YILDIZ TEKNİK ÜNİVERSİTESİ

BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ

Adı Soyadı: Enes Gündüz

Numarası : 16011112

Konu : Dinamik programlama

1-) YÖNTEM

Verilen reklam verilerine göre süreleri çakışmadan en fazla kazanç sağlayan reklamları ve bu reklamların toplam kazancı istenmektedir.

Giriş bilgisi olarak her reklamın başlangıç zamanını, süresini ve ödeyeceği fiyatı “Sample.txt” dosyasından alındı.

```
char fileName[] = "Sample.txt";
char line[512];
int i, j;
struct Reklam dizi[80];

int sayi1, sayi2, sayi3, c=0;

FILE *fs = fopen(fileName, "r");  // FILE pointer
if (!fs) {
  printf("Dosya acilamadi!");
  return 0;
}
```

Sayı1 başlangıç zamanını, sayı2 süresini, sayı3 ise ödeyeceği fiyatı tutar.

```
fseek(fs, 0, SEEK_SET);
do{

    fscanf(fs, "%d", &sayi1);
    fscanf(fs, "%d", &sayi2);
    fscanf(fs, "%d", &sayi3);



    dizi[c].baslangic=sayi1;
    dizi[c].sure=sayi2;
    dizi[c].fiyat=sayi3;
    dizi[c].reklamID=c+1;
    dizi[c].bitis= dizi[c].baslangic + dizi[c].sure;



    printf("%d. Nokta: %d ve %d ve %d \n", c+1, sayi1, sayi2, sayi3);
    c++;

} while (fgets(line, 512, fs));

  printf("C: %d\n",c);
  fclose(fs);
```

Burada reklamları bitiş sürelerine göre büyükten küçüğe doğru sıraladık.

```
void bsortDesc(struct Reklam dizi[],int N)
{
    int i,j;
    struct Reklam temp;

    for(i=0;i<N-1;i++)
    {
        for(j=0;j<(N-1-i);j++)
        {
            if(dizi[j].bitis > dizi[j+1].bitis)
            {
                temp = dizi[j];
                dizi[j] = dizi[j+1];
                dizi[j+1] = temp;
            }
        }
    }
}
```

Reklamlar bitiş sürelerine göre sıralandıktan sonra çakışma olup olmadığı kontrol edildi.

```
int binarySearch(struct Reklam ads[],int index)
{
    // lo= 0.index , hi= dizideki sonuncu index
    int lo = 0 , hi = index-1;

    while(lo<=hi)
    {
        int mid = (lo + hi) / 2;
        if(ads[mid].bitis <= ads[index].baslangic)
        {
            if(ads[mid+1].bitis <= ads[index].baslangic)
                lo = mid + 1;
            else
                return mid;
        }
        else
            hi=mid-1;
    }
    return-1;
}
```

Reklamlardan kazançlı olanlar alındı ve max kazanç bulundu.

```
int kazancliYol(struct Reklam dizi[], int n)
{
    bsortDesc(dizi,n);

    int*table = (int*)malloc(n*sizeof(int));;
    table[0]=dizi[0].fiyat;
    int i;

     for(i=1;i<n;i++)
    {

        int P = dizi[i].fiyat;
        int l = binarySearch(dizi,i);
        if(l!= -1){
            P+= table[1];
        }else{
            printf("Kazancli bulunan diziler: %d\n", i);
        }
        int max;
        if(P>table[i-1]){max=P;
        }else{max=table[i-1];
        }

        table[i]=max;
    }

    int sonuc = table[n-1];
    free(table);

    return sonuc;
}
```

Karmaşıklık Hesabı:

```
void bsortDesc(struct Reklam dizi[],int N)
{
    int i,j;
    struct Reklam temp;

    for(i=0;i<N-1;i++)
    {
        for(j=0;j<(N-1-i);j++)
        {
            if(dizi[j].bitis > dizi[j+1].bitis)
            {
                temp = dizi[j];
                dizi[j] = dizi[j+1];
                dizi[j+1] = temp;
            }
        }
    }
}
```

Burada bubblesortun karmaşıklığı;

Bubblesortta her eleman yanındaki elemanla kontrol edilip sizi sonuna kadar kontrol ilerler, bu işlem dizinin her elemanı için yapıldığından en kötü durumda WorstCase 0(n^2) olur. AverageCase i de O(n^2) olur ama BestCase dizinin sıralı gelme durumunda O(n) olur.

Dinamik programlamada binary search kullandık;

```
int binarySearch(struct Reklam ads[],int index)
{
    // lo= 0.index , hi= dizideki sonuncu index
    int lo = 0 , hi = index-1;

    while(lo<=hi)
    {
        int mid = (lo + hi) / 2;
        if(ads[mid].bitis <= ads[index].baslangic)
        {
            if(ads[mid+1].bitis <= ads[index].baslangic)
                lo = mid + 1;
            else
                return mid;
        }
        else
            hi=mid-1;
    }
    return-1;
}
```

Binary Search çalışma zamanı olarak iyidir. Her iterasyonda arama uzayını yarıya indirmek üzere tasarlanmıştır.

Binary Search algoritmasının zaman karmaşıklığı O(logn)'dir.

2-) UYGULAMA

Kodun çalıştırıldıktan sonraki hali şekildeki gibidir