Kurs Adı: Algoritma Analizi

Öğrenci Adı ve Soyadı: Arda Kaşıkçı

Proje Konusu: Hashing

Yöntem:

Verilen değerlere göre hash ve cache yapılarını tanımladım. Ardından verilen dosyadan satır satır kişiye ait bilgileri alıp, her kişi için insert fonksiyonunu çalıştırdım. İnsert fonksiyonunda öncelikle verilen kimlik numarasına horner metodunu uyguladım. Horner metodundan gelen değerin hash boyutuna göre modunu alıp key değerine ulaştım. Sonrasında double hashing için artış değerini hesapladım. Bulduğum key değeri üzerinden hash'te gezerek alınan kişi cache'te var mı diye kontrol sağladım. Kişi zaten cache'te var ise kişi bilgilerini yazdırıp, kişiyi cache'te başa alıp hashteki sıra değerini güncelledim. Kişi cache'te yok ise, cache'in doluluk durumuna göre kişiyi cache'te başa ekleyip hash'te uygun adrese yerleştirdim ardından hash'te flag değerini=1 yaptım ve sıra bilgilerini güncelledim. Hash tablosunda bir satır boş ise flag=0 , dolu ise flag=1, delete ise flag=2 olarak kodladım.

Zaman ve Yer Karmaşıklıkları:

Cache belleğe ekleme karmaşıklığı: O(1)
Cache bellekten silme karmaşıklığı: O(N)
Cache bellekte arama karmaşıklığı: O(1)
Cache bellek yer karmaşıklığı: O(N)

Uygulama:

Cache Size=4

Hash Size=7 için

```
Cache uzunlugunu giriniz.
4
Hash Tablosu Uzunlugunu giriniz.
7
Kisi KimlikNo: 43321 Isim: ZUBEYDE HARMANBASI Dogum Yili: 2001 Adres: izmir
Kisi KimlikNo: 33445 Isim: ACELYA SENLIK Dogum Yili: 1990 Adres: adana
```

Girilen değerler için Hash Tablosundaki değişimler:

```
HASH TABLOSU
0. BOS
1. BOS
2. BOS
3. BOS
4. BOS
5. BOS
6. KimlikNo: 12345 Sira: 0
HASH TABLOSU
0. BOS
1. BOS
2. BOS
3. KimlikNo: 32145 Sira: 0
4. BOS
5. BOS
6. KimlikNo: 12345 Sira: 1
HASH TABLOSU
0. BOS
1. BOS
2. BOS
3. KimlikNo: 32145 Sira: 1
4. KimlikNo: 43213 Sira: 0
5. BOS
6. KimlikNo: 12345 Sira: 2
HASH TABLOSU
0. BOS
1. BOS
2. KimlikNo: 45543 Sira: 0
3. KimlikNo: 32145 Sira: 2
4. KimlikNo: 43213 Sira: 1
5. BOS
6. KimlikNo: 12345 Sira: 3
HASH TABLOSU
0. KimlikNo: 43321 Sira: 0
1. BOS
2. KimlikNo: 45543 Sira: 1
3. KimlikNo: 32145 Sira: 3
4. KimlikNo: 43213 Sira: 2
5. BOS
6. BOS
HASH TABLOSU
0. KimlikNo: 43321 Sira: 1
1. BOS
2. KimlikNo: 45543 Sira: 2
3. BOS
4. KimlikNo: 43213 Sira: 3
5. BOS
6. KimlikNo: 54213 Sira: 0
Kisi KimlikNo: 43321 Isim: ZUBEYDE HARMANBASI Dogum Yili: 2001 Adres: izmir
```

```
HASH TABLOSU
0. KimlikNo: 43321 Sira: 0
1. BOS
2. KimlikNo: 45543 Sira: 2
3. BOS
4. KimlikNo: 43213 Sira: 3
5. BOS
6. KimlikNo: 54213 Sira: 1
HASH TABLOSU
0. KimlikNo: 43321 Sira: 1
1. KimlikNo: 33445 Sira: 0
2. KimlikNo: 45543 Sira: 3
3. BOS
4. BOS
5. BOS
6. KimlikNo: 54213 Sira: 2
HASH TABLOSU
0. KimlikNo: 43321 Sira: 2
1. KimlikNo: 33445 Sira: 1
2. BOS
3. BOS
4. BOS
5. KimlikNo: 12345 Sira: 0
6. KimlikNo: 54213 Sira: 3
Kisi KimlikNo: 33445 Isim: ACELYA SENLIK Dogum Yili: 1990 Adres: adana
HASH TABLOSU
0. KimlikNo: 43321 Sira: 2
1. KimlikNo: 33445 Sira: 0
2. BOS
3. BOS
4. BOS
5. KimlikNo: 12345 Sira: 1
6. KimlikNo: 54213 Sira: 3
```

