Data Structure and Algorithm I

Homework #5

Due: 17pm, Thursday, December 30, 2010

Submit the answers for problem 2-5 through the CEIBA system (electronic copy) or to the TA in R432 (hard copy). You also need to submit the answers of problem 1 through the CEIBA system. If you choose to submit your writing problems through CEIBA,

1. please combine the answers of all writing problems into only one file in the doc/docx or pdf format, e.g., hw4_writing.pdf; otherwise, you will only get the score of one problem (the one that the TA chooses);
2. please submit your programming report together with your source codes (in the zip archive); if the programming report is submitted with the answers of the writing problems, you will NOT get the score for the report; and
3. please write down your name and school ID in the header of both of the above two documents, e.g., b98902xxx Your_Name

Problem 1. (40%) In Chapter 8, we introduce static hashing. In this problem, you are required to implement the division hash function coupled with chaining that handles the situation of collisions. Your program should take the input from the standard input device (stdin) and please use the following input/output format:

Input format:
The first input line is the size of hash table, i.e. b buckets, which is also the divisor, D
The following input lines represent the keys(strings) inserted into the hash table sequentially.
Each bucket has only one slot and the string is no more than 20 characters.

Example 1. The following is an input example.

```
26 --> the number of buckets, b, and also the divisor, D.
```

```
acos --> the 1st key inserted
atan --> the 2st key inserted
atoi
char
atoi
ceil
bdnr
exp  --> the 8st key inserted
```

In your program, you should use the **Program 8.1** in the textbook to convert strings, s, to integers, k. Then, the remainder of k divided by D is used as the home bucket for s. Additionally, you will need to consider the case that a key already exists in the hash table before inserting.

Output format:
For each key(string), you should output the key(string), existence(y/n), collision(y/n), bucket number and the position in the chain. The following is an output example.

Example 2. Continue with *Example* 1.

```
acos n n 6 1
atan n n 4 1
atoi n n 13 1
char n n 24 1
atoi y
ceil n n 23 1
bdnr n y 6 2
exp n n 21 1
```

Notice that the second "atoi" has already existed in the hash table, so you only output "y" for the existence. Because "bdnr" has the same remainder value as "acos", 6, you should output "y" for collision and put it behind "acos". Consequently, the position of "bdnr" is 2.

You must upload your homework in the format of a compressed zip file to the CEIBA, and the zip file should include the following three files:

1. The source code (.c file),
2. A shell script to compile the source (.sh), and
3. A document in PDF format to describe how your program/algorithm works.

Your score of 40% is divided into two parts: correctness (with 4 test cases)(32%) and explanations in the document(8%).

Problem 2. (35%)

1. (5%) Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of each iteration of the for loop of *insertionSort*(Program 7.5 in the textbook)
2. (5%) Draw a figure similar to Figure7.1 in the textbook starting with list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) − *quickSort*
3. (5%) Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of each phase of *mergeSort*(Program 7.9)
4. (5%) Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of the first *for* loop as well as at the end of each iteration of the second *for* loop of *heapSort*(Program 7.13)
5. (5%) Write the status of the list (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) at the end of each pass of *radixSort*(Program 7.14).Use r=10.
6. (5%) Give an example that *quickSort* may run in $O(n^2)$.
7. (5%) Give an example that *insertionSort* can run faster than *mergeSort*.

Problem 3. (10%) Now we want to sort n integers in the range 0 to $n^2 - 1$ in O(n) time by *radixSort*. Show what value of d you would and why. (For example, when $r = 10$ you can use $d = log_{10}n^2$) Explain why the time complexity of your algorithm is O(n).

– you can refer to section 7.7 in the textbook in p. 356 to p. 359 –

Problem 4. (15%) Consider a hash function $h(k) = k\%D$, where D is not given. We want to figure out what value of D is being used. We wish to achieve this using *minimum* attempts, where an attempt consists of supplying the number k (please supply k in the range [1,50]) and observing $h(k)$. Write down your attempts and explain briefly why your attempt can achieve our goal.

1. (5%) D is known to be a prime number in the range [10,20].
2. (10%) D of the form 2^t, where t is an integer in [1,5].

– you can refer to section 8.2.2 and 8.2.2.1 in the textbook in p. 398 –