Lab09-Turing Machine and Reduction

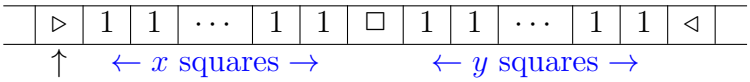
Algorithm: Analysis and Theory (X033533), Xiaofeng Gao, Spring 2018.

∗ If there is any problem, please contact TA Mingding Liao.

∗

1. Design an one-tape TM M that compute the function $f(x) = x \mod y$, says, the reminder of dividing x by y, where x and y are positive integer. The alphabet is $\{1, 0, \square, \triangleright, \triangleleft\}$, where the input on the tape are x "1"'s, $\square$ and y "1"'s. Below is the initial configurations for input x and y. The result is the number of "1"'s on the tape with pattern of $\triangleright 111 \cdots 111 \triangleleft$. First describe your design and then write the specifications of M in the form like $\langle q_S, \triangleright \rangle \rightarrow \langle q_1, \triangleright, R \rangle$ and then explain the transition functions in detail. Moreover, you should draw the state transition diagram.

Initial Configurations



Solution:

f(x)= x mod y

The input to our tape will be u(x) and the output will be u(x mod y). Our strategy is to mark all the 1s as we scan to the right using our states to remember how many 1s we have seen mod y. Then we will remove the marks from that many 1s and scan back to the left over the remaining marked 1s. Then we will delete all the marked ones and halt with x mod y on the tape. We will define the TM as follows:

$Q = q_0, q_1, ..., q_f$, where q_0 is the start state and q_f the only final state,

$\sum = 1$

$\Gamma = 1, i, \square$ where the input on the tape are x "1"$'s$,$\square$ and y "1"$'s$

Let's take a finite set of y to be 5.

so x mod y $\rightarrow$ x mod 5

σ as defined follow:

$\sigma(q_0, 1) = (q_1, i, R)$, Mark a one and move to the state of 1 mod 5

$\sigma(q_1, 1) = (q_2, i, R)$, Mark a one and move to the state of 2 mod 5

$\sigma(q_2, 1) = (q_3, i, R)$, Mark a one and move to the state of 3 mod 5

$\sigma(q_3, 1) = (q_4, i, R)$, Mark a one and move to the state of 4 mod 5

$\sigma(q_4, 1) = (q_0, i, R)$, Mark a one and move to the state of 0 mod 5, repeating until the end

$\sigma(q_4, \square) = (q_4, \square, L)$,

$\sigma(q_3, \square) = (q_3, \square, L)$

$\sigma(q_2, \square) = (q_2, \square, L)$,

$\sigma(q_1, \square) = (q_1, \square, L)$

$\sigma(q_0, \square) = (q_0, \square, L)$ whichever state we ended in we remember

$\sigma(q_4, i) = (q_3, 1, L)$

$\sigma(q_3, i) = (q_2, 1, L)$,

$\sigma(q_2, i) = (q_1, 1, L)$,

$\sigma(q_1, i) = (q_0, 1, L)$, and unmark that many 1s (this will be our answer)

$\sigma(q_4, i) = (q_3, i, L)$ We then scan back at the beginning

$\sigma(q_0, \square) = (q_5, \square, L)$,

$\sigma(q_5, i) = (q_5, \square, L)$, and remove the marked 1s

$\sigma(q_5, \square) = (q_5, 1, L)$, until we find an unmarked one

$\sigma(q_0, \square) = (q_f, \square, L)$, and then halt

Another Explanation:
the unary representation of x mod y by erasing some of the 1s from a's unary representation, and all of the 1s from b's unary representation. Observe that x mod y = (x - y) mod y, at least when x >= y; when x < y, then x mod y = b. This observation suggests that we can erase by1s from a's unary representation until we have fewer than y 1s remaining, at which point we erase the 1s from y's representation and halt-accept.
Steps to Follow:
-move right until you find a blank.
-move one step to the left.
-you are now looking at the last 1 in b's representation.
-mark this as Y and move left until you find 0.
-move left until you find a 1 or blank.
-you are now looking at the last 1 in x's representation, or blank.
-if 1, mark this as X and move right until you find Y.
-if blank, x < y; change all Xs to 1s and all 0s, 1s and Y to blanks. halt-accept.
-move one step to the left.
-you are now looking at the last 1 in b's representation, or 0. -if 1, continue as above.
-if 0, b < a; change all Xs to 0s, all Ys to 1s, and restart from the beginning

