Linear Collision

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define size 10

struct DataItem

{

int data;

int key;

};

struct DataItem ht[size];

int hashcode(int val, int i)

{

int h1 = val % size;

int h = (h1 + i) % size;

return h;

}

int found;

void insertval();

void searchval();

void deleteval();

void display();

int main()

{

int i;

printf("Initially \n");

for (i = 1; i<=size; i++)

ht[i].key = i;

for (i = 0; i < size; i++)

ht[i].data = -1;

display();

insertval();

searchval();

deleteval();

return 0;

}

void display()

{

int i;

printf("Value \t Key \n");

for (i = 0; i < size; i++)

printf("%d \t %d \n", ht[i].data, ht[i].key);

}

void insertval()

{

int i, j, temp, index,x;

printf("How many elements \n");

scanf("%d",&x);

printf("Enter elem \n");

for (j = 0; j < x; j++)

{

scanf("%d", &temp);

index = hashcode(temp, 0);

if (ht[index].data == -1) // no Collision

{

ht[index].data = temp;

ht[index].key = index;

}

else // Collision -Probing starts

{

for (i = 1; i < size; i++)

{

index = hashcode(temp, i);

if (ht[index].data == -1)

{

ht[index].data = temp;

ht[index].key = index;

break;

}

}

}

}

printf("\n");

display();

}

void searchval()

{

int val,flag,index,i,k;

printf("Enlem to search \n");

scanf("%d",&val);

index=hashcode(val, 0);

if(ht[index].data==val) // Found for probe i=0

{

flag=1;

k=ht[index].key;

}

else //try for probes with i=1,2,3...

{

for (i = 1; i < size; i++)

{

index = hashcode(val, i);

if (ht[index].data == val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==0) //Sequential

{

for(i=0;i<size;i++)

{

if(ht[i].data==val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==1) //search from start to end

{

printf("Item found at pos %d \n",k);

found=1;

}

else

{

printf("Not found \n");

found =0;

}

}

void deleteval()

{

int val,flag,index,i,k;

printf("Enlem to delete \n");

scanf("%d",&val);

index=hashcode(val, 0);

if(ht[index].data==val) // Found for probe i=0

{

flag=1;

k=ht[index].key;

}

else //try for probes with i=1,2,3...

{

for (i = 1; i < size; i++)

{

index = hashcode(val, i);

if (ht[index].data == val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==0) //Sequential

{

for(i=0;i<size;i++)

{

if(ht[i].data==val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==1) //search from start to end

{

//printf("Item found at pos %d",k);

found=1;

}

else

{

// printf("Not found");

found =0;

}

if(found==1)

{

printf(" \n Deleted %d from loc %d",ht[k].data,ht[k].key);

ht[k].data=-1;

}

}

Quadratic Collision

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#define size 10

int c1=1,c2=3;

struct DataItem

{

int data;

int key;

};

struct DataItem ht[size];

int hashcode(int val, int i)

{

int h1 = val % size;

int h = (h1 +(c1\*i)+(c2\*i\*i))% size;

return h;

}

int found;

void insertval();

void searchval();

void deleteval();

void display();

int main()

{

int i;

printf("Initially \n");

for (i = 1; i<=size; i++)

ht[i].key = i;

for (i = 0; i < size; i++)

ht[i].data = -1;

display();

insertval();

//searchval();

//deleteval();

return 0;

}

void display()

{

int i;

printf("Value \t Key \n");

for (i = 0; i < size; i++)

printf("%d \t %d \n", ht[i].data, ht[i].key);

}

void insertval()

{

int i, j, temp, index,x;

printf("How many elements \n");

scanf("%d",&x);

printf("Enter elem \n");

for (j = 0; j < x; j++)

{

scanf("%d", &temp);

/\*index = hashcode(temp, 0);

if (ht[index].data == -1) // no Collision

{

ht[index].data = temp;

ht[index].key = index;

}\*/

// Collision -Probing starts

{

for (i = 0; i < size; i++)

{

index = hashcode(temp, i);

if (ht[index].data == -1)

{

ht[index].data = temp;

ht[index].key = index;

break;

}

}

}

}

printf("\n");

display();

}

void searchval()

{

int val,flag,index,i,k;

printf("Enlem to search");

scanf("%d",&val);

index=hashcode(val, 0);

if(ht[index].data==val) // Found for probe i=0

{

flag=1;

k=ht[index].key;

}

else //try for probes with i=1,2,3...

{

for (i = 1; i < size; i++)

{

index = hashcode(val, i);

if (ht[index].data == val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==0) //Sequential

{

for(i=0;i<size;i++)

{

if(ht[i].data==val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==1) //search from start to end

{

printf("Item found at pos %d",k);

found=1;

}

else

{

printf("Not found");

found =0;

}

}

void deleteval()

{

int val,flag,index,i,k;

printf("Enlem to delete");

scanf("%d",&val);

index=hashcode(val, 0);

if(ht[index].data==val) // Found for probe i=0

{

flag=1;

k=ht[index].key;

}

else //try for probes with i=1,2,3...

{

for (i = 1; i < size; i++)

{

index = hashcode(val, i);

if (ht[index].data == val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==0) //Sequential

{

for(i=0;i<size;i++)

{

if(ht[i].data==val)

{

flag=1;

k=ht[index].key;

break;

}

}

}

if(flag==1) //search from start to end

{

//printf("Item found at pos %d",k);

found=1;

}

else

{

// printf("Not found");

found =0;

}

if(found==1)

{

printf("Deleted %d from loc %d",ht[k].data,ht[k].key);

ht[k].data=-1;

}

}