Data Structures Fall 2023
Lab 13
Hash Tables Implementation

There are two things to understand while making hash tables,

1. What is the best hash function?
2. ***How is the data organized in the hash table?***

In this lab, we will only discuss how data is organized in hash tables

There are two ways that data is organized with the help of the hash table. In the first approach, the data is stored within the hash table, that is, in an array.
In the second approach, the data is stored in linked lists and the hash table is an array of pointers to those linked lists.

The hash table HT is, usually, divided into, say b buckets HT[0], HT [1], . . ., HT [b – 1]. Each bucket is capable of holding, say r items. Thus, it follows that br ¼ m, where m is the size of HT. Generally, r ¼ 1 and so each bucket can hold one item. The hash function h maps the key X onto an integer t, that is, h(X) ¼ t, such that 0 h(X) b – 1.

In this lab task, we will only focus on the array-based implementation.

Task One:

Array-based Hash Tables

You need to make the following functions and classes.

1. Class HashEntry { } //This will contain key and value
2. Class HashMap { } //This will contain a 2d Array of HashEntry Type
    a. ***HashMap()*** //Initalize the constructor
    {}
    b. ***HashFunction(int key***) // There are different types of hash functions , for now we will go with key%table_size
    c. ***Insert(key , value)*** //Now we will use linear probing to insert new value
    //Follwing is an algorithm
    hIndex = hashFunction(insertKey);
    found = false;
    while (HT[hIndex] != emptyKey && !found)
    if (HT[hIndex].key == key)
    found = true;
    else
    hIndex = (hIndex + 1) % HTSize;
    if (found)
    cerr << "Duplicate items are not allowed." << endl;
    else
    HT[hIndex] = newItem;
    d. ***Search(int key)*** //Search for the key and return true if found else false
    e. ~HashMap() //Destructor to free allocated memory

Task Two:

In this task, you are asked to make insertions in the table using ***random probing.*** This method uses a random number generator to find the next available slot. The ith slot in the probe sequence is (h(X) + ri) % HTSize where ri is the ith value in a random permutation of the numbers 1 to HTSize – 1. Also display the generated random numbers in each iteration.

Task Three:

In the process of inserting an item with key X into a hash table using quadratic probing, assume that the hash function places the item at position t, where $0 \leq t \leq$ HTSize – 1, and this position is already occupied. In quadratic probing, the search for an available slot begins at position t and continues with increments based on a quadratic sequence. Specifically, the probe sequence follows the pattern: t, (t + 1) % HTSize, (t + 2^2) % HTSize, (t + 3^2) % HTSize, ..., (t + i^2) % HTSize, where i is the current step in the probing process. This method ensures a systematic exploration of positions in the hash table until an empty slot is found.