Reference:
https://stackoverflow.com/questions/46409398/a-mod-b-function-turing-machine
http://www.public.asu.edu/~ccolbou/src/355hw5s12sol.pdf

2. $\mathbb{A}$ and $\mathbb{B}$ are two domains other than $\mathbb{N}$. Assume there exist intuitively computable functions $\alpha : \mathbb{A} \to \mathbb{N}$ and $\alpha^{-1} : \mathbb{N} \to \mathbb{A}$ as encoding and decoding functions from $\mathbb{A}$ to $\mathbb{N}$ respectively (we have $\beta : \mathbb{B} \to \mathbb{N}$ and $\beta^{-1} : \mathbb{N} \to \mathbb{B}$ similarly). Then answer the following questions.

   (a) To prove that $f : \mathbb{A} \to \mathbb{B}$ is computable, we need to construct an $f^* : \mathbb{N} \to \mathbb{N}$ and analyze its computability. How to calculate f^*?

   (b) Reversely, for a computable function $g : \mathbb{N} \to \mathbb{N}$, we can obtain an intuitively computable function $g' : \mathbb{A} \to \mathbb{B}$ from g. What is g' in notation of g, α and β?

   (c) Find a new coding function γ (with the help of α and β) to deal with computability on domain $\mathbb{A} \times \mathbb{B}$. (Hint: Consider the "zig-zag" mapping in Figure 1.)

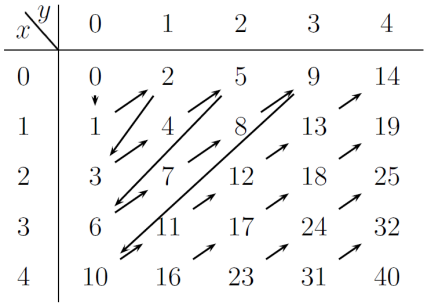


图 1: Zig Zag Mapping

Solution:

Suppose D is an object domain. A coding of D is an explicit and effective injection $\alpha : D \to \mathbb{N}$.
We actually say that an object $d \in D$ is coded by the natural number $\alpha(d)$
A function $f : D \to D$ extends to a numeric function $f^* : \mathbb{N} \to \mathbb{N}$.
We say that f is computable if f^* is computable.
$f^* = \alpha \circ f \circ \alpha^{-1}$
Hence for a computable function $g : \mathbb{N} \to \mathbb{N}$ A function $g' : \mathbb{A} \to \mathbb{B}$ extends to a numeric function $g' : \mathbb{N} \to \mathbb{N}$.
We say that g' is computable in notation if g is computable.

$f^* = \beta \circ f \circ \alpha^{-1}$
For a coding function γ to deal with the computability on domain $\mathbb{A}x\mathbb{B}$
We can write a function containing all the values from the figure 1. for the computation of $\mathbb{A}$ and $\mathbb{B}$.
function Compute($\mathbb{A}$, $\mathbb{B}$):
Contains all the rows and columns from x and y from the fig.
Reference: From Lecture Slides and Reference notes

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

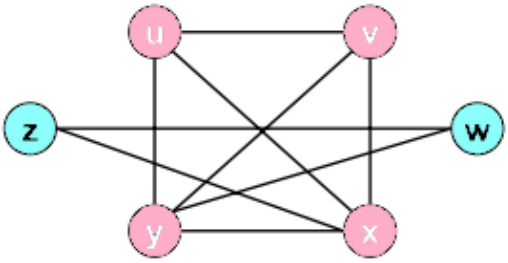
   (a) *CLIQUE*: Given a graph $G = (V, E)$ and a positive integer k, is there a clique (subsets of vertices which are all adjacent to each other) whose size is no less than k?

