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.
∗ If there is any problem, please contact TA Mingran Peng.
∗ Name:Tianyao Shi Student ID:517021910623 Email: sthowling@sjtu.edu.cn

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 $\triangleright 111 \cdots 111 \triangleleft$.

Init:		$\triangleright$	1	1	1	1	1	1	1	$\square$	1	1	1	$\triangleleft$	
		↑													
		q_S													

   (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) As is shown in the figure, the cardinality of x and y is represented using number of 1's on the tape. A natural thought for computing $x - y$ would be that first move right to spot the start of y, for every 1 in y, remve it first, then move left until we spot a 1 in x, wipe it out. Afterwards, move right to find the new start of $y - 1$ 1's. Repeat this process until all y 1's is removed and we can only find a $triangleleft$. Remove it, rewind to the end of the result and add a $\triangleright$ after it. According to this design, we can give a set of transition functions as followed:

$\langle q_S, \triangleright\rangle \to \langle q_1, \triangleright, R\rangle$	starting state, now prepared to move until the start of y spotted
$\langle q_1, 1\rangle \to \langle q_1, 1, R\rangle$	skip x 1's
$\langle q_1, \square\rangle \to \langle q_2, \square, R\rangle$	x 1's skipped, now ready to remove y 1's
$\langle q_2, 1\rangle \to \langle q_2, \square, L\rangle$	remove y 1's and rewind
$\langle q_2, \square\rangle \to \langle q_2, \square, L\rangle$	rewind, skipping $\square$
$\langle q_2, 1\rangle \to \langle q_3, \square, R\rangle$	remove the current last 1 of x, then to find the 'new' start of y
$\langle q_3, \square\rangle \to \langle q_3, \square, R\rangle$	move forward, skipping $\square$
$\langle q_3, 1\rangle \to \langle q_2, \square, L\rangle$	'new' start spotted, loop
$\langle q_2, \triangleleft\rangle \to \langle q_4, \square, L\rangle$	all y 1's removed, preparing to halt
$\langle q_4, \square\rangle \to \langle q_4, \square, L\rangle$	skipping $\square$
$\langle q_4, 1\rangle \to \langle q_5, 1, R\rangle$	end of result spotted, next to halt
$\langle q_5, \square\rangle \to \langle q_H, \triangleleft, R\rangle$	halt the machine

   (b) The state transition diagram is shown in Figure 1.

   (c) The whole process is shown in Figure 2. □

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 ?

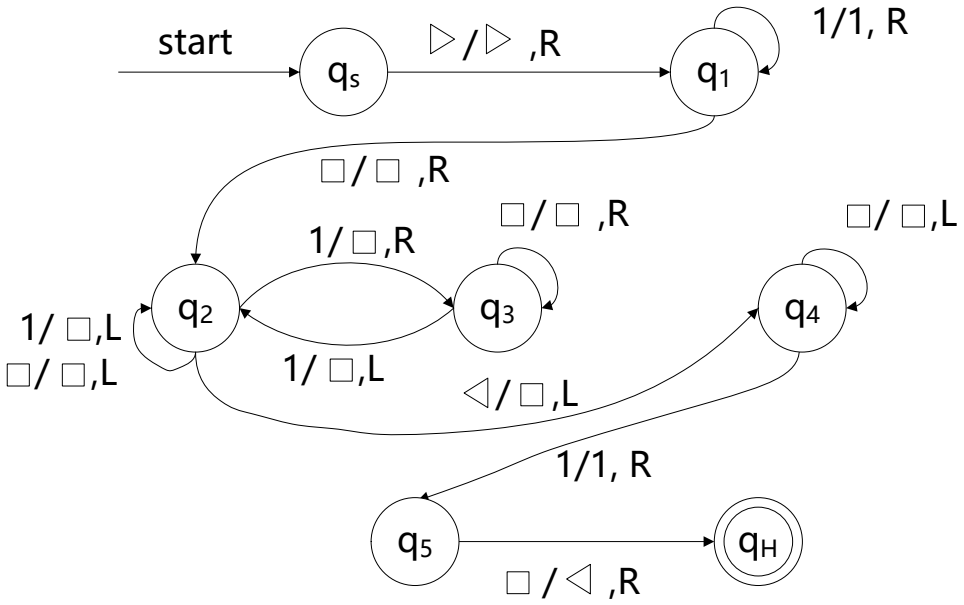

Figure 1: State transition diagram of this one-tape turing machine to perform function $f(x, y) = x - y$.

