CENG213
Data Structures

Middle East Technical University
Department of Computer Engineering

Lab 6

In this lab, we will be studying two open addressing methods of hashing:

- *Linear Probing*
- *Quadratic Probing*

Open addressing

- The general formula is as follows:
- $h_i(x) = (hash(x) + f(i))$ mod TableSize
- The function f is the collision resolution strategy.
- If a collision occurs, alternative cells are tried until an empty cell is found.

Linear Probing

- f is a linear function of i, we will use f(i)= i
- $h_i(x) = (hash(x) + i)$ mod TableSize
- you will try cells (h + 0), (h + 1), (h+2), . . . (h + i)

TASK1:Linear Probing

Your first task is to implement **hash**, **insert** and **get** functions.
Open the following *lab6* files:

- *myhash1.h*
- *main1.cpp*

hash Function

In this function you will implement the following formula to calculate the **hash** value for the given key. (hash function 3 in your lecture slides)

$$hash(key) = \left(\sum_{i=0}^{KeySize-1} Key[KeySize - i - 1] * 13^i \right) \text{ mod tableSize}$$

- E.g. assume that **tableSize** is 7
  - hash('ali') = (105 * 1 + 108*13 + 97*13^2) % 7 = 3
    a=97
    l=108
    i=105
  - hash('1')= 49*1 % 7 = 0
  - hash('2')= 50*1 % 7 = 1
  - hash('3')= 51*1 % 7 = 2
  - hash('8')= 56*1 % 7 = 0

insert Function

- In this recitation, the hash table was created as a **vector of pairs**. $vector < pair < string, int >> table$
  - **first** and **second** is used to access the 1.nd and 2.nd pair elements respectively.
  - We use the first item in the pair to store the string **key** and the second item to store an integer **value**.
  - E.g. table[i].first, table[i].second
- You will use the following function to **insert** an item to hash table: insert(string &key, int val)
  - Calculate the **hash** for the key.
  - If the corresponding row in the hash table is **empty** then insert the given key-value pair to that row.
  - If the **key** to be searched is found in the corresponding row, then update the integer **value** of the row. E.g. table[i].second=val
  - If the corresponding row in the hash table is not empty and does not contain the given key value then sequentially scan the array (with wraparound) until an empty cell is found or key value is found.

get Function

The **get** algorithm follows the same probe sequence as the insert algorithm.

- calculate the **hash** for the key.
- If the searched **key** is found in the hash table, then return its **value**. e.g. table[i].second
- If the corresponding row in the hash table is not empty and does not contain the given key value then sequentially scan the array (with wraparound) until an empty cell is found or key value is found.
- If the **key** to be searched is **(**not) found then return -1.

TASK2:Quadratic Probing

- Open the following *lab6* files:
  - *myhash2.h*
  - *main2.cpp*
- Again, you have to implement **insert** and **get** functions.
  - You will use the same **hash** function that you have implemented in Task1.
  - Copy and past the **hash** function code to myhash2.h.
- In Quadratic Probing collision function is quadratic.
  - We will use $f(i) = i^2$.
- $h_i(x) = (hash(x) + i^2)$ mod TableSize
- you will try cells $(h + 0^2), (h + 1^2), (h + 2^2), ...(h + i^2)$

This is the end of the lab sessions for Ceng213.
Good luck in the exams :)