Algoritma Analizi Dersi 1.Ödev Raporu

Böl ve Yönet Algoritması Uygulaması

Bu uygulamada Merge Sort sıralama algoritmasında manipülasyonlar yapılarak önceki durumun korunarak yeni bir duruma göre sıralama yapılması gerçekleştirilmiştir.

M.Yasin SAĞLAM

15011804

ALGORİTMA ANALİZİ

GRUP 1

Algoritma Analizi Dersi 1.Ödev Raporu

Böl ve Yönet Algoritması Uygulaması

Yöntem

Verilen örnek problemde bir otobüs firmasının düzenlemiş olduğu otobüs seferlerinin şehir ve saat olarak bilgileri aşağıdaki resimdeki gibi verilmiştir.

Sırasız		**Şehre göre sıralı**		**Şehre göre sıralı iken saate göre sıralı**	
Ankara	09:00	Adana	10:00	Adana	10:00
İzmir	12:00	Ankara	09:00	Ankara	09:00
Ankara	16:00	Ankara	16:00	Ankara	14:00
İstanbul	19:00	Ankara	14:00	Ankara	16:00
İstanbul	11:00	İstanbul	19:00	İstanbul	11:00
İstanbul	13:00	İstanbul	11:00	İstanbul	13:00
Ankara	14:00	İstanbul	13:00	İstanbul	19:00
İzmir	10:00	İzmir	12:00	İzmir	10:00
Adana	10:00	İzmir	10:00	İzmir	12:00

Problemde ise ana hatlarıyla Merge Sort algoritması iki kez kullanılarak örnekteki gibi ilk önce şehirlere göre sonra ise hem şehir hem de saatlere göre Merge Sort sıralama algoritması üzerinde gerekli değişiklikler yapılarak sıralanması istenmektedir. Bunların yanında gerekli bilgilerin input olarak dosyadan okunabileceği gibi kullanıcıdan da program esnasında eklenebilmesinin yazılacak olan C programında bulundurulması beklenmektedir. Yazılan programda şehir ve saat bilgileri LIST adı verilen bir structta tanımlanmıştır ve sefer ekleneceği zaman (dosyadan okuma veya kullanıcı yoluyla girdi olarak) dynamic memory allocation yoluyla boyut genişletilerek inputlar alınmıştır. Bunun yanında kullanıcıdan saat girdisi alınırken saat in uygun formatlara göre girilip girilmediği de kontrol edilmiştir. Örnek verilecek olursa a4:43, 34:99 gibi saatlerin girilememesinin kontrolü yazılan time_control() fonksiyonuyla handle edilmiştir. Sonrasında ise merge modülü ve bu modülü ve kendisini rekürsif olarak çağıracak olan Mergesort modülü sadece şehir ismine göre büyük küçük harfe duyarlı olmaksızın sıralama yapacak şekilde dizayn edilmiş ve kodlanmıştır. İkinci durum olan önceki durumdan alınan şehre göre sıralı olan sefer listesindeki şehir sıralaması bozulmaksızın saatlere göre de sıralanması için

merge2 modülü eklenen kontrolle şehir adı aynı olması durumunda sefer saatleri kıyaslanarak yedek diziye atılacak şekilde değiştirilmiştir ve bu modülü ve kendisini rekürsif olarak çağıracak olan Mergesort2 modülü dizayn edilmiş ve kodlanmıştır.

Önemli değişiklikleri içeren Merge2 modulü ve açıklaması

```
void merge2(LIST *arr, int l_index, int mid, int r_index){
	LIST *temp;
		temp=(LIST*)malloc(sizeof(LIST)*(r_index-l_index+1));
			if(!temp){
				system("COLOR c");
				printf("Not enough space for merge sort!!! Quitting...");
				exit(0);
			}
	int i,j,k;
	i=l_index;
	j=mid+1;
	k=0;
	while(i<=mid && j<=r_index){
		if(strcasecmp(arr[i].city_name,arr[j].city_name)==0 && strcmp(arr[i].hour,arr[j].hour)<0) // eklenen siralama kosulu  şehir adları küçükse saati kıyasla ve atama yap
			temp[k++]=arr[i++];
		else if(strcasecmp(arr[i].city_name,arr[j].city_name)<0) //şehir isimleri eşit değilse  ada göre sıralı olacak şekilde atama yap
			temp[k++]=arr[i++];
		else
			temp[k++]=arr[j++];
	}
	while(i<=mid)
		temp[k++]=arr[i++];
	while(j<=r_index)
		temp[k++]=arr[j++];
	k--;
	while(k>=0){
		arr[l_index+k]=temp[k];
		k--;
	}
free(temp);
}
```