Kaynak Kod:

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
typedef struct kisi{//Kisi bilgilerini tutan struct
		char kimlik_no[10];
		char ad[10];
		char soyad[15];
		int yil;
		char adres[15];
		struct kisi* next;
}KISI;
typedef struct cache{//Cache structı: Kisi nodeunu ve cache'in anlik boyutunu gosterir
		int size;
		KISI* head;
}CACHE;

typedef struct hash//Hash Tablosu icin struct
{
		int flag;//flag = 0 : kisi yok flag = 1 : kisi kayitli flag = 2 : kisi silinmis
		int sira;
		char kimlik_no[10];

}HASH;

CACHE* cache_tanimla(int n){// Cache allocation
		CACHE* cache = (CACHE*)malloc(sizeof(CACHE));
		cache->size=0;
		cache->head=NULL;
		return cache;
}
HASH* hash_tanimla(HASH* hash , int m){//Hash allocatiin

		hash = (HASH *)malloc(m*sizeof(HASH));
  int i;
  for (i = 0; i < m; i++)
  {
		hash[i].sira= -1;//Hash gozleri bos oldugundan duzenleme.
			hash[i].flag = 0;//Flag=0 bos anlaminda
  }
  return hash;
}
int horner_key(char kimlik_no[]){//Kimlik no'yu horner kuraliyla int degere cevirme
		int i,key=0,n;
		n=strlen(kimlik_no);
		for(i=n;i>0;i--){
			key=key+pow(31,i-1)*((kimlik_no[n-i]-'0')+1);
		}
		return key;
}
int search(HASH* hash,char kimlik_no[],int m){//Gonderilen kimlik no hashte var mi?
		int i,buldu=0;
		int key = horner_key(kimlik_no);
		int adr = key%m;
		int artis=1+(key%(m-1));
		while(i<m && hash[adr].flag!=0 && !buldu){
			if(hash[adr].flag==1 && !strcmp(hash[adr].kimlik_no,kimlik_no)){//flag=1 ve kimlik no mevcut ise
				buldu=1;//Bulunca cikmasi icin flag.
			}else{
				i++;
				adr=(adr+artis)%m;
			}
		}
		if(buldu==1){
			return adr;//Kimlik no hashte varsa adresini dondurur.
		}else{
			return -1;//Yoksa -1 gonderir.
		}
}
int hash_dolu(HASH* hash, int m){//Cache size ve hash sizein degerlerinde hata olur ise hash tablosu doldu mu kontrolu.
		int i;
		while(i<m && hash[i].flag==1){
			i++;
		}
		if(i==m){
			return 1;
		}else{
			return 0;
		}
}
```

```
void insert(HASH* hash,CACHE* cache, int m,int n, KISI* kisi){

	int kontrol=hash_dolu(hash,m);//Hash Dolu mu?
	if(kontrol){
		printf("Hash Tablosu Dolu.\n");
		return;
	}
	int i=0,adr2;
	int key = horner_key(kisi->kimlik_no);
	int adr = key%m;
	int artis=1+(key%(m-1));
	KISI* tmp;
	KISI* init;
	adr = search(hash,kisi->kimlik_no,m);//Alinan key hashte zaten var mi?

	if(adr!=-1){
		//Alinan kimlik no hashte var ise buradan devam eder.
		tmp=cache->head;
		if(hash[adr].sira==0){
			//Alinan kisi cachein zaten basinda ise printleyip doner.
			printf("Kisi Kimlik_No: %s Isim: %s %s Dogum Yili: %d Adres: %s\n",tmp->kimlik_no,tmp->ad,tmp->soyad,tmp->yil,tmp->adres);
			return;
		}else{
			//Cachede kisiyi basa alip sirasini guncelleme islemleri
			while(strcmp(tmp->next->kimlik_no,hash[adr].kimlik_no)!=0){
				tmp=tmp->next;
			}

			init=tmp->next;
			if(init->next!=NULL){
				tmp->next=init->next;
			}else{
				tmp->next=NULL;
			}

			init->next=cache->head;
			cache->head=init;
			printf("Kisi KimlikNo: %s Isim: %s %s Dogum Yili: %d Adres: %s\n",init->kimlik_no,init->ad,init->soyad,init->yil,init->adres);
			hash[adr].sira=0;//Kisiyi cache'de basa alip sirasini guncelledim.

