Algoritma Analizi ve Tasarımı Dersi

2022-2023 Bahar Dönemi

**ÖDEV # RAPORU**

Teslim Tarihi: 15/05/2023

Öğrenci Adı: Kasım İslam TATLI

Öğrenci Numarası: 202802055

1. Giriş

Knapsack (sırt çantası) probleminin çözümü için gerekli dinamik programlama algoritmasını yazmak. Yazdığımız algoritmada dosyaların boyutuna göre hızını hesaplayıp çıkarım yapmak.

1. Geliştirme ve Çalıştırma Ortamları

🡪Visual Studio 2022, C# dili.

1. Algoritma Akışı

-

function KnapsackProblem(filePath, includeState):

values = [] // değerleri atanır

weights = [] // ağırlıkları atanır

includeState = [] // eklenme durumunu atanır

itemCount = 0 // item sayısı

maxCapasity=0 // çantanın maximum kapasitesi

file = open(filePath,”r”) // dosyanın okunması sağlanır

firstLine = file.readline().split(‘ ‘) // boşluğa göre ilk satır alınır

itemCount = parseInt(firstLine[0]) //item sayısı alınır

maxCapacity = parseInt(firtsLine[1].trim()) //çanta kapasitesi alınır

for i in range(itemCount): // değerler ve ağırlıklar ilgili listelere eklenir

line = file.readline().split(' ')

values.append(parseInt(line[0].trim()))

weights.append(parseInt(line[1].trim()))

size = itemCount // item sayısı atanır

dynamicProgTable = createArray(n + 1, maxCapacity + 1) //dinamik programlama tablosunu oluşturuldu

for i in range(1, size + 1): //dinamik çözüm

for w in range(1, maxCapacity + 1):

if weights[i - 1] <= w:

dynamicProgTable[i][w] = max(values[i - 1] + dynamicProgTable[i - 1][w - weights[i - 1]], dynamicProgTable[i - 1][w]) // / Öğeyi çantaya eklemenin veya eklemeden geçmenin maksimum değerini seçilir

else:

dynamicProgTable[i][w] = dynamicProgTable[i - 1][w] // Öğeyi çantaya ekleyemiyoruz, bir önceki öğe durumunu al

optimalValue = dynamicProgTable[size][maxCapacity] //optimal değer atanır

includedItems = [] //eklenecek itemler için dizi oluşturulur

k = maxCapacity

for i in range(n, 0, -1):

if optimalValue != dynamicProgTable[i - 1][k]:

includedItems.append(i) //dahil edilen itemler eklenir

optimalValue -= values[i - 1]

k -= weights[i - 1]

includeState.append(1) // dahil edilme durumunda 1 eklenir

else:

includeState.append(0) //dahil edilmeme durumunda 0 eklenir

1. Sonuç

Knapsack problemi kısaca sırt çantası mantığına dayanan ağırlık, değer birimleri üzerinde en mantıklı birimleri saklamak, almak, veya taşımak anlamına gelir. Günlük hayatta bir hırsızı örneği verirsek, o hırsız için mantıklı olan çantasına hafif ama değerli ürünleri almaktır. Tabi o çantanın bir sınırı vardır, o çantaya ne kadar değerli ve hafif ürünleri alırsa hırsız o kadar kazanır.

Bu projemizde de knapsack problemini uygulayıp optimal değerleri bulmaya çalıştık.

Problemin en bariz kullanıldığı yer Lojistik ve Dağıtımdır. Ürünlerin taşınması için en mantıklı kombinasyonu bulmak için knapsack problemi kullanılabilir.

Yatırım Portföyü: Yatırımcılar portföylerini hazırlarken en erken ve etkili kar kalemini secmeye çalışırlar. Burada knapsack problemi kullanılabilir.

Bunların yanında Reklam kampanyaları ve Proje planlamalarında kullanılabilir.

ekran görüntüsü, diyagram, çizgi, öykü gelişim çizgisi; kumpas; grafiğini çıkarma içeren bir resim

Açıklama otomatik olarak oluşturuldu

Veriler artıkça şüphesiz çalışma zamanının artacağını bekliyordum. 19 tane veri bulunan dosya işlenirken 0,5 saniye, 200 tane veri bulunan dosya işlenirken 0,54 saniye, 10000 tane veri bulunan dosya işlenirken tam 284 saniyeye ihtiyaç duyuldu.