Algorithm Design and Analysis
Homework #5

Due: 14:20 December 20, 2012

Homework submission instructions

- Submit your programming assignment (problem 1) to the Judgegirl System (http://katrina.csie.ntu.edu.tw/judgegirl/). Also, use SVN to commit your report (named “report.pdf”) in the same folder of your code.

- Submit the answer of the writing problems by one of the following ways:

  1. Submit an electronic copy to the CEIBA system before the deadline; or
  2. Submit a hard copy right before the class on the day of the deadline.

  If you submit by both ways, the TA will randomly select only one of them for grading.

- Please make sure that you write down your name and school ID in the header of your documents. Otherwise, the TA will not grade them.

- If you choose to submit the answers of the writing problems via CEIBA, please combine the answers of all writing problems into only one file in the pdf format, with the file name in the format of “hw5 [ID].pdf” (e.g. “hw5 b99902010.pdf”). Otherwise, the TA will not grade them.

- What should be included in the report:

  1. Explain how your program works in detail.
  2. Derive the time complexity of your program and briefly explain why. No need for formal proof.
  3. The reference of your report.

- If you are stuck with any problem, you are welcome to ask the TAs via e-mails or in person during the office hour. The TA may discuss with you and give you some useful hints. :D

Problem 1 – Puzzle

Description

There are 16 sliding tiles inside a 4×4 frame, each with a number from 0 to F on it (hexadecimal). Four buttons, R_0, R_1, R_2, R_3, are on the right of the frame. The $(i+1)$-th row will rotate right by one tile when R_i is pressed, i.e., every tile in that row will move to the location on its right, except the rightmost tile, which will move to the leftmost location in that row. Another four buttons, C_0, C_1, C_2, C_3, are at the bottom. Similarly, the $(i+1)$-th column will rotate down by one tile when Ci is pressed, i.e., every tile in that column will move to the location below it, except the bottommost tile, which will move to the topmost location in that column.

Given an arrangement of the tiles, please find an optimal solution (fewest number of moves) to rearrange the tiles in order by pressing buttons.

Input

Four lines containing four hexadecimal digits in each line, representing the initial configuration of the frame. The neighboring digits are separated by one space character ‘ ‘.

The input is always legal, i.e., each of the digits from 0 to F occurs exactly once in the given input data.

Output

Output an integer X, the minimum number of steps required to reach the goal, in the first line. Then the next X lines should show the sequence of buttons that should be pressed to move the tiles from the initial configuration to the final configuration with all tiles in proper order.

If there are two or more optimal solutions, choose the one with minimum lexicographic order.

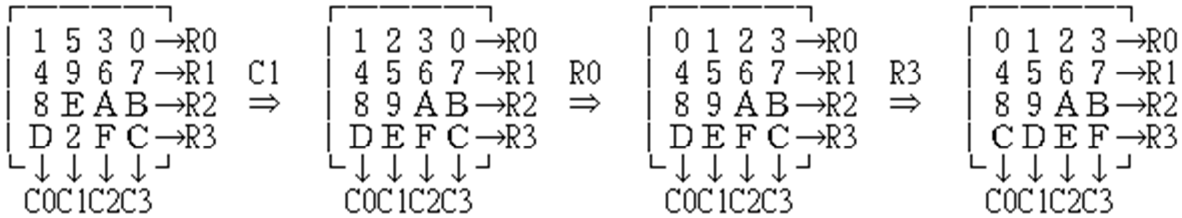
Sample Input

```
1 5 3 0
4 9 6 7
8 E A B
D 2 F C
```

Sample Output

```
3
C1
R0
R3
```

Hint



```
┌───────┐          ┌───────┐          ┌───────┐          ┌───────┐
│ 1 5 3 0 →R0      │ 1 2 3 0 →R0      │ 0 1 2 3 →R0      │ 0 1 2 3 →R0
│ 4 9 6 7 →R1  C1  │ 4 5 6 7 →R1  R0  │ 4 5 6 7 →R1  R3  │ 4 5 6 7 →R1
│ 8 E A B →R2  ⇒   │ 8 9 A B →R2  ⇒   │ 8 9 A B →R2  ⇒   │ 8 9 A B →R2
│ D 2 F C →R3      │ D E F C →R3      │ D E F C →R3      │ C D E F →R3
└↓ ↓ ↓ ↓┘          └↓ ↓ ↓ ↓┘          └↓ ↓ ↓ ↓┘          └↓ ↓ ↓ ↓┘
 C0C1C2C3           C0C1C2C3           C0C1C2C3           C0C1C2C3
```

Use an A*/IDA* algorithm to solve this problem. Please look for relevant information on the Internet; this is an algorithm that is commonly taught in an Artificial Intelligence course. Being able to learn an algorithm all by yourself from the information you found is cool, don't you think? :P

Grade

Judge result (10%)
Report & code (10%)

<u>In the report, please also explain/show that your algorithm can indeed produce the optimal solution, i.e., generate a solution with the minimum number of steps.</u>

Problem 2 – New Policeman's Confusion

From a police station located on a straight road which is infinite in both directions, a thief has stolen a police car. A while later, a policeman discovers the car is missing and starts to pursue the thief on foot. His maximum running speed equals to two times the maximum speed of the car.

