Algorithm Design and Analysis
Homework #4

Due: 2:20pm, Thursday, November 28, 2013

TA email: ada@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, commit your source code to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "`hw4`" and put the source code file in it. The filename of the source code should be "`main.c`"; you will get some penalties in your grade if your submission does not follow the naming rule.
  You have to login to judgegirl system (http://katrina.csie.ntu.edu.tw/judgegirl) to make it judge your program.
- You need to submit your `report of Problem 1` and your `answer to other Problems` via the SVN server (electronic copy, all in a single pdf file named "`hw4_[student ID].pdf`") or to the TA at the beginning of class on the due date (hard copy).
- For hard copies
  - Please use A4 paper.
  - Please write down `your name` and `the page number` on every page of your homework.
  - Please do not use paper with things printed on the other side.
    You are encouraged to write down your answers on both sides of the paper.
  - Please don't use staples to bind you papers.
    The TA will provide you paperclips when you hand in your homework.
  - Please take your homework out of the L-folder when you hand it in, or you might lose your L-folder.

  You may get some penalties in your grade if you do not follow the above rules.
- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, for each problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem.
- **NO LATE SUBMISSION IS ALLOWED** for the homework submission in hard copies - no score will be given for the part that is submitted after the deadline. For submissions via SVN (including the programming assignment and electronic copies of the writing problems), up to one day of delay is allowed; however, the score of the part that is submitted after the deadline will get some penalties according to the following rule (the time will be in seconds):

$$\text{LATE SCORE} = \text{ORIGINAL SCORE} \times (1 - DelayTime/86400)$$

Problem 1. Dating Crisis

(30%)

In the CSIE department, there are some shy boys who want to date with girls. Each boy may have a crush on several girls, but they can only choose one girl as his girlfriend. Although there may be several boy who would like to select the same girl, that girl must also only choose one boy as her boyfriend. Given a series of secret love relations (only boys to girls), please match the boys to the girls to get the maximum number of couples.

Report(10%)

Please write down how you derived the algorithm, how you implemented it, and analyze the running time of your algorithm using asymptotic notations. In your running time analysis, please use only |V| and |E| to describe your final answer, using |flow| is not allowed.

Coding(20%)

Input format:

The first line contains the total number of secret love relations, N, and the total number of people, $M(M \leq 750)$. The following N lines contain the secret love relations (Boy's name <3 Girl's name). We also guarantee that the length of each name ≤ 10.

Note: the Boy and Girl's names will use the UTF-8 encoding, so please look for information about how to read it from the input on the Internet. Note that there is no space character (' ') between a name and "<3".

Note: A Boy's name will not be the same as a girl's name

Output format:

Please output the maximum number of couples.

Sample input:

2 3
蔡蔡 <3 欣欣
穆穆 <3 欣欣

Sample output:

1

Sample input:

2 3
蔡蔡 <3 欣欣
蔡蔡 <3 穆穆

Sample output:

1

Problem 2. Prove the closeness of P (16%)

Show that the class P, viewed as a set of languages, is closed under the following operations. That is, if $L_1,\ L_2 \in$ P, then

(1) (4%) $L_1 - L_2 \in$ P

(2) (4%) $L_1 L_2 \in$ P

(3) (4%) $\bar{L}_1 \in$ P

(4) (4%) $L_1^* \in$ P.

Problem 3. Counter (20%)

We have seen in the lecture that the amortized cost of each increment operation of a k-bit binary counter is $O(1)$. Now, we will try to support a few more operations in the counter. For simplicity, we only consider the cost of flipping bits in the computation of running time, as this is the most time-consuming part. For example, if an algorithm checks the values of all k bits but only flips one bit, then we still conclude that this algorithm runs in $O(1)$.

1. (8%) Increment-2^i operation

   Denote the value of the counter as v. After conducting an increment-2^i operation, the value of the counter will become $(v + 2^i) \bmod 2^k$, where $i \in \{0, \ldots, k-1\}$.

   (a) Write down the pseudo code of the increment-2^i operation.

   (b) Derive its worst-case running time using the O-notation.

   (c) Assume that there are n operations, consisting of both the increment and the increment-2^i operations, are performed on an initially zero counter. Show that the amortized cost of each operation is $O(1)$. You may use any analyzing method that we have introduced in the lecture.

2. (8%) Shifting-right operation

   Denote the value of the counter as v. After conducting a shifting-right operation, the value of the counter will become $v >> 1$, and the least significant bit is dropped.

   (a) Write down the pseudo code of shifting-right operation.

