29/02/2024

PAGE No.
DATE / /

Day 3 OF DSA

* Tasks of Hashtaples-

checkbox -

- [x] Basic operations : Insertion, deletion, searching
- [x] Collision Handling : Chaining, separate Chaining.
- [x] Implement basic hash table operations : insertion, deletion, searching.
- [x] solve problems related to hash table collision Handling.

Theory Part

PAGE No.
DATE 11

* Hash table

A hash table is defined as a data structure used to insert, look up & remove key-value pairs quickly.

* Hashing :-

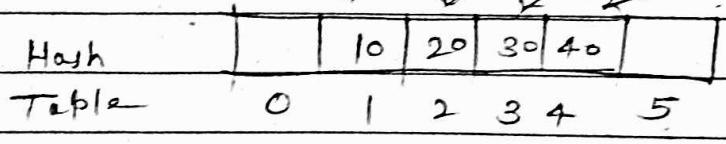
Hashing refers to the process of generating a fixed size output from an input of variable size using the mathematical formulas known as hash functions.

*

List = [10, 20, 30, 40]

H(x) = [x.%.10]

10%10 20%10 30%10 40%10

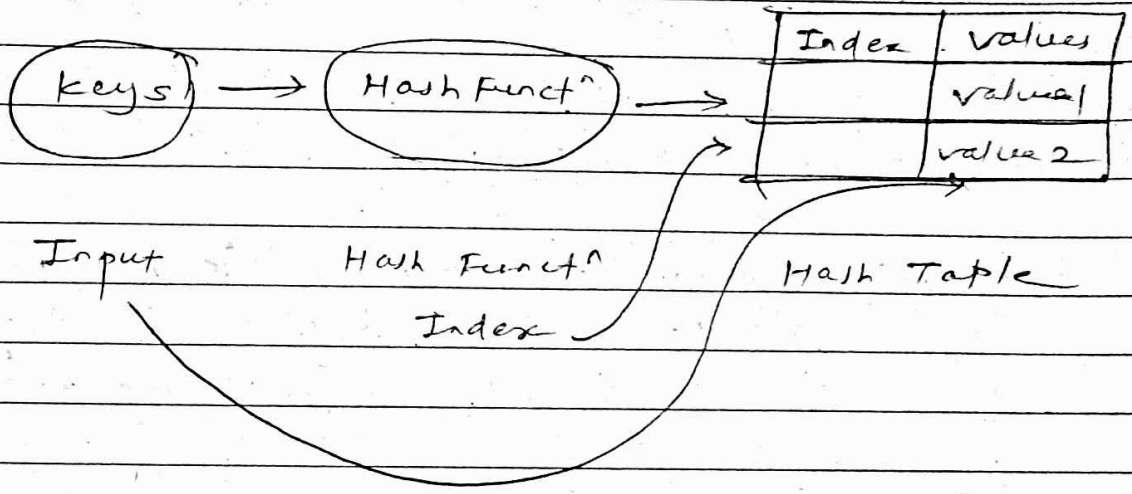


Time complexity = O(1)
For Insertion, Deletion & searching

PAGE No.
DATE / /

* Components of Hashing.

1) Key :-
2) Hash Function :-
3) Hash Table :-



* Collision :-

When we inserting new value with a desired index but there is already a value
Then we called that term
Collision

key = 10 → 10 % 5 = 0
key = 5 → 5 % 5 = 0

Index	value
0	10

* Advantages :-

1) key - value support
2) Fast data retrieval
3) Efficiency
4) Memory Usage

PAGE No.
DATE / /

* Hash Functions:-

A Hash Function is a Function that converts a given numeric or alphanumeric key to a small practical integer value

