\leq_p$ *KNAPSACK*.

   Proof.

   (a)
   - Instance of Set Partition: Given Set A.
     Instance of Subset Sum: Same Set A, aimed sum B.
   - To Prove:
     $\Rightarrow$ There exist a aimed sum B', which makes Set Partition problem become a Subset Sum problem.
     $\Leftarrow$ Subset Sum problem can be specified into a Set Partition problem.
   - Prove:
     For Given Set A, we can compute $\Theta = \sum_{a=1}^{|A|} s(a)$, then make $B' = \frac{1}{2}\Theta$. A' have the same value with $A - A' \Rightarrow A'$ have the aimed value B'.

   (b)
   - Instance of Subset Sum: Same Set A, aimed sum B_1.
     Instance of Knapsack: Same Set A, size upper bounder B_2, value lower bounder K.
   - To Prove:
     $\Rightarrow$ There exist B_2, K and function $v(a)$, which makes Subset Sum problem become a Knapsack problem.
     $\Leftarrow$ Knapsack problem can be specified into a Subset Sum problem.
   - Prove:
     For Given Set A, we can set $B_2 = K = B_1$, function $v = s \Leftarrow \sum_{a \in A'} s(a) \leq B_2$ and $\sum_{a \in A'} s(a) \geq B_2 \Rightarrow \sum_{a \in A'} s(a) = B_2$, this is subset sum problem.

4. *3-SAT*: Given a set U of variables, a collection C of clauses over U such that each clause $c \in C$ has $|c| = 3$, is there a satisfying truth assignment for C?

   Prove *3-SAT* $\leq_p$ *CLIQUE*.

Proof.

- **Configuration.** Since we want to prove a reduction between logic problem($3-SAT$) and graph problem(k-CLIQUE), we must establish a graph model G between them, which satisfys: it have nodes for all (clause, literal) pairs and edges between all non-contradictory nodes in different clauses.
- **Claim.** $3-SAT$ satisfiable iff G has a k-CLIQUE.
- **Example.** We give a synergy paradigm below:

$$w = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3)$$

For general, k represent number of clauses, in this case, $k = 3$, We label them with c_1, c_2, c_3 in this example.
We can draw the graph model in Figure.3

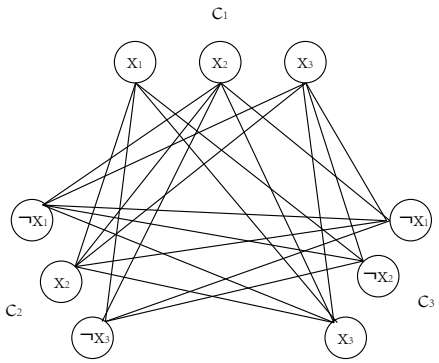


Figure 3: **Graph Model G for The Example**

We can pick a 3-KLIQUE($k = 3$) to see its correctness. (In Figure.4, red-line CLIQUE). Once this CLIQUE exists, we can set all nodes in this CLIQUE $True$, which means:

$$x_1 = False, x_2 = False, x_3 = True$$

which satisfys 3-SAT.

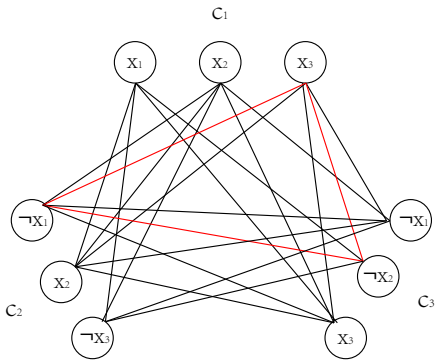


Figure 4: **A Certificate Clique**

- **To Prove.** $3-SAT$ satisfiable iff G has a k-CLIQUE.
- **Prove.**

  $\Rightarrow$: Assume the formula is satisfiable which means satisfying assignment gives one literal in each clause, all with non-contradictory assignments. Therefore, it yields a k-clique, which makes one $True$ literal in every clause satisfied.

  $\Leftarrow$: Assume there is a k-clique in Graph model G, which yields one node per clause, none contradictory. Once at least one k-CLIQUE exist, it yeilds more than one consistent assignments satisfying all clauses of synergy paradigm w and make 3-SAT established.

5. Algorithm class is a democratic class. Denote class as a finite set S containing every students. Now students decided to raise a student union $S' \subseteq S$ with $|S'| \leq K$.

   As for the members of the union, there are many different opinions. An opinion is a set $S_o \subseteq S$. Note that number of opinions has nothing to do with number of students.

   The question is whether there exists such student union $S' \subseteq S$ with $|S'| \leq K$, that S' contains at least one element from each opinion. We call this problem *ELECTION* problem, prove that it is NP-complete.

   Proof.

