

Homework 2

February 17, 2004

Due Date: February 24

1. (15 pts) In the first two parts of this problem we look at hashing. Assume the hash table is called **table** and is of size 8. Here are the hash values that you will need:

	22	30	45	47	51	73
baseHash	0	5	5	5	0	4
stepHash	1	3	1	7	5	5

- (a) Show the result of inserting items with keys of 73,22,51,30,45, and 47 when using chaining to resolve collisions.
- (b) Show the result of inserting items with keys of 47, 22, 51, 30 (in that order) using open addressing with double hashing to resolve collisions.
- (c) Illustrate the operation of the **Partition** algorithm on the array: 8,17,12,5,3,10 when partitioning around the 10. Indicate where the swaps occur during the algorithm's execution and show the state of the array after each swap.
2. (15 pts) This problem is designed to be sure you understand the definition of expected value and can use it to determine the expected number of times a portion of code is executed. There is a practice problem on the webpage like this so you should look at that if you feel like you need more guidance before starting. Here is code for binary search which assumes that A is sorted such that $A[i] \leq A[i+1]$. The initial call made is `binarySearch(x,A,0,9)`.

```
boolean binarySearch(x, A, p, r)
```

```
1 if (p == r)
2   return (x == A[p])
3 q = [(p + r)/2]
4 if (x == A[q])
5   return true
6 if x < A[q]
7   binarySearch(x, A, p, q - 1)
8 else
9   binarySearch(x, A, q + 1, r)
```

You are to compute the expected number of times that x is compared to an element of A (in line 2,4 or 6) when searching in array $A[0..9]$ for an element x where for $0 \leq i \leq 9$, x is $A[i]$ with a probability of $1/10$. Using the formal definition of expected value, compute the exact value.

3. (20 points) In order to have an *extensible* array (such as Java's `ArrayList` that has no upper limit on the size, the most efficient implementation is to double the array size (copying over the elements in use) when the array is full. We study an extensible array here in the context of open addressing but the same basic analysis would hold for Java's `ArrayList` implementation.

Suppose that you have a hashtable with open addressing that is initially set to a size of 2. Whenever the number of items stored in the hashtable reaches $1/2$ of the current array size you are to double the array size and must then rebuild the array since the hash values have

changed. Assume that there are 2^k elements inserted (and none deleted). Compute the number of re-insertions needed during the doubling and then report the average cost per element of the doubling by dividing the number of re-insertions by the number of elements (2^k). This is an example of amortized analysis in which you periodically do a lot of work (the doubling operation) but we can show that the work per insertion operation is still small. In other words, the cost of each doubling operation is amortized over a large number of insertions.

4. (50 points) For each of the following two situations you are to pick a data structure that is best suited for it and that meets the given specifications. The following components should be clearly given in each of your solutions. See the course web page for a sample solution for a problem of this type.

- You should very clearly describe your data structure choice, including all decisions about how the data structure is to be applied (e.g. what is used as the key, what is the associated data, what is the table size and collision resolution choice for each hashtable,...).
- You should clearly describe how each of the provided operations will be implemented AND analyze the efficiency for each of the operations. You only need to describe any new methods or variations of methods covered in class that you need. For the methods covered in class (e.g. insertion into a hashtable) you don't need to describe how it or derive the expected time complexity. You can just specify what standard method you are using (e.g. insertion) along with its parameters (e.g. what is given as the key, and what is given as the associated data).
- Briefly discuss your choice of data structures included the hashtable size and the method for collision resolution.

HINT: You will often find a need when you have two data structures that store related information, to have elements in one data structure hold a reference to the corresponding element in the other data structure. For example, you may find it useful to add a component within the elements (i.e. the `DictionaryRecord` class from Lab 2) of one hashtable that is a reference to the elements in another hashtable.

(a) You have an inventory system in which each item has unique barcode that is an 8 digit (base-10) number along with associated information such as the item name, number in stock, etc. Design a data structure to implement the following methods in the stated time. You should clearly describe your data structure and how each of the below methods will be implemented. Be sure to go over the expected time complexity for your implementation of each method.

Operation	Expected Time Requirement
<code>insertItem(int barcode, itemData data)</code>	$O(1)$
<code>removeItem(int barcode)</code>	$O(1)$
<code>findItem(int barcode)</code>	$O(1)$
<code>printSortedInventory</code>	$O(n)$

where n is the number of items currently in the system. The semantics of `insertItem` and `removeItem` should be clear. The method `findItem` should return the data provided when the given barcode was inserted. Finally, `printSortedInventory` should print a list of all items sorted by barcode along with the associated information.

- (b) You have been hired as a consultant by University X. They have an enrollment of s students where s is approximately 25,000. Each student has an id number which is a 9 digit number that is unique for that student. In addition for each student there is a `StudentRecord` that includes a name, an email address, a mailing address, current courses (initially empty) and a transcript. Each course also has unique id (e.g. E61241SP03 which could then be converted to a unique integer). For each course there is a `CourseRecord` that includes a professor, classroom and class list (initially empty). Each semester, University X teaches approximately t courses where t is approximately 4500. The size of course i is denoted by n_i where $5 \leq n_i \leq 300$. Finally, you can assume that each student takes no more than 10 courses in a semester.

You are to design a data structure to implement the following operations in the specified *expected* time. So any operation that must run in expected constant time cannot depend on the size of s , t , or n_i for any i . From the names of the methods it should be clear what is expected of them. For `printSortedRoster` the roster must be sorted alphabetically by the students' names (and also include their id and email address). If you are uncertain about the semantics of any other methods please ask. You should assume that both the `insertStudent`, `removeStudent`, `registerStudent`, and `withdrawStudent` methods are called frequently, whereas the `insertCourse` method is not called very frequently. Also, notice that courses are never removed.

Operation	Expected Time Requirement
<code>insertStudent(int studentID, StudentRecord data)</code>	$O(1)$
<code>removeStudent(int studentID)</code>	$O(1)$
<code>insertCourse(int courseID, CourseRecord data)</code>	$O(1)$
<code>registerStudent(int studentID, int courseID)</code>	$O(1)$
<code>withdrawStudent(int studentID, int courseID)</code>	$O(1)$
<code>printSortedRoster(int courseID)</code>	$O(n_i \log n_i)$ for course i

Challenge Problems:

Only work on these after you have completed the required problems. Note that you can obtain 100% without doing either of these.

5* (2 points)

The following program determines the maximum value in an unordered array $A[0..n-1]$.

```

1 max = -∞
2 (int i = 0; i < n; i++)
3     if A[i] > max
4         then max = A[i]
```

What is the expected number of times the assignment in line 4 is executed? (Assume that all numbers in A are drawn randomly from the interval $[0,1]$.) You may find it useful to let s_1, s_2, \dots, s_n be n random variables, where s_i represents the number of times (0 or 1) that line 4 is executed during the i th iteration of the **for** loop. Since $s = s_1 + s_2 + \dots + s_n$, by linearity of expectations

$$E(s) = E(s_1) + \dots + E(s_n).$$

6* (3 points) Generalize your work from Problem 2 to compute the expected number of elements examined when the array has $n = 2^r - 1$ elements (for r an integer).