Informatics II, Spring 2023, Exercise 10

Publication of exercise: May 08, 2023
Publication of solution: May 14, 2023
Exercise classes: May 15 - 19, 2023

Task 1

Consider a hash table with 11 slots, and use chaining to resolve conflicts. For each hash function $h_i(k)$ below, draw the hash table after inserting the following values: 6, 30, 41, 25, 17, 10, 60, 14, 52, 5

(a) $h_1(k) = (k + 5) \mod 11$ (**division method**)

(b) $h_2(k) = \lfloor 8(k * 0.618 \mod 1) \rfloor$ (**multiplication method**)

Task 2

Now consider a Hash Table with open addressing. Insert the values {6, 30, 41, 25, 17, 10, 60, 15, 52, 5} using the provided hash functions $h_3(k, i)$ and $h_4(k, i)$.

(a) $h_3(k, i) = (k + i) \mod 11$ (**linear probing**)

(b) $h_4(k, i) = (k \mod 11 + i * (k \mod 7)) \mod 11$ (**double hashing probing**)

Task 3: Coding

Implement a hash table with m slots, where m is a positive integer. All keys are positive integers. Use double hashing to resolve collisions.

$$
\begin{aligned}
h(k, i) = (h_1(k) + i * h_2(k)) \mod m \\
h_1(k) = (k \mod m) + 1 \\
h_2(k) = m' - (k \mod m') \\
m' = m - 1
\end{aligned}
$$

The following hints will help you through the implementation.

(a) Define the number m that defines the size of the hash table. Set m to 7.

(b) The function `void init(int A[])` fills all slots of the hash table A qith the value 0.

(c) The hashing function `int h(int k, int i)` that receives the key k and the probe number i and returns the hashed key.

(d) The function `void insert(int A[], int key)` inserts the key into the hash table A.

(e) The function `int search(int A[], int key)` that returns -1 if the key was not found in the hash table A. Otherwise, it should return the index of the key in the hash table.

(f) The function `void printHash(int A[])` prints the table size and all non-empty slots of the hash table A accompanied with their index and the key.

Mathematical functions available in the library math.h can be used and don't need to be redefined. For testing purposes, use a table size of 7, add the values 1313, 1314, 1315, 2000, 2001 and 2002 and print your hash table. Search for the values 2000, 10, 1314, 1313 and 337 and print the results.

Task 4 [21 FS Final Exam]

Consider a hash table `HT` consisting of m slots, using open addressing **with linear probing and a step of size 1**. Write a C code for the function `int HTDelete(int k)`, which deletes key k from the hash table and also rearranges the other elements. The rearrangement leaves the table exactly as it would have been had key k never been inserted i.e., you cannot simply set the status as deleted.

The Hash table `HT` is defined as follows:

```
struct elem{
    int key;
    int status;      // 0:OCCUPIED, -1: EMPTY
};

struct elem HT[m];
```

The following example shows the state of a hash table after calling `HTDelete(11)`. The hash table uses linear probing with step of size 1 and a hash function $h(k) = k\%m$, with m=5.

Slot	Status	Key
0	-1	-1
1	0	11
2	0	22
3	0	31
4	0	2

After Deleting 11 ⟹

Slot	Status	Key
0	-1	-1
1	0	31
2	0	22
3	0	2
4	-1	-1

Table 1: Illustration of function call `HTDelete(11)` on the hash table.

You can use the following helper functions:

1. `int hash(int k,int i)` - return the hash slot for key k and step of size i.
2. `int HTInsert(int k)` - inserts key k in the hash table HT and return the slot number. Returns -1 if hash table is full.

Note: Initially all slots are empty and keys are initialized with -1. The function `HTDelete(int k)` returns the slot number of a deleted key k and -1 if key k does not exist in the hash table.