Homework 1 of Lexture Computational Complexity
February 22, 2021
Tanglin

Problem 1. Construct a single-tape Turing Machine, which can find out which number is greater than the other for two n-bits natural numbers x, y in time $O(n^2)$.

Solution:

By the definition of Turing Machine, let alphabet symbol set

$$A = \{0, 1, S, M, \Lambda, X, \text{Less}, \text{Greater}, \text{Equal}\}$$

and status set

$$Q = \{q_0, q_1, r_s, r_0, r_1, r_{00}, r_{11}, f_e, f_{ee}, f\}$$

where q_0 is the initial status, f the final status.

First, let me show how we should set the tape in order to calculate this problem.

$\cdots$	Λ	Λ	Λ	S	b	b	$\cdots$	b	M	b	b	$\cdots$	b	Λ	Λ	$\cdots$

Figure 1: The tape with initial cell S, where b is 0 or 1.

The second cell to the left of S is result bit, and cell S is the initial cell.

It is necessary to explain elements of the alphabet set:

0	—	the binary number 0.
1	—	the binary number 1.
Λ	—	the empty symbol.
S	—	the initial position.
M	—	the separate symbol which distinguishes two binary numbers.
X	—	the symbol denotes that the bit was extracted.
Greater	—	result symbol denotes the first number is greater.
Less	—	result symbol denotes the first number is less.
Equal	—	result symbol denotes the two numbers are equal.

The statuses are difficult to explain in detail, I will show the transposition table, and the transposition graph to help us understand it. I can only little explain some of them, q_0 is the initial status, r_s means the head prepare to get the highest bit of a number(the first or the second), r_0 or r_1 means that the head get the bit 0 or 1 and then it will set the bit to X and skip all the cell until meets Λ, then it changes to status r_{00} or r_{11}, then it go to the left cell and set the bit if the cell with Λ, or it compares the bit with information in head, if the cell is Λ, it denotes that this head got bit from the first number, then the status changes to q_1 which means the head will go and get the corresponding bit of the second number, or a new loop will start. The status f_e and f_{ee} used to deal with the case that two numbers are equal.

This is the transposition table, in other words, the function

$$\delta : A \times Q \to A \times Q \times \{-1, 0, 1\}$$

I will give the full map:

1. Start from the initial cell with initial status q_0.

$$(S, q_0) \mapsto (S, r_s, 1)$$

2. Get the last unextracted bit of the first number and set it to X, then send it to the result bit.

$$\begin{array}{lll} (0, r_s) \mapsto (X, r_0, -1) & (1, r_s) \mapsto (X, r_1, -1) & (X, r_s) \mapsto (X, r_s, 1) \\ (S, r_0) \mapsto (S, r_0, -1) & (X, r_0) \mapsto (X, r_0, -1) & \\ (S, r_1) \mapsto (S, r_1, -1) & (X, r_1) \mapsto (X, r_1, -1) & \\ (\Lambda, r_0) \mapsto (\Lambda, r_{00}, -1) & (\Lambda, r_1) \mapsto (\Lambda, r_{11}, -1) & \\ (\Lambda, r_{00}) \mapsto (0, q_1, 1) & (\Lambda, r_{11}) \mapsto (1, q_1, 1) & \end{array}$$