* Types of Hash Function

1) Division Method- $h(k) = k \bmod M$

Eg. $1276 \,\%\, 11 = 0$ $\quad k$ = key value

M = Hash Table size

2) Mid square = $h(k) = h(k * k)$

$h(60) = h(3600)$

$= 60$

3) Digit Folding Method: $S = k_1 + k_2 + k_3 \ldots$

Eg. $k = 12345$

$S = 12 + 34 + 5$

$S = 51$

4) Multiplication Method = $h(k) = \text{Floor}(M(kA \bmod 1))$

$k = 123$

$A = 0.15$

$M = 10$

$= \text{Floor}(10(123 \times 0.15 \bmod 1))$

$= (10(18.45 \,\%\, 1))$

$= (10(0.45))$

$4.5 = 4$

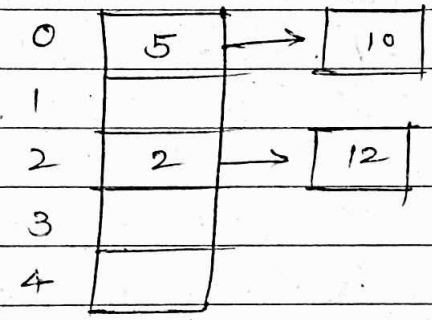
PAGE No. | DATE / /

* Handling collision :-

Separate Chaining (Open Hashing)

Open Addressing (Closed Hashing)
- Linear Probing
- Quadratic probing
- Double Hashing

1) Separate chaining = Using linked list



2) Open Addressing : We check for next empty location.

- Linear probing

3) - quadratic → For every hash funct^n we do this $(h(x) + i^2)$

- Double Hashing :-

$$h(k,i) = (h1(k) + i * h2(k)) \% n$$

i = non-Negative integer

k = key which is being hashed

n = hash table size

PAGE No.
DATE 1 1

* Load Factor :- Base value = 0.75

Load Factor = Total elements in hash table / size of hash table

* Rehashing :-

If load factor value increases by 0.75 then we double the size of Hash Table.

* unordered set :-

An unordered_set is an unordered associative container implemented using hash table

Time complexity = O(1)

Syntax :-

unordered_set <data_type> name;

Functs :-

1. size() & empty() For capacity
2. find () For searching
3. insert () & erase() For modificat°.

PAGE No.
DATE / /

* unordered_map :-

unordered_map is like a key value & mapped value combinat^n.

Time Complexity : O(1)

unordered_map <int, string> umap;

key value map name

Problem Part 1

PAGE No.
DATE / /

1) Count Distinct (Unique) elements in an array.

Input = {10, 20, 20, 10, 30, 10}

Output = to 3.

Approach :-

1) Take a int variable initialized with 0 & create unordered_set with name s.
2) Then traverse the map.
   if element is not present
   put it in hashtable & increment result
3) After ending of loop return result.

Time complexity : O(n) Space complexity : O(n)

PAGE No.
DATE / /

2) Counting Frequencies of Array elements :-

Input: arr[] = { 1, 1, 1, 2, 2, 3, 3 }

1	3
2	2
3	2

Approach :-

1) Initialized unordered map with 2 Integers

2) Iterate Loop & move every array ele in map

3) At the end
traverse loop.
print m.First & m.second

that's it.

Time complexity :	O(n)
Space complexity :	O(n)

PAGE No.
DATE / /

* Union of two unsorted Arrays.

Approach:-

1) Initialized map with 2 Integers

2) Insert the 1st array elements in map with index using 1st loop

3) Insert the 2nd array elements in map with index using 2nd loop

4) Using itr print first key of map & we get a union

Time complexity: O(m * log(m) + n * log(n))

Space complexity : O(m+n).

PAGE No.
DATE 1 1

* Intersection of two Arrays:-

Approach:-

1) Initialized with set.

2) Insert all elements of arr1 in set

3) Traverse in loop from 0 to 2nd Array's length & checks the arr2 element in ~~set~~ Set
if it is present then print that arr[i] element.

Time Complexity : O(nlogn)

Space Complexity : O(n)

PAGE No.
DATE / /

* Check if pair with given sum exists in Array (Two sum)

Approach:-

(1) Initialized unordered_set with 1 integer

(2) traversed array size
Take a temp variable
& store sum - arr[i]
mean for particular element
What is the value we needed that will be store in temp.

(3) After that we finding that temp in our set
if we get we return yes if not
then push into set.

Time complexity : O(N)
Space complexity : O(N)