Programació Avançada

Tècniques de disseny d'algorismes

Examen Curs 2016-17

Exercici 1: Tècnica Voraç

Tècnica: Voraç

- Un llibreter acaba d'obrir un negoci de venda de llibres de diferents temàtiques que s'ofereixen al públic ubicades en diferents prestatgeries.
- Es vol optimitzar l'espai del local i per tant es pretén **minimitzar** el nombre de prestatgeries a usar.
- **És imprescindible que tots els llibres d'una mateixa temàtica estiguin ubicats en una mateixa prestatgeria**.

Tècnica: Voraç

Més consideracions:

- Una prestatgeria pot estar ocupada per llibres de diferents temàtiques mentre tingui espai suficient per tots els seus corresponents llibres.
- Totes les prestatgeries tenen la mateixa capacitat i tots els llibres ocupen el mateix espai.
- Si es dona el cas de què una prestatgeria no té suficient capacitat per contenir tots els llibres d'una única temàtica aquesta no s'ubicarà a la botiga. El programa en haurà d'informar, indicant les temàtiques que no s'ha pogut ubicar i per tant no es posaran a la venda.

Tècnica: Voraç

- Es vol escriure un **programa voraç** que determini la distribució de les temàtiques en les diferents prestatgeries, **minimitzant** el nombre d'elles.

- La **funció de selecció** que heu d'usar és la següent:

  L'estratègia voraç ha de recorre totes les temàtiques a col·locar i ho farà de **major a menor número d'unitats** de llibres alhora d'assignar la prestatgeria.

  La prestatgeria seleccionada serà aquella que tingui espai lliure suficient per la temàtica completa i que a més deixi el menor espai lliure. Si no hi té cabuda en cap de les prestatgeries que ja tenen temàtiques assignades aleshores agafarà una prestatgeria nova (una buida, que no conté res). Si la temàtica necessita més espai que la capacitat màxima d'una prestatgeria aquesta temàtica no apareixerà a la solució final.

Tècnica: Voraç

```
public class Tematica implements Comparable{
    private String nom;
    private int numLlibres;
    public Tematica(String nom, int num){
        this.nom=nom; this.numLlibres=num;
    }
    public String getNom(){return nom;}
    public int getNumLlibres(){return numLlibres;}
    public int compareTo(Object o){
        Tematica t= (Tematica)o;
        if (numLlibres<t.numLlibres) return -1;
        if (numLlibres==t.numLlibres) return 0;
        return 1;
    }
    public String toString() {
        return "Nom de la temàtica " + nom +" amb: "+ numLlibres;
    }
}
```

Tècnica: Voraç

```
public class Prestatgeria {
    public static int capacitatTotal;
    private int espaiOcupat;
    private List<Tematica> contingut; // temàtiques que té assignades

    public Prestatgeria() {
        espaiOcupat = 0;
        contingut = new ArrayList<Tematica>();
    }
    public void addTematica(String tematica, int mida) {
        this.addTematica(new Tematica(tematica, mida));
    }
    public void addTematica(Tematica p) {
        contingut.add(p);
    }
    public int getEspaiOcupat() {
        return espaiOcupat;
    }
    public void ocuparEspai(int espai) {
        espaiOcupat += espai;
    }
    public String toString() {
        String r="";
        for (int i=0; i<contingut.size(); i++){
            r+=contingut.get(i)+"\n";
        }
        return r;
    }
}
```

Notació <E>

Tècnica: Voraç

```
public class Aplicacio {
    private Tematica[] candidats;
```

Afegir atributs per emmagatzemar la solució trobada

```
    private void omplenaDades() {
```

Aquest mètode demana les dades a l'usuari, totes les dades referents a les temàtiques i número de llibres que la composen deixant aquestes dades dins de l'atribut candidats.
El suposeu implementat

```
    }
    public Aplicacio(int quants) {
        candidats = new Tematica[quants];
```

Crea i/o inicialitza els atributs afegits

```
        omplenaDades();
    }
    public static void main(String args[]) {
        System.out
                .println("Indica la capacitat que tenen les prestatgeries que tens");
        Prestatgeria.capacitatTotal = Keyboard.readInt();
        System.out.println("Quantes temàtiques diferents vols ubicar? ");
        int quants = Keyboard.readInt();
```

Crea un objecte Aplicacio, invoca al mètode voraç i mostra la solució trobada

```
    }
```

