Hashing is a process to speed up searching.

Data:

7

10

13

19

14

20

33

44

15

//stores information into a hashtable

//sample table size is 5

|  |
| --- |
|  |
|  |
|  |
|  |
|  |

H[0]🡪[10 15]->[10 20]->[10|/]

H[1]🡪[/]

i=data%5 H[2]🡪[7|/]

H[3]🡪[13]->[13|/]

H[4]🡪[19]

class HASH

{

private:

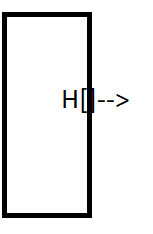
//we have a linked list so we need declaration for each node

struct node

{

int info;

node \*next;

 };//end struct node

node \*H[5]

public:

void clearHashTable()

{

for(int i=0; i<5; ++i)

{

H[i]=NULL;

}//endfor

}//end clearHashTable

int hashFx(int x) //hash function

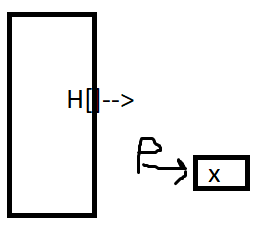
{return x%5;}//end hashFx

void push(int x)

{

//find te insertion place first so call hashfunction

int i = hashFx(x);

 node \*p = new node; // p->[x] refer to image

p->info = x;

~~p->next=NULL;~~

//insert this new node in front of H[i];

p->next=H[i]; H[i]=p;

}//end push

void search(int x)

{

//send it to hash function bec we searching if it belongs

//to H[0], H[1] etc. . .

int i = hashFunc(x);

//don’t want to lose your head node so create a copy

node \*p = H[i];

while(p!=NULL) //in case value is not there. visit all nodes

{

if(p->info==x){break;}

p=p->next;

}//endwhile

if(p==NULL)

{cout<<x<<”Does not exist\n”;}//endif

else

{cout<<x<<”is found\n”;}//endelse

}//end search

void displayH()

{

//node \*p;? ? ?

for(int i =0; i < 5; ++i)

{

p=H[i];

cout<<”H[“<<i<<”]->”;

while(p!=NULL)

{

cout<<p->info<<”->”;

p=p->next;

}//endwhile

cout<<”NULL\n”;

//output will be like

/\*

H[0]->150>20->10->NULL

H[1]->NULL

H[2]->7->NULL

H[3]->33->13->NULL;

H[4]->44->14->19->NULL;

\*/

}//endfor

}//end displayH

}//end clash HASH

//sample in main

int main()

{

int a[9]={7, 10, 13, 19, 14, 20, 33, 44, 15};

HASH h;//in this object we have hashtable of size 5

h.clearHashTable(); //set all pointers to NULL

for(int i=0; i<9; ++i)

{

h.pushH(a[i]);

}//endfor

h.displayH();//display out hash table

//-----we want to search an item------

int item;

cout<<”Enter the value of item you want to search: “;

cin>>item;

h.search(item);

}//end main

/\*to make like a hash table to have id and balance you’ll hav eto add those to your struct

sample

H[i]->[Bob|200.25|(next)] \*/

//the value of i comes from the hash function, if u give him a number it will divide it by the size of the hash table to give it back but since Bob is not a number

Data

Bob

Tom

Mary

John

Bill

Tim

..etc

//how to make searching faster

//we do it alphabetically

//use the asci of the first character

name=”Bob”

i=int(name[i])%26; //is 66 because int asci of ‘B’ is 66

//or the asci of the first two chars to make the hashtable smaller

i = int(name[i]) + int(name[i]) % 10

//this is if u don’t want to use just first letter, we can grab the 2 first letters and we can change the modulator to anything not 26.. 10, or 7.. whatever

//-----------what if we have social security 111-22-3333----------

//same concept, up to you is to look digits

//you just want to distribute data evenly as much as possible

RADIX SORT

ex: sort the ff data:

235

100

55

166

698

700

345

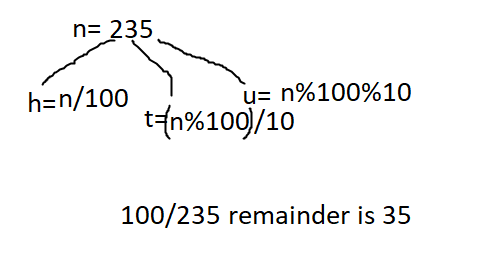
567

9

88

200

//how to extract 235

//n=235

extract the number based on the digit

unit digit

0[100-700-200

1[

2[

3[

4[

5[ ->235-55-345

6[166

7[567

8[88

9[9

235

100

55

166

698

700

345

567

9

88

200

0[100-700-200

1[

2[

3[

4[

5[ ->235-55-345

6[166

7[567

8[88

9[9

100

700

200

235

55

345

166

88

9

list data

0[->100->700->200->9

1[

2[

3[->235

4[->345

5[ ->55

6[->166->567

7[

8[->88

9[

100

700

200

235

55

345

166

88

9

distribute on the 10th digit

0[->9->55

1[->100->166

2[->200->235

3->345

4

5[->567

6

7[->700

8

9

0[->100->700->200->9

1[

2[

3[->235

4[->345

5[ ->55

6[->166->567

7[

8[->88

9[

list data

distribution based

----------🡪

hundred digit

100

700

200

235

55

345

166

88

9

0[->9->55

1[->100->166

2[->200->235

3->345

4

5[->567

6

7[->700

8

9

list data --🡪

9

55

88

100

166

200

235

345

567

700