18011706

Burak Boz

2019-2020 Bahar Yarıyılı

BLM2512 Veri Yapıları ve Algoritmalar 1. Ödevi

Konu : Cache Buffer (Önbellek Tamponu) Tasarımı

Problem: Bu ödevde, double linked list (ikili linkli liste) veri yapısını kullanarak bir cache buffer sistemi tasarlamanız ve gerçeklemeniz istenmektedir. Gerçekleştireceğiniz cache buffer, bir network ortamında yapılan web sayfası isteklerini, hızlı ve kaliteli hizmet verebilmek için depolamak ve servis için önceliklendirmek amacı ile kullanılacaktır.

Çözüm:

Kullanıcıdan eşik değeri ve tampon bellek üzerinde tutulması istenilen maksimum istek sayısı alınır ve başlanır.

Site istekleri dosya.txt üzerinden satır satır okunur ve her satır bir istek olarak kaydedilir.

Her satır okunduğunda eklenen node sayısı kullanıcıdan alınan maksimum değerin altında ise düğüm ekleme işlemine geçilir.

Eğer düğüm daha önceden eklenmişse sayacı arttırılır. Eğer düğüm hiç eklenmemişse ve node sayısı maksimumdan düşük ise, node tampon belleğe eklenir.

Eklenen düğüm eğer eşik değerinden fazla ise tampon bellek üzerinde en başa çekilir.

<u>Kullanılan fonksiyonlar:</u>

- int kontrolEt(char veri[100],int esikDegeri) //Düğüm oluşturmak için gerekli koşulların kontrolü
- struct node* dugumOlustur(char veri[100]) //Düğümün oluşmasını sağlayan fonksiyonumuz
- int kontrolEt(char veri[100],int esikDegeri) //Düğüm oluşturmak için gerekli koşulların kontrolü
- void sonaEkle(char veri[100],int esikDegeri)//En sona yeni bir düğüm eklemek için kullanığımız fonksiyon
- void basaEkle(char veri[100]) //Eğer hiç düğümümüz yoksa ilk olarak bu fonksiyonla ilk düğümü ekleyeceğiz

- void sondanSil() //Eğer kullanıcının girdiği maksimum istek sayısına ulaşıldıysa sondaki düğümü silmek için bu fonksiyonu kullanacağız
- void yazdir() //Düğümlerimizi yazdırmak için bu fonksiyonu kullanıyoruz
- struct node* dugumOlustur(char veri[100]) //Düğümün oluşmasını sağlayan fonksiyonumuz

Düğüm Yapımız:

```
struct node {

   char req[100];          //Web sitesi isteğimiz
   int sayac=0;            //Web sitesine gönderilen istek adedi
   struct node *next;      //Bir sonraki düğümün adresi
   struct node *prev;      //Bir önceki düğümün adresi
};
```

Kodlar:

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <string.h>
#include <malloc.h>
struct node {//Ana düğüm yapımız

   char req[100];
   int sayac=0;
   struct node *next;
   struct node *prev;
};
```

```
struct node* start = NULL;



struct node* dugumOlustur(char veri[100]) //Düğümün oluşmasını sağlayan fonksiyonumuz
{
   struct node* yeniDugum = (struct node*)malloc(sizeof(struct node));//Bellek üzerinde düğümümüz için alan açıyoruz
   strcpy(yeniDugum->req,veri);//düğüme gireceğimiz site adresini düğümün gerekli alanına atıyoruz
   yeniDugum->sayac=1;//ilk defa eklenen düğüm için sayacını 1 olarak ilklendiriyoruz
   //yeniDugum->req = tmpVeri;
   yeniDugum->next = NULL;
   yeniDugum->prev = NULL;

   return yeniDugum;
}

int kontrolEt(char veri[100],int esikDegeri)//Düğüm oluşturmak için gerekli koşulların kontrolü
{
        int kontrolSayac=0;
        char tmpVeri[100];
        strcpy(tmpVeri,veri);
        struct node* temp = start;
   while (temp->next != NULL && strcmp(temp->req,tmpVeri))//Gelen site adresi ile mevcut düğümlerin adreslerini bu döngü içerisinde kontrol edeceğiz
   {
      temp = temp->next;
        }
```

