**DESIGN & ANALYSIS OF ALGORITHMS**

**CUCKOO HASHING ALGORITHM**

By :

21pc17 – Jyothish.K.S

21pc19 – Nanda Pranesh.S

21pc25 – Varun.S

Definition

It is a type of hashing where two hash functions h1 and h2 are used such that each of the key can hash two possible locations.

Abstract:

This structure uses 2 hash tables along with 2 hash functions defined on them .Each function indexes into a single table. Further improvements can be done to this so that insertion and look up takes constant time , O(1) in worst case when provided with strong hash functions.

Source Code:

//DESIGN & ANALYSIS OF ALGORITHMS

//PACKAGE BY

//21PC17 - JYOTHISH.K.S

//21PC19 - NANDA PRANESH.S

//21PC25 - VARUN.S

//Cuckoo Hashing is one of the Hashing algorithms

//which takes o(1) time for insertion , searching & deletion

//which might depend on the hash function used

#include<bits/stdc++.h>

#define TNO 2

#define MOD 6

using namespace std;

//Defining a 2D matrix format for creating the 2 hash tables

//here in your program we restrict ourselves to creating 2 hash tables

//no.of hash tables can be increased to improve performance

//TNO defines the no.of hash tables

int cuckooTable[TNO][MOD];

int POSITION[TNO];

//This function is used to create/fill in the values into the hash tables present

void fillTable()

{

for(int j = 0;j<MOD;j++)

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

cuckooTable[i][j] = INT\_MIN;

}

//This function is used to print the Hash tables

void printTable()

{

cout<<"Hash Tables are : \n"<<endl;

for(int i =0;i<TNO;i++,printf("\n"))

{

int k = i+1;

cout<<"Table "<<k<<"-> ";

for(int j =0;j<MOD;j++)

{

if(cuckooTable[i][j]==INT\_MIN)

cout<<" N ";

else

cout<<" "<<cuckooTable[i][j];

}

}

cout<<endl;

}

//This function to generate the hash formula

int getHashValue(int function, int key)

{

switch (function)

{

case 1:

return key%MOD;

case 2:

return (key/MOD)%MOD;

}

}

//This function is used for placing the values in their respective places

void getArrange(int key,int id,int c,int n)

{

if(c==n)

{

cout<<key<<"do not have a position\n"<<endl;

//if cycle is present then we need to rehash

return;

}

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

{

POSITION[i] = getHashValue(i+1,key);

if(cuckooTable[i][POSITION[i]]==key)

return;

}

if(cuckooTable[id][POSITION[id]]!=INT\_MIN)

{

int dis = cuckooTable[id][POSITION[id]];

cuckooTable[id][POSITION[id]]=key;

getArrange(dis,(id+1)%TNO,c+1,n);

}

else

cuckooTable[id][POSITION[id]]=key;

}

//This function is used to arrange the values in the hash table

void cuckooFunction(int keys[],int n)

{

fillTable();

for(int i =0,c=0;i<n;i++,c=0)

getArrange(keys[i],0,c,n);

//calling the function to print the hash tables

printTable();

}

//Main Function

int main()

{

cout<<"\n\n\n\t\t\t\*\*\* CUCKOO HASHING ALGORITHM \*\*\*"<<endl;

cout<<"\t\t\t ------ ------- --------- \n"<<endl;

int n,m;

cout<<"Enter the Size(n) of 1st Input : \n"<<endl;

cin>>n;

int keyTable1[n];

cout<<"Enter the Elements : \n"<<endl;

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

{

cin>>keyTable1[i];

}

cout<<"Successfully Entered the 1st Input!"<<endl;

cuckooFunction(keyTable1,n);

cout<<"Enter the Size(m) of the 2nd Input : \n"<<endl;

cin>>m;

int keyTable2[m];

cout<<"Enter the Elements : \n"<<endl;

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

{

cin>>keyTable2[i];

}

cout<<"Successfully Entered the 2nd Input!"<<endl;

cout<<"Hashing Begins"<<endl;

cuckooFunction(keyTable2,m);

return 0;

}

Time Complexity:

* Search & Deletion take O(1) worst case.
* Insertion is amortized with O(1) expected time.

Applications:

* Used to solve scheduling problems
* Password checking
* Used to solve design optimization problems

Improvements:

* Dynamic Resizing
* Stashing

Limits:

* Few common hash functions perform poorly
* Rehashing may be expensive

