Tècnica de backtracking

Exercici 28 de la pàgina 84

Tècnica de backtracking

Cortines tecnoCampus

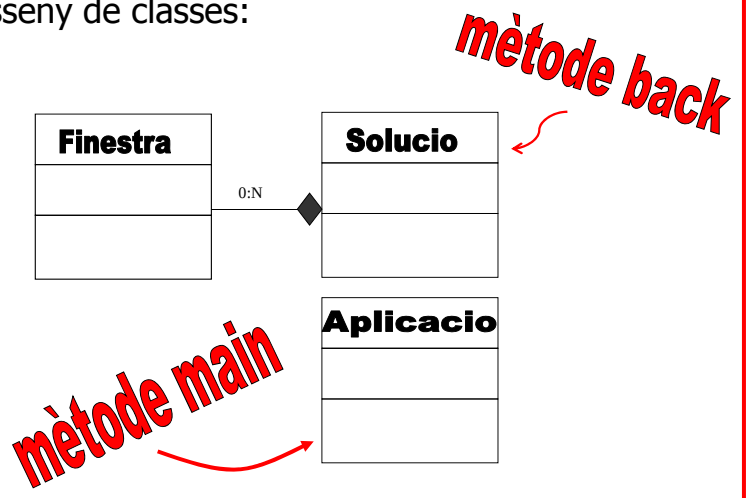
- **Enunciat**: Cortinatge dels edificis del tecnoCampus.

  Dades:
  - finestres amb la seva amplada
  - cada finestra necessita el **doble** de la seva amplada independent de l'alçada
  - metres de la peça de roba sencera

- **L'objectiu** és minimitzar costos, per això és necessari que determineu el **nombre mínim de peces** de roba que s'hauran de comprar per la seva confecció. **Esquema millor solució.**
- **Restricció**: Les peces s'han de comprar senceres i mai es podrà retornar els metres sobrants.

Tècnica de backtracking

- Disseny de classes:



Cortines tecnoCampus

Tècnica de backtracking

```
public class Finestra{
    private int identificador; private float alçada;
    private float amplada;
    public Finestra(int identificador){ this(identificador,0,0);}
    public Finestra(int identificador, float amplada, float alçada){
        this.identificador=identificador; this.amplada=amplada;
        this.alçada=alçada;}
    public int getIdentificador(){ return identificador;}
    public float getAmplada(){ return amplada;}
    public float getAlçada(){ return alçada;}
    public void setAmplada(float amplada){ this.amplada=amplada;}
    public void setAlçada(float alçada){ this.alçada=alçada;}
    public String toString(){
        return identificador+" "+alçada+" "+amplada; }
} //fi classe
```

Cortines tecnoCampus

Tècnica de backtracking

Cortines tecnoCampus

A decidir!

```
public class Solucio {
        private int numFinestres; // dimensió real de la taula següent
        private Finestra TotesFinestres[]; // totes les finestres
        private ????? millor[]; //solució millor
        private ????? solucio[];  // solució  en construcció
        public  static void TotesDades(Finestra TotesFinestres[]) {
            /*es determinen les dades de totes les finestres dels edificis
            del TecnoCampus. Aquestes dades les emmagatzema
            consecutivament sobre el paràmetre que té el mètode */   }
        public Solucio(int numFin, float mPeça){ /* Exercici 3*/ }
        public String toString(){ /* Exercici 4*/ }
        public ??? backSolucio( /*paràmetres*/ ){ /*Exercici 6*/ }
} //fi classe
public class Aplicacio{
        public static void main (String args[]){   /*Exercici 5*/}}
```

Tècnica de backtracking - Anàlisi

Cortines tecnoCampus

- **Backtracking?:** **és** un problema d'optimització, s'ha de trobar la millor distribució de peça-tela/cortina minimitzant la quantitat de peces de roba a usar, cal aplicar l'esquema de trobar la millor solució.
- **Decisió:** per cada finestra de quina peça li assignem la roba.
- **Acceptable**: Sempre disposarem de totes les peces per escollir però haurem de controlar-ne la quantitat que en resta en tot moment per a cadascuna de les peces.
- **Arbre:** **L'Amplada** nombre de finestres, doncs el pitjor cas ve donat quan es necessita un peça per cada finestra. **L'Alçada** de l'arbre és exacta, número de finestres.
- **Solució** quant per a totes les finestres tinguem decisió assignada, és a dir quan arribem a una fulla de l'arbre, i **completable** mentre no hi arribem.
- **Sempre trobarem solució**, a no ser que la peça de roba tingui una llargària inferior al doble de la finestra d'amplada màxima, que evidentment es suposarà que aquesta situació no es dona.
- **No usarem marcatge**, sempre que la llargària de la peça ho permeti, d'una mateixa peça poden fer més d'una cortina, i per tant les peces poden repetir-se. **Si controlarem la quantia** disponible de cadascuna de les peces en tot moment.

