Türkiye Cumhuriyeti, Balıkesir Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü

Programlama Dilleri Dersi Dönem Sonu Sınavı, 26 Aralık 2017, 15:00, Sınav Süresi: 60 Dakika

Adı-Soyadı: **Cevap Anahtarı** **Öğrenci No:**

USB Teslim Ettim ☐ **USB Teslim Etmedim** ☐ **(……..'nın USB diskindedir)**

Soru	1 (Proje-40P)	2 (60P)	Toplam
Puan			

NOT: *Cevabınızı tek bir class dosyası içinde oluşturunuz ve class dosyanızın ismine adınızın-soyadınızın ilk harflerini yazdıktan sonra devamında öğrenci numaranızı ekleyiniz. Örneğin 201410404000 numaralı Kemal Aslan isimli öğrencinin hem projesinin hem de class dosyasının adı KA201410404000 olmalıdır. Sınav sonunda projenize ait class dosyasını (.java uzantılı dosya) üzerinde adınız, soyadınız ve öğrenci numaranız yazan USB disk içerisinde soru kağıdınızla birlikte teslim ediniz. Tedbir amacıyla, class dosyasına ek olarak, kodlarınızı bir Word veya text dosyası içerisine de kopyalayarak bu dosyayı da aynı şekilde isimlendirip USB disk içerisine atınız. Kendi USB diskini kullanmayan veya üzerinde ismi yazmayan USB disk teslim edenlerin dosyalarının kaybolması durumunda sorumluluk tamamen kendilerine aittir. Bu yönergeye uygun olmayarak yapılan teslimlerde puanınızın bir kısmını veya tamamını kaybedebilirsiniz. Başarılar.*

Yrd. Doç. Dr. İbrahim KÜÇÜKKOÇ

SORULAR

1 – Proje kapsamında çözdüğünüz problem ve geliştirdiğiniz programla ilgili kısaca bilgi veriniz.

Cevap 1:
Kişiye özel cevap.

2 – Seyahat etmek üzere valizini hazırlayan bir turist, havayolu şirketinden kaynaklanan valiz ağırlığı kısıtı sebebiyle yanına alabileceği eşyalardan bazılarını elemek durumundadır. Turistin yanına alabileceği eşyalar, bu eşyaların faydaları ve ağırlıkları aşağıdaki tabloda verilmiştir.

Eşya	1	2	3	4	5	6	7	8
Fayda (x10 TL)	5	8	3	6	7	9	4	5
Ağırlık (kg)	7	8	4	10	4	6	4	6

Havayolu şirketi, valiz ağırlığının maksimum **32 kg** olabileceğini belirtmektedir. Turistin, yanına alacağı eşyaların toplam faydasını maksimize etmek amacıyla hangi eşyaları alması gerektiğini belirleyecek bir *Brute Force (BF)* algoritması geliştiriniz (*NOT: BF algoritması ile ilgili detaylı bilgi arka sayfadadır*).

Programda kullanılması gereken tanımlamalar aşağıdaki gibidir:

```
static final int n = 8; // Eşya sayısı, sabit
static final int W =32; // Çanta kapasitesi, sabit
static final int[] Faydalar = { 5, 8, 3, 6, 7, 9, 4, 5}; //TL
static final int[] Agirliklar = { 7, 8, 4, 10, 4, 6, 4, 6};//kg
static int[] CozumAtamalar = new int[n]; //Her bir çözümü tutacak olan dizi; dizinin
                          elemanı 0 ise, ilgili eşya alınmadı, 1 ise, ilgili eşya alındı
static int cozumFayda; //Her bir olası çözümün faydası
static int cozumAgirlik; //Her bir olası çözümün ağırlığı
static int[] bestAtamalar = new int[n]; //En iyi çözümü tutacak olan dizi; dizinin elemanı
                                    0 ise, ilgili eşya alınmadı, 1 ise, ilgili eşya alındı
static int bestFayda; //En iyi çözümün faydası
static int bestAgirlik; //En iyi çözümün ağırlığı
```

Programınızda tüm ihtimalleri (*uygun veya uygun olmayan tüm olası çözümleri*) üreterek, bunların *cozumFayda* ve *cozumAgirlik* değerlerini hesaplattırıp, arasından kapasite kısıtını sağlayan ve faydayı maksimize eden en iyi çözümü bulmanız gerekmektedir (eğer aynı faydaya sahip iki farklı çözüm varsa, daha az ağırlığa sahip olan çözüm tercih edilmelidir).

İpucu: Tüm olası çözümleri üretmek için aşağıdaki kod bloğunu kullanıp bir `binary` String elde edebilirsiniz. Sonrasında bu Stringe göre, *CozumAtamalar[]* dizinizi oluşturup/düzenleyip, çözümünüzün faydasını (alınan eşyaların toplam faydası) ve ağırlığını (alınan eşyaların toplam ağırlığı) hesaplamanız gerekmektedir. Bir eşyanın alınıp alınmadığı, o eşyanın *CozumAtamalar[]* dizisindeki ilgili indeks karşılığının değerine bağlıdır (1 ise alındı, 0 ise alınmadı). Bunu tüm olası çözümler için yaparak, mevcut en iyi çözümle kıyaslamanız ve en iyi çözüme ait *bestFayda*, *bestAgirlik* ve *bestAtamalar[]* değerlerini güncel tutmanız gerekmektedir.

```
String binary = null;
for (int i=0; i<Math.pow(2, n);i++) {
   binary = Integer.toBinaryString(i);
   while (binary.length()<n) {
      binary = "0" + binary;
   }
   //System.out.println(binary);
}
```

