Lesson-11-Dictionary Class Notes & Review

Dictionary:

In real world Dictionary having words (Keys) associated with its meaning and definition(values).

In computing, the operations we need to perform on a collection most often are searching for a given item, adding a new item, and deleting an item from the collection. It's maintained collection of unique keys associated with its values like real world dictionary. Example getting a student information using student Id. SSN is an ID for the person to get their information.

The advantages of this data structure are to perform Insertion, Deletion and Searching efficient and faster.

Slide – 5

1. Faculty can teach Multiple courses.

   Renuka = { FPP, MPP, MDP, DS, Algorithms}

2. Product ID with multiple reviews

3. Log file can have multiple entries.

   Log file: log files record the activities of a system, which helps in understanding the system's behavior, debugging issues, and maintaining security and compliance.

Example: Keys are log levels ('error', 'warning', 'info'), and each key is associated with an array of log messages.

```
logEntries = {
 // Key--> Values
  'error'→ ['Error message 1', 'Error message 2'],
  'warning'→['Warning message 1', 'Warning message 2'],
  'info'→ ['Info message 1', 'Info message 2']
};

// Adding another entry to the 'error' key
logEntries['error'].push('Error message 3');
```

logEntries is a Dictionary ADT here.

DLL Implementation of Log file & its performance

Insertion in the beginning and end: O(1)

Remove(k) -> O(n)

findKey(k) --> O(n)

Slide-6

Unordered Dictionary ADT Applications

1. Addressbook – Each entry in an address book has the key person associated with Address includes billing/shipping address, email, phone number
2. Credit card Authorization – Credit card number as key associated with the transaction details. Helps to authorize the transaction.
3. Mapping Host names: In networking, there is a need to map human-readable host names cs16.net hostname as key associated to value of numerical IP addresses 128.148.34.101

Slide-10

Hash table is an unordered Dictionary
Table Size N = 11

K,O	11, "Tom"										21, "Anne"
Ind	0	1	2	3	4	5	6	7	8	9	10

Formula = h(k) = Key % N

Insert(21, "Anne")

H(21) = 21 % 11 = 10, Store this entry at the index 10

Insert(11, "Tom")

H(11) = 11 % 11 = 0, Store this entry at the index 0

If it's an int you can do modulus, Key can be of any type. Convert your object type to int value by writing your own hash function to return int value.

Find(11)

H(k) = 11% 11= 0 --> return “Tom”

Insert(22,”Bruce”)
H(k) = 22 % 11 = 0

In this point, key 11 and Key 22 got the same index, it’s called collision. You can apply Collision Strategies. I am applying Separate chaining. Each index use Linked List. Now you can store key 11 and 22 in the same index using Linked list.

22 Bruce

K,O	11, “Tom”										21, “Anne”
Ind	0	1	2	3	4	5	6	7	8	9	10

Slide -14

Collisions occur when different elements are mapped to the same cell

Collision handling Strategies

1. Separate Chaining
2. Open Addressing
    a. Linear Probing
    b. Quadratic Probing
    c. Double Hashing

Linear Probing

If you got a collision in the index, keep find the next empty cell.
N = 11

K,O	11, “Tom”	22 Renuka									21, “Anne”
Ind	0	1	2	3	4	5	6	7	8	9	10

Insert(22,”Renuka”)
H(k) = 22 % 11 = 0(apply linear Probing)
If index 1 is free insert

Slide-19 – Find Algorithm using Linear Probing.

FindValue(22) k = 22

findItem(22)

I = h(k) = 22 % 11 = 0
P=0
X = I + p mod n – 0 + 0 mod 11 = 0
Item = arr[x] – 11, Tom
11 == 22 False
P = p+ 1 = 1

P=1
X = I + p mod n – 0 + 1 mod 11 = 1
Item = arr[1] – 22, Renuka
22 == 22 return Renuka

Remove(22)

K,O	11, "Tom"	"Availble"									21, "Anne"
Ind	0	1	2	3	4	5	6	7	8	9	10

Insert(33, "Bob")

33%11 = 0 insert 33 at index 1
Index 1 is available,

K,O	11, "Tom"	33									21, "Anne"
Ind	0	1	2	3	4	5	6	7	8	9	10

Remove(11)

K,O	Available	33									21, "Anne"
Ind	0	1	2	3	4	5	6	7	8	9	10

Slide-21

Quadratic probing Disadvantages

Primary clustering refers to the tendency for keys to hash to the same slot and remain there.

