Department of Electrical and Computer Engineering McGill University

ECSE-322 Computer Engineering

18 January 2008

Problem Set 2

1. Hashing:

We need to use a hashing function in order to store, in an array of strings, family names of the 10 top students of a class of 1000 students. We have the following family of hashing functions:

$H_k(e)$ = ((e)ModM+k)Mod M, where k is the hashing function number, and M is the number of positions in the array.

(a) Give some examples of what we can use as the key: *e*.
(b) What size array should we dedicate for storage?
(c) Describe when hashing is best suited to store data (in general). Is it well suited for this problem?
(d) Explain why the 1st Mod function is useful? The 2nd?
(e) How would you go about retrieving such data?
(f) What problem with hashing does bucket hashing help solve?
(g) Show the array of hashed elements for the following names using e=(sum(ASCII(1st letter of name)+ ASCII(last letter of name))):
  Names: Daniels, Arnold, Beaudet, Stephens, Durocher, Atkins, Moore, Cameron, Ini, Lanf.
  (i) Using the family of hashing function described
  (ii) Using bucket hashing

2. Design a data structure for representing signed integer numbers of arbitrary size, with efficient algorithms for addition and comparison. What is the running time of your algorithms?

3 Determine the performance of Quicksort and Bubblesort in the following particular cases:
  (a) All the elements of the array have the same value.
  (b) The elements of the array are already sorted in increasing order.
  (c) The elements of the array are sorted in decreasing order.

4. Extend the array vectoring formula in the notes to a 6-dimensional array E:
  (a) For an array of 4 x 6 x 5 x 3 x 7 x 2, what is the index of
    (i) E[2,1,4,2,5,1]
    (ii) E[0,0,0,0,0,1]

(iii) E[0,3,0,0,1,1]

(b) How much memory is required to store the array?

(c) What is the time required for index computation?

(d) Develop an indexing function for the form of vectoring for sparse multi dimensional arrays. This function should return the presence or lack of an entry in a specific array position.

5. An MxN array is stored by rows. What minimum amount of temporary storage is required to rewrite it so it is stored by columns? How much time is needed? Can time be traded for temporary storage in this case?

6. In a square sparse matrix (of dimension several hundred), only the non-zero terms are to be stored. The row and column locations are to be indicated in associated integer arrays in as compact a form as possible, i.e. by using vectoring if possible. If an integer occupies 2 bytes and a real number occupies 4 bytes. Determine the approximate level of sparsity (in terms of the percentage of the full matrix) below which it is not worth storing only the non-zeroes.

7. Write the conditions for testing a queue for emptiness. Assume a queue is implemented as an array (NAME) of size k. How many elements may be stored in the queue? Draw pictures illustrating the queue and typical positions for the pointers FRONT and REAR when the queue (a) is empty, (b) contains one element, and (c) is full.

Notation: Items are added at the FRONT and removed at the REAR.

8. The row and column locations of a sparse matrix are to be indicated in associated integer arrays, possibly with vectoring. Determine the approximate levels of sparsity (in terms of the percentage of the full matrix) for which it is best to use array storage, sparse matrix, or vectoring.

Extra Challenging Questions for the Brave:

9. A copy of the address book of a handheld computer is installed on a desktop PC. Both the handheld and the PC versions of the address book can be updated independently by editing a record, inserting a new record, or deleting an existing record. Design an ADT (Abstract Data Type) for the address book that supports the updates above, as well as a “hot sync” feature that transfers the latest changes on each record from the PC to the handheld and from the handheld to the PC.

10. A web browser is configured to hold the most recently visited web pages in a buffer of N1 Kbytes. Assume the average page size is N2 Kbytes and that the hit rate (number of page requests that can be served from the cache over the total number of page requests) is R. Is it more efficient to implement the buffer as an array sorted by visiting date, an unsorted array, a sorted linked list, an unsorted linked list, or a hash table? What are the tradeoffs?

Hint:

Let N = [N1/N2]. Assume that the browser replaces the oldest page when it retrieves a new page. If a page can be served from the buffer, the visiting time of that page is updated. Under these assumptions, the time to serve a page is:

serve = R*retrieve + (1-R)(findold + replace)

where retrieve is the time to find a page in the buffer by its URL, findold is the time to find the page with the earliest visitation date in the buffer, and replace is the time to substitute the oldest page by a new page.