T.C.
SAKARYA ÜNİVERSİTESİ
FEN BİLİMLERİ ENSTİTÜSÜ
BİLGİSAYAR VE BİLİŞİM MÜHENDİSLİĞİ ANABİLİM DALI
BİLİŞİM TEKNOLOJİLERİ PR. (YL) (UZAKTAN EĞİTİM)

VERİ YAPILARI VE ALGORİTMALAR 2.ÖDEV
SIRT ÇANTASI PROBLEMİ

Hazırlayan
SEDAT ÖZTÜRK
E235013168

Öğretim Üyesi
Prof. Dr. NEJAT YUMUŞAK

MAYIS 2024

Sırt Çantası Problemi

Sırt Çantası problemi, belirli bir ağırlık kapasitesine sahip bir çantaya, toplam değeri maksimize edecek şekilde eşyaların nasıl yerleştirileceğini belirleyen bir optimizasyon problemidir. Problemin iki yaygın türü vardır: 0/1 Sırt Çantası Problemi ve Kesirli Sırt Çantası Problemi.

0/1 Sırt Çantası Problemi: Bu versiyon da, eşyalar bölünemez; yani ya tamamen çantaya konulur ya da hiç konulmaz. Bu problem, genellikle dinamik programlama ile çözülür. Problemin çözümü için, eşyaların değerlerine ve ağırlıklarına göre bir matris oluşturulur ve bu matris, çantanın kapasitesine göre doldurulur. Matrisin her hücresi, o hücreye kadar olan eşyalarla ve o hücredeki ağırlık limitiyle elde edilebilecek maksimum değeri temsil eder.

Kesirli Sırt Çantası Problemi: Bu versiyon da, eşyalar bölünebilir; yani eşyaların bir kısmı çantaya konulabilir. Bu problem genellikle greedy (aç gözlü) algoritma ile çözülür. Eşyalar değer/ağırlık oranına göre sıralanır ve en yüksek orana sahip eşyalardan başlanarak çanta doldurulur.

Her iki problem de, çeşitli gerçek dünya senaryolarında karar verme süreçlerinde kullanılır. Örneğin, yatırım portföy optimizasyonu, yük taşımacılığı ve hatta bilgisayar bilimlerinde veri sıkıştırma gibi alanlarda uygulanabilir.

Dinamik Programlama Yöntemi ile

Algoritma Adımları

1. **Alt Problemleri Tanımlayın**: Her bir eşyanın çantaya alınıp alınmaması durumunu temsil eden alt problemleri tanımlayın.
2. **Tablo Oluşturun**: Eşyaların (i) ve çanta kapasitesinin (W) her bir kombinasyonu için bir tablo oluşturun.

EŞYA BİLGİLERİ (I)			**AĞIRLIK (W)**					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
			0					
			0					
			0					
			0					

3. **Tabloyu Doldurun**: Tabloyu doldurmak için aşağıdaki adımları izleyin;
    a. Tablonun ilk satırını ve sütununu 0 ile başlatın.
    b. Her eşya için (i), 1'den W'ye kadar olan her çanta kapasitesi (w) için, eşyanın çantaya alınıp alınmaması durumunda elde edilecek maksimum değeri hesaplayın.
    c. Eğer eşyanın ağırlığı (w_i) mevcut çanta kapasitesinden (w) fazla ise, bu eşya alınamaz. Önceki eşyanın değerini (i-1, w) alın.
    d. Eğer eşya alınabilirse, eşyanın değerini (v_i) ve çantadan eşyanın ağırlığını çıkardığınızda kalan kapasite için önceki maksimum değeri (i-1, w-w_i) ekleyin. Bu değeri, eşya alınmadan önceki maksimum değerle (i-1, w) karşılaştırın ve daha büyük olanı seçin.

Zaman Karmaşıklığı: Her bir eşya için, her bir kapasite değeri için hesaplama yapılır. Eğer (n) eşya ve (W) maksimum kapasite ise, algoritmanın zaman karmaşıklığı $O(nW)$ olur.

Bellek Karmaşıklığı: Dinamik programlama tablosu, (n) eşya ve (W) kapasite için bir hücre içerir. Bu nedenle, bellek karmaşıklığı da $O(nW)$ olur.