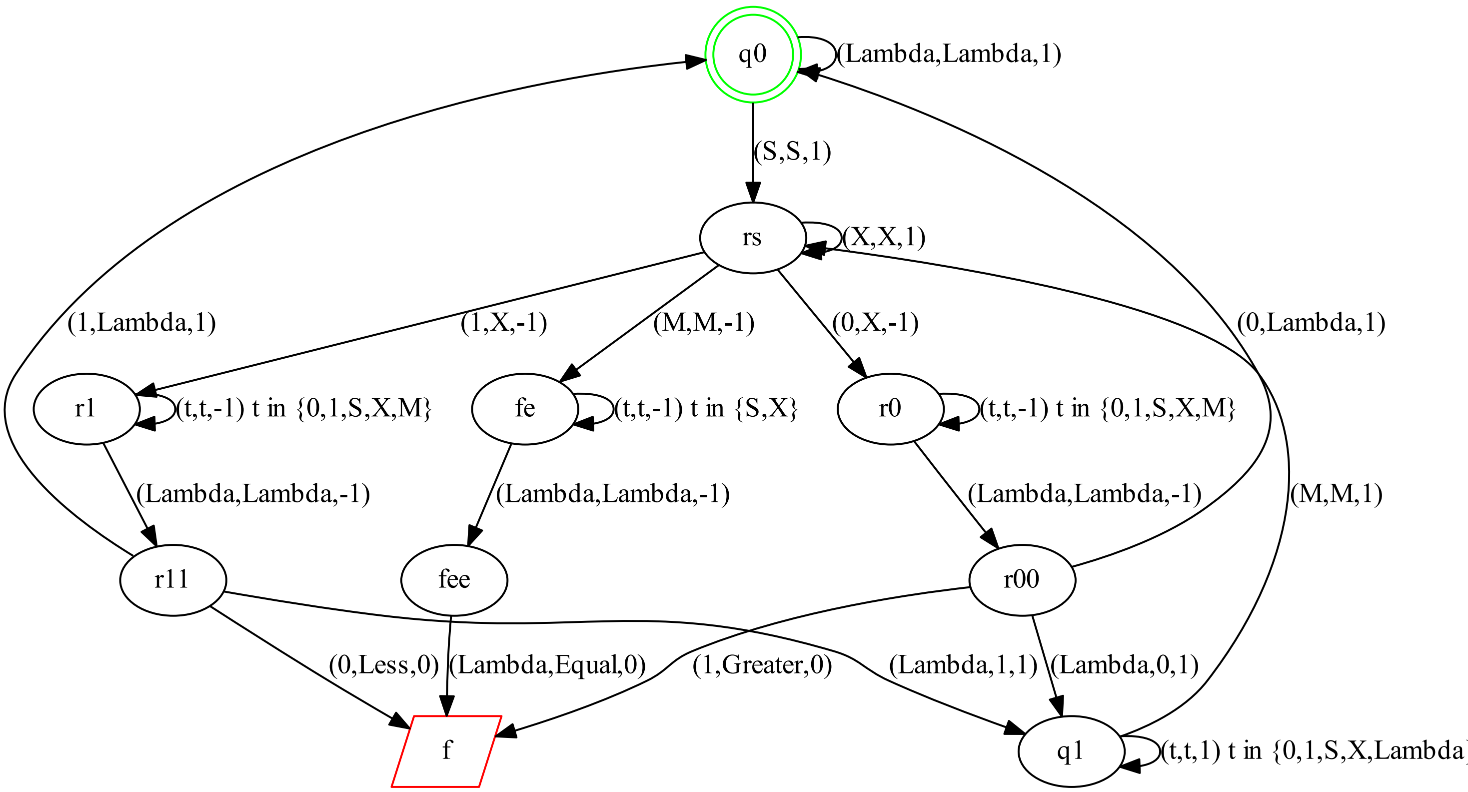
3. Get the last unextracted bit of the second number and set it X, then send the information to the result bit.

$$\begin{array}{ll} (\Lambda, q_1) \mapsto (\Lambda, q_1, 1) & (S, q_1) \mapsto (S, q_1, 1) \\ (0, q_1) \mapsto (0, q_1, 1) & (1, q_1) \mapsto (1, q_1, 1) \\ (X, q_1) \mapsto (X, q_1, 1) & \\ (M, q_1) \mapsto (M, r_s, 1) & \\ (M, r_0) \mapsto (M, r_0, -1) & (M, r_1) \mapsto (M, r_1, -1) \\ (0, r_0) \mapsto (0, r_0, -1) & (0, r_1) \mapsto (0, r_1, -1) \\ (1, r_0) \mapsto (1, r_0, -1) & (1, r_1) \mapsto (1, r_1, -1) \end{array}$$

4. Compare the two bits and halt when compare all the bits.

$$\begin{array}{ll} (0, r_{00}) \mapsto (\Lambda, q_0, 1) & (1, r_{11}) \mapsto (\Lambda, q_0, 1) \\ (\Lambda, q_0) \mapsto (\Lambda, q_0, 1) & \\ (0, r_{11}) \mapsto (\text{Less}, f, 0) & (1, r_{00}) \mapsto (\text{Greater}, f, 0) \\ (M, r_s) \mapsto (M, f_e, -1) & (X, f_e) \mapsto (X, f_e, -1) \\ (S, f_e) \mapsto (S, f_e, -1) & (\Lambda, f_e) \mapsto (\Lambda, f_{ee}, -1) \\ (\Lambda, f_{ee}) \mapsto (\text{Equal}, f, 0) & \end{array}$$