Uygulama

Örnek Sonuçlar

1-)Verilen Örneklerle

```
Order by Cityname

0.      Adana           10:00
1.      Ankara          14:00
2.      Ankara          16:00
3.      Ankara          09:00
4.      Istanbul        15:00
5.      Istanbul        11:00
6.      Istanbul        19:00
7.      Izmir           10:00
8.      Izmir           12:00
```

```
Order by Cityname and hour

0.      Adana           10:00
1.      Ankara          09:00
2.      Ankara          14:00
3.      Ankara          16:00
4.      Istanbul        11:00
5.      Istanbul        15:00
6.      Istanbul        19:00
7.      Izmir           10:00
8.      Izmir           12:00
```

2-)Input Eklenerek

```
Order by Cityname

0.      Adana           09:00
1.      Adana           10:00
2.      Ankara          14:00
3.      Ankara          16:00
4.      Ankara          09:00
5.      Istanbul        14:40
6.      Istanbul        15:00
7.      Istanbul        11:00
8.      Istanbul        19:00
9.      Izmir           10:00
10.     Izmir           12:00
11.     Mus             10:00
12.     Mus             12:00
13.     Mus             12:42
14.     Sivas           14:00
15.     Sivas           16:00
16.     Teksas          03:00
17.     Teksas          22:00
18.     Teksas          19:00
19.     Zonguldak       10:00
20.     Zonguldak       13:00
21.     Zonguldak       11:00
```

```
Order by Cityname and hour

0.      Adana           09:00
1.      Adana           10:00
2.      Ankara          09:00
3.      Ankara          14:00
4.      Ankara          16:00
5.      Istanbul        11:00
6.      Istanbul        14:40
7.      Istanbul        15:00
8.      Istanbul        19:00
9.      Izmir           10:00
10.     Izmir           12:00
11.     Mus             10:00
12.     Mus             12:00
13.     Mus             12:42
14.     Sivas           14:00
15.     Sivas           16:00
16.     Teksas          03:00
17.     Teksas          19:00
18.     Teksas          22:00
19.     Zonguldak       10:00
20.     Zonguldak       11:00
21.     Zonguldak       13:00
```

Bonus

Sıralamanın önceki durum korunarak yapılabileceği ve yapılamayacağı durumlara verilen ad ingilizce literatürde stability veya stable olarak geçmektedir.

Stable Olan Sorting Algoritmaları ; Merge Sort haricinde Insertion Sort ve Bubble Sort olarak iki örnek,

Stable Olmayan Sorting Algoritmaları ise Selection Sort ve Heap Sort iki örneği olmak üzere toplam 4 örnekle açıklayabiliriz.

Sonuç

Merge Sort Algoritmasının **recurrence bağıntısı** : **N > 1 ve D (1) = 0 olduğu durum için**

D (N) = 2 D (N / 2) + N olduğundan D (N) = N lg N olacaktır ve karmaşıklığı N *lgN dir. Gerçekleştirilen bu çözümde aynı algoritmayı 1 kez daha çağırarak sıralama yaptığımız için karmaşıklığımız 2*N*lg N e çıkmaktadır. Fakat kabaca katsayılar yok edildiğinden karmaşıklığımızı N*lg N olarak söyleyebiliriz.

