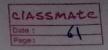
Assignment No. 6.



Title - linear probing with & without replacement.

Aim To leasn & implement the hashing with linear probing with & without replacement.

Problem Statement: Implement all the functions of dictionary
using hashing. Data: set of (key, value) paire.

keys are mapped to values. Key must be
comparable, keys must be unique. Standard
operation: insert (key), Find (key), Delete (key)

(use Lineur probing with & without replacement
calculate overage search rost for both.)

objective = i) To leash l'implement the hoshing data structure.

ii) To understand the use of hoshing l'advantage

of hoshing over tree

iii) To understand different hoshing functions with

different collision resolution strategies.

nenws palday ?

Theory :

Hoshing: Hoshing is an important data structure is designed to use a special function called hash function which is used to map a given value with a particuluse key for faster access of elements. The efficiency of hushing mapping depends on the efficiency of hush function used.

Let howh function H(x) maps the value at index X.10 in an array. For e.g. the values \$11,12,13,14,153 only be stored at position \$1,2,3,4,53 in hostile.

· Collision in hashing -

- · In hashing, hash function used to compute hash value to for a key.
- · Hash value is then used as an index to store the key.
- Hosh function may return some hosh value for two or more keys. It moons the collision occurs.
 - · A collision occur when two non-idential identifies
- · collision regulation strategy = 1

In linear probing when collision occars

we linearly search for next bucket we keep

probing until on empty bucket is found.

· With replacement -

In this we imeally seall be next built

hath function = h(x) = X1.5

0	10	10-1-5 = 0
1 /	15	15.15 = 0
2	20	201.5 = 0
3	40	401.5 = 0
4	14	141.5=4

- linear probing with replacement:

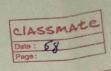
In this type of strategy, if collision occur then we linearly search for next available bucket a store the value in it. If hash value is calculated of the incorrect value is stored in the bucket of previous value is stored in appropriate bucket of previous value is stored in next available bucket.

eg. 13,23,20,14,10

	10	0	20	201.5=0	,	
		1	23	231.5=3		
		2	10	101.5= 0		
21		3	13	13.1.5=3		
		4	14	147.5=4		

	Operations:									
	i) Ingest (koy, value):									
	First hush value of key is calculated & then									
	data is inserted at that hash address									
	in hush table.									
	0 10									
	h(x) = x / 10 31									
	2 12 h(x) = 311.10 2 12									
	3 23 1 3 23									
	Have no engilles ti supplients to sout soft at									
	Be Fore After invertin									
	ingerting 31									
1	A his books a saley topos pring all bell 110									
	with the brooks and the little of the little									
11	Deletion =									
	Deletion of ky from hun table required									
	to colculate much address & value to be									
	deleted & making that bucket empty.									
	Out of the second of the secon									
	0 10									
	131 h(x) = x.1.10 1 31									
	2 (22) 22 (12)									
	3 43 = 2									
Refore doleting After deleting										
	22									
	22									

4-15-94	
	7 7 (01)
	if (hashTable [iv.n] -> key == key)
	ent it
	14+
	end for
	end if
	1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
	Step 4:) Keturn =
	step 4:) Return -1 step 5:) Stop
2)	Dalaling
	Deleting =
	closed the key to be deleted.
	classiff consch the key to be deleted & stok
	Step 1:) Read the key to be deleted. Step 2:) Search the key to be deleted & store its position, say k
3	Step 3.) If (KI = empty)
	Step 3.) If (k) = empty) make the high Table (k) of empty.
	else print "key not found"
	VIII.
	Step 4:) Stop.



	-	-		
-	1		1	
Can	1	110	IUN	
1 (01)	1	us	101	

i) hash tables are fuster in most coses but general problem with high tables there O(1) complexity is not quarenteed.

ii) for addition there is a point when table become full , then take need to be enlarged which

requires moving all of its elements.

iii) Hah put the operation & get the operation of time complexity O(1) with assumption that keyvalue pais are well distributed across the buckets.

iv) Time complexity:

Average case worst cal seasch (O(1) O(n)0(1) O(n)Ingest Delete o(1) 0(n)

Scanned with CamScanner