Logo

Description automatically generated

**2021-2022 Güz Yarıyılı**

**Algoritma Analizi**

**Ödev – 2: Dinamik Programlama**

**Rayene Bech**

**18011115**

1. **Yöntem**

Bu problem, dinamik programlama yaklaşım ile çözülmektedir.

Amacımız, verilen ilanlar arasında, birbirini çakışmayan ilanlar, maksimum kazanç elde edecek şekilde seçilecek ilanlar bulmak.

Bu problem çözmek için kullanılan veri yapılar bunlardır:

* İlanın bilgileri tutmak için struct yapısı kullanmaktadır

**typedef struct{**

**int index;**

**int start;**

**int duration;**

**int value;**

**}addv;**

* Bütün ilanların bilgileri tutmak için array of struct kullanılmaktadır.
* Her seferinde toplam kazanç hesaplayıp tutmak için bir dizi kullanılmaktadır (**F[n]**)
* Seçilecek olan ilanlar tutmak için ayrıca bir dizi kullanmaktadır(**S[n]**). Bu dizinin her elemanı aslında bir linked list olarak tanımlandı. Şu ana kadar seçilen ilanların indisleri tutmaktadır. Ekleme işlemleri kolaylık açısından linked lists tercih edilmiştir.

Bu problem aşağıdaki adımlar ile çözülmektedir:

* İlk başta dizi ilanların bitiş zamanına göre küçükten büyüğe sıralanır . Onun için Merge Sort yaklaşımı kullanılmaktadır. Worst Case = Average Case = Best Case = O(n\* Log n)
* Hiçbir reklam yoksa, maksimum kazancımız 0 olacaktır.
* Tek bir ilan varsa, maksimum kazanç o ilanın kazancına eşittir olacaktır.
* Bir i ilana geldiğinde önceki son çakışmayan ilanın kazancına toplanır. Eğer elde edilen toplam önceki (i-1) kazancından daha büyük ise F[i] yeni hesaplanan değere sahip olmaktadır. Değilse, F[i]=F[i-1] olacaktır. Bunu aşağıdaki formül ile özetleyebiliriz:

**F[i]= max((arr[i].value +F[son çakışmayan]) , F[i-1] )**

* Her sefer, eğer bu ilan seçilecekse, indisi linked list’e eklenir. Seçilmezse, bir önceki linked list kopyalanır.
* İşlem sonunda maksimum kazanç F[n]’de, seçilecek ilanlar ise S[n]’de.

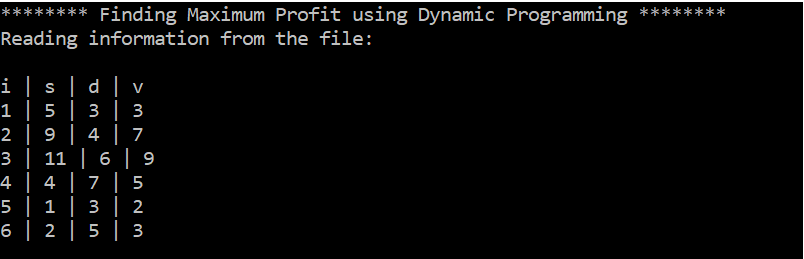
**Zaman Karmaşıklığı:**

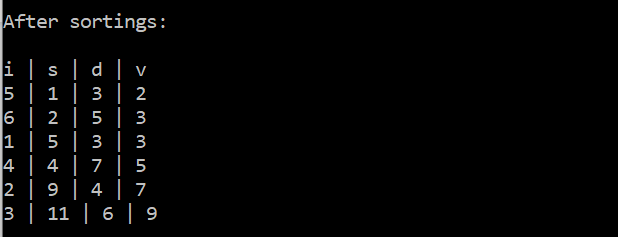
Maksimum kazanç hesaplarken diziyi baştan sonuna kadar **(0’den n’e )** gezilir. Her **i eleman** için, son çakışmayan son eleman bulmak için yine geriye doğru **(i’den 0’a)** gezilir. En kötü durumlarda 0’a kadar araştırılacak (i.elemanın bütün elemanlar ile çakışıyorsa veya son çakışmadığı eleman 0.eleman ise). O yüzden Zaman karmaşıklığı **O(n^2)** ile verilebilir.

Sıralamada Merge Sort kullanıldığı için zaman karmaşıklığı **O(n\* Log n)**

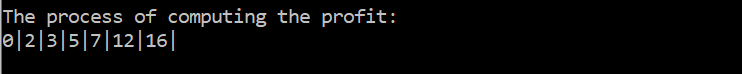
Toplam karmaşıklığı yine **O(n^2)** olmaktadır

1. **Uygulama**

* İlk başta, verilen txt dosyasından veriler bir diziye okunup yazdırılmaktadır. Dizi struct yapısına sahiptir. Her ilan için start, duration ve value değerleri vardır.
* Sonra, ilanların bitiş zamanlara göre küçükten büyüğe merge sort kullanarak sıralanmaktadır. Sıralandıktan sonra da sonuç yazdırılmaktadır:



* Sonra, Profit hesaplamak için F fonksiyonu çağırılır ve hesaplama aşağıda gibi yapılır:



**F[i]= max((arr[i].value +F[son çakışmayan]) , F[i-1] )**

0 ilan için: **F[0] = 0; S[0]=NULL;**

1 ilan için: ilk ilan olduğu için seçilir

**F[1]= max(arr[1].value, F[0])= max(2,0)=2;**

**S[1]= arr[1].index**; şu ana kadar 1. indisi seçildi.

2 ilan için: 1 ve 2 çakışıyor ama 2.ilan profit değeri daha büyük olduğu için seçilir

**F[2]= max(arr[2].value, F[1])=max(3,2)=3**

**S[2]= arr[2].index;** şu ana kadar 2. indisi seçildi

3 ilan için: 3’ün çakışmadığı son reklam 1, F[1]+arr[3].value>F[2] olduğu için F[3]=5 olur.

**F[3]=max(arr[3].value +F[1], F[2])= max(3+2,3)= 5**

**S[3]= arr[3].index -> arr[1].index ;** şu ana kadar 1. ve 3.indisler seçildi

4 ilan için: 4’ün çakışmadığı son reklam 1, F[1]+arr[4].value>F[3] olduğu için F[4]=7 olur.

**F[4]=max(arr[4].value +F[1], F[3])= max(5+2,5)= 7**

**S[4]= arr[4].index -> arr[1].index ;** şu ana kadar 1. ve 4.indisler seçildi

5 ilan için: 5’in çakışmadığı son reklam 3, F[3]+arr[5].value>F[4] olduğu için F[5]=12 olur

**F[5]= max(arr[5].value+F[3], F[4])= max(7+5,7)=12**

**S[5]= arr[5].index -> arr[3].index-> arr[1].index** şu ana kadar 1., 3. ve 5.indisler seçildi

6 ilan için: 6’ın çakışmadığı son reklam 4, F[4]+arr[6].value>F[5] olduğu için F[6]=16 olur

**F[6]= max(arr[6].value+F[4], F[5])= max(9+7,12)=16**

**S[6]= arr[6].index -> arr[4].index-> arr[1].index;** şu ana kadar 1., 4. ve 6.indisler seçildi

* Ve sonuç olarak F[6]=16 en büyük profit olur. Seçilen ilanlar ise: 5,4 ve 3 olur