These are all, and then I give the graph.


q0
(Lambda,Lambda,1)
(S,S,1)
rs
(X,X,1)
(1,Lambda,1)
(1,X,-1)
(M,M,-1)
(0,X,-1)
(0,Lambda,1)
r1
(t,t,-1) t in {0,1,S,X,M}
fe
(t,t,-1) t in {S,X}
r0
(t,t,-1) t in {0,1,S,X,M}
(Lambda,Lambda,-1)
(Lambda,Lambda,-1)
(Lambda,Lambda,-1)
(M,M,1)
r11
fee
r00
(0,Less,0)
(Lambda,Equal,0)
(1,Greater,0)
(Lambda,1,1)
(Lambda,0,1)
f
q1
(t,t,1) t in {0,1,S,X,Lambda}

The arrow from one circle(status p) to another(status q) with triple $(a, b, c \in \{-1, 0, 1\})$ means that, the head with status p and the cell with content a, it will move to the left($c = -1$), right($c = 1$) or stay($c = 0$), and change the content a to b.

I have written a Python script to simulate this Turing Machine, and it will work well, the code in the attachment, you can test it with different inputs.

PROBLEM 2. Construct a single-tape Turing Machine, which can find out which sum of two n-bits natural numbers x, y in time $O(n^2)$.

Solution:

I build a machine with 31 status, it is a little complex. The annotation you can find in the solution of the first problem. I just give the necessary explanation.

The alphabet set and status set:

$$A = \{0, 1, X, \Lambda, R, M, S\}$$

$$\begin{gathered} Q = \{q_0, g_r, r, r_g, r_{g1}, r_{g0}, r_{g11}, r_{g00}, \\ g_l, l, l_g, l_{g1}, l_{g0}, l_{g11}, l_{g00}, l_{g0p}, l_{g1p}, l_p, l_{pp}, \\ r_f, f_f, f_{f0}, f_{f1}, f_1, f, \\ 0_{check_R}, 1_{check_R}, R_{set1}, R_{set0}, \\ 0_{CR}, 1_{CR}\} \end{gathered}$$

Where R means that cell on the left of R is used to store overflow bit.

The tape is of form:

$\cdots$	Λ	0	R	Λ	$\cdots$	Λ	S	b	$\cdots$	b	M	b	$\cdots$	b	Λ	$\cdots$

Figure 2: The tape with initial cell S, where b is 0 or 1.

It's worth noting that, we have to leave enough cells from R to S, because these cells are used to store result, in other words, if we find sum of two n-bits number, we have to leave at least $n + 1$ cells between R and S.

Before giving the transposition table and graph, I will explain these statues in brief. The status q_0 is the initial status, g_r, r, r_g are the statuses for prepare to get the bit of the second number, r_{g0}, r_{g1} denote that head get bit information 0 or 1, r_{g00}, r_{g11} mean that the head has brought the information to result area,

$g_l, l, l_g, l_{g0}, l_{g1}, l_{g00}, l_{g11}$ are similar, they are for the first number. $l_{g0p}, l_{g1p}, l_p, l_{pp}$ are used to deal with set overflow bit, $0_{check_R}, 1_{check_R}, R_{set0}, R_{set1}, 1_{CR}, 0_{CR}$ are used to deal with overflow bit when new bit comes. $r_f, f_f, f_{f0}, f_{f1}, f_1, f$ are used to deal with the final case and halt.