Cortines tecnoCampus

Backtracking – Més atributs

Per controlar més eficientment la solució del problema, afegeixo a la classe Solució els següents atributs:

```
private float metresPeça;  //indicada per
                           // l'usuari
private int numPecesMillor;   //Millor solució
private int numPecesActual;  //de la solució
                           // en construcció
private float []sobrantMillor; //metres sobrants
                           // de cada peça en la millor
```

Cortines tecnoCampus

Backtracking- Tipus atributs solució

Nova classe Parell, privada dins de la classe Solucio, amb atributs:

→ Finestra

→ Identificador peça roba **- cada peça amb un valor numèric**
// Є [0,numFinestres-1]

```
private class Parell{
    private Finestra finestra;private int identificador;
    public Parell(Finestra finestra){
        this.finestra=finestra; this.identificador=-1;}
    public void setIdentificador(int identificador){
        this.identificador=identificador;}
    public Finestra getFinestra(){ return finestra;}
    public int getIdentificador(){ return identificador;}
} // fi classe privada
private Parell millor[]; //millor solució
private Parell solucio[];  // solució en construcció
```

Backtracking - Constructor

Cortines tecnoCampus

```
public Solucio(int numFinestres, float metres){
    this.metresPeça=metres; //metres de cada peça de roba
    this.numFinestres=numFinestres;
    TotesFinestres=new Finestra[numFinestres];
    TotesDades(TotesFinestres); //omplenem dades
    solucio= new Parell[numFinestres];
    millor=new Parell[numFinestres];
    for (int i=0; numFinestres>i; i++){
                millor[i]= new Parell(TotesFinestres[i]);
                solucio[i]= new Parell(TotesFinestres[i]);
    }
    numPecesMillor=numFinestres+1;  //provocar el primer canvi !!!
    numPecesActual=0;  //de la solució amb construcció
    sobrantMillor=new float[numFinestres]; //sobrants Millor solució
} // fi constructor
```

Important!

El mètode tostring() ha d'imprimir de la millor solució el sobrant de cada rollo.

Backtracking - toString

Cortines tecnoCampus

```
public String toString(){
        String r="";
        if (numPecesMillor==numFinestres+1) return "No hi ha solució";
        else{    float metresSobrants=0;
                 for (int i=0;i<numFinestres; i++)
                          r+=millor[i].getFinestra()+"té assignada la peça
                                                  "+ millor[i].getIdentificador();
        r+="El nombre total de peces a comprar és de:
        "+numPecesMillor+"/n El sobrant de cada peça és el següent:\n";
        for (int i=0;i<numFinestres; i++){
                 if (sobrantMillor[i]!=metresPeça){
                          r+="Peça "+i + sobrantMillor[i] + "\n";
                          metresSobrants+=sobrantMillor[i];}
                 /*en cas contrari la peça no s'ha usat*/ } //fi for
        r+="amb un sobrat total de "+ metresSobrants; return r;}
```

Backtracking - main

Cortines tecnoCampus

```
public static void main (String args[]){
        int numFinestres= Keyboard.readInt();
        float metresPeça= Keyboard.readFloat();
        Solucio s=new Solucio(numFinestres,
                                        metresPeça);
        float marcats[]=new float[numFinestres];
                        //compte !!!, no boolean
        for (int i=0; i<numFinestres; i++)
                        marcats[i]=metresPeça;
        s.backSolucio(0,marcats);
        System.out.println(s);
}
```

Important

Backtracking – Mètode

Cortines tecnoCampus

```
public void backSolucio(int k, float[] marcats){  //  Esquema Millor
    for (int i=0; i<numFinestres; i++){ //Recorregut
        if (marcats[i]>=TotesFinestres[k].getAmplada()*2){  //hi ha prou de la i-èss
                if (marcats[i]==metresPeça)numPecesActual++; //nova peça
                solucio[k].setIdentificador(i);
                marcats[i]-= TotesFinestres[k].getAmplada()*2;
                if (k==numFinestres-1){ //solució
                        if (numPecesActual<numPecesMillor){ //Millor
                            for (int j=0; j<numFinestres; j++)
                                millor[j].setIdentificador(solucio[j].
                                                getIdentificador());
                            for (int j=0; j<numFinestres; j++)  //sobrants
                                sobrantMillor[j]=marcats[j];
                            numPecesMillor= numPecesActual;} //fi millor
                }
                else if (numPecesActual<numPecesMillor) //Podem!!!
                                backSolucio(k+1,marcats);
                solucio[k].setIdentificador(-1); //desfer
                marcats[i]+= TotesFinestres[k].getAmplada()*2;
                if(marcats[i]==metresPeça)numPecesActual--;
                //havíem encetat una nova
        } /* fi acceptable*/ } /*fi  for*/ } // fi procediment
```

i → peça de roba
k → finestra