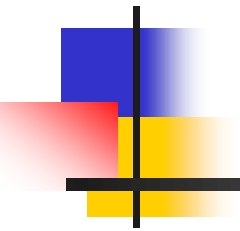

Programació Avançada

Tècniques de disseny d'algorismes

Exercici 26 pàgina 80

Tècnica: Backtracking

Armari

- Organitzar armari
- Joan és un desastre per a combinar la roba, ha decidit **organitzar les roba per conjunts**, posant en cada penjador el **pantaló i camisa que més conjunten.**
- Per a cadascun dels pantalons li ha indicat com de bé o de malament li combina amb cadascuna de les camises, aquesta informació s'ha codificat numèricament amb un valor de l'interval [0,9], indicant amb un valor de 0 que no hi combina gens i amb un valor de 9 que hi combina a la perfecció.
- **Mai** s'acceptarà un conjunt de pantaló-camisa amb una combinació numèrica **inferior a 3** (és una **restricció** del problema), serà millor deixar la prenda sense parella i en una propera sortida comprar més prendes.

Tècnica: Backtracking

Armari

- Es demana d'escriure un programa que trobi 2 solucions al problema, **les dues han de complir la restricció anterior**, però:
  - → una de elles ha de **maximitzar el valor numèric total** dels conjunts realitzats (aquest valor serà la suma del grau de combinació de la camisa amb el pantaló de cadascun dels conjunts) i,
  - → l'altra, la que pugui formar el **màxim** nombre d'aparellaments pantaló-camisa, independentment del grau de combinació però, que com a mínim, **hi hagi un aparellament d'un pantaló de pinces amb una camisa que no té botons**.
- S'ha de fer un **únic** mètode de backtracking per trobar ambdues solucions, encara que les solucions a trobar tinguin un criteri d'optimització diferent **(mateix espai de cerca)**.

Tècnica: Backtracking

Prenda.java

```
public class Prenda {
    private int identificador;
    private String descripcio; private int anyCompra;
    public Prenda( int identificador, int anyCompra){
        this.identificador=identificador;
        this.anyCompra=anyCompra;this.descripcio="No hi ha";
    }
    public Prenda( int identificador){this(identificador, 0);}
    public int getIdentificador(){ return identificador;}
    public int getAnyCompra(){ return anyCompra;}
    public void setIdentificador(int identificador){
        this.identificador=identificador;}
    public void setAnyCompra(int anyCompra){
        this.anyCompra=anyCompra;}
```

Tècnica: Backtracking

```
    public void setDescripcio(String des){ this.descripcio=des;}
    public boolean equals(Object o){
        if (o instanceof Prenda)
            return this.identificador==((Prenda)o).identificador;
        else return false;
    }
}
```

Prenda.java

Tècnica: Backtracking

Pantalo.java

```
public class Pantalo extends Prenda {
    private boolean pinces;
    private boolean portaCinturo;
    public Pantalo( int identificador, int anyCompra){
        super(identificador,anyCompra);pinces=false;
        portaCinturo=false;
    }
    public void setPinces( boolean pinces){
        this.pinces=pinces;
    }
    public void setPortaCinturo( boolean teCinturo){
        this.portaCinturo=teCinturo;}
    public boolean getPinces(){ return pinces;}
    public boolean getPortaCinturo(){
        return this.portaCinturo;}
    }
```

Tècnica: Backtracking

Camisa.java

```
public class Camisa extends Prenda {
    private boolean teBotons;
    public Camisa(int identificador, int anyCompra){
        super(identificador, anyCompra);
        teBotons=true;
    }
    public void setTeBotons( boolean teBotons){
        this.teBotons=teBotons;
    }
    public boolean getTeBotons(){
        return this.teBotons;
    }
}
```

Tècnica: Backtracking

Solucio.java

```
public class Solucio {
        private int numPantalons; // dimensió real de la taula següent
        private Pantalo TotsPantalons[]; // tots els pantalons
        private int numCamises; // dimensió real de la taula següent
        private Camisa TotesCamises[]; //totes les camises
        private int [][]grauAfinitat; /* les files representen pantalons i les
        columnes les camises, a la posició grauAfinitat[i][j] s'hi troba
        l'afinitat del pantaló i-èssim amb la camisa j-èssima – sempre serà
        un valor pertanyent a l'interval tancat [0,9].*/
        private ????? millor[]; //solució millor
        private ????? solucio[];  // solució  en construcció
        public  static void TotesDades(Pantalo TotsPantalons[], Camisa
        TotesCamises[]) {
            /*es determinen les dades de tots els pantalons i camises que té el Joan.
            Aquestes dades les emmagatzema consecutivament sobre els paràmetres
            TotsPantalons i TotesCamises respectivament. */       }
```

