[SERKAN DOLDUR](javascript:;)

E235013188

FEN BİLİMLERİ ENSTİTÜSÜ

BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI

BİLİŞİM TEKNOLOJİLERİ PR. (YL) (UZAKTAN EĞİTİM)

EBT 512 VERİ YAPILARI VE ALGORİTMALAR 2.ÖDEV

**a) Dinamik Programlama Yöntemi ile Sırt Çantası Problemi:**

Sırt çantası problemi, belirli bir ağırlık sınırlaması altında değerleri maksimize etmeye çalışırken verilen nesneler arasında optimal bir seçim yapmayı hedefler.

Dinamik programlama, bu problemin çözümünde oldukça etkilidir.

**Algoritma Adımları:**

Bir matris oluşturulur. Bu matrisin satırları, her bir nesnenin seçilip seçilmediğini (0/1) temsil eder. Sütunlar, farklı ağırlık kapasitelerini temsil eder.

İlk satır ve ilk sütun sıfırlarla doldurulur.

Her bir hücre için, o anki nesnenin ağırlığı kapasiteyi aşmıyorsa, iki durum değerlendirilir: nesneyi almamak veya almak.

Optimal değeri bulmak için alt problemlerin sonuçları kullanılır.

Matrisin sağ alt köşesindeki değer en yüksek toplam değeri verir.

**Zaman Karmaşıklığı:**

Dinamik programlama, her bir nesne ve ağırlık kapasitesi için bir hücre hesaplar. Toplam zaman karmaşıklığı O(nW) olur, n: nesne sayısı, W: ağırlık kapasitesi.

**Bellek Karmaşıklığı:**

Matrisin boyutu nesne sayısı ve ağırlık kapasitesine bağlıdır, bu nedenle bellek karmaşıklığı O(nW) olur.

**Örnek:**

Nesne 1: Değer = 10, Ağırlık = 2

Nesne 2: Değer = 6, Ağırlık = 3

Nesne 3: Değer = 12, Ağırlık = 1

**Adımlar:**

1.Bir matris oluşturuyoruz. Satırlar nesneleri, sütunlar ise sırt çantasının farklı ağırlık kapasitelerini temsil edecek.

2.İlk satır ve ilk sütun sıfırlarla doldurulur.

**Adım 0:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| --------------------------- | | | | | | | |
| 0 | | | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | | | 0 |  |  |  |  |  |
| 2 | | | 0 |  |  |  |  |  |
| 3 | | | 0 |  |  |  |  |  |

3.Her bir nesne için, o anki ağırlık kapasitesine kadar olan durumlar değerlendirilir ve matris güncellenir.

**1.Nesne**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| ---------------------------------- | | | | | | | |
| 0 | | | 0 | 0 | 10 | 10 | 10 | 10 |
| 1 | | | 0 |  |  |  |  |  |
| 2 | | | 0 |  |  |  |  |  |
| 3 | | | 0 |  |  |  |  |  |

**2.Nesne**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| ---------------------------------- | | | | | | | |
| 0 | | | 0 | 0 | 10 | 10 | 16 | 16 |
| 1 | | | 0 |  |  |  |  |  |
| 2 | | | 0 |  |  |  |  |  |
| 3 | | | 0 |  |  |  |  |  |

**3.Nesne**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | 0 | 1 | 2 | 3 | 4 | 5 |
| ------------------------------------ | | | | | | | |
| 0 | | | 0 | 12 | 12 | 22 | 22 | 28 |
| 1 | | | 0 |  |  |  |  |  |
| 2 | | | 0 |  |  |  |  |  |
| 3 | | | 0 |  |  |  |  |  |

4.Matrisin sağ alt köşesindeki değer en yüksek toplam değeri verir. Bu durumda en yüksek değer 28.

**b) Greedy Yaklaşımı ile Sırt Çantası Problemi:**

Greedy yaklaşımı, her adımda en iyi seçimi yapmaya çalışır, ancak bu her zaman global olarak en iyiyi sağlamaz.