Örnek: Kapasitesi 5 kg olan bir çantaya eşyalar yerleştirilecektir. Çantanın kapasitesi tam kullanılmalı ve içerisindeki eşyaların değeri maksimum olması gerekmektedir. Aşağıda verilen tabloda eşyaların ağırlıkları ve değerleri verilmiştir.

Eşya	Ağırlık (kg)	Değer
A	2	6
B	1	5
C	3	10
D	2	8

Çözüm:

1. Verilen bilgilere göre tanımlama verileri doldurulur.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
2	6	A	0					
1	5	B	0					
3	10	C	0					
2	8	D	0					

2. A eşyası için mavi tablodaki ağırlık sütunlarını (1, 2, 3, 4, 5) dolduracağız.
    1. A eşyasının ağırlığı 2 olması sebebiyle bu sütuna 0 yazıyoruz. A eşyasını alamıyoruz.
    2. A eşyasının ağırlığı 2 olması sebebiyle ve bir üst hücrede yazan değerden daha avantajlı olduğu için A eşyasının değerini (6) bu sütuna yazıyoruz.
    3. A eşyasının ağırlığı 3 den küçük olduğu için ve çantada şu anda A eşyası olduğu için bu sütuna 6 değerini yazıyoruz.
    4. A eşyasının ağırlığı 3 den küçük olduğu için ve çantada şu anda A eşyası olduğu için bu sütuna 6 değerini yazıyoruz.
    5. A eşyasının ağırlığı 3 den küçük olduğu için ve çantada şu anda A eşyası olduğu için bu sütuna 6 değerini yazıyoruz.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
2	6	A	0	0	6	6	6	6
1	5	B	0					
3	10	C	0					
2	8	D	0					

3. B eşyası için mavi tablodaki ağırlık sütunlarını (1, 2, 3, 4, 5) dolduracağız.
    1. B eşyasının ağırlığı 1 olması sebebiyle bu sütuna bu sütuna B eşyasının değerini (5) bu sütuna yazıyoruz.
    2. B eşyasının ağırlığı 2 den küçük olması sebebiyle alınabilirdi. Ancak çantadaki A eşyası var ve onun değeri 5 den büyük olduğu için bu sütuna bir üstteki değeri yazıyoruz.
    3. Bu sütunun ağırlığına A ve B eşyasının sığdırabileceğimiz için A ve B eşyalarının değer toplamını yazıyoruz.

4. A ve B eşyasının toplam ağırlığı bu sütundan az olduğu için bu sütuna A ve B eşyalarının değer toplamını yazıyoruz.
5. A ve B eşyasının toplam ağırlığı bu sütundan az olduğu için bu sütuna A ve B eşyalarının değer toplamını yazıyoruz.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
2	6	A	0	0	6	6	6	6
1	5	B	0	5	6	11	11	11
3	10	C	0					
2	8	D	0					

4. C eşyası için mavi tablodaki ağırlık sütunlarını (1, 2, 3, 4, 5) dolduracağız.
    1. C eşyasının ağırlığı bu sütundan fazla olması sebebiyle bir üst sütundaki değeri bu sütuna yazıyoruz.
    2. C eşyasının ağırlığı bu sütundan fazla olması sebebiyle bir üst sütundaki değeri bu sütuna yazıyoruz.
    3. C eşyasının ağırlığı bu sütunla aynı ancak C eşyasının değeri bir üst sütundaki değerden az olduğu için bu sütuna bir üst sütundaki değeri yazacağız.
    4. Bu sütun C ve B eşyalarının ağırlığını toplamı kadardır. C ve B eşyalarının değerler toplamı 15 olması ve bir üstteki sütundan büyük olması sebebiyle bu sütuna C ve B eşyalarının değerler toplamı yazılır.
    5. Bu sütun C ve A eşyalarının ağırlığı toplamı kadardır. C ve A eşyalarının değerleri toplamı 16 olması ve bir üstteki sütundan büyük olması sebebiyle bu sütuna C ve A eşyalarının değerler toplamı yazılır.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
2	6	A	0	0	6	6	6	6
1	5	B	0	5	6	11	11	11
3	10	C	0	5	6	11	15	16
2	8	D	0					

