Algoritma Analizi ve Tasarımı Dersi

2022-2023 Bahar Dönemi

**ÖDEV # RAPORU**

Teslim Tarihi: 11.05.2023

Öğrenci Adı: Mahmut Bayram Köylü

Öğrenci Numarası: 222802011

1. Giriş

Ödevin amacı knapsack probleminin çözümünü, büyük boyutlu dosyaların oluşturduğu memory sorununu iyi bir şekilde yönetebilmeyi, optimum zaman karmaşıklığında problemi çözebilmeyi öğrenmek.

Ödev için geliştirilen algoritmalar:

10.000 öğelik dosyada memory sorunu oluştuğundan dolayı, 10.000 öğelik dosya için ayrı bir algoritma kullanmak üzere 2 farklı algoritma kullandım.

1)

1. “ks\_4(19, 200)\_0” adlı dosyayı oku.
2. İlk satırı “features” listesine aktar.
3. “lines” adlı 2D liste tanımla ve tüm öğelerin ağırlık ve değerlerini ata.
4. “weighted\_knapsack” fonksiyonunu tanımla ve “W”(maksimum ağırlık) ve “lines”(tüm ağırlık ve değerleri) parametrelerini al.
5. “n” değişkenini oluşturup içine dosyadaki öğe sayısını ata.
6. Dinamik programlama tablosu oluşturmak için (n+1) x (W+1) boyutunda bir matris tanımla.
7. Matrisin i. satırını hesapla:

* Eğer “M” matrisinin (i-1). öğenin ağırlığı w’den büyükse, matrisin i. satırı (i-1). satırın aynı sütununa eşittir.
* Eğer “lines” matrisinin (i-1). öğesinin ağırlığı w’den küçükse, o zaman “M” matrisinin i. satırı w. sütununa, max fonksiyonunu kullanarak “M” matrisinin (i-1). satırı w. sütunundaki değeri veya “lines” matrisinin (i-1). satırdaki ‘value’ değeri + “M” matrisinin (i-1). satır, w’den “lines” matrisinin (i-1). satırındaki ağırlığın çıkarılmasıyla elde edilen sayınıncı sütunda bulunan değerinden en büyük olanını ata.

1. “M” matrisinin son satır ve son sütunundaki değeri yazdır.

2)

1. “ks\_10000\_0” adlı dosyayı oku.
2. İlk satırı “features” listesine aktar.
3. “lines” listesini oluşturmak için gerekli boşlukları içeren bir 2D liste tanımla.
4. Dosyanın geri kalanını okuyarak, “lines” 2D listesine tüm öğelerin ağırlık ve değerlerini ata.
5. “weighted\_knapsack” fonksiyonunu tanımla ve “W”(maksimum ağırlık) ve “lines”(tüm ağırlık ve değerleri) parametrelerini al.
6. “M” adlı bir liste oluştur ve listenin boyutunu öğelerin sayısına eşitle.
7. Her öğe için bir döngü oluştur ve “M” listesine, öğenin ağırlık ve değerleri kullanılarak hesaplanan oranı ekle.
8. “M” listesini küçükten büyüğe sırala ve aynı sırayla “lines” listesini de sırala.
9. “topWeight” ve “topValue” adlı, 0 ile başlayan iki değişken tanımla.(values değerini 0 ve 1’leri yazdırmakta kullandığımdan yazmadım.)
10. En büyük orandan başlayacak şekilde tüm öğeler için bir döngü oluştur ve her öğenin ağırlığını ve değerini “topWeight” ve “topValue” değişkenlerine ekle fakat toplam ağırlık 1 milyondan büyük olduğunda döngüyü durdur ve son eklenen value ve weight değerlerini “topWeight” ve “topValue” değişkenlerinden çıkar.
11. “topValue” değişkenini yazdır.

Giriş olarak; ks\_4\_0, ks\_19\_0, ks\_200\_0, ks\_10000\_0 dosyalarındaki value ve weight verilerini aldım.

Çıkış olarak yukarıdaki dosya sırasıyla 19, 12248, 100236, 824482 verilerini aldım.

1. Geliştirme ve Çalıştırma Ortamları

Geliştirme ortamı olarak PyCharm ve programlama dili olarak Python kullandım.

1. Algoritma Akışı

Yukarıda belirttiğim sebepten ötürü 2 farklı algoritma kullandım.

1)

Input: n, w1, w2, ..., wn, v1, v2, ..., vn, features, lines, W

Output: M[n, W]

for i = 0 to n+1

for w = 0 to W+1

if(wi > w)

M[i, w] = max (M[i-1, w], vi + M[i-1, w-wi])

print M[n, W]

2)

Input: n, w1, w2, ..., wn, v1, v2, ..., vn, features, lines, topWeight, topValue

Output: final

for i = 0 to n

if(wi < 1000000)

oran = wi/vi

M[i] = oran

for index = 0 to n

min\_index = index

for j = index + 1 to n

if M[j] < M[min\_index]

min\_index = j  
 M[min\_index] swap with M[index]

lines[min\_index] swap with lines[index]

for j = n to 0

topWeight = topWeight + wj

topValue = topValue + vj

if topWeight > 1000000

topWeight = topWeight – wj

topValue = topValue – vj

break

print topValue

1. Sonuç

* Knapsack problemi kısıtlı imkanlarla en iyiye ulaşmak için kullanılır. Günlük hayattan örnek verecek olursak, bir mağazadan veya marketten sınırlı bütçemizi kullanarak evin ihtiyaçları için en iyi ürünleri almak.

Robotik, havacılık, lojistik, … vb. birçok alanda kullanılıyor.

çizelge içeren bir resim

Açıklama otomatik olarak oluşturuldu

19 ve 200 boyutlu dosyalar için brute-force algoritma kullandığım için boyutlarına göre uzun sürdüler fakat en iyi sonucu verdiler. 10000 boyutlu dosya için greedy algoritma kullandığım için boyutuna göre uzun sürmedi fakat en iyi sonuca ulaşıp ulaşmadığım bilinmiyor.