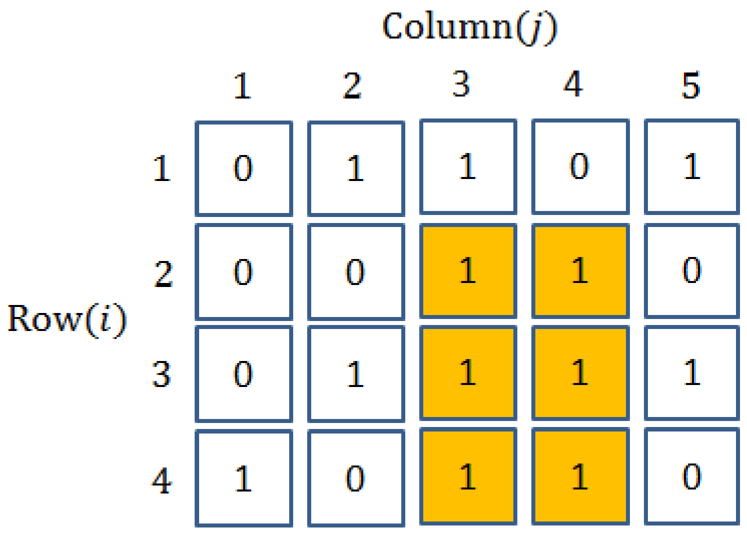
   (b) Derive its worst-case running time using the O-notation.

   (c) Assume that there are n operations, consisting of the increment and the shifting-right operations, are performed on an initially zero counter. Show that the amortized cost of each operation is $O(1)$. You may use any analyzing method that we have introduced in the lecture.

3. (4%) Assume that there are n operations, consisting of the increment, the increment-2^i and the shifting-right operations, are performed on an initially zero counter. Is the amortized cost of each operation still $O(1)$? If yes, prove it; otherwise, give a counterexample.

Problem 4. Maximum Sub-Rectangle (16%)

1. (8%) Given a non-negative integer array of size n, we would like to find the sub-array with the "maximum contribution of minimum member value", which is defined as the length multiplies the minimum value in the sub-array. For example, if the array is {0,4,2,3,3,3,3,2,1,0}, then the sub-array can be {4} with sum 4 (4×1), or {4,2,3,3,3,3,2} with sum 14 (2×7), or {3,3,3,3} with sum 12 (3×4), {4,2,3,3,3,3,2,1} with sum 8 (1×8). Accordingly, the sub-array with the "maximum contribution of minimum member value" is {4,2,3,3,3,3,2}. Please design an algorithm to find this sub-array and use amortized analysis to show that the time complexity of the algorithm is $O(n)$.



2. (8%) In last semester's DSA course, there was a problem (dsa_13spring hw1 problem5) which asks you to find the maximum sub-rectangle with all 1's in a given $m \times n$ $\{0,1\}^*$ matrix, and the given algorithm's time complexity is $O(mn^3)$. Now, based on your answer of problem 4.1, please design and describe an algorithm with $O(mn)$ to find the area of the maximum sub-rectangle. For example, the area of the maximum sub-rectangle in the figure above is 6.

***Problem* 5.** Multiple Sorted Arrays (18%)

Sorted array is an array data structure in which the elements are sorted by some specified order. If there are n elements in the sorted array, we can find the k-th largest element in $O(1)$ time or check if an element is in the array by performing a binary search which runs in $O(\log n)$ time. However, insertion and deletion could cost $O(n)$ time if the element inserted or deleted is not at the end of the array.

Now we propose a new data structure which has k sorted array $A_0, A_1, A_2, ..., A_{k-1}$ where the size of array A_i would be 2^i. Each of the array can be either empty or full.

For example, if $n = 13 = 1101_2$, A_3, A_2, A_0 would be full (A_3 contains 8 elements. A_2 contains 4 elements. A_0 contains 1 element.) and A_1 would be empty.

Given n, the number of elements, and the sorted arrays, $A_0, A_1, A_2, A_3, ..., A_{k-1}$.

(Assume that k is large enough so that $2^k - 1$ is always larger than the number of elements.)

1. (4%) Describe an algorithm to check if an element x is in the data structure.
   (3%) Analyze its worst-case running time.
2. (4%) Describe an algorithm to insert an element x into the data structure.
   (3%) Analyze its worst-case running time and amortized running time.
3. (4%) Describe an algorithm to delete the element $A_i[j]$. Assume that $0 \le i < k$ and A_i is not empty and $0 \le j < 2^i$.