			//KAYDIRDIKTAN SONRA HASHTE SIRALARI GÜNCELLE
			for(i=1;i<cache->size;i++){
				init=init->next;
				adr2 = search(hash,init->kimlik_no,m);
				hash[adr2].sira=i;
			}
			return;
		}
	}else{
		//Alinan kimlik no hashte yok ise.
		adr=key%m;
		while(hash[adr].flag==1){
			adr=(adr+artis)%m;;
		}

		if(cache->size<n){//Cache dolu degil ise cache sizeini arttirdim
			cache->size++;
		}else{
			//CACHE dolu ise cachein sonundaki elemani silip. Hashte flag degerini deleted yaptim.
			tmp=cache->head;
			int boyut=(cache->size)-1;
			for(i=1;i<boyut;i++){
				tmp=tmp->next;
			}
			adr2=search(hash,tmp->next->kimlik_no,m);
			tmp->next=NULL;
			hash[adr2].flag=2;
		}
		//Yeni kisiyi cache'e ve hashe ekledim.
		KISI* yeni=(KISI*)malloc(sizeof(KISI));
		strcpy(yeni->kimlik_no,kisi->kimlik_no);
		strcpy(yeni->ad,kisi->ad);
		strcpy(yeni->soyad,kisi->soyad);
		strcpy(yeni->adres,kisi->adres);
		yeni->yil=kisi->yil;
		yeni->next=NULL;
		if(cache->head==NULL){
			cache->head=yeni;
		}else{
			yeni->next=cache->head;
			cache->head=yeni;
		}

		hash[adr].flag=1;
		strcpy(hash[adr].kimlik_no,kisi->kimlik_no);
		hash[adr].sira=0;
```

```
                        //KAYDIRDIKTAN SONRA HASHTE SIRALARI GÜNCELLE
                        init=cache->head;
                        for(i=1;i<cache->size;i++){
                                init=init->next;
                                adr2 = search(hash,init->kimlik_no,m);
                                hash[adr2].sira=i;
                        }
                        return;
                }
}
void hash_yazdir(HASH* hash, int m){//Debug icin her adimda hashi yazdirmak icin kullandim.
                int i;
                for(i=0;i<m;i++){
                        if(hash[i].flag==1){
                                printf("%d. KimlikNo: %s Sira: %d\n",i,hash[i].kimlik_no,hash[i].sira);
                        }else{
                                printf("%d. BOS\n",i);
                        }
                }
}
void read_file(char filename[], HASH* hash,CACHE* cache, int m,int n){//Dosyadan kisi bilgilerini alir.
                FILE *fp;
                int i=0;
                KISI* kisi=(KISI*)malloc(sizeof(KISI));
                kisi->next=NULL;
                fp=fopen(filename,"r");
                if(!fp){
                        printf("'%s' isimli dosya bulunmuyor.\n",filename);
        return;
                }
                rewind(fp);
                while(!feof(fp)){
                        fscanf(fp,"%s %s %s %d %s\n",kisi->kimlik_no,kisi->ad,kisi->soyad,&kisi->yil,kisi->adres);//Dosyadan okuma yapip structa aliyorum
                        insert(hash,cache,m,n,kisi);//Aldigim kisiyi hashe eklemek veya cache kontrolu icin kullnadigim fonksiyon.

                        //HASH KONTROLU ICIN TABLO YAZDIRMAK ICIN KULLANDIM.
                        //printf("HASH TABLOSU\n");
                        //hash_yazdir(hash,m);
                }
                fclose(fp);
}

int main(){
                int n,m;//N cache uzunulugu M hash uzunlugu
                char filename[20]="test.txt";
                HASH *hash;
                CACHE* cache;
                printf("Cache uzunlugunu giriniz.\n");
                scanf("%d",&n);
                printf("Hash Tablosu Uzunlugunu giriniz. \n");
                scanf("%d",&m);
                //Cache'i ve Hash'i Tanimladim.
                cache = cache_tanimla(n);
                hash = hash_tanimla(hash,m);
                //Dosyadan okumaya basladim.
                read_file(filename,hash,cache,m,n);
                return 0;
}
```