5. D eşyası için mavi tablodaki ağırlık sütunlarını (1, 2, 3, 4, 5) dolduracağız.
    1. D eşyasının ağırlığı bu sütundan fazla olması sebebiyle bir üst sütundaki değeri bu sütuna yazıyoruz.
    2. D eşyasının ağırlığı bu sütun ile aynı ve bir üst sütundaki değerden fazla olduğu için bu sütuna D eşyasının değeri yazılır.
    3. Bu sütun D ve B eşyalarının ağırlığı toplamı kadardır. D ve B eşyasının değerler toplamı 13 olması ve bir üstteki sütundan büyük olması sebebiyle bu sütuna D ve B eşyalarını değerler toplamı yazılır.
    4. Bu sütun D ve A eşyalarının ağırlığı toplamı kadardır. Ancak D ve A eşyalarının değerler toplamı 14 olması ve bir üstteki sütundan küçük olması sebebiyle bir üstteki hücrenin değeri yazılır.
    5. Bu sütun C v D eşyasının ağırlığı toplamı kadardır. Ayrıca bu sütun D, B ve A eşyalarının ağırlığı toplamı kadardır. Burada karar şu şekilde verilir. Hangi eşyaların değer toplamı daha fazla ise o eşyanın değer toplamı tercih edilir. C ve D eşyalarını değer toplamı 18 dir. D, B ve A eşyalarının

değer toplamı 19 dur. D, B ve A eşyaların değer toplamı bir üst hücreden büyük olması sebebiyle bu sütuna D, B ve A eşyalarını değer toplamı yazılır.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)					
			0	**1**	**2**	**3**	**4**	**5**
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0
2	6	A	0	0	6	6	6	6
1	5	B	0	5	6	11	11	11
3	10	C	0	5	6	11	15	16
2	8	D	0	5	8	13	15	19

Yukarıdaki tablo dinamik programlama göre çantanın kapasitesi tam doldurulduğunda çantada maksimum 19 değerlilikte eşya konulabildiğini göstermektedir. Hangi eşyaların alındığını bulma durumu ise geriye dönük olarak gösterilebilir. Bu gösterim şekli şu şekildedir;

a. D eşyası için bir önceki hücreden ve bir üstteki hücreden farklı bir değer aldığı için bir üst hücreye çıkıp D eşyası ağırlığı kadar geriye dönülür. Yani bir üst hücre olan 16 ya gidilir ve D eşyası ağırlığı kadar hücre sola gidilir. Buradan 11 değerine ulaşılır. 11 değeri ile D eşyasının değeri 8 toplanır. Eğer 19 değerine ulaşıldıysa D eşyası seçilmiştir.
b. C eşyasının seçilip seçilmediği anlamak için bir önceki madde de kaldığımız hücreden devam edilir. Yani 11 değeri bir önceki hücreden geldiği tespit edildiği için C eşyası seçilmemiştir.
c. B eşyasının seçilip seçilmediği anlamak için kaldığımız hücreden bir üst hücreye çıkılır. Yani 11 değerine ulaşılır. Bu değer bir üst veya bir önceki hücreden gelmediği görüldüğü için bir üst hücreye çıkılır ve B eşyasının ağırlığı kadar hücre geri gidilir. Burada 6 değerine ulaşılır ve 6 değeri B eşyasının değeri ile toplanır. Eğer 11 değerine ulaşıldıysa B eşyası seçilmiştir.
d. A eşyasının seçilip seçilmediğini anlamak için bir önceki madde de kaldığımız hücreden devam edilir. Bir üst hücre ve bir önceki hücre 0 olması sebebiyle A eşyasının da seçildiği anlaşılır.

EŞYA BİLGİLERİ (I)			AĞIRLIK (W)						Seçilme Durumu
			0	**1**	**2**	**3**	**4**	**5**	
Eşya Ağırlığı	**Eşya Değeri**	**Eşya Adı**	0	0	0	0	0	0	
2	6	A	0	0	6	6	6	6	✓
1	5	B	0	5	6	11	11	11	✓
3	10	C	0	5	6	11	15	16	X
2	8	D	0	5	8	13	15	19	✓

Çantamı maksimum değer F(D, 5) = 19 ve içerisinde A, B ve D eşyaları ile doldurabilirim.