There are the transposition table:

1. Start from initial cell with initial status.

$$\begin{array}{ll} (S, q_0) \mapsto (S, g_r, 1) & (1, g_r) \mapsto (1, g_r, 1) \\ (0, g_r) \mapsto (0, g_r, 1) & (M, g_r) \mapsto (M, r, 1) \\ (1, r) \mapsto (1, r, 1) & (0, r) \mapsto (0, r, 1) \\ (\Lambda, r) \mapsto (\Lambda, r_g, -1) & (X, r) \mapsto (X, r_g, -1) \end{array}$$

2. Get the first bit of the rest bits of the second number, and set it to X.

$$(1, r_g) \mapsto (X, r_{g1}, -1) \quad (0, r_g) \mapsto (X, r_{g0}, -1)$$

3. Skip the other bits.

$$(p, q) \mapsto (p, q, -1), \text{where } p \in \{1, 0, X, M, S\}, q \in \{r_{g1}, r_{g0}\}.$$

4. Set the corresponding bit by information of head.

$$\begin{array}{ll} (\Lambda, r_{g0}) \mapsto (\Lambda, r_{g00}, -1) & (\Lambda, r_{g1}) \mapsto (\Lambda, r_{g11}, -1) \\ (\Lambda, r_{g00}) \mapsto (0, 0_check_R, -1) & (\Lambda, r_{g11}) \mapsto (1, 1_check_R, -1) \end{array}$$

5. Skip set bit till Lambda while present status with r_{g00} or r_{g11}.

$$(p, q) \mapsto (p, q, -1), \text{where } p \in \{0, 1\}, q \in \{r_{g00}, r_{g11}\}.$$

6. Skip all alphabet till 'R' while present status is 0_{check_R} or 1_{check_R}.

$$(p, q) \mapsto (p, q, -1), \text{where } p \in \{0, 1, \Lambda\}, q \in \{0_{check_R}, 1_{check_R}\}.$$

7. Check overflow and handle these cases.

$$\begin{array}{ll} (R, 0_{check_R}) \mapsto (R, 0_{CR}, -1) & (R, 1_{check_R}) \mapsto (R, 1_{CR}, -1) \\ (0, 0_{CR}) \mapsto (0, g_l, 1) & (0, 1_{CR}) \mapsto (0, g_l, 1) \\ (1, 0_{CR}) \mapsto (0, R_{set1}, 1) & (1, 1_{CR}) \mapsto (1, R_{set0}, 1) \\ (\Lambda, R_{set0}) \mapsto (\Lambda, R_{set0}, 1) & (\Lambda, R_{set1}) \mapsto (\Lambda, R_{set1}, 1) \\ (R, R_{set0}) \mapsto (R, R_{set0}, 1) & (R, R_{set1}) \mapsto (R, R_{set1}, 1) \\ (0, R_{set0}) \mapsto (0, g_l, 1) & (0, R_{set1}) \mapsto (1, g_l, 1) \\ (1, R_{set0}) \mapsto (0, g_l, 1) & (1, R_{set1}) \mapsto (1, g_l, 1) \\ (0, r_{g00}) \mapsto (0, r_{g00}, -1) & (1, r_{g00}) \mapsto (1, r_{g00}, -1) \\ (R, r_{g00}) \mapsto (1, g_l, 1) & \end{array}$$

8. Go back to the initial cell and prepare to get the corresponding bit in the first number.

$$(p, g_l) \mapsto (p, g_l, 1), \text{where } p \in \{0, 1, \Lambda, R\}.$$

9. To get the corresponding bit of the first number.

$$\begin{array}{ll} (S, g_l) \mapsto (S, l, 1) & (1, l) \mapsto (1, l, 1) \\ (0, l) \mapsto (0, l, 1) & (M, l) \mapsto (M, l_g, -1) \\ (X, l) \mapsto (X, l_g, -1) & \\ (1, l_g) \mapsto (X, l_{g1}, -1) & (0, l_g) \mapsto (X, l_{g0}, -1) \end{array}$$

(p, q) $\mapsto (p, q, -1), \text{where } p \in \{1, 0, S\}, q \in \{l_{g1}, l_{g0}\}$.

10. Put the bit we got to the result bits and handle some cases.

$$(\Lambda, l_{g1}) \mapsto (\Lambda, l_{g11}, -1) \quad (\Lambda, l_{g0}) \mapsto (\Lambda, l_{g00}, -1)$$

11. Skip the last bits.

$$(p, q) \mapsto (p, q, -1), \text{where } p \in \{0, 1\}, q \in \{l_{g00}, l_{g11}\}.$$