```
        if(strcmp(temp->req,tmpVeri) == 0)//Gelen site adresi ile düğümdeki site adresinin aynı olup olmadığını kontrol ediyoruz
          {
                temp->sayac++;//Eğer düğüm mevcutsa sayaç değerini 1 arttırıyoruz
                kontrolSayac++;

                if(temp->sayac>=esikDegeri)
                {
                        //Başa çek
                        if(temp != start){
                                temp->prev->next=temp->next;
                                if(temp->next != NULL){
                                        temp->next->prev=temp->prev;
                                    }
                                temp->next=start;
                                start->prev = temp;
                                start=temp;
                                temp->prev=NULL;
                                }

                        }
                }
        if      (kontrolSayac!=0)
        {
                return 1;
        }
        return 0;
}
```

```
void sonaEkle(char veri[100],int esikDegeri)//En sona yeni bir düğüm eklemek için kullanığımız fonksiyon
{
   struct node* sonaEklenecek = dugumOlustur(veri);



   if (start == NULL)//Listemizin içerisinde hiç eleman yoksa ilk elemanımızı ekliyoruz
   {
      start = sonaEklenecek;
   }


   else//Eğer liste içerisinde eleman varsa
   {
         if(kontrolEt(veri,esikDegeri))//kontrolEt fonksiyonundan 1 dönerse düğüm eklemiyoruz
         {


                  }
                   else{//kontrolEt fonksiyonundan 0 dönerse düğüm ekliyoruz


               struct node* temp = start;//Start değerini kaybetmemek için temp değerine kopyalıyoruz



               while (temp->next != NULL)//Son düğümün next değerini buluyoruz
               {
                  temp = temp->next;
               }
```

```
            sonaEklenecek->prev = temp;//Sona ekleyeceğimiz düğümün prev değerine liste içerisindeki var olan son değeri ekliyoruz



            temp->next = sonaEklenecek;     //Listede var olan son düğümün next değerine yeni ekleyeceğimiz düğümü işaret ediyoruz
                }


   }
}




void basaEkle(char veri[100])//Eğer hiç düğümümüz yoksa ilk olarak bu fonksiyonla ilk düğümü ekleyeceğiz
{
   struct node* basaEklenecek = dugumOlustur(veri);


   if (start == NULL)
   {
      start = basaEklenecek;
      return;
   }


   start->prev = basaEklenecek;
   basaEklenecek->next = start;
   start = basaEklenecek;
}


void sondanSil()//Eğer kullanıcının girdiği maksimum istek sayısına ulaşıldıysa sondaki düğümü silmek için bu fonksiyonu kullanacağız
```

```
{
   if (start == NULL)
   {
      printf("\n Liste zaten bos ....");
      return;
   }


   struct node* temp = start;
   while (temp->next != NULL)
   {
      temp = temp->next;
   }

   struct node* onceki = temp->prev;
   free(temp);
   onceki->next = NULL;
}

void yazdir()//Düğümlerimizi yazdırmak için bu fonksiyonu kullanıyoruz
{
   struct node* temp = start;

   while (temp != NULL)
   {
      printf("Site: %s\t Sayac: %d\n", temp->req, temp->sayac);
      //printf(" %d", temp->sayac);
      temp = temp->next;
   }
```

```
}


int main(){
	int nodeSayac=0,esik,maxAdet;
	printf("Esik degeri giriniz:");
	scanf("%d", &esik);
	printf("Tutulabilecek maksimum site sayisini giriniz:");
	scanf("%d", &maxAdet);
	FILE *fp=fopen("dosya.txt","r");
	char bilgi[1000];
	while(!feof(fp)){
		fscanf(fp,"%s",bilgi);

		if (nodeSayac<maxAdet)
		{

			sonaEkle(bilgi,esik);
			nodeSayac++;
		}
		else
		{
			sondanSil();
			sonaEkle(bilgi,esik);
		}


		//printf("Eklenen : %s \n",bilgi);
		//printf("%s \n",bilgi);
```

```
	}

	yazdir();


	char cevap;
	printf("\nBellekteki verilerin silinmesini istiyormusunuz:(y: evet / n:hayir)");
	cevap = getch();
	if(cevap =='Y' || cevap=='y')
	{
		//belleği boşalt
		while(start->next != NULL){
			sondanSil();
		}

		free(start);
		printf("\nVeriler temizlendi");

	}else if(cevap=='N' || cevap=='n')
	{
		//Veriler dursun
	}
	return 0;

}
```