Kod

```
1.  /**
2.  @file
3.
4.  Bu uygulamada Merge Sort siralama algoritmasinda manipulasyonlar yapýlarak onceki durumun koru
    narak yeni bir duruma göre siralama yapilmasi gerceklestirilmistir.
5.
6.  @author
7.
8.  Name              :       Muhammed Yasin SAGLAM
9.  Student No        :       15011804
10. Date              :       15/10/2017
11. E-Mail            :       myasinsaglam1907@gmail.com
12. Compiler Used     :       GCC
13. IDE               :       DEV-C++(Version 5.11)
14. Operating System  :       Windows 10 Educational Edition
15. */
16. #include<stdio.h>
17. #include<stdlib.h>
18. #include<time.h>
19. #include<strings.h>
20. #include<string.h>
21. #include<locale.h>
22. #define CSIZE 20 //maksimum sehir adi karakter uzunlugunu tutan makro
23. #define filename "seferler.txt" //dosya adini tutan makro
24. //ilgili struct olusturuluyor
25. typedef struct {
26.     char city_name[CSIZE]; //sehir adi
27.     char hour[6]; //saat bilgisi
28. }LIST;
29.
30.
31. void merge(LIST *,int ,int ,int ); //ilk merge modulu sadece sehir ismine gore kontrol yapar
```

```
32. void Merge_sort(LIST *,int, int ); //ilk MergeSort fonksiyonu adece sehir ismine gore siralama
     yapan merge modulunu cagirir
33. void merge2(LIST *,int ,int ,int ); //ikinci merge modulu ismin yaninda eklenen bir kosulla sa
    atlere de bakarak kiyaslama yapar
34. void Merge_sort2(LIST *,int, int ); //ikinci MergeSort fonksiyonu hem sehir ismine hemde saate
      gore siralama yapan merge2 modulunu cagirir
35. char* time_control();
36.
37. int main(){
38.     setlocale(LC_ALL, "Turkish");
39.     FILE *fp; //file pointer
40.     LIST *list; //list pointer.
41.     int list_count=0; //baslangicta dosyadan okunacak eleman sayisini tutan degisken
42.     int choice,i; //while kontrol icin choice ve cevrim degiskeni olan i
43.     int add_size=0,new_size; //sonradan kullanici input girdiginde eklenecek input sayisini ve
     toplami tutan degiskenler
44.
45.     list = (LIST*)malloc(sizeof(LIST)); //liste icin 1 elemanlik yer ayriliyor
46.     if(!list){
47.         printf("Allocation error!!! Quitting...");
48.         exit(0);
49.     }
50.
51.     fp = fopen(filename,"r"); //dosya okuma modunda acilir
52.     if(!fp){
53.         printf("file not opened !!! Quitting...");
54.         exit(0);
55.     }
56.     else{
57.         while(!feof(fp)){
58.             fscanf(fp,"%s",list[list_count].city_name); //tum sehir adini oku
59.         //  printf("%s\n",list[list_count].city_name);
60.             fscanf(fp,"%s",list[list_count].hour); //sefer saatini oku
61.         //  printf("%s\n",list[list_count].hour);
62.             list_count++;  //liste boyutunu arttir
63.             list=(LIST*)realloc(list,sizeof(LIST)*(list_count+1));  //listeyi memory allocatio
    n yaparak genislet
64.             if(!list){
65.                 printf("Allocation error!!! Quitting...");
66.                 exit(0);
67.             }
68.         }
69.     }
70.     fclose(fp);
71.
72.     new_size=list_count;
73.
74.     printf("\n1.Add input & Merge Sort\n2.Merge Sort\n\nPlease enter the choice (0 for exit):
    ");
75.     scanf("%d",&choice);                    //Reading choice from the user
76.     system("CLS");
77.     while(choice!=0){
78.         if(choice ==1){ //kullanicinin input girmesini saglayan secenek
79.             printf("\nHow many city and hour information will you add ?\n");
80.             scanf("%d",&add_size);  //eklenecek sefer sayisini kullanicidan oku
81.             new_size+=add_size; //toplam boyutu arttir
82.             list=(LIST*)realloc(list,sizeof(LIST)*(new_size)); //listeyi genislet realloc ile

83.             if(!list){
84.                 printf("Allocation error!!! Quitting...");
85.                 exit(0);
```

```
86.             }
87.             for(i=new_size-add_size;i<new_size;i++){ //eklenecek sefer sayisi kadar dngu
88.                 printf("\nEnter city name : ");
89.                 scanf("%s",list[i].city_name); //sehir adini ilgili adrese oku
90.                 printf("\nEnter hour information : ");
91.                 strcpy(list[i].hour,time_control()); //saat bilgisini formata uygun olacak sek
    ilde kontrol ederek oku
92.                 //printf("%s",list[i].hour);
93.             }
94.             printf("\n\nOrder by Cityname\n");
95.             Merge_sort(list,0,new_size-
    1);//sehre gore sirali halini yazdiran Merge_sort fonksiyonunu cagir
96.             for (i=0;i<new_size;i++){  //listeyi ekrana yazdir
97.                 printf("\n%d.\t%-10s\t%s",i,list[i].city_name,list[i].hour);
98.             }
