

DGAL

shrikrushna Zirape 21286

Assignment - 06 problem Stutement! limpliment Function of the dictionard using bashing and hundle Collision using chaining with without replacement. earning Objective! - impliment hashtable to impliment dictionary use chaining to reduce Complexity Learning Outcomer. - impliment a dictionary using hugh tuble analyze the result. heory. hashing is a technique or process of rappind keys, value into a hash tuble using husbfunction the efficiency of mapping depends on that of Fine hush Function used. seprate chaining is the idea to make euch cell of the hush table point to a linked list of second that same hush Function Value. · Advantage of Chaining - Eimple to Impliment. - Hash Tuble is dynamic.



PICT, PUNE	
	Disadvantage of chaining.
	1) Hastage of storage space as
	Some indices may never used. 2) Search Complexity will be O(n) for
	2) Search Complexity will be O(n) for
	longer inputs.
	Pseudo (de:
	Class Node
	& etajud kediousy
	meaning Nodec
	\$ /kpywood = "";
	meaning =" "
	next = Mull!
	E .
	Node (String 1, string m)
	& KET 10029 = K
	meaniny = m
	next = Null 1
	2
	Friend class HashTable;
	3
	class HashTable
	& Static Const int size = 23.
	Mode ph [size];
	in+ len=0.



