DATA STRUCTURES AND ITS APPLICATIONS UE21CS252A

Kusuma K V
Department of Computer Science & Engineering

DATA STRUCTURES AND ITS APPLICATIONS

Hashing: Collision Resolution
Linear Probing

Kusuma K V
Department of Computer Science & Engineering

Hashing: Linear Probing

Linear Probing (open addressing, closed hashing) resolves collision by finding the next vacant spot in the hash table, in other words, it uses the below formula to resolve collision:

h(key) = (h(key) + i) % tableSize

where i = 1, 2, 3, …

Hashing: Linear Probing

Insert 7, 20, 41, 31, 18, 8, 9 into a hash table of size 7.

Use key%tableSize as the hash function & linear probing(closed hashing)to resolve collision.

h(7) = 7 % 7 = 0

h(20) = 20 % 7 = 6

h(41) = 41 % 7 = 6 collision,

0	1	2	3	4	5	6
7	41	8	31	18	9	20

Resolving collision: h(41)=((6+1)%7)=0, h(41)=((6+2)%7)=1

h(31) = 31 % 7 = 3

h(18) = 18 % 7 = 4

h(8) = 8 % 7 = 1 collision

Resolving collision: h(8)=((1+1)%7)=2

h(9) = 9 % 7 = 2 collision

Resolving collision: h(9)=((2+1)%7)=3, h(9)=((2+2)%7)=4, h(9)=((2+3)%7)=5

Hashing: Linear Probing

```
void initTable(NODE ht[],int *n)
{
        for(int i=0;i<SIZE;i++)
                ht[i].mark=-1;

        *n=0;
}
```

Hashing: Linear Probing

```
int insertRecord(NODE ht[],int rNo,char name[],int *n) {
        if(SIZE==*n)
                return 0;
        int index=rNo%SIZE;             //hash function
        while(ht[index].mark!=-1)
                index=(index+1)%SIZE;
        ht[index].rNo=rNo;
        strcpy(ht[index].name,name);
        ht[index].mark=1;
        (*n)++;
        return 1;
}
```

Hashing: Linear Probing

```
int deleteRecord(NODE ht[],int rNo,int *n){
   if(*n==0)
         return 0;

   int index=rNo%SIZE;       //hash function
   int i=0;
   while(ht[index].rNo!=rNo && i<*n)
   {
         index=(index+1)%SIZE;
         if(ht[index].mark==1)
                  i++;
   }
   if(ht[index].rNo==rNo && ht[index].mark==1)
   {
         ht[index].mark=-1;
         (*n)--;
         return 1;
   }
   return 0;
}
```

DATA STRUCTURES AND ITS APPLICATIONS

Hashing: Linear Probing

```
int searchRecord(NODE ht[],int rNo,int n){
    if(n==0)
            return 0;

    int index=rNo%SIZE;        //hash function
    int i=0;
    while(ht[index].rNo!=rNo && i<n)
    {
            index=(index+1)%SIZE;
            if(ht[index].mark==1)
                    i++;
    }
    if(ht[index].rNo==rNo && ht[index].mark==1)
    {
            strcpy(name,ht[index].name);
            return 1;
    }
    return 0;
}
```

THANK YOU

Kusuma K V

Department of Computer Science & Engineering

kusumakv@pes.edu