Assignment 1

C.Sriram Saran - EE18BTECH11007

Download all Codes from

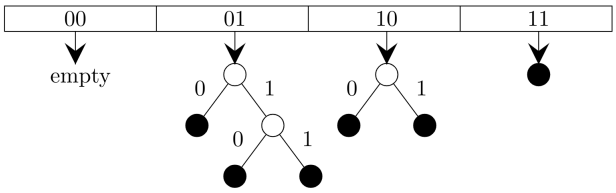
https://github.com/csrs2000/EE4013/blob/main/Assignment−1/codes

Download all latex-tikz codes from

https://github.com/csrs2000/EE4013/blob/main/Assignment−1/Assignment1.tex

1 Problem

Consider a dynamic hashing approach for 4-bit integer keys:
1)There is a main hash table of size 4.
2)The 2 least significant bits of a key is used to index into the main hash table.
3)Initially, the main hash table entries are empty.
4)Thereafter, when more keys are hashed into it, to resolve collisions, the set of all keys corresponding to a main hash table entry is organized as a binary tree that grows on demand.
5)First, the 3rd least significant bit is used to divide the keys into left and right subtrees.
6)To resolve more collisions, each node of the binary tree is further sub-divided into left and right subtrees based on the 4th least significant bit.
7)A split is done only if it is needed, i.e., only when there is a collision.
Consider the following state of the hash table.



Which of the following sequences of key insertions can cause the above state of the hash table (assume the keys are in decimal notation)?
(A) 5,9,4,13,10,7
(B) 9,5,10,6,7,1
(C) 10,9,6,7,5,13
(D) 9,5,13,6,10,14

1.1 Solution

Answer:Option-(C)
Explanation
By using above rules we get below insights-
$\implies$ Sequence given in option (A) is not possible, because of entry 4 (= 0100) whose 2 LSB are 00 but the 00 column is empty.
$\implies$ Sequence given in option (B) is not possible, because of entry 1 (=0001) and 9 (=1001) These have 3rd LSB collision but in the given table theres no collision in the 3rd LSB of 01 column.
$\implies$ for Sequence given in option (C) there is no key with 00 indexing,5 (=0101),13 (=1101) have 3rd LSB collision ,9 (=1001) has no collision over 3rd LSB so these 3 form 2nd column elements,6 (=0110),10 (=1010) form the elements of 3rd column,7 (=0111) forms element of the last column
$\therefore$ hence this is the correct option
$\implies$ Sequence given in option (D) is not possible, because of entry 6 (=0110), 10 (=1010), and 14 (=1110) in the third column,but the third column only has 2 entries.
Below C-Program implements the above described dynamic 4 bit hashing table

https://github.com/csrs2000/EE4013/blob/main/Assignment−1/codes/hashtable.c

2 Hash Table

2.1 *Hashing*

- Hashing is the process of mapping large amount of data item to smaller table with the help of hashing function.
- It is a technique to convert a range of key values into a range of indexes of an array.
- It is used to facilitate the next level searching method when compared with the linear or binary search.
- Hashing allows to update and retrieve any data entry in a constant time O(1).
- Constant time O(1) means the operation does not depend on the size of the data.

- Hashing is used with a database to enable items to be retrieved more quickly.

2.2 Hash Function

- A fixed process converts a key to a hash key is known as a Hash Function.
- This function takes a key and maps it to a value of a certain length which is called a Hash value or Hash.

2.3 Hash Table

- Hash table or hash map is a data structure used to store key-value pairs.
- It is a collection of items stored to make it easy to find them later.
- It uses a hash function to compute an index into an array of buckets or slots from which the desired value can be found.
- Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of the below hash table which is stored in an array of size 20,and the following items are to be stored in the (key,value) format.

Key	Hash	Array Index
2	2%20	2
1	1%20	1
42	42%20	2
14	14%20	14

- **Linear Probing**: As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing. In the below example we obtain new array indices after linear probing

Key	Hash	Array Index after linear probing
2	2%20	2
1	1%20	1
42	42%20	3
14	14%20	14

2.4 Operations in Hash Table

Following are the basic primary operations of a hash table.

- Search: Searches an element in a hash table.
- Insert: inserts an element in a hash table.
- delete: Deletes an element from a hash table.

Search Operation: Whenever an element is to be searched, compute the hash code of the key passed and locate the element using that hash code as index in the array. Use linear probing to get the element ahead if the element is not found at the computed hash code.

Insert Operation: Whenever an element is to be inserted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing for empty location, if an element is found at the computed hash code.

Delete Operation: Whenever an element is to be deleted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing to get the element ahead if an element is not found at the computed hash code. When found, store a dummy item there to keep the performance of the hash table intact.

below table shows the time complexity of hash table operations

Algorithm	Average case	Worst case
search	O(1)	O(N)
insert	O(1)	O(N)
delete	O(1)	O(N)

2.5 Application of Hash Table

Let us look at the example of phonebook to understand how hash table can be used

Using the Hash Table we can implement following functions of Phone directory. each record consists of ID, Name, Telephone number.

- **Create Record**:This method takes details from the user like Name and Telephone number and create new record in the hash table.
- **Display Record**:This method displays all the existing records in the phone book
- **Search Record**:This method takes the id of the record and prints the name and phone number in the book
- **update Record**:This method takes the ID of the record and updates the name and phone number with new values.
- **Delete Record**:This method takes the key of the record and deletes the record.

Approach

We create a hash table, For inserting records,deleting, searching, or updating an entity, the User ID is asked and on the basis of request,details are displayed or processed. If the record is not found, then an appropriate message is displayed.

Collision is the major problem in the hashing technique. When a collision occurs, the details are searched for an open or unoccupied element using linear probing. Steps for inserting entities in a hash table:

- If the location is empty, directly insert the entity.
- If mapped location is occupied then keep probing until an empty slot is found. Once an empty slot is found, insert the entity.

Example- lets consider a phonebook of size 4 which is shown below

ID	Array Index	Name	Phone no
1234	1234%4=2	A	9199999999
3122	3121%4=1	B	9199999911

create record: now for example lets say the phone book takes ID=1122,Name=C,Phone no=9381261459 as inputs for creating new record then the phonebook gets updated as follows,we can clearly see that to avoid collision linear probing allocates index no-3 which is the next available slot -

ID	Array Index	Name	Phone no
1234	1234%4=2	A	9199999999
3122	3121%4=1	B	9199999911
1122	1122%4=2 but index=3	C	9381261459

update record: now for example lets say the phone book takes ID=1122,Name=C,Phone no=9111111459 as inputs for updating existing record then the phonebook gets updated as follows-

ID	Array Index	Name	Phone no
1234	1234%4=2	A	9199999999
3122	3121%4=1	B	9199999911
1122	1122%4=2 but index=3	C	9111111459

Delete record: now for example lets say the phone book takes ID=3121,Name=B as inputs for deleting a record then the phonebook gets updated as follows

ID	Array Index	Name	Phone no
1234	1234%4=2	A	9199999999
1122	1122%4=2 but index=3	C	9381261459

Below C++ Program implements the phone book using hashing

https://github.com/csrs2000/EE4013/blob/main/Assignment−1/codes/phonebook.cpp