# Algorithm hasfunction. (aring key) MY K = 0 For (int i = 0; i x sey). length(); i++) § K + = Key[i] & return K). size. Algorithm incert (string key) Algorithm incert (string key) String mean). Node & (= Hull; Gearch (key, Soc) If (t = Hull) & prin ("Element elready exist") return 3 Lent + in (index = hasfunction (key)) If (ht [index] = Hull) do funde & new node. ht [index] = new node. return glice else funde & temp = ht lindex]; while (teemp > next := Hull) funde & temp = ht lindex]; while (teemp > next := Hull)	For (inti=o; i x ded) length(); int) { For (inti=o; i x ded) length(); int) { K + = Key[i] { Redurn k isize. Algorithm insert (string key) For (key, 80c) If (ct = Hull; Gearch (key, 80c) If (ct = Hull) { Proto C' Element elrendy exist") return { In (index = has function (key)) If (ht [index] = new node (key, mean) ht [index] = new node. return elso elso elso elso in (leemp > next; Hull) function (leemp > next; Hull) function (leemp > next; Hull) function (leemp > next; Hull)	PICT, PUNE	
For (int i = 0 : i x arey length() int	For (inti=o; i x ded) length(); int) { For (inti=o; i x ded) length(); int) { K + = Key[i] { Redurn k isize. Algorithm insert (string key) For (key, 80c) If (ct = Hull; Gearch (key, 80c) If (ct = Hull) { Proto C' Element elrendy exist") return { In (index = has function (key)) If (ht [index] = new node (key, mean) ht [index] = new node. return elso elso elso elso in (leemp > next; Hull) function (leemp > next; Hull) function (leemp > next; Hull) function (leemp > next; Hull)		# Algorithm husturation. (otating key)
reduce temp > next; = null Elso Elso Algorithm insert (string key), String mean). Alde & C = Hull; Geardn (key, &oc) If (c = Hull) & print c" Element elready exist") return Elment = hasfunction (key) If (ht [index = hasfunction (key)) If (ht [index] = new node (key, mean)) ht [index] = new node.; return elso elso temp = temp > next; = Hull temp = temp > next;	Temp = temp > next; temp = temp > next; temp		1nx .K=0
reduce temp > next; = null Elso Elso Algorithm insert (string key), String mean). Alde & C = Hull; Geardn (key, &oc) If (c = Hull) & print c" Element elready exist") return Elment = hasfunction (key) If (ht [index = hasfunction (key)) If (ht [index] = new node (key, mean)) ht [index] = new node.; return elso elso temp = temp > next; = Hull temp = temp > next;	Temp = temp > next; temp = temp > next; temp		For Cinti=o i & expl. length() , itt) &
Algorithm insert (string key, String mean). Node & (= Hull; Search (key, boc) If ((t = Hull) & potal ("Element chready exist") return in (index = has function (key)) If (ht [index] = new node (key, mean) ht [index] = new node. return else else in the etemp > next = Hull in the etemp > next = Hull temp = temp temp = temp > next = Hull temp = temp temp = temp temp = temp temp = temp t	Algorithm insert (etring key, Stoing mean). Node & C = Hull; Securch (key, 80c) If (ct = Hull) & poth ("Element ctready exist") if (ht [index = hasfunction (key)) If (ht [index] = new node (key, mean) the hold & newhold = new node (key, mean) Algorithm insert = Hull) do Ende & newhold = new node (key, mean) Teturn Else Else Else Lade & temp > next = Hull Lade & temp > next		K+= Key[i] &
Temb = femb > vert; while (feemb > vert; myle (feemb > vert; mode (fe	temb = femb > vert; femb = femb > vert; print (feemb > vert; fent + inf(index = partunation (ken)) it (pt [index] = hall qo fundex bearde = vero uppe (ken) it (pt [index] = vero uppe (ken) fent + inf(index = partunation (ken)) fent = partunation (ken) fent = partunation (ken) fent = partunation (ken) fent = partunation (ken) temp = femb > vert; t		redurn K >. 5/20.
femb = femb > vert; while (feemb > vert; mylie (feemb > vert; mode (feel) fent + inflinder = free vorde (feel) fent + inflinder = free vorde; fent + gent - fement ferende (feel) gent - fement ferende (feel) fent - femb > vert; mean) fent - femb - f	temb = femb > vert; femb = femb > vert; print (feemb > vert; fent + inf(index = partunation (ken)) it (pt [index] = hall qo fundex bearde = vero uppe (ken) it (pt [index] = vero uppe (ken) fent + inf(index = partunation (ken)) fent = partunation (ken) fent = partunation (ken) fent = partunation (ken) fent = partunation (ken) temp = femb > vert; t		Algorithm insert (string kell
Else Else Else Else Fello	Temp = temp > next; temp = te		stoing mean).
Securch (key, 80c) if ((t= +un)) & proto ("Element Chready exist") return ind (index = hasfunction (key)) if (ht[index] = +uni) do funde x new mode = new node (key, mean) return g else else while (teemp > next '= +uni) temp = temp > next '= +uni)	Search (key, 80c) If ((t= +uu)) & poth ("Element chready exist") return ? !ent+ in(index = hasfunction (key)) if (htlindex] = +uull do f Hode & newhode = new node (key, mean) htlindex] = new node. return ? else shile (teemp > next := +uull) Entry = +emp > next := +uull		
Securch (key, 80c) if ((t= +un)) & proto ("Element Chready exist") return ind (index = hasfunction (key)) if (ht[index] = +uni) do funde x new mode = new node (key, mean) return g else else while (teemp > next '= +uni) temp = temp > next '= +uni)	Search (key, 80c) If ((t= +uu)) & poth ("Element chready exist") return ? !ent+ in(index = hasfunction (key)) if (htlindex] = +uull do f Hode & newhode = new node (key, mean) htlindex] = new node. return ? else shile (teemp > next := +uull) Entry = +emp > next := +uull		Mode to C = Hull.
Temp = temp > next; temp = temp > next; temp = temp > next;	temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next;		Search (key, ≻)
temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next; temp = temp > next;	Else \$ Continued to the property of the p		IF(C+=+un) >
Jent+ int(index = hasfunction (key)) if (ht[index] == hull) do int(index = hasfunction (key)) felse else abile (teemp > next := hull) temp = temp > next := hull) temp = temp > next := hull)	Lent + lent + int(index = hasfunction (key)) if (ht[index] == hull) do ht[index] = new node(key, mean) return else tade & temp = ht lindex] while (teemp > next = hull) it (ht[index] = hull) int(index = hasfunction (key)) int(index = hasfunction (key)		before Element Element on Star
lent+ in(index = hasfunction (key)) if (ht[index] == hull) do § Hode × newHode = new node (key, mean) ht [index] = new node. return g else shile (teemp > next := hull) temp = temp > next := hull)	lent + in (index = hasfunction (key)) if (ht[index] == hull) do § Hode & newhode = new node (key, mean) ht [index] = new node. return else in (index = hasfunction (key)) telse in (index = hasfunction (key)) telse in (index = hasfunction (key)) return in (index = hasfunction (key)) temp = hasfunction (key) in (index = hasfunction (key)) temp = hasfunction (key) in (index = hasfunction (key)) temp = hasfunction (key) in (index = hasfunction (key)) in (inde		(searl)
int(index = hasfunction (key)) if (ht[index] == Hull] do { Hode x new Hode = new node (key , mean) return else while (teemp > next := Hull) Ende x temp = ht lind ex]; while (teemp > next := Hull)	int(index = hasfunction (key)) if (ht[index] == Hull) do f Hode & new Hode = new node (key , mean) return else time & temp > next := Hull while (teemp > next := Hull temp = temp > next := Hull		3
Else Femb = femb > vert; Mode & femb > vert; Mode & femb = pt ling ex]; Mode & temb = pt ling ex]; Mode & temb = pt ling ex]; Else Femb = femb > vert; = mail Femb = femb > vert;	Else Findex = temp > next; while (teemp > next; temp = temp > next; temp = temp > next;		
Else Femb = femb > vert; Mode & femb > vert; Mode & femb = pt ling ex]; Mode & temb = pt ling ex]; Mode & temb = pt ling ex]; Else Femb = femb > vert; = mail Femb = femb > vert;	Else Findex = temp > next; while (teemp > next; temp = temp > next; temp = temp > next;		int(index = hasfunction(kou))
else # Hode & temp > next; while (teemp > next; - mull) \$ temp = temp > next;	else Else While (teemp > next; - Hull) Temp = temp > next; Temp = temp > next;		1+ (D1) index (== HIII) 10
else Ande se temp > next; while (teemp > next; = mull) temp = temp > next;	else Ande & temp = ht lindex]; while (teemp > next; = hull) temp = temp > next;		2 Hode & Dew Hode = Dein andolling
else Mode & temp = ht lindex]; while (teemp > next := hull) temp = temp > next;	else Else Made & temp = ht lindex]; while (teemp > next := hull) temp = temp > next;		ht lindex] - new node: ' mean
else Mode & temp = ht lindex]; while (teemp > next; = Hull) temp = temp > next;	else Mode & temp > next; while (teemp > next; = Hull) Elemp = temp > next;		LEFOL V
Hade & temp = ht lind ex]; while (teemp > next := Hull) { temp = temp > next;	Hade & temp = ht lind ex]; while (teemp > next := Hull) { temp = temp > next;		3
Hade & temp = ht lind ex]; while (teemp > next := Hull) { temp = temp > next;	Hade & temp = ht lindex]; while (teemp > next; = Hull) { temp = temp > next;		
femb = femb > next;	femb = femb > next;		else
femb = femb > next;	femb = femb > next;		5
femb = femb > next;	femb = femb > next;		Hade & temp = ht linder ?
temp = temp > next.	temp = temp > next.		while (teemp > next = Mull)
temp = temp > next;	femp = temp > next;		
5	5		temp = temp > next;
			9