99.             printf("\n\nOrder by Cityname and hour\n");
100.                Merge_sort2(list,0,new_size-
    1);//hem sehre hemde saate gore sirali halini yazdiran Merge_sort2 fonksiyonunu cagir
101.                for (i=0;i<new_size;i++){//listeyi ekrana yazdir
102.                    printf("\n%d.\t%-10s\t%s",i,list[i].city_name,list[i].hour);
103.                }
104.                printf("\nTotal times is %d",new_size);
105.            }
106.
107.            if(choice ==2){ //Input girmeksizin siralama yapan secenek
108.                printf("\n\nOrder by Cityname\n");
109.                Merge_sort(list,0,new_size-
    1); //sehre gore sirali halini yazdiran Merge_sort fonksiyonunu cagir
110.                for (i=0;i<new_size;i++){
111.                    printf("\n%d.\t%-10s\t%s",i,list[i].city_name,list[i].hour);
112.                }
113.                printf("\n\nOrder by Cityname and hour\n");
114.                Merge_sort2(list,0,new_size-
    1); //hem sehre hemde saate gore sirali halini yazdiran Merge_sort2 fonksiyonunu cagir
115.                for (i=0;i<new_size;i++){
116.                    printf("\n%d.\t%-10s\t%s",i,list[i].city_name,list[i].hour);
117.                }
118.                printf("\nTotal times is %d",new_size);
119.            }
120.            printf("\n1.Add input & Merge Sort\n2.Merge Sort\n\nPlease enter the choice (0
    for exit): ");
121.            scanf("%d",&choice);                    //kullanicidan secimi oku
122.            system("CLS");
123.        }
124.
125.        free(list); //liste icin ayrilan yeri serbest birak
126.        system("PAUSE");
127.        return 0; //program cikisi
128.    }
129.
130.    //saat bilgisini uygun formatli olarak kullanicidan alan ve disari donduren fonksiyon
131.    char* time_control(){
132.        char *time,ch;
133.        time = (char*)malloc(sizeof(char)*5);
134.        int correct = 0;
135.        while(!correct){
136.            ch=getche();
137.            if(isalpha(ch) || (int)(ch-'0')>2 || (int)(ch-
    '0')<0){ //girilen saat bilgisininin ilk karakterini kontrol et {0,2 arasinda olmali}
138.                system("CLS");
139.                printf("\nPlease enter digits or enter correct values for time\n");
```

```
140.                }
141.                else{
142.                    time[0]=ch;
143.                    correct=1;
144.                }
145.            }
146.
147.            while(correct){
148.                ch=getche();
149.                if(isalpha(ch) || ((int)(time[0]-'0')==2) && (int)(ch-'0')>3 || (int)(ch-
    '0')<0){//girilen saat bilgisininin ikinci karakterini kontrol et {ilki iki ise 3 ten buyuk ol
    amaz 0 dan kucuk olamaz}
150.                    system("CLS");
151.                    printf("Please enter digits or enter correct values for time\n");
152.                    printf("%c",time[0]);
153.                }
154.                else{
155.                    time[1]=ch;
156.                    correct=0;
157.                }
158.            }
159.            printf(":");
160.            time[2]=':';
161.
162.            while(!correct){
163.                ch=getche();
164.                if(isalpha(ch) || (int)(ch-'0')>5 || (int)(ch-
    '0')<0){//girilen saat bilgisininin ucuncu karakterini kontrol et {0,5 arasinda olmali }
165.                    system("CLS");
166.                    printf("Please enter digits or enter correct values for time\n");
167.                    printf("%s",time);
168.                }
169.                else{
170.                    time[3]=ch;
171.                    correct=1;
172.                }
173.            }
174.
175.            while(correct){
176.                ch=getche();
177.                if(isalpha(ch)  || (int)(ch-
    '0')<0){ //girilen saat bilgisininin ucuncu karakterini kontrol et {0,9 arasinda olmali }
178.                    system("CLS");
179.                    printf("Please enter digits or enter correct values for time\n");
180.                    printf("%s",time);
181.                }
182.                else{
183.                    time[4]=ch;
184.                    correct=0;
185.                }
186.            }
187.            return time; // dogru alinan input olan saati dondur
188.        }
189.
190.        /**
191.            @param *arr     array that will be merged
192.            @param l_index      left index of array
193.            @param mid      middle index of array
194.            @param r_index      right index of array
195.        */
196.        void merge(LIST *arr, int l_index, int mid, int r_index){