   - **Configuration**:
     - S contains n elements(students).
     - There are m kinds of opinion subsets S_0, marked with $S_{0(1)}$, $S_{0(2)}$, $S_{0(3)}$, $\cdots S_{0(m)}$.
   - **ELECTION $\in$ NP**:

     To Prove ELECTION $\in$ NP, we need to claim there is a poly-time certifier $C(S,t)$. For any subset $S' \in S$, $C(S',t) = True$ iff S' contains at least one elements of every opinion subset S_0. Vise verse.

     $\Rightarrow$ Once we have a subsets $S' \in S$, we can use brute-force to check whether S' contains at least one elements in $S_{0(t)}$, it takes $O(n^2)$ for each S_0 check, it means for certificier $C(S,t)$, $t = O(mn^2)$, which confirm to poly-time constarin. Therefore, ELECTION $\in$ NP problem.
   - **Set-Covering $\leq_p$ ELECTION**:

     Set-Covering Problem belongs to Karp's 21 NP-complete problems, it means:

     $$Set-Covering \in NPC$$

     According to proven Claim:

     $$X \in NPC, \;\; Y \in NP, \;\; X \leq_p Y \Rightarrow Y \in NPC.$$

     $ELECTION \in NP$ is given above. So we need to prove Set-Covering $\leq_p$ ELECTION:

     $\Rightarrow$: For every element s_w in S, it may be contained by several opinion subset:

     $$s_w \in S_{0(t)}, \forall w \in \{1, \cdots n\} \; (\; for \; some \; t \in \{1, \cdots m\}, w \in \{1, \cdots n\} \;)$$

     It can be marked:

     $$C_w = \{S_{0(t)} \cdots\} \;\; (for \; some \; t \; \in \; \{1, \cdots m\})$$

C_w means the set of set which contains s_w, and C_w can be $\emptyset$ iff s_w isn't contained in any opinion subset $S_{0(t)}(\forall t \in \{1, \cdots m\})$.
Then, we can get the universe set:

$$U = \{S_{0(1)},\ S_{0(2)},\ S_{0(3)},\ \cdots\ S_{0(m)}\}$$

And the collection of sets

$$C_u = \{C_1,\ C_2,\ C_3,\ C_4, \cdots\ C_n\}$$

Then, ELECTION problem is converted into Set-Covering problem, aimed to cover the universe set U with setset union of C_u. And, origin ELECTION problem exist satisfied answer iff Set-Covering exist a collection of $\leq K$ of these sets whose union is equal to C_u.
$\Leftarrow$(correctness): Once these sets' union is equal to C_u, it means all the opinion subset have at least one elements in the colection union. Moreover, the colection $\leq K$ means we selection no more than K elements, which confirms to origin ELECTION problem.

- **Example**:
  We give a example like Figure.5. There are four opinions and six students, and An edge between opinion and student means this opinion construct an union which contains this student, for instance, in this case(Figure.5), opinion1 construct an union containing student1, student2 and student3.
  Then, we can number these opinion with 1 to 4, which are combined to universe set U. (In my proof above, opinion1(include other opinions) is a set like {1,2,3}, and C_1 contains this set as an element, but we simplify it in our example.)
  Moreover, we can construct a set C_i for each student i, once this student belongs to this opinion sebset, for example, $C_1 = \{1, 2, 3\}$ since these three opinion want to invite student1 to their own union. Finally, we can convert our problem to: Can we find no more than k subsets in G_u, which covers U. THis is obviously a NP-complete problem, so is the origin problem.

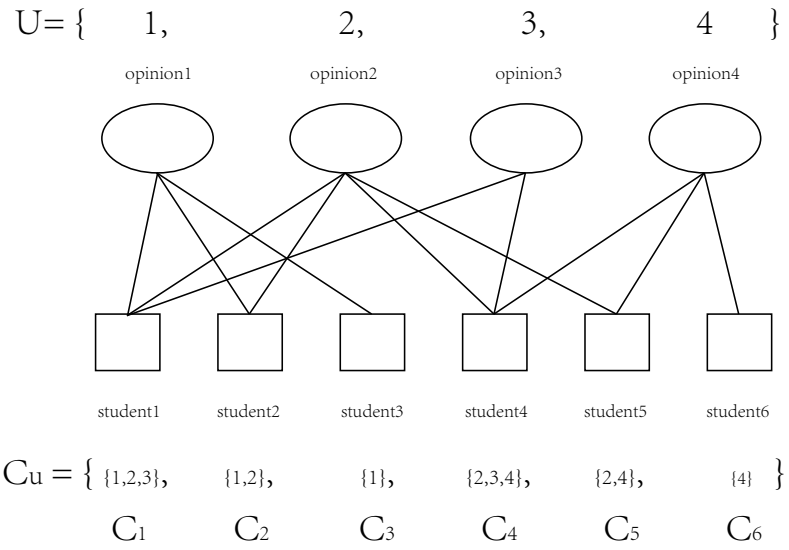


Figure 5: **How to build U and C_u**