PICT, PUNE	
	temp > next = new Mode (key, mean);
	2 .
	Algorithm Gearch (key, Noder & Current)
	O q
	int (omp=0
	index = hash function (key);
	Hode & temp - ht [index];
	while (temp != Hall) do
	5
	is (temp > keyword) == key
	٩ '
	corrent = femb;
	point (" Element Found");
	seran
	39
	Comp++
	temp = temp > next.
	3
	\$
	Algorith delete Mode (string key)
	Mode to C=Hull.
	seurch (key, &nc);
	12 (C== HAII) 90 3
	paint (" Element not found ")
	return ?
-	



PICT, PUNE	
	Node temp = ht[index]
	if (ht [index] > Keywood == key)do
	§ ht[index] = temp → hext;
	delete(temp);
	return;
	2
	while (temp > next > key wood! = key)
	4
	temp = temp > next;
	3
	Noder n=temp >next;
	temp > next = temp> next > next;
	delete(n)
	return
	3
	Algorithm display()
	For (i = 0 to i = 6 i ze-1) do
	3
	if (Y+Li] i = MAID GD
	5
	Hoder temp= ht[i]
	print ("Entoies with index) 3];
	while (temp 1= Hull)
	& point (temp> kgrood and temp> next!)
	temp: temp> next;
	3 5



PICT, PUNE		
	eles & point ("No entry For" i)	1
	433	,
	Test Cases!	
	input. Res output	Resul 1
\	1600+ >	
	day: animal 9 cat animal	
	cost animal 10 mango Frent	PUSS .
	rose! Flower It doy animal	
	manyo!fruit.	
	delete>	
	rose "doleted"	
	mouse. "not found"	
	Conclusion! -	
	Thus He have implimented	
	dictionary using Heishtable	
	in control oucescrally.	