Secondary clustering happens when keys that hash to the same index, but don't have the same primary probe location, and end up with secondary positions after probing. When searching for a key or deleting a key, the search process must traverse these clusters to find the occupied slots. It is increasing the average search time. This can lead to longer response times and reduced efficiency, especially with larger hash tables.

Steps to follow

1. Find h(k) = key % N, where N = Table Size. If the slot is free insert the key in that available slot(Primary Hashing)
2. If the primary hashed index is occupied, find the next available slot by applying the given formula for performing Quadratic probing collision strategy.
    a. A[(i + j^2) mod N], for j = 1, 2, … until an empty slot is found.

Step by Step Execution

Insert keys 18, 41, 22, 44, 59, 32, 31, 73, in this order using Quadratic Probing.

Table Size N = 13
h(k) = Key % N

Secondary hashing
A[(i + j^2) mod N], j = 1,2,3,….

Key													
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

1. Insert(18)
    h(18) = 18 % 13 = 5, Slot is empty and insert the key at index 5

Key						18							
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

2. Insert(41)
    h(41) = 41 % 13 = 2, Slot is empty and insert the key at index 2

Key			41			18							
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

3. Insert(22)
    h(22) = 22 % 13 = 9, Slot is empty and insert the key at index 9

Key			41			18				22			
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

4. Insert(44)
    h(44) = 44 % 13 = 5, Slot is occupied at index 5. i = 5. Go for secondary hashing.

A[(i + j^2) mod N], j = 1
A[(5 + 1) mod 13] = A[6 % 13] = 6, Slot is empty and insert the key at index 6

Key			41			18	44			22			
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

5. Insert(59)
    h(59) = 59 % 13 = 7, Slot is empty and insert the key at index 7

Key			41			18	44	59		22			
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

6. Insert(32)
    h(32) = 32 % 13 = 6, Slot is occupied at index 6. i = 6. Go for secondary hashing.
    A[(i + j^2) mod N], j = 1
    A[(6+1) % 13] = 7 % 13 = 7, Slot is occupied at index 7. i = 6, j = 2
    A[(6+2^2) % 13] = 10 % 13 = 10

Key			41			18	44	59		22	32		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

7. Insert(31)
    h(31) = 31 % 13 = 5, Slot is occupied at index 5. i = 5. Go for secondary hashing.
    A[(i + j^2) mod N]
    A[(5 + 1) mod 13] = A[6 % 13] = 6, Slot is occupied at index 6. j=1
    A[(5 + 4) mod 13] = A[9 % 13] = 9, Slot is occupied at index 6. j=2
    A[(5 + 9) mod 13] = A[14 % 13] = 1, j=3,
    Slot is empty and insert the key at index 1

Key		31	41			18	44	59		22		32	
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

8. Insert(73)
    h(73) = 73 % 13 = 8, Slot is empty and insert the key at index 8

Key		31	41			18	44	59	73	22		32	73
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

Slide-22

Why q is prime in the probing?

Answer: A prime number is chosen to ensure a good spread of values and to minimize the number of repeated patterns. This helps in evenly distributing keys across the table.