**Algoritma Adımları:**

Nesneler değer/ağırlık oranına göre sıralanır.

Sırt çantasına sığacak kadar en değerli nesne eklenir.

Sığacak kadar eklenemeyen nesneler için aynı işlem tekrarlanır.

Sığacak nesne kalmayana kadar devam edilir.

**Zaman Karmaşıklığı:**

Nesnelerin sıralanması O(n log n) karmaşıklığına sahiptir, ardından nesneler sıralandıktan sonra her bir nesnenin kontrolü O(n) zaman alır. Toplam zaman karmaşıklığı O(n log n + n) olur, yani O(n log n) olur.

**Bellek Karmaşıklığı:**

Greedy algoritma, sadece sıralama için biraz ek bellek gerektirir. O(n) ek bellek tüketebilir.

**Örnek:**

Nesne 1: Değer = 10, Ağırlık = 7

Nesne 2: Değer = 5, Ağırlık = 3

Nesne 3: Değer = 15, Ağırlık = 8

Nesne 4: Değer = 7, Ağırlık = 4

**Adımlar:**

1.Nesneleri değer/ağırlık oranına göre sıralıyoruz.

2. Sıralı nesneleri sırayla ele alıyoruz ve her bir nesneyi alıp almayacağımızı belirliyoruz. Eğer çantaya sığacaksa alıyoruz, sığmıyorsa almıyoruz.

**Adım 1: Nesne Sıralaması:**

Nesne 3: Değer = 15, Ağırlık = 8 (Değer/Ağırlık = 1.875)

Nesne 4: Değer = 10, Ağırlık = 7 (Değer/Ağırlık = 1.429)

Nesne 2: Değer = 7, Ağırlık = 4 (Değer/Ağırlık = 1.75)

Nesne 1: Değer = 5, Ağırlık = 3 (Değer/Ağırlık = 1.666)

**Adım 2: Nesneleri Çantaya Ekleyelim:**

Başlangıçta çanta boş, çanta kapasitesi 15.

Nesne 3: Değer = 15, Ağırlık = 8

Çantanın boş ağırlığı: 0, değeri: 0

Nesne 3'ü çantaya koyarsak:

Çantanın ağırlığı: 8, değeri: 15

Nesne 4: Değer = 10, Ağırlık = 7

Çantanın boş ağırlığı: 0, değeri: 0

Nesne 4'ü çantaya koyarsak:

Çantanın ağırlığı: 7, değeri: 10

Nesne 2: Değer = 7, Ağırlık = 4

Çantanın ağırlığı: 7, değeri: 10

Nesne 2'yi çantaya koyarsak:

Çantanın ağırlığı: 11, değeri: 17

Nesne 1: Değer = 5, Ağırlık = 3

Çantanın ağırlığı: 11, değeri: 17

Nesne 1'i çantaya koyarsak:

Çantanın ağırlığı: 14, değeri: 22

**Sonuç:** Çantaya en değerli eşyaları seçerek koyduğumuzda, toplam değer 22 olur ve toplam ağırlık 14 olur. Bu, greedy yaklaşımı ile çözümümüzdür.

**Karşılaştırma ve Yorumlar:**

**Dinamik programlama ile sırt çantası problemi, her durumu analiz ederek global optimal bir çözüm sunar. Ancak bu, daha fazla hesaplama ve bellek gerektirir.**

**Greedy yaklaşımı ise basit ve hızlıdır, ancak her zaman en iyi sonucu vermez. Bazı durumlarda global optimal çözüme ulaşamayabilir.**

**Küçük boyutlu problemler için her iki yöntem de kullanılabilir.**

**Ancak büyük boyutlu problemlerde dinamik programlama daha iyi performans gösterirken, Greedy yaklaşımı daha hızlı sonuç verebilir ve daha az bellek tüketebilir.**