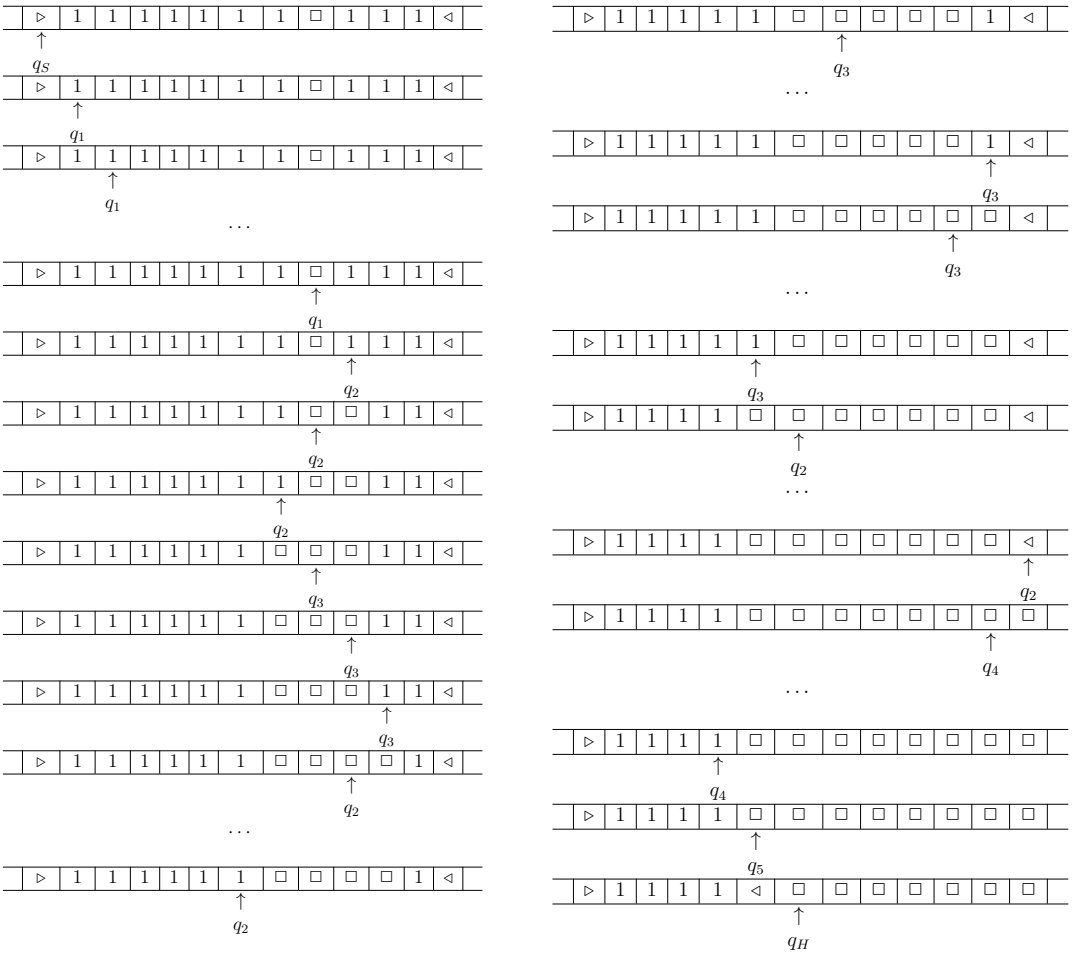

Figure 2: The whole process of running TM with input $x = 7, y = 3$

(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 of *PARTITION* problem is a subset $A' \subseteq A$ given in explicit form, such that we can add up $s(a)$ for every $a \in A'$ and $A - A'$.

Certifier A certifier of *PARTITION* problem is the following algorithm:

Algorithm 1: *PARTITION* Certifier

input : A finite set A, an array of size $s(a)$ for each $a \in A$, an explicit subset $A' \subseteq A$

```
1  sum1 ← 0, sum2 ← 0;
2  foreach a ∈ A' do
3      if a ∉ A then
4          return false;
5      sum1 ← sum1 + s(a);
6  foreach a ∈ A − A' do
7      sum2 ← sum2 + s(a);
8  if sum1 = sum2 then
9      return true;
10 else
11     return false;
```

(b)**Certificate** A certificate of *CLIQUE* problem is a subset $V' \subseteq V$ given in explicit form, such that we can check if everytwo vertices in V' are joined by an edge in E.

Certifier A certifier of *CLIQUE* problem is the following algorithm:

Algorithm 2: *CLIQUE* Certifier

input : A graph $G = (V, E)$,an integer $K < |V|$, an explicit subset $V' \subseteq V$

```
1  subsetsize ← 0;
2  foreach v ∈ V' do
3      if v ∉ V then
4          return false;
5      subsetsize ← subsetsize + 1;
6  if subsetsize < K then
7      return false;
8  Sort vertices in V' according to some criteria;
9  for i ← 1 to subsetsize do
10     for j ← i to subsetsize do
11         if (v_i, v_j) ∉ E then ; // v_i, v_j are the i-th and j-th vertices in
               the sorted V'
12
13         return false;
14 return true;
```

(c)**Certificate** A certificate of *ZERO-ONE INTEGER PROGRAMMING* problem is an integer $n-$vector x with elements in the set 0,1.