However, he knows neither in which direction the thief has gone nor how long ago the car was stolen.

1) (5%) Design a strategy for the policeman to catch the thief in finite time.
2) (5%) If actually the thief stole the car T seconds ago(but the policeman doesn't know it). For your strategy, please analyze the running time of the policeman, in terms of T, to catching the thief.
3) (5%) Besides the possibility of the car being stolen by a thief, think of another possibility: the car has disappeared from the world due to some natural miracle power! Due to this possibility, the police now want to know whether there is a thief or not in finite time. Does your previous strategy help? Why?

Problem 3 – NP's Closure

Let $L_a,\ L_b \in P$ be two languages that are in the complexity class P. Show that the following languages still belong to P:

1) (5%) $L_1 = \overline{L_a} = \{x | x \notin L_a\}$ (Complement operation)
2) (5%) $L_2 = L_a L_b = \{x_a x_b |\ x_a \in L_a, x_b \in L_b\}$ (Concatenation operation)

Now, consider $L_a,\ L_b \in NP$ be two languages that are in the complexity class NP. Show that the following languages also belong to NP:

3) (10%) $L_3 = L_a \cup L_b = \{x | x \in L_a\ or\ x \in L_b\}$ (Union operation)
4) (10%) $L_4 = L_a^* = \{\varepsilon\} \cup L_a \cup L_a^2 \cup L_a^3 \cup \ldots$ (Kleene star operation)

Problem 4 – Numerous Purchase Coupons

Let us revisit the problem 'money' in HW4 again. The time complexity of your solution is probably $O(NM)$ or $O(NMlgM)$. For the following, you can choose either of these two time complexities as the time complexity of your algorithm.

Recall that the input contains:

M //The total value of the coupons
N //The number of kinds of items
$n_1, b_1, s_1;\ \dots;\ n_N, b_N, s_N$ //Number of item, price of buying, price for sale

Where $0 \le n_i, b_i, s_i \le M$ for all $i = 1,2..N$. Assume that we can encode every integer x as a binary string of length $O(lgx)$.

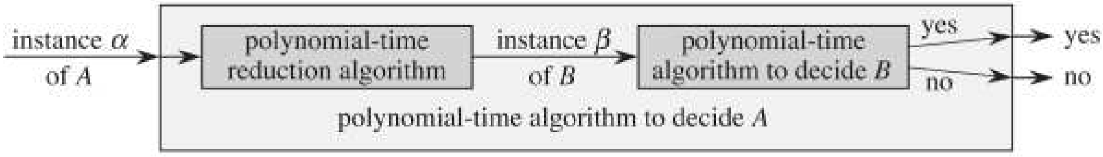
1) (10%) Let L be the input length of the 'money' problem. Give an asymptotically upper bound of L in terms of M, N, i.e., find a $f(M,N)$ such that $L = O(f(M,N))$. Make your bound as tight as possible.
2) (10%) Is your dynamic-programming algorithm for the 'money' problem really a polynomial-time algorithm? Explain your answer.

Problem 5 – Sorry, I am a NPC.

The warrior *LostEclipse0721* and the magician *~*MisaCookie*~* visited a NPC's(non-player character) home and found N valuable stuffs. The number N, the values of these stuffs $v_1,\ v_2, \dots v_N$, and a weapon price T are given as the input. Now they are facing two problems:

*~*MisaCookie*~*:"I wonder that whether these stuffs can be divided into two partitions — one for you and the other for me — such that the total values of each partition are exactly same.($Q1$)"

LostEclipse0721:"I must enhance my weapon before fighting the next boss. So I'd like to take some stuffs whose total values are exactly T. The remaining part is yours. However, does such partition really exist?($Q2$)"


instance α of A
polynomial-time reduction algorithm
instance β of B
polynomial-time algorithm to decide B
yes
no
yes
no
polynomial-time algorithm to decide A

1) (5%) Assume we have an algorithm $ALGO2$ that can solve $Q2$. Please design a reduction algorithm to transform an instance of a $Q1$ problem into an instance of a $Q2$ problem. What is the time complexity of your reduction algorithm?
2) (10%) Conversely, you are given an algorithm $ALGO1$ for solving $Q1$. Please design an algorithm to solve $Q2$ by using $ALGO1$.
3) (Bonus 5%) Please prove that the two problems mentioned above are polynomial-time reducible to the 'money' problem in HW4.