```

```
197.            LIST *temp;
198.                temp=(LIST*)malloc(sizeof(LIST)*(r_index-
    l_index+1)); //yedek dizi icin memory allocation
199.                    if(!temp){
200.                        system("COLOR c");
201.                        printf("Not enough space for merge sort!!! Quitting...");
202.                        exit(0);
203.                    }
204.            int i,j,k;
205.            i=l_index;
206.            j=mid+1;
207.            k=0;
208.            while(i<=mid && j<=r_index){
209.
210.                if( strcasecmp(arr[i].city_name,arr[j].city_name)<0) // isme gore sirali olacak
     sekilde buyuk kucuk harf dikkate onemsiz olarak kontrol yapiliyor
211.                    temp[k++]=arr[i++]; //ismen kucuk olan yedek dizide ilk siralara aliniyor

212.                else
213.                    temp[k++]=arr[j++]; //ismen kucuk olan yedek dizide ilk siralara aliniyor

214.            }
215.            while(i<=mid) //eleman kalmissa solda atama yap
216.                temp[k++]=arr[i++];
217.            while(j<=r_index) //eleman kalmissa sagda atama yap
218.                temp[k++]=arr[j++];
219.            k--;
220.            while(k>=0){ //yedek diziyi gercege yaz
221.                arr[l_index+k]=temp[k];
222.                k--;
223.            }
224.
225.        free(temp); //yedek diziyi free yap
226.        }
227.
228.        /**
229.            @param *arr     array that will be merge sorted
230.            @param  left        start index of array
231.            @param right        end index of array
232.        */
233.        void Merge_sort(LIST *arr,int left,int right){
234.            if(left<right){ //durma kosulu
235.                int mid =(left+right)/2; //ortayi bul
236.                Merge_sort(arr,left,mid); //sol taraf icin cagir
237.                Merge_sort(arr,mid+1,right); //sag taraf icin cagir
238.                merge(arr,left,mid,right); //merge modulu ile kontrol ve atamalari yap
239.            }
240.        }
241.
242.        /**
243.            @param *arr     array that will be merged
244.            @param l_index      left index of array
245.            @param mid      middle index of array
246.            @param r_index      right index of array
247.        */
248.        void merge2(LIST *arr, int l_index, int mid, int r_index){
249.            LIST *temp;
250.                temp=(LIST*)malloc(sizeof(LIST)*(r_index-
    l_index+1));//yedek dizi icin memory allocation
251.                    if(!temp){
252.                        system("COLOR c");
```

```
253.                        printf("Not enough space for merge sort!!! Quitting...");
254.                        exit(0);
255.                    }
256.            int i,j,k;
257.            i=l_index;
258.            j=mid+1;
259.            k=0;
260.            while(i<=mid && j<=r_index){
261.                if( strcasecmp(arr[i].city_name,arr[j].city_name)==0 && strcmp(arr[i].hour,arr[
    j].hour)<0) //sehir adlari ayni ise saatlere gore kiyasla
262.                    temp[k++]=arr[i++];
263.                else if(strcasecmp(arr[i].city_name,arr[j].city_name)<0) //farkli ise isme gore
     sirali olacak sekilde buyuk kucuk harf dikkate onemsiz olarak kontrol yapiliyor
264.                    temp[k++]=arr[i++]; //ismen kucuk olan yedek dizide ilk siralara aliniyor
265.                else
266.                    temp[k++]=arr[j++]; //ismen kucuk olan yedek dizide ilk siralara aliniyor
267.            }
268.            while(i<=mid)//eleman kalmissa solda atama yap bitir
269.                temp[k++]=arr[i++];
270.            while(j<=r_index)//eleman kalmissa sagda atama yap bitir
271.                temp[k++]=arr[j++];
272.            k--;
273.            while(k>=0){  //yedek diziyi gercege yaz
274.                arr[l_index+k]=temp[k];
275.                k--;
276.            }
277.
278.        free(temp);//yedek diziyi free yap
279.        }
280.
281.        /**
282.            @param *arr     array that will be merge sorted
283.            @param  left        start index of array
284.            @param right        end index of array
285.        */
286.        void Merge_sort2(LIST *arr,int left,int right){
287.            if(left<right){//durma kosulu
288.                int mid =(left+right)/2;//ortayi bul
289.                Merge_sort2(arr,left,mid); //sol taraf icin cagir
290.                Merge_sort2(arr,mid+1,right);//sag taraf icin cagir
291.                merge2(arr,left,mid,right);//merge modulu ile kontrol ve atamalari yap
292.            }
293.        }
```