Slide-23- Step by Step Execution

Steps to follow,

1. Find h(k) = key % N, where N = Table Size. If the slot is free insert the key in that available slot (Primary Hashing)
2. If the primary hashed index is occupied, find the next available slot by applying the given formula for performing double hashing collision strategy.

    a. Empty cell index = $(\boldsymbol{i} + \boldsymbol{j}*\boldsymbol{d}(\boldsymbol{k}))$ mod $\boldsymbol{N}$ ***[for j = 0, 1, … , N – 1]***

    b. The secondary hash function:
    $d(k) = q$ - $(k$ mod $q)$ where $q < N$ *and* q is a prime

Insert keys 18, 41, 22, 44, 59, 32, 31, 73, in this order using Double Hashing.
Table Size N = 13, Secondary hashing q = 7 (prime)
h(k) = Key % N

Key													
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

1. Insert(18)
    h(18) = 18 % 13 = 5, Slot is empty and insert the key at index 5

Key						18							
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

2. Insert(41)
   h(41) = 41 % 13 = 2, Slot is empty and insert the key at index 2

Key			41			18							
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

3. Insert(22)
   h(22) = 22 % 13 = 9, Slot is empty and insert the key at index 9

Key			41			18				22			
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

4. Insert(44)
   h(44) = 44 % 13 = 5, Slot is occupied at index 5. i = 5. Go for secondary hashing.
   $d(k) = q$ **-** $(k \bmod q)$, here k = 44, q = 7
   d(44) = 7 – (44 % 7) = 7 – 2 = 5. [Prob length]

   Empty cell index = (***i*** + ***j*d***(***k***)) mod ***N* , *j* = *1***
   = (5 + 1 * 5) mod 13 = 10 mod 13 = 10
   Index 10 is empty, Insert the key at index 10

The indexes processed are 5 and 10 are called probes.

Key			41			18				22	44		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

5. Insert(59)
   h(59) = 59 % 13 = 7, Slot is empty and insert the key at index 7

Key			41			18		59		22	44		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

6. Insert(32)
   h(32) = 32 % 13 = 6, Slot is empty and insert the key at index 6

Key			41			18	32	59		22	44		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

7. Insert(31)

h(31) = 31 % 13 = 5, Slot is occupied at index 5. i = 5. Go for secondary hashing.

d(*k*) = *q* - (*k* mod *q*), here k = 31, q = 7

d(31) = 7 – (31 % 7) = 7 – 3 = 4. [Prob length]

Empty cell index = (***i*** + ***j*d***(***k***)) mod ***N* , *j* = *1***

= (5 + 1 * 4) mod 13 = 9 mod 13 [Index 9 is Occupied]

Empty cell index = (***i*** + ***j*d***(***k***)) mod ***N* , *j* = *2***

= (5 + 2 * 4) mod 13 = 13 % 13 = 0

Index 0 is empty, Insert the key at index 0

The indexes processed are 5, 9, 0 are called probes.

Key	31		41			18	32	59		22	44		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

8. Insert(73)

h(73) = 73 % 13 = 8, Slot is empty and insert the key at index 8

Key	31		41			18	32	59	73	22	44		
Ind	0	1	2	3	4	5	6	7	8	9	10	11	12

Slide-26

- Load factor(α) is n/N where n is the number items in the table and N is the table size
- When the load factor goes above .75, the table is resized and the items are rehashed.
- **Rehashing:** A new hash table is created with a larger capacity. The size of the new table is often double the size of the old one. Each element in the original table is rehashed according to the new table size.
- Assuming that the hash values are like random numbers, it can be shown that the expected number of probes for an insertion with open addressing is

  1 / (1 - α)

Example:

Let's say we have a hash table with 10 slots and it currently holds 7 items. This means the load factor α is 0.7 (7 items / 10 slots).

To find the expected number of probes for the next insertion:

1 / (1 - 0.7) = 1 / (0.3) ≈ 3.33 probes in open addressing to find the next slot.

Good Hash Table

- Keep table size is greater than the input size(Eg: n=75, N=100)
- Keep N as Prime number.
- Maintain load factor to rehash.
- Handle collision strategy in your implementation.

Lesson-11- Review Questions

1. Different ways to organize the Dictionaries?
2. List out the operations on Dictionary ADT.
3. Describe about Log files.
4. How Hash table mechanism works? (Slide-10)
5. What is Load factor and why it is important? (Slide-15)
6. What is Rehashing? (Slide-15)
7. Able to know how various kinds of Collision Strategies works? Refer your homework.
8. What is Lookup Table and how it is implemented?