Greedy Yaklaşımı ile

Algoritma Adımları

1. Her eşya için değer/ağırlık oranını hesaplanır.
2. Bu birim değere göre eşyalar azalan şeklinde sıralanır.
3. Sıralanan eşyalar, çanta kapasitesi dolana kadar veya tüm eşyalar tükenene kadar çantaya eklenir.

Zaman Karmaşıklığı: Genellikle **O(n log n)** olarak ifade edilir. Bu, algoritmanın öğeleri değer/ağırlık oranına göre sıralaması ve ardından bu sıralamayı kullanarak çantaya eklenecek öğeleri seçmesi gerektiği için böyledir.

Bellek Karmaşıklığı: Genellikle **O(n)** olarak ifade edilir, çünkü algoritma tüm öğelerin listesini ve çanta kapasitesini hafızada tutmalıdır.

Örnek: 12 Kg olan bir çantaya eşyalar yerleştirilecektir. Aşağıda verilen tabloda eşyaların ağırlıkları ve değerleri verilmiştir. Bu durumda çantanın kapasitesini aşmayacak fakat çantayı en değerli kılacak seçim nasıl yapılmalıdır.

Eşya	Ağırlık	Değer
A	20	10
B	1	8
C	6	12
D	4	1
E	10	10

Çözüm:

1. Birim değeri her eşya için hesaplayacak bir sütun oluşturulur. Bu sütuna her eşya için değer/ağırlık oranını hesaplarız.

Eşya	Ağırlık	Değer	Birim Değer
A	20	10	10 / 20 = 0.50
B	1	8	8 / 1 = 8
C	6	12	12 / 6 = 2
D	4	1	1 / 4 = 0.25
E	10	10	10 / 10 = 1

2. Hesaplanan birim değere göre eşyalar azalan şeklinde sıralanır. Birim değeri en çok olan eşya B dir. Birim değeri en az olan eşya D dir.

Eşya	Ağırlık	Değer	Birim Değer
B	1	8	8 / 1 = 8
C	6	12	12 / 6 = 2
E	10	10	10 / 10 = 1
A	20	10	10 / 20 = 0.50
D	4	1	1 / 4 = 0.25

3. Sırasıyla en değerli eşyalardan en değersize doğru çanta kapasitesi dolan kadar ekleme yapılmalıdır. Eğer değersizler mümkünse alınmamalıdır. B ve C malzemeleri seçildiğinde çanta kapasitesinin dolmasına 5 birim kalmaktadır. E eşyası alındığında çanta kapasitesi aşılmış olacaktır. Bu durumda E eşyasının kapasiteyi aşmayacak şekilde 5 birim alınır. Eşya Değeri sütunu birim değer ile seçilen miktarın çarpımıyla hesaplanır. Sonuç olarak çantaya 1 birim B, 6 birim C ve 5 birim E eşyası seçildiğinde çantaya 25 değerinde eşya doldurulur.

Eşya	Birim Değer	Ağırlık	Seçilen Ağırlık	Eşya Değeri
B	8	1	1	8 * 1 = 8
C	2	6	6	2 * 6 = 12
E	1	10	5	1 * 5 = 5
A	0.50	20	0	0.5 * 0 = 0
D	0.25	4	0	0.25 * 0 = 0
		TOPLAM	**12**	**25**

Karşılaştırma ve Yorum

Dinamik programlama yöntemi, problemi tam olarak çözer ve uygun bir çözüm sağlar. Ancak, büyük ağırlık kapasiteleri ve eşya sayısı için zaman ve bellek kullanımı yüksek olabilir. Greedy yaklaşımı ise daha hızlıdır ve çok az bellek kullanır, ancak her zaman uygun bir çözüm sağlamaz. Greedy yöntemi, eşyaların bölünebilir olduğu durumlarda veya değer/ağırlık oranlarının benzersiz olduğu durumlarda iyi sonuçlar verebilir. Ancak, genel olarak, dinamik programlama sırt çantası problemi için daha güvenilir bir çözüm yöntemidir.