University of Colombo School of Computing

SCS 1308 - Foundations of Algorithms

Take-Home

Instructions : Try the following questions and upload your answer script/code files as a zip file to the given link in the UGVLE on/before 02nd February at 6 pm. Note: Rename your zip file with your index number and name. (i.e: indexNo_Name.zip).

1. A software engineer tasked with developing a simple library management system that uses a hash table to manage book records. Each book has a unique ID (integer), and you need to implement a hash table to store and retrieve book information efficiently. The system should handle collisions using the division method and linear probing.

   I. Create a structure Book in C programming with the following fields:

   - ➔ int id: Unique book ID.
   - ➔ char title[100]: Book title.
   - ➔ char author[50]: Author's name.

   II. Write the C code to define the Book structure and a hash table array of size 10 to store these records.

   III. Write the C function int hashFunction(int key, int size) that takes a book ID and the hash table size as inputs and returns the hash index.

   IV. Write the C function void insertBook(Book *table, int size, Book book) to insert a book record into the hash table.

   V. Write the C function void displayTable(Book *table, int size) to display all records in the hash table.

   VI. Assume the hash table size is 10, and the following book IDs are inserted: 12, 22, 32, 42, and 52. Analyze the hash table's state after each insertion, showing the index and content.Illustrate the hash table's state after each book insertion using linear probing.

   VII. What are the potential performance issues when the hash table becomes almost full, and how could these be mitigated? (Discuss the efficiency of using linear probing in the hash table for a scenario where the number of book records approaches the hash table's size.)

2. A Sri Lankan company tasked with building a secure password management system. The system stores user passwords in a hash table, using a simple hash function (based on ASCII sum) and the division method to determine the index. Collisions are resolved through linear probing. Below are tasks requiring you to perform calculations and apply theoretical knowledge in practical scenarios.

   I. The company registers a user named ”Kamal” with the password ”Kamal@2025”. The ASCII values of each character in the password are summed, and the division method with a table size of 10 is used to find the hash index. Calculate the hash index for the password ”Kamal@2025”.Calculate the sum and then apply the division method to find the index

      ➔ ASCII sum of characters: ***K=75, a=97, m=109, a=97, l=108, @=64, 2=50, 0=48, 2=50, 5=53***

   II. Users ”Nimal” and ”Sunil” have passwords that also hash to the same index using the ASCII sum method and the division method with a table size of 10. Their passwords are ”Nimal@123” and ”Sunil@456”.Given that the hash index for ”Nimal@123” is 1 (same as ”Kamal@2025”), show how linear probing resolves this collision by calculating the next available index for ”Nimal”. (ASCII sum of ”Nimal@123” = 711.)

   III. If Kamal decides to update his password to ”Kamal@2026” ASCII sum is 757 . Calculate the new hash index for the updated password using the same method.

   IV. When the hash table becomes 70% full, it needs resizing. The current table size is 10, and there are 7 users. Determine the new table size by doubling the current size and rehash the password ”Sunil@456” into the new table. Calculate the new table size and the new hash index for ”Sunil@456” (ASCII sum of ”Sunil@456 = 746)

3. Suppose you have a hash table of length m = 16, implemented using open addressing. Show the result of inserting the set of keys {42, 45, 7, 61, 32, 4, 13, 27, 48} into the table using linear probing and quadratic probing. Calculate the number of collisions for each of the three methods. Use the following hash functions:

   Linear probing: $h(k, i) = (k + i) \bmod m$

   Quadratic probing: $h(k, i) = (k + i/2 + i^2/2) \bmod m$