Tècnica: Voraç

```
public ??? solucio(???){
```

Implementació mètode voraç. Decidiu vosaltres el retorn i paràmetres. En la seva execució és imprescindible que s'invoqui al mètode següent corresponent a la implementació de la funció de selecció, el que determina el següent candidat a considerar.

```
}
private ??? funcioSeleccio(???){
```

Implementació del mètode privat que aplica la funció de selecció indicada en l'enunciat. Determineu vosaltres el retorn i els paràmetres necessaris.

```
}
public String toString(){
```

Ha de crear i retornar una cadena amb la solució trobada. Ha d'indicar el número de prestatgeries necessàries, per a cadascuna d'elles les temàtiques que allotja i finalment les temàtiques que no es poden ubicar i per tant no es posen a la venda.

```
    }
}
```

Tècnica: Voraç

Exemple d'execució 1:

```
<terminated> Aplicacio [Java Application] C:\Program Files\Java\jre1.8.0_101\bin\ja
Indica la capacitat que tenen les prestatgeries que tens
30
Quantes temàtiques diferents vols ubicar?
8
Dades Entrades
**************
Nom de la temàtica tema2 amb: 20 llibres
Nom de la temàtica tema1 amb: 25 llibres
Nom de la temàtica tema0 amb: 30 llibres
Nom de la temàtica tema5 amb: 15 llibres
Nom de la temàtica tema4 amb: 15 llibres
Nom de la temàtica tema3 amb: 5 llibres
Nom de la temàtica tema6 amb: 10 llibres
Nom de la temàtica tema7 amb: 10 llibres
```

No s'ha de mostrar. S'ha posat perquè pugeu comprovar el resultat trobat

```
Necessitem 5 prestatgeries
Distribució:
Prestatgeria 0 hi ubiquem:
Nom de la temàtica tema0 amb: 30

Prestatgeria 1 hi ubiquem:
Nom de la temàtica tema1 amb: 25
Nom de la temàtica tema3 amb: 5

Prestatgeria 2 hi ubiquem:
Nom de la temàtica tema2 amb: 20
Nom de la temàtica tema7 amb: 10

Prestatgeria 3 hi ubiquem:
Nom de la temàtica tema4 amb: 15
Nom de la temàtica tema5 amb: 15

Prestatgeria 4 hi ubiquem:
Nom de la temàtica tema6 amb: 10

Temàtiques no ubicades:
Totes estan ubicades
```

Visualització de la Solució trobada

Tècnica: Voraç

Exemple d'execució 2:

```
<terminated> Aplicacio [Java Application] C:\Program Files\Java\jre1.8.0_101\bi
Indica la capacitat que tenen les prestatgeries que tens
20
Quantes temàtiques diferents vols ubicar?
8
Dades Entrades
**************
Nom de la temàtica tema2 amb: 20 llibres
Nom de la temàtica tema1 amb: 25 llibres
Nom de la temàtica tema0 amb: 30 llibres
Nom de la temàtica tema5 amb: 15 llibres
Nom de la temàtica tema4 amb: 15 llibres
Nom de la temàtica tema3 amb: 5 llibres
Nom de la temàtica tema6 amb: 10 llibres
Nom de la temàtica tema7 amb: 10 llibres

Necessitem 4 prestatgeries
Distribució:
Prestatgeria 0 hi ubiquem:
Nom de la temàtica tema2 amb: 20

Prestatgeria 1 hi ubiquem:
Nom de la temàtica tema4 amb: 15
Nom de la temàtica tema3 amb: 5

Prestatgeria 2 hi ubiquem:
Nom de la temàtica tema5 amb: 15

Prestatgeria 3 hi ubiquem:
Nom de la temàtica tema7 amb: 10
Nom de la temàtica tema6 amb: 10


Temàtiques no ubicades:

tema1
tema0
```

Solució trobada

Tècnica: Voraç. Enunciat

Fes

1.- (**0.5 punts**) Raona si la funció de selecció indicada trobarà o no la millor solució al problema. Justifica la resposta.

2.- (**3.5 punts**) Afegeix i implementa els mètodes pendents de la classe Aplicacio:

→2.1. - Indica els atributs que afegeixes. Important explicar l'ús que en faràs.

→2.2. - Inicialitza i/o crea en el constructor els atributs afegits.