Programınız, yukarıda verilen girdi ile çalıştırılıp sonlandığında aşağıdaki şekilde çıktı vermelidir:

```
>>Algoritma Sonlandırıldı
>>bestAtamalar: 0 1 1 0 1 1 1 1
>>bestFayda: 36
>>bestAgirlik: 32
```

Cevap 2:

```
public class BruteForceBinary {

    static final int n = 8; // Eşya sayısı, sabit
    static final int W =32; // Çanta kapasitesi, sabit
    static final int[] Faydalar = { 5, 8, 3, 6, 7, 9, 4, 5};
    static final int[] Agirliklar = { 7, 8, 4, 10, 4, 6, 4, 6};//kg
    static int[] CozumAtamalar = new int[n];
    static int cozumFayda;
    static int cozumAgirlik;

    static int[] bestAtamalar = new int[n];
    static int bestFayda;
    static int bestAgirlik;

    public static void main(String[] args) {
        System.out.println("Algoritma Başlatıldı ve Başlangıç Parametreleri Ayarlandı");
        bestFayda=-9999;
        bestAgirlik=-9999;
        String binary = null;

        for (int i=0; i<Math.pow(2, n);i++) {
            binary = Integer.toBinaryString(i);
            while (binary.length()<n) {
                binary = "0" + binary;
            }
            //System.out.println(binary);
```

```
                    for (int k=0; k<n;k++) {
                        CozumAtamalar[k]=Integer.valueOf(binary.substring(k, k+1));
                    }
                    //Yaz(CozumAtamalar, "CozumAtamalar");
                    FaydaHesapla(CozumAtamalar);
                }
                SonucYazdir();
        }

        private static void SonucYazdir() {
                System.out.println("Algoritma Sonlandırıldı");
                Yaz(bestAtamalar, "bestAtamalar");
                System.out.println("bestFayda: " + bestFayda);
                System.out.println("bestAgirlik: " + bestAgirlik);
        }

        private static void FaydaHesapla(int[] CozumAtamalar) {
                //Yaz(CozumAtamalar, "CozumAtamalar");
                cozumFayda=0;
                cozumAgirlik=0;
                for (int i=0; i<CozumAtamalar.length; i++) {
                    if (CozumAtamalar[i] == 1) {
                        cozumFayda+=Faydalar[i];
                        cozumAgirlik+=Agirliklar[i];
                    }
                }
                EnIyiKontrolEt(CozumAtamalar);
        }

        private static void EnIyiKontrolEt(int[] CozumAtamalar) {
                if ((cozumFayda > bestFayda && cozumAgirlik <=W) || (cozumFayda ==bestFayda &&
                cozumAgirlik <=W && cozumAgirlik <bestAgirlik )) {
                    bestFayda = cozumFayda;
                    bestAgirlik=cozumAgirlik;
                    for (int i = 0; i < n; i++) {
                        bestAtamalar[i] = CozumAtamalar[i];
                    }
                }


        }

        private static void Yaz(int[] dizi, String diziAdi) {
                System.out.print(diziAdi + ": ");
                for (int i=0; i<dizi.length; i++) {
                    System.out.print(dizi[i] + " ");
                }
                System.out.println();
        }
}
```

Brute Force (BF) Algoritması Çalışma Prensibi

BF algoritması, kaba kuvvet çözüm olarak da bilinir ve bir problemin çözümü için olabilecek tüm ihtimallerin tek tek denenerek içlerinden en iyisinin (amaç fonksiyonunu en iyileyen çözümün) belirlenmesi prensibine dayanır. *Örneğin,* yukarıdaki problem için eşya sayısı 3 olsaydı (*n=3*), ve bu eşyalara ait *Faydalar = {9, 8, 3}*; *Agirliklar = {7, 8, 4}* ve kapasite kısıtı da *W=12* (kg) olsaydı; 2^3=8 adet olası çözüm söz konusu olacaktı:

CozumAtamalar={0,0,0}	cozumFayda=0	cozumAgirlik=0
CozumAtamalar={0,0,1}	cozumFayda=3	cozumAgirlik=4
CozumAtamalar={0,1,0}	cozumFayda=8	cozumAgirlik=8
CozumAtamalar={0,1,1}	cozumFayda=11	cozumAgirlik=12
CozumAtamalar={1,0,0}	cozumFayda=9	cozumAgirlik=7
CozumAtamalar={1,0,1} *	**cozumFayda=12***	**cozumAgirlik=11***
CozumAtamalar={1,1,0}	cozumFayda=17	cozumAgirlik=15
CozumAtamalar={1,1,1}	cozumFayda=20	cozumAgirlik=19

Tüm bu olası çözümlerden, verilen kapasite kısıtını (12kg) aşmayan çözümler arasından en büyük faydaya sahip olan çözüm, probleminin en iyi çözümüdür. Örneğimizde *cozumFayda=12* ve *cozumAgirlik=11* değerlerine sahip *CozumAtamalar={1,0,1}* çözümü, problemimizin en iyi çözümüdür (tabloda * ile işaretlenmiştir). Yani, 1 ve 3 numaralı eşyalar alınırsa, kapasite kısıtını sağlayarak maksimum fayda elde edilmiş olunur. ***ÖNEMLİ: cozumAgirlik=15* ve *cozumAgirlik=19* değerlerine sahip çözümler, daha yüksek fayda sağlamasına karşın, kapasiteyi aştıkları için (>12kg) uygun çözüm değillerdir.**