Data Structures and its Applications

V R BADRI PRASAD
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Hashing – Closed Hashing – Linear Probing
- Insert Operation
- Display Operation

V R BADRI PRASAD
Department of Computer Science & Engineering

Closed Hashing – Linear Probing

Hash Table	
Index	**Data**
0	15
1	46
2	72
3	18
4	34

- Consider key elements as 34, 46, 72, 15, 18
- The data is stored in the hash table as shown .

  - 34 mod 5 = 4, 34 is stored at index 4.
  - 46 mod 5 = 1, 46 is stored at index 1.
  - 72 mod 5 = 2 , 72 is stored at index 2.
  - 15 mod 5 = 0 , 15 is stored at index 0.
  - 18 mod 5 = 3 , 18 is stored at index 3.

Say if 57 is to be stored, then

Compute hash / index value : **57 % 5 = 2.**

However, it exceeds the capacity of the hash table. Hence it cannot be stored.

To overcome this problem, increase the size of the hash table.

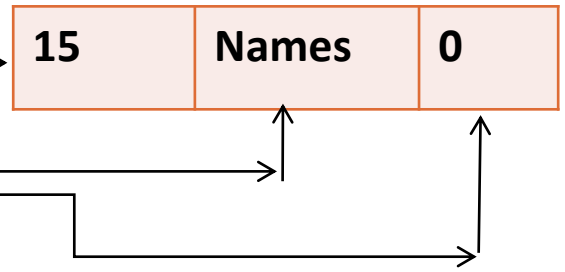
Data Structures and its Applications

Hashing : Linear Probing - Node Creation

The method is same as separate chaining for creation if the node.

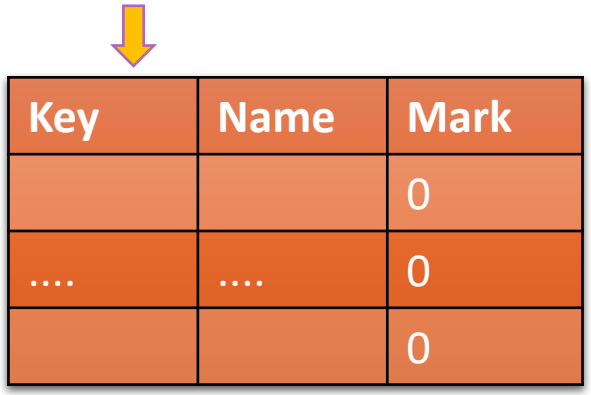
```
struct element
{
  int key;
  char name[100];
  int mark;
};
```

Sample node



Allocation of memory

hashtable = (struct element *)(malloc(tablesz* sizeof(struct element)));



Data Structures and its Applications

Hashing – Linear Probing : Insert Operation

Consider key elements as 34, 46, 72, 15, 18

- 34 mod 5 = 4, 34 is stored at index 4.
- 46 mod 5 = 1, 46 is stored at index 1

Key	Name	Mark
--	--	0
46	DEF	1
--	--	0
--	--	0
34	MNP	1

Let table size be 5.
if count == 5 then Table is Full. Cannot insert
The code is as follows:

```
void insert_to_hash(struct element *ht, int size, int key, char *name, int *count)
  {
     int index;
    if(size==*count)
   {
     printf("Table full.. cannot insert\n");
     return;
   }
```

Hashing – Linear Probing : Insert Operation

Consider key elements as 34, 46, 71, 15, 18

- 34 mod 5 = 4, 34 is stored at index 4.
- 46 mod 5 = 1, 46 is stored at index 1

Next number is 71.

Index is 71 % 5 = 1.

Since , index '1' is non empty location, search for first empty location in the sequence.

I.e., location with index value 1+1 =2.

Now, Location 2 is empty.

- Store the key value in that location.
- Mark location 2 as 1.

The code for the same is as follows:

Array Index	Key	Name	Mark
0	--	--	0
1	46	DEF	1
2	71	ABC	0
3	--	--	0
4	34	MNP	1

Hashing – Linear probing : Insert Operation

Find the index value using the statement

```
index=key % size;
```

Find the first empty location in the hash table and store the data sent from the main program

```
while(ht[index].mark !=0)
   index=(index+1)%size;

ht[index].key=key;
strcpy(ht[index].name,name);
ht[index].mark=1;
```

Increment count by 1

```
(*count)++;
return;
}
```

Key	Name	Mark
15	ABC	1
46	DEF	1
71	GHI	1
18	JKL	1
34	MNP	1

Other elements are stored in the memory as shown. Mark field is set to 1.

An element is to be removed from the hash table. How to delete..?

Hashing – Linear Probing – Deletion of an element

```
void delete_from_hash(struct element *ht,int size,int key, int *count)
 {
   int i,index;
   printf("count = %d\n",*count);

//  If Table is empty display table is empty.

   if(*count==0)
   {
    printf("table empty..\n");
    return;
    }

//  if Mark is '0', indicates the element is not present in the hash table

// Otherwise:
```

Key	Name	Mark
15	ABC	0
46	DEF	0
71	GHI	0
18	JKL	0
34	MNP	0

Data Structures and its Applications

Hashing – Linear Probing – Deletion of an element

```
//   Search for the element to be deleted

    index = key % size;
    i=0;
while(i<*count)
            {
                        if (ht[index].mark==1)
                        {  // indicates element is present
                                    if(ht[index].key==key) // if found
                                    { ht[index].mark=0;    // Delete
                                                (*count)--;
                                                return;
                                    }
                                    i++;  }
  index=(index+1)%size;          // search for the element in the
 }                                //consecutive memory location

 printf("key not found..");
 return;
 }
```

Key	Name	Mark
15	ABC	1
46	DEF	1
71	GHI	1
18	JKL	1
34	MNP	1

Data Structures and its Applications

Hashing – Linear Probing – Deletion of an element

// Search for the element to be deleted

Ex: Let's delete 71.
First, search for 71at location 71 % 5 = 1.

At memory location 1, check the value stored with the key value.

It is the same.
Now, Set the mark value to '0'.

This is deletion of an element from the hash table.

Key	Name	Mark
15	ABC	1
46	DEF	1
71	GHI	0
18	JKL	1
34	MNP	1

THANK YOU

V R BADRI PRASAD

Department of Computer Science & Engineering

badriprasad@pes.edu