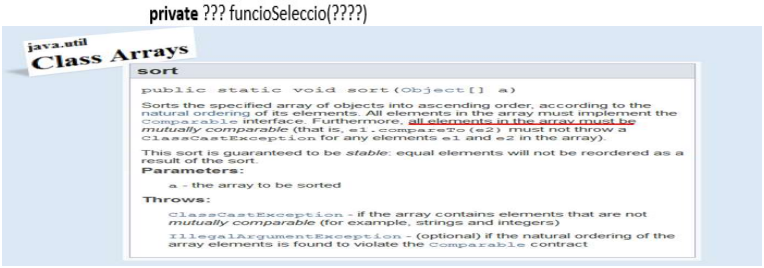
→2.3. - Completa el main creant l'objecte i invocat al mètode voraç i mostrant a pantalla la solució trobada.

→2.4.- Escriu el mètode voraç. Completa el prototipus indicant retorn i paràmetres. Imprescindible que invoqui al mètode que implementa la funció de selecció.

public ??? solucio(????????)

→2.5.- Escriu el mètode funció de selecció. Completa el prototipus indicant retorn i paràmetres. Imprescindible que implementi l'estratègia indicada en l'enunciat com a funció de selecció.

private ??? funcioSeleccio(????)



Tècnica: Voraç. Solució

Apartat 1

Sempre trobarà la millor solució al problema. El fet de tractar les temàtiques de més gran a més petita i assignar-li la prestatgeria que deixa el menor espai lliure garanteix de trobar sempre la millor solució, per tant mai resoldríem aquest exercici amb bactracking ja que trigaria més per trobar la mateixa solució.

Tècnica: Voraç. Solució

Apartat 2

```
public class Aplicacio {
    private Tematica[] candidats;
    //atributs afegits
    private Prestatgeria[] prestatgeries;
    private int quantes; // dim real
    private String noUbicats;

    public Aplicacio(int quants) {
        candidats = new Tematica[quants];
        omplenaDades();
        //Creem i inicialitzem atributs afegits
        prestatgeries = new Prestatgeria[quants]; // pitjor una per cada tema
        quantes = 0;
        noUbicats="";
    }
```

Afegit!

Tècnica: Voraç. Solució

```
public static void main(String args[]) {
    System.out
            .println("Indica la capacitat que tenen les prestatgeries que tens");
    Prestatgeria.capacitatTotal = Keyboard.readInt();
    System.out.println("Quantes temàtiques diferents vols ubicar? ");
    int quants = Keyboard.readInt();
    //Completar main
    Aplicacio m = new Aplicacio(quants);
    System.out.println("\nNecessitem " + m.solucio() + " prestatgeries");
    System.out.println("Distribució:\n"+ m);
    System.out.println("Temàtiques no ubicades: ");
    if (m.noUbicats!="")System.out.println("\n"+m.noUbicats);
    else System.out.println("Totes estan ubicades");
}
```

Tècnica: Voraç. Solució

```
public int solucio() {
    // ordena de menys a més capacitat les temàtiques
    Arrays.sort(candidats);
    for (int i = candidats.length - 1; i >= 0; i--) {
        int quina = funcioSeleccio(candidats[i]);
        switch (quina) {
        case -1: // no la ubiquem
            noUbicats+= candidats[i].getNom()+"\n";
            break;
        case -2: // estanteria nova
            prestatgeries[quantes] = new Prestatgeria();
            prestatgeries[quantes].addTematica(candidats[i]);
            prestatgeries[quantes]
                    .ocuparEspai(candidats[i].getNumLlibres());
            quantes++;
            break;
        default: // ho possem a quina
            prestatgeries[quina].addTematica(candidats[i]);
            prestatgeries[quina].ocuparEspai(candidats[i].getNumLlibres());
            break;
        }
    }
    return quantes;
}
```

Tècnica: Voraç. Solució

```
private int funcioSeleccio(Tematica p) {
    // buscar la prestatgeria on hi cap i sobre menys espai
    if (p.getNumLlibres() > Prestatgeria.capacitatTotal) return -1;//no hi cap
    int quina = -1;
    int espaiSobrant = Prestatgeria.capacitatTotal;
    for (int j = 0; j < quantes; j++) {
        int total = prestatgeries[j].getEspaiOcupat() + p.getNumLlibres();
        if (total <= Prestatgeria.capacitatTotal) {
            if (Prestatgeria.capacitatTotal - total < espaiSobrant) {
                quina = j;
                espaiSobrant = Prestatgeria.capacitatTotal - total;
            }
        }
    }
    if (quina == -1) return -2; // una nova
    else return quina;
}
```

Tècnica: Voraç. Solució

```
public String toString() {
    String re = "";
    for (int i = 0; i < quantes; i++) {
        re += "Prestatgeria " + i + " hi ubiquem: \n"
                + prestatgeries[i].toString()+"\n";
    }
    return re;
}
```