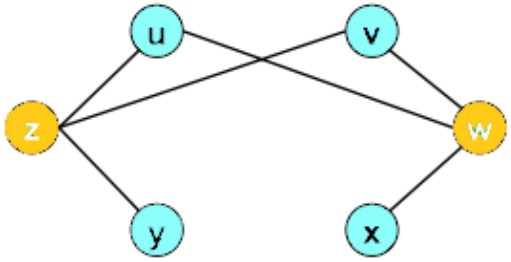
   (b) *SET PACKING*: Given a finite set U, a positive integer k and several subsets $U_1, U_2, \ldots, U_m$ of U, is there k or more subsets which are disjoint with each other?

   (c) *STEINER TREE IN GRAPHS*: Given a graph $G = (V, E)$, a weight $w(e) \in Z_0^+$ for each $e \in E$, a subset $R \subset V$, and a positive integer bound B, is there a subtree of G that includes all the vertices of R and such that the sum of the weights of the edges in the subtree is no more than B.

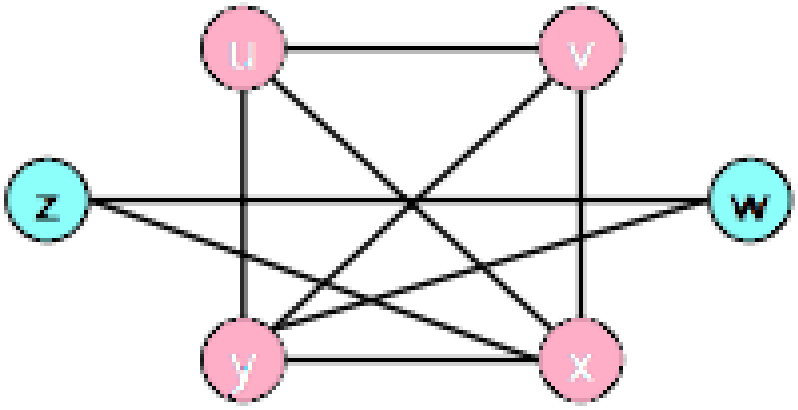
   Solution:
   (i): Certificate: A subset of vertices which has all the adjacent to each other hence making it a clique to itself. In case of Certifier, We check all the pairs in the subset to see either they are adjacent or not, if in case they are not adjacent we can simply return the results through boolean.
   (ii): Certificate: A set S=U_i... which satisfies the condition U_i , $U_j \in$ S.
   if and only if $i \neq j$ then we can say that U_i INTERSECT U_j is equal to $\emptyset$ and $|S| >= k$
   Hence for Certifier: We will check all the pairs in S to see if the U_i INTERSECT U_j is equal to empty set and we can return the results by simply boolean expressions.
   (iii): Certificate: We will see a subtree of G will includes all the vertices of R and the the weights of the edges in the sub-tree, the weights of these edges there sum will not be greater than that of B. In case of the Certifier: We will check if the subtree of G includes all the verticies of R by return the results through boolean expressions. Then we can see if the sum of these weights in the edges of the subtree is not greater than that of B.
   https://www.andrew.cmu.edu/user/avigad/Teaching/candi_notes.pdf
   https://www.andrew.cmu.edu/user/avigad/Teaching/

