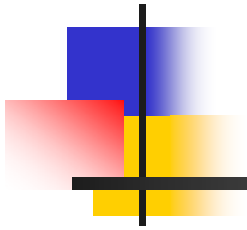


Programació Avançada

Exercici 29 pàgina 86

Festa Aniversari

Tècnica Voraç

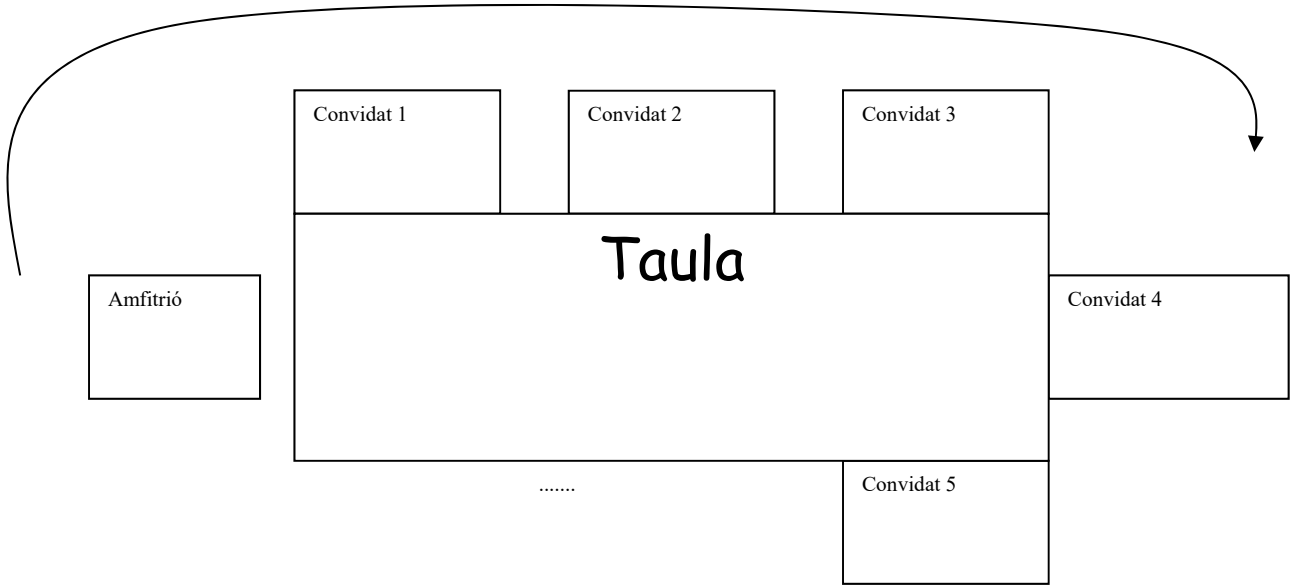
Tècnica Voraç: Aniversari

Enunciat

- Amb motiu del setantè aniversari de l'avi Pep, la família li vol organitzar un dinar sorpresa.
- S'ha de decidir la distribució dels convidats en la taula, per això es tindran en compte els graus d'afinitat entre els convidats, aquesta informació s'ha codificat numèricament amb un valor sencer dins l'interval [0,5], essent el valor de 5 el millor grau d'afinitat i 0 el pitjor.
- Es tracta d'escriure un **algorisme per trobar la millor distribució de convidats, amb suma total d'afinitats màxima**.
- Al cap de la taula s'hi assentarà l'amfitrió, l'algorisme ha de d'anar determinant a partir d'aquest i per a cadascun dels convidats que anem assentant a qui assentem al seu costat, féu aquesta distribució seguint el sentit de les agulles del rellotge.

Tècnica Voraç: Aniversari

Enunciat



Dades: nombre de convidats i nom de cadascun. Afinitats

Objectiu: distribuir-los maximitzant l'afinitat total (suma)

Tècnica Voraç: Aniversari

Anàlisi

Anàlisi

- 1.-**Candidats**: cadascun dels convidats, excepte l'amfitrió.
- 2.- **Funció selecció**: com es vol maximitzar el grau total d'afinitats començant per l'amfitrió buscarem el seu millor veí X a assentar a la seva esquerra (amb qui té major grau d'afinitat), del X el seu millor Y, i així successivament fins que tots els convidats estiguin assentats.
- 3.- **No** necessàriament trobarà la millor solució, però **sempre hi haurà solució.**

Tècnica Voraç: Aniversari

Estructura

```
public class Aniversari{
   private int afinitats[][]; //serà simètrica
   private String convidats[]; //noms convidats
   private int quants; // posicions plenes de la taula
   de convidats
   public Aniversari(int quantsConvidats){
       /*creació magatzems+entrada dades*/ }
   public static void main(String args[]){
   /*declaració i creació objectes i variables
   crida al mètode voraç + visualització de resultats*/ }
   public ????? Voraç(?????){
   /*crida obligatòria a la funció de selecció*/ }
   public ???? FuncioSeleccio(?????){
       /*determinar millor candidat*/ }
} // fi classe
```

Aniversari - Constructor

Solució

```
public Aniversari( int quantsConvidats){
    quants=quantsConvidats;                                   Entrada de dades
    convidats=new String[quants];
    System.out.println("Especifica el nom de l'amfitrió");
    convidats[0]=Keyboard.readString();
    for (int i=1; i<quantsConvidats; i++)
        convidats[i]=Keyboard.readString();
    sol= new int [quants]; //índex convidat!!!!!
    sol[0]=0; //assentem amfitrió primer lloc
    afinitats= new int [quants][quants];
    //a la diagonal no cal posar res !
    for (int i=0; i<quants; i++)
        for (int j=i; j<quants; j++){ //triangle superior
            if (i==j) afinitats[j][i