CSE331 Computer Organization HW1 Report

Muhammed ÖZKAN 151044084

Programın yazım aşamasında hazırladığım ve kullandığım data yapısı

setu için:

value	length	1	2	3	...	...	...	n
index	0	1	2	3	...	...	...	n

arrayin uzunluk bilgisini kendi hariç negatif bir sayı olarak 0. elemanda saklar. Yukarıdaki örnekte length in değerinin "-n" olması gerekmetedir.

arr0…..arr9 için:

value	length	1	2	3	...	...	n	iv
index	0	1	2	3	...	...	n	n+1

Bu yapıda da aynı şekilde arrayin uzunluk bilgisini kendi hariç negatif bir sayı olarak 0. elemanda saklar. Yukarıdaki örnekte length in değerinin "-n" olması gerekmetedir. Ek olarak iv kısmında ise intersection prosedürünün kullanması için arrayin son elemanının programın başlangıcın da -1 olarak belirlenmesi gerekmektedir. Ilgili prosedür bu alana U setiyle o setin benzer eleman sayısını bu alanda tutacaktır.

Örnek bir girdi şu şekilde olmalı

```
arr9:   .word -6, 7, 3, 4, 5, 6, 2, -1

arr2:   .word -4, 3, 4, 5, 6, -1
```

Ayrıca length değeri 0 yapılarak ilgili array 0 elemanlı olarak tanımlanabilir ve ilgili işlemlere dahil edilmeyebilir.

```
arr2:   .word 0, 3, 4, 5, 6, -1
```

Bu durumada arr2 işlemlere dahil edilmeyecektir.

```
arr:    .word arr0, arr1, arr2, arr3, arr4, arr5, arr6, arr7, arr8, arr9
```

Bu yapı ise tanımlanan arraylere daha hızlı ulaşım ve işlem gerçekleştirmek için yapılmıştır ve ilk 10 elemanı üzerinde işlem yapacak şekilde mips programı yazılmıştır.(C deki double pointer gibi)

Programı debug ettiğim süre boyunca farkettiğim bir tane hatasının olduğunu gördüm bu hata ise şu durumlarda gerçekleşiyor. Benzerlik sayılarının hesaplanması aşamasından sonra bu sayılar büyükten küçüğe doğru sıralandığında aynı benzerlik sayısına sahip birden fazla küme varsa program

yukarıdan aşağıya doğru sırayla gittiğinden kümelerin tanımlama sırasına göre programın seçtiği kümeler değişiklik gösterebiliyor. Bir örnekle göstermek gerekirse aşağıdaki tanımlada sonuç arr2,arr0,arr1 iken aslında arr1,arr2 en optimal sonuç oluyor. Ama aşağıdaki girdide arr0 ile arr1 in içeriklerini değiştirdiğimizde istediğimiz sonucu elde ediyoruz. Bu problem dışında başka bir sorun tespit etmedim.

```
setu:   .word -5, 1, 2, 3, 4, 5
arr0:   .word -2, 4, 5, -1
arr1:   .word -2, 1, 5, -1
arr2:   .word -3, 2, 3, 4, -1
arr3:   .word 0, -1
arr4:   .word 0, -1
arr5:   .word 0, -1
arr6:   .word 0, -1
arr7:   .word 0, -1
arr8:   .word 0, -1
arr9:   .word 0, -1
```

Programın algoritmasının daha iyi anlaşılması için program yazım aşamasında hazırladığım c kodunu ek olarak ekliyorum.

Prosedürler

- **Main:** Sırayla intersection ve sort prosedürleri çağrılır ve en çok benzer eleman bulunandan en aza doğru sıralanır. Daha sonra ise yeni boş bir küme oluşturularak sıralanan kümeler vasıtasıyla ulaşması gereken (U) kümeye en az küme(S_1,S_2…) kullanarak ulaşılmaya çalışır. Hedefe ulaşıldığı şu şekilde kontrol edilir, ulaşılmak istenilen kümenin eleman sayısı, yeni kümenin eleman sayısına eşit ise işlemden çıkılır ve kullanılan kümeler sırayla ekrana yazdırılır.

- **Intersection:** $a0 ve $a1 ile gönderilen adresler içerisindeki arraylerde bulunan değerleri tek tek karşılaştırıp benzer eleman sayısını ilgili arrayin son elemanına kayıt eder.

- **Sort:** $a0 ile gönderilen adresin içindeki adreslerde bulunan arraylerin son elemanlarında bulunan U kümesiyle kesişme sayısına göre büyükten küçüğe doğru sıralar.Sıralama algoritması olarak Bubble Sort kullanılmıştır.

- **Printarr:** $a0 ile gönderilen adresteki değerin 1. elemanından başlayarak 0. Elemandaki negatif sayı kadar integer değeri sırayla “printint” prosedürünü çağırarak konsola bastırır.

- **Printint:** $a0 ile gönderilen değeri integer olarak syscall ile konsola bastırır.

- **Printstring:** $a0 ile gönderilen değeri string olarak syscall ile konsola bastırır.

Örnek ekran görüntüleri

- İlk satır ulaşılması istenen set, ikinci ve diğer satırlar kullanılan set lerin içeriğidir.

```
--------Min Set Cover Program Begin--------
U= 8 2 3 4 5 6 7 1 9

7 3 4 5 6 2
6 7 8
1 2 9
---------Min Set Cover Program End---------


-- program is finished running --
```

```
--------Min Set Cover Program Begin--------
U= 1 2 3 4 5

1 2 4
3 4 5
---------Min Set Cover Program End---------


-- program is finished running --
```

```
--------Min Set Cover Program Begin--------
U= 8 2 3 4 5

3 4 5
2 3
4 6 8
---------Min Set Cover Program End---------


-- program is finished running --
```

```
--------Min Set Cover Program Begin--------
U= 1 2 3 4 5 6 7 8 9

7 3 4 5 6 2
1 2 8 9
---------Min Set Cover Program End---------


-- program is finished running --
```