Certifier A certifier of *ZERO-ONE INTEGER PROGRAMMING* problem is the following algorithm:

Algorithm 3: *ZERO-ONE INTEGER PROGRAMMING* Certifier

input : An integer $m \times n$ matrix A, an integer m-vector b, an integer $n-$vector x with elements in the set 0,1

```
1  for i ← 1 to n do
2      if x[i] ∉ {0, 1} then
3          return false;
4  for i ← 1 to m do
5      result[i] ← 0;
6  for i ← 1 to m do
7      for j ← 1 to n do
8          result[i] ← result[i] + A[i][j] × x[j];
9      if result[i] > b[i] then
10         return false;
11 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)Notice that the difference between *PARTITION* and *SUBSET SUM* is that instance of *SUBSET SUM* has an additinal interger B which is irrelavent to A. Let us define that $\sum\limits_{a \in A} s(a) = C, \sum\limits_{a \in A'} s(a) = D$. We then prove that A' is a partition of A iff A' satisfies *SUBSET SUM* with $B = C - D$.

"⇒": If A' is a partition of A, $D = \sum\limits_{a \in A - A'} s(a) = \sum\limits_{a \in A} s(a) - \sum\limits_{a \in A'} s(a) = C - D = B$.

"⇐": If A' satisfies *SUBSET SUM* with $B = C - D$, $D = B = C - D = \sum\limits_{a \in A} s(a) - \sum\limits_{a \in A'} s(a) = \sum\limits_{a \in A - A'} s(a)$. Then A' is a partition of A. □

(b)Let instance of *KNAPSACK* be that $v(a) = s(a)$, $B = K$. Then for any subset $A' \subseteq A$, it is a yes instance in *SUBSET SUM* iff it is a yes instance in this according *KNAPSACK*.

"⇒": If A' is a yes instance in *SUBSET SUM*, we have $\sum\limits_{a \in A'} s(a) = B = K$. Then we have $\sum\limits_{a \in A'} s(a) \leq B$ and $\sum\limits_{a \in A'} s(a) = \sum\limits_{a \in A'} v(a) \geq K$. Thus A' is a yes instance in *KNAPSACK*.

"⇐": If A' is a yes instance in *KNAPSACK*, we have $\sum_{a\in A'} s(a) \leq B$, $\sum_{a\in A'} s(a) = \sum_{a\in A'} v(a) \geq K = B$, which leads to $\sum_{a\in A'} s(a) = B$. Thus A' is a yes instance in *SUBSET SUM*. □

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. For every instance Φ with n variables and k clauses in *3-SAT*, we construct a graph G accordingly and set the integer $K = k$, such that we derive an instance of *CLIQUE*. For each literal in each clause, construct a vertex, and each clause from a 3-vertex gadget. Then connect each vertex to all vertices in other clauses except it negation(s), and the construction is done.

   We then prove that answer of the instance in *3-SAT* is yes iff the contructed instance in *CLIQUE* is yes.

   "⇒": If Φ is satisfiable, there exists a set of truth values of variables $x_1, x_2, \cdots, x_n$ s.t. every clause in Φ is true, i.e. at least one literal in each clause is set to true. Select one true literal in each clause, to form a subset V' of V in the graph, then $|V'| = k \geq K$. And according to the construction process, vertices between different clauses are disconnected only when they are negations to each other, which cannot happen since one literal and its negation cannot be true simultaneously. Thus every two vertices in V' is joined by an edge in E, which leads the answer of *CLIQUE* to be yes.