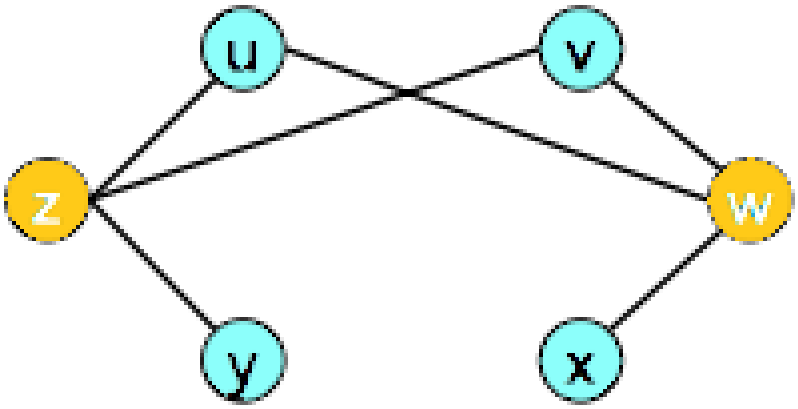
4. Prove VERTEX COVER $\equiv_p$ CLIQUE
   Solution: We will proof this with one of the examples we find online.
   Given an undirected graph G = (V, E), its complement is $G' = (V, E')$, where $E' =$ (v, w) : (v, w) $\notin$ E.
   G has a clique of size k if and only if G' has a vertex cover of size $|V| - k$.
   Proof:





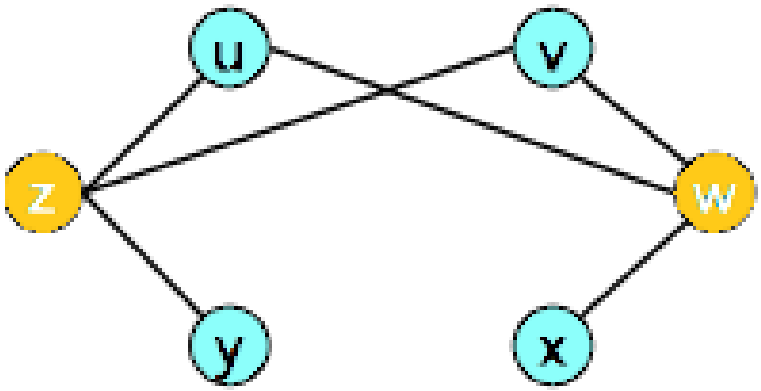
G Clique $= u, v, x, y$ Vertex cover $= w, z$
Suppose G has a clique S with $|S| = k$.
Consider $|S^{'}| = V - S$
$|S^{'}| = |V|$ - k.
To show $|S^{'}|$ is a cover, consider any edge (v, w) $\in |E^{'}|$
- Then edge (v, w) $\notin |E^{'}|$
- Atleast one of v or w is not in S since S forms a clique
- At least one of v or w is in $S^{'}$
- hence (v,w) is covered by $S^{'}$

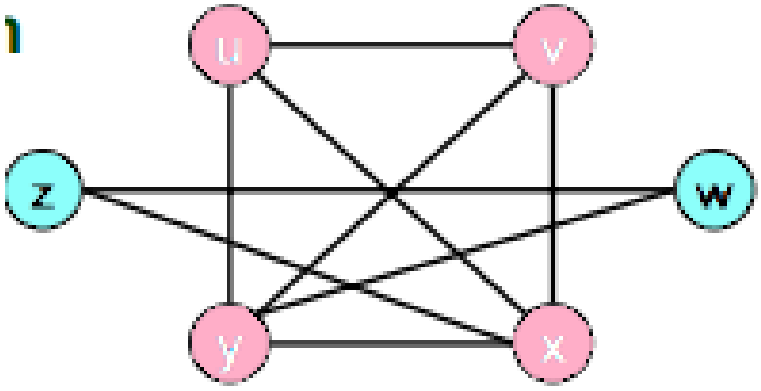




Suppose $G^{'}$ has a cover $S^{'}$ with $|S^{'}| = |V| - k$. Consider $S = V - S^{'}$.

Clearly $|S| = k$.
To show S is a clique, consider some edge (v, w) $\notin E'$
-if (v, w) $\notin E'$ then either v $\in S'$,w $\in S'$, or both
-By contrapositive, if v $\notin S'$ and w v $\notin S'$,
- Then (v, w) v $\in E'$
-Thus S is a clique in G





Reference:
http://www.cs.princeton.edu/~wayne/cs423/lectures/reductions-poly-4up.pdf
http://theory.stanford.edu/~trevisan/cs154-12/np-reductions.pdf