12. Handling these cases.

$$\begin{array}{ll} (\Lambda, l_{g00}) \mapsto (\Lambda, l_{g0p}, 1) & (\Lambda, l_{g11}) \mapsto (\Lambda, l_{g1p}, 1) \\ (0, l_{g0p}) \mapsto (0, q_0, 1) & (0, l_{g1p}) \mapsto (1, q_0, 1) \\ (1, l_{g0p}) \mapsto (1, q_0, 1) & (1, l_{g1p}) \mapsto (0, l_p, -1) \\ (\Lambda, l_p) \mapsto (\Lambda, l_p, -1) & (R, l_p) \mapsto (R, l_{pp}, -1) \\ (0, l_{pp}) \mapsto (1, q_0, 1) & \end{array}$$

13. Go back to the initial position and execute the next bit.

$$(p, q_0) \mapsto (p, q_0, 1), \text{where } p \in \{0, 1, \Lambda, R\}.$$

14. When we handle all the bits, then halt.

$$(X, g_r) \mapsto (X, g_r, 1) \quad (M, r_g) \mapsto (M, r_f, -1)$$

(p, r$_f$) $\mapsto (p, r_f, -1), \text{where } p \in \{0, 1, X, S, \Lambda\}$.

$$\begin{array}{ll} (R, r_f) \mapsto (R, f_f, -1) & (0, f_f) \mapsto (0, f_{f0}, 1) \\ (1, f_f) \mapsto (0, f_{f1}, 1) & \end{array}$$

(p, q) $\mapsto (p, q, 1), \text{where } p \in \{\Lambda, R\}, q \in \{f_{f0}, f_{f1}\}$.

$$\begin{array}{ll} (0, f_{f0}) \mapsto (0, f, 0) & (1, f_{f0}) \mapsto (1, f, 0) \\ (0, f_{f1}) \mapsto (0, f_1, -1) & (\Lambda, f_1) \mapsto (1, f, 0) \\ (1, f_{f1}) \mapsto (1, f_1, -1) & \end{array}$$

And then the transposition graph.


q0
(t,t,1) t in {0,1,R,Lambda}
(S,S,1)
gr
(t,t,1) t in {0,1,X}
(M,M,1)
r
(1,1,1)
(0,0,1)
(Lambda,Lambda,-1)
(X,X,-1)
rg
(M,M,-1)
(1,X,-1)
(0,X,-1)
rf
(t,t,-1) t in {0,1,X,S,Lambda}
rg1
(t,t,-1) t in {0,1,X,S,M}
rg0
(t,t,-1) t in {0,1,X,S,M}
(R,R,-1)
(Lambda,Lambda,-1)
(Lambda,Lambda,-1)
ff
rg11
(t,t,-1) t in {0,1}
rg00
(t,t,-1) t in {0,1}
(1,0,1)
(Lambda,1,-1)
(Lambda,0,-1)
(0,0,1)
ff1
(t,t,1) t in {R,Lambda}
1_check_R
(t,t,-1) t in {0,1,Lambda}
0_check_R
(t,t,-1) t in {0,1,Lambda}
(1,1,-1)
(0,0,-1)
(R,R,-1)
(R,R,-1)
(1,1,1)
(0,0,1)
ff0
(t,t,1) t in {R,Lambda}
f1
1_CR
(R,1,1)
0_CR
(0,1,1)
(0,0,0)
(1,1,0)
(Lambda,1,0)
(1,1,1)
(1,0,1)
f
(0,0,1)
R_set_0
(t,t,1) t in {R,Lambda}
(0,0,1)
R_set_1
(t,t,1) t in {R,Lambda}
(0,1,1)
(1,0,1)
(0,0,1)
(0,1,1)
(1,1,1)
gl
(t,t,1) t in {0,1,R,Lambda}
(S,S,1)
l
(t,t,1) t in {0,1}
(M,M,-1)
(X,X,-1)
lg
(0,X,-1)
(1,X,-1)
lg0
(t,t,-1) t in {0,1,S}
lg1
(t,t,-1) t in {0,1,S}
(Lambda,Lambda,-1)
(Lambda,Lambda,-1)
lg00
(t,t,-1) t in {0,1}
lg11
(t,t,-1) t in {0,1}
(Lambda,Lambda,1)
(Lambda,Lambda,1)
lg0p
lg1p
(1,0,-1)
lp
(Lambda,Lambda,-1)
(R,R,-1)
lpp

I also write a simulation for this Adder Turing machine, you can find it in the attachment and test it. It works well.