   "⇐": If there exists a subset $V' \subseteq V$ with $|V'| \geq K = k$, such that every two vertices in V' is joined by an edge in E, then we actually have $|V'| = k$. If $|V'| > k$, then there is at least a clause that has more than one vertex in V'. But it contradicts that there does not exist an edge joining two vertices in the same clause. Therefore $|V'| = k$, and each clause contain exactly 1 vertex. We then set the according literals to true, and Φ is then satisfiable.

   Figure 3 is an illustration of the construction.

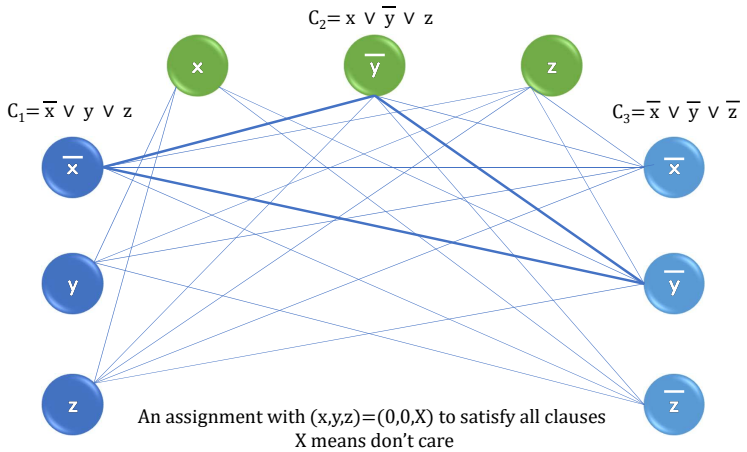

Figure 3: An illustration of the instance constructing process, from *3-SAT* to *CLIQUE*.

□

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. *ELECTION* is actually a famous type of problem called *HITTING-SET*. We give the formalization of this problem: given a finite collection of sets $\{S_1, S_2, \cdots, S_n\}$ and a budget K, where $S_o (1 \leq o \leq n)$ is a subset of the universal set S , whether there exists a set S' s.t. $\forall o, S_o \cap S' \neq \emptyset$ and $|S'| < K$.

Claim. *VERTEX-COVER* $\leq_P$ *HITTING-SET*.

Given an undirected graph $G = (V, E)$ and an integer K as an instance of *VERTEX-COVER*, let us construct an instance of *HITTING-SET*. Let vertice $v \in V$ become set elements in S, and edge $e_o = (v_i, v_j) \in E$ become a set $S_o = \{v_i, v_j\}$. The integer K becomes the budget of required set. All edges, transformed into sets, then become the collection in *HITTING-SET*. And all vertices become the elements of universal set S.

Then we prove that the answer of the instance in *VERTEX-COVER* is yes iff the contructed instance in *HITTING-SET* is yes.

"⇒": If there exists a vertex cover V' with $|V'| < K$ in graph G, then we can pick every vertex in V' to form a set S'. Actually $V' = S'$, and $|S'| = |V'| < k$. Since V' is a vertex cover, then for every edge $e_o = (v_i, v_j) \in E$, $v_i \in V'$ or $v_j \in V'$. That means for every set $S_o = \{v_i, v_j\}$, $S' \cap S_o \neq \emptyset$. Thus there exists a hitting set S' for this instance.

"⇐": If there exists a hitting set S' for the given collection $\{S_1, S_2, \cdots, S_n\}$ with respect to the universal set S and a budget K, then for every set $S_o = \{v_i, v_j\}$ in the collection, $S' \cap S_o \neq \emptyset$. That is to say, $v_i \in S'$ or $v_j \in S'$. Then for the original edge $e_o = (v_i, v_j) \in E$, e_o is covered by $V' = S'$. Therefore V' is a vertex cover, and $|V'| = |S'| < K$.

Figure 4 is a illustration of *VERTEX-COVER* and *HITTING-SET* with an example.

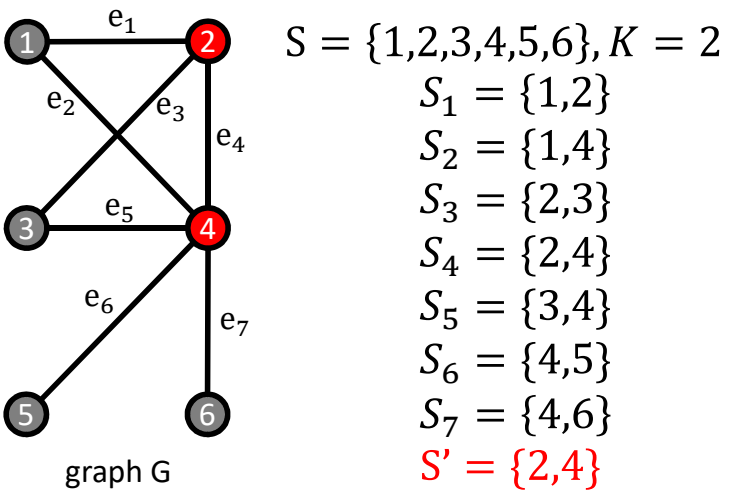

Figure 4: An illustration of the instance constructing process mentioned above.

Now that we proved *VERTEX-COVER* $\leq_P$ *HITTING-SET*, and we know that *VERTEX-COVER* is NP-complete, according to the definition of NP-completeness and transitivity of $\leq_P$, *HITTING-SET*, i.e. *ELECTION* is also NP-complete. □

Remark: You need to include your .pdf and .tex files in your uploaded .zip file.