Dictionary n Hashing

date 2/27/2023

* Dictionary *

- list of pointers; elements are grouped by something common (i.e. last digit of num = index of array)

* operations:

1.) insert(x, A) - adds element x in set A, if $x \notin A$

2.) delete(x, A) - removes element x in set A, if $x \in A$

3.) member(x, A) - returns true (or non-zero) if $x \in A$
else returns false (zero) if $x \notin A$

4.) Initialize(A) - initializes set A to be empty.

5.) make Null - makes set A empty.

• Implementations:

1.) Linked List - O(N)

2.) Array - O(N)

3.) Cursor based - O(N)

4.) Hashing - O(1) improved ver.

* Hashing *

↳ uses a function called hash().

↳ assigns a "hash value" to an element.

↳ determines: a.) exact location of element (closed hashing)

b.) starting point in searching for the location of the element (open hashing)

* kinds of Hashing

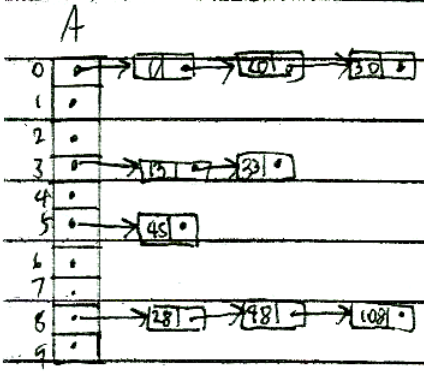
1.) OpenHashing (External) - allows set to be stored in potentially unlimited space
- array of Linked Lists

2.) Closed Hashing (Internal) - uses a fixed space for storage & thus limits the size of set
- Array

Open Hashing

date 2/27/2023

Set A = {0, 13, 20, 28, 30, 33, 45, 48, 108}



- Data structure:
  - > array of sets (or groups)
  - > each set can be array or linked list

NOTE: uses % modulo to get hash value

- Hash() function
  - → returns an INTEGER value = the SUBSET (group) in which the element is a member of.

- Example:
  1.) Group integer elements according to ones digit
  2.) Group names according to 1st letter of name.

- Exercise 1 → Hash func. which accepts an integer as parameter & returns the digit in the ones place

- Code:

```
int hash (int x){
  return (x % MAX);
}
```

1.b → group the hashed values

Ones digit	hash value
0, 1	→ 0
2, 3	→ 1
4, 5	→ 2
6, 7	→ 3
8, 9	→ 4

- Code:

```
int hash (int x){
  return (x % 10)/2;
}
```

Strathmore

date 2/27/2023

2) Hash function which accepts the last name as parameter & returns:

0 if the 1st letter is A

1 if the 1st letter is B

...

25 if the list letter is Z

•Code:

```
int hash(char name[]){
    return toupper(name[0]) - 'A';
}
```

•Assignment.

1.) The hash func. accepts as parameter an integer x & returns the digit in the hundreds place of x.

• Code:

```
int hash(int x){
  return (x/100) %10;
}
```

2.) the hash func. accepts as parameter an integer x & returns a hash value between 0 to 18. The hash value is the remainder when the sum of all the digits in x is divided by 19.

```
int hash(int x){
int hashVal;
for(hashVal=0; x!=0; x/=10){
    hashVal = hashVal + x %10;
}
return(hashVal %19);
}
```

Strathmore

date 2/27/2023

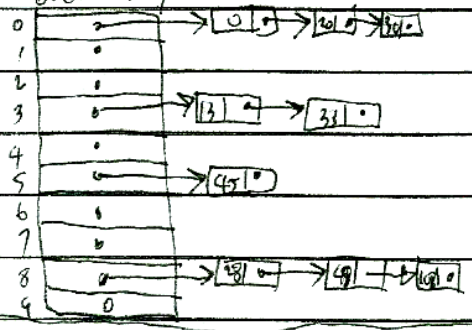
3.) The hash func. accepts as parameter a name & returns a hash value between 0 to 48. The hash value is the remainder when the sum of the ASCII values of all the letters in the name is divided by 49.

• Code:

```
int hash (char name[]){
    int retval, ndx, length = strlen(name);
    for(ndx= 0; ndx < length; ndx++){
        retval = retval + name[ndx];
    }
    return(retval % 49);
}
```

• Exercise #2:

Dictionary A



1.) Write an appropriate data type def.

```
#define SIZE 10
typedef struct node{ typedef LIST Dictionary[SIZE]
    int elem;
    struct node* link;
}*LIST;
```

// do initDictionary

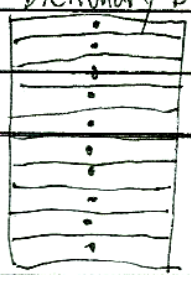
a.) func-header:

• Void initDictionary (Dictionary D)

b.) func-call:

```
• Dictionary D;
  InitDictionary(D)
```

c.) Exec Stack:



d.) Code:

```
void initDictionary (Dictionary D){
    int index;
    for (index=0; index< SIZE; index++){
        D[index] = NULL;
    }
}
```

date _____

```
//insert()
    void insertDict(Dictionary D, int x){
      int hashVal = hash(x);
      LIST temp = (LIST) malloc (size of(struct node));
      LIST *trav = &D[hashVal];

      for(; (*trav)!=NULL && (*trav)->data <=x; trav=&(*trav)->link){}
        temp->data = x;
        temp->link = *trav;
        *trav = temp;
    }

//delete
    void deleteDict(Dictionary D, int x){
      int hashVal = hash(x);
      LIST temp; LIST *trav;

      if(D[hashVal]!=NULL){
        for(trav=&D[hashVal]; (*trav)!=NULL & (*trav)->data!=x;
                                trav = &(*trav)->link){}
        temp = *trav;
        *trav = temp->link;
        free(temp);
      }
    }
```

date _____

// member()

```
int isMember (Dictionary D, int x){
  int hashVal = hash(x);
  LIST trav;

  for (trav = D[hashVal]; trav != NULL && trav->data != x;
                          trav = trav->link){}
  return (trav != NULL)? 1:0;
}
```

• Sum Notes:

• If the hash func. evenly distributes N elements to S subsets (or groups):

a.) How many on the average is the # of elements in each subset? - N/S

b.) Running time of operations insert, delete, & member? - O(N/S)

Staedtmore