Tècnica: Backtracking

Solucio.java

```
    public Solucio(int numPantalons, int numCamises){
            /* Exercici 3*/
    }
    public String toString(){ /* Exercici 4*/
            /*sentències  que generen una cadena amb les 2 solucions
             trobades */
    }
    public ??? back2Solucions( /*paràmetres*/ ){  /*Exercici 6*/  }
} //fi classe
```

Aplicacio.java

```
public class Aplicacio{
        public static void main (String args[]){   /*Exercici 5*/
        }
}
```

Podeu afegir els atributs i/o mètodes que trobeu escaients en qualsevol de les classes,
→pels mètodes afegits cal fer la seva implementació i,
→pels atributs, cal explicar la seva funcionalitat.

Tècnica: Backtracking-Exercici 1

Anàlisi del problema

- Podem aplicar la tècnica del backtracking doncs la solució del problema la podem expressar com un conjunt de decisions. Tinguem present que **és un problema d'optimització**, les dues solucions a trobar són d'optimització, cal aplicar l'esquema de trobar la millor solució.
- En cada nivell, decidim **a cada pantaló X quina camisa li assignem**. Sempre disposarem de totes les camises per escollir però haurem de controlar-ne la repetició, doncs de cadascuna només en tenim un exemplar. **L'Amplada** de l'arbre: nombre de camises+1, doncs **cal considerar el cas de què cap camisa combini amb el pantaló tractat al nivell**. **L'Alçada** de l'arbre és exacta, fins que per a cadascun dels pantalons no haguem decidit quina camisa li assignem, donant la possibilitat de que sigui cap, no tindrem solució, per tant l'alçada coincideix amb el nombre de pantalons.
- **Serà solució quant tots els pantalons tinguin decisió assignada, és a dir quan arribem a una fulla de l'arbre**.

Backtracking-Exercici 2

Tipus taula Solucions

Enunciat: Determineu, decidiu vosaltres el tipus dels atributs taula anomenats *solucio* i *millor.*

Solució: muntar tripletes (com l'exercici de les factories)
Pantaló - Camisa solució 1 - Camisa solució 2

```
private class Tripleta{
    private Pantalo pantalo; private Camisa camisa1;
    private Camisa camisa2;
    public Tripleta(Pantalo pantalo){
          this.pantalo=pantalo; this.camisa1=null; this.camisa2=null;}
    public void setCamisa1(Camisa camisa){ this.camisa1=camisa;}
    public void setCamisa2(Camisa camisa){ this.camisa2=camisa;}
    public Pantalo getPantalo(){ return pantalo;}
    public Camisa getCamisa1(){ return camisa1;}
    public Camisa getCamisa2(){ return camisa2;}
} // fi classe privada
```

Per tant la declaració queda així:

```
private Tripleta millor[]; //millor solució
private Tripleta solucio[];  // solució en construcció
```

Backtracking-Exercici 3

constructor

```
public Solucio( int numPantalons, int numCamises){
        this.numPantalons=numPantalons;
        TotsPantalons=new Pantalo[numPantalons];
        this.numCamises=numCamises;
        TotsCamises= new Camisa[numCamises];
        TotesDades(TotsPantalons, TotesCamises); //omplenem
                                                 // dades
        solucio= new Tripleta[numPantalons];
        millor= new Tripleta[numPantalons];
        for (int i=0; numPantalons>i; i++){
                millor[i]= new Tripleta(TotsPantalons[i]);
                solucio[i]= new Tripleta(TotsPantalons[i]);
        }
```

Backtracking-Exercici 3

```
grauAfinitat= new int [numPantalons][numCamises];
for ( int i=0; i<numPantalons; i++)
        for ( int j=0; j<numCamises; j++)
                grauAfinitat[i][j]=(int)(9*Math.random());
} // fi constructor
```

constructor

Backtracking-Exercici 5

main

```
public static void main (String args[]){
    int pantalons= Keyboard.readInt();
    camises= Keyboard.readInt();
    Solucio s=new Solucio(pantalons, camises);
    boolean marcats[]=new boolean[camises];
    for (int i=0; i<camises; i++)
        marcats[i]=false;
    s.back2Solucions(0,marcats);
    System.out.println(s);
}
```

Podría ser un atribut

Backtracking-Consideracions

Per controlar millor les millors solucions del problema, d'una manera eficient afegeixo a la classe Solució els següents atributs:

```
private int acumuladorGrauMillor;  //solució 1
private int acumuladorGrauActual;  //de la solució amb construcció
private boolean hiEsCombinacioDemanada; /*solució 2 camisa
                                          sense botons – pantalons pinces*/
private int comptadorParellesFetesMillor; //solució 2
private int comptadorParellesFetesActual; //solució actual
```

En el constructor de la classe s'inicialitzaran aquests atributs de la següent manera:

```
acumuladorGrauMillor=-1; acumuladorGrauActual=0;
hiEsCombinacioDemanada=false;
comptadorParellesFetesMillor=-1;
comptadorParellesFetesActual=0;
```

Afegeixo atributs

Backtracking-Exercici 4

toString

```
public String toString(){
        String r="";
        if (acumuladorGrauMillor==-1)  //No existeix primera solució
                r+="No existeix primera solució";
        else for ( int i=0; i<numPantalons; i++){
                r+=millor[i].getPantalo().toString()+" ";
                if (millor[i].getCamisa1()!=null)
                        r+=millor[i].getCamisa1().toString();
                else r+="queda desaparellat \n";
              } // fi for
        if (!hiEsCombinacioDemanada) //No existeix segona solució
                r+="No existeix segona solució";
        else for ( int i=0; i<numPantalons; i++){
                r+=millor[i].getPantalo().toString()+" ";
                if (millor[i].getCamisa2()!=null)
                        r+=millor[i].getCamisa2().toString();
                else r+="queda desaparellat \n";
              } // fi for
        return r;
} //es necessari que les classes tinguin redefinit el mètode toString
```

Backtracking-Exercici 6

backtracking

```
public void back2Solucions(int k, boolean[] marcats){
    //  Esquema Millor
    int i=0; boolean valorAnterior=false; //per desfer
    while (i<numCamises+1){ //Recorregut
            // i==numCamises →sempre acceptable, cap camisa
             if (i==numCamises || !marcats[i] && grauAfinitat[k][i]>=3){
                    // lliure + assignar camisa i al pantaló k
                    if (i!=numCamises){
                            solucio[k].setCamisa1(TotesCamises[i]);
                            solucio[k].setCamisa2(TotesCamises[i]);
                            marcats[i]=true;
                            comptadorParellesFetesActual++;
                            acumuladorGrauActual+=grauAfinitat[k][i];
                            valorAnterior=hiEsCombinacioDemanada;
                            if (!hiEsCombinacioDemanada &&
                                    TotsPantalons[k].getPinces() &&
                                    !TotesCamises[i].getTeBotons())
                                      hiEsCombinacioDemanada=true;
                    } // if assignem camisa
```

Backtracking-Exercici 6

backtracking

```
if (k==numPantalons-1){ //solució
        //Millor primera solució
        if (acumuladorGrauActual>acumuladorGrauMillor){
            for (int j=0; j<numPantalons; j++)
                    millor[j].setCamisa1(solucio[j].getCamisa1());
            acumuladorGrauMillor= acumuladorGrauActual;
        }
        //Millor segona solució
        if (comptadorParellesFetesMillor<
                                comptadorParellesFetesActual &&
                                        hiEsCombinacioDemanada){
            for (int j=0; j<numPantalons; j++)
                    millor[j].setCamisa2(solucio[j].getCamisa2());
            comptadorParellesFetesMillor=
                                        comptadorParellesFetesActual;
         }
}
else back2Solucions(k+1,marcats);
```

Backtracking-Exercici 6

```
            // desfer- deixar tot com estava abans d'acceptar
            if (i!=numCamises){
                    solucio[k].setCamisa1(null);
                    solucio[k].setCamisa2(null);
                    acumuladorGrauActual-=grauAfinitat[k][i];
                    marcats[i]=false;
                    hiEsCombinacioDemanada=valorAnterior;
                    comptadorParellesFetesActual--;
            } //fi desfer
    } // fi acceptable
  } //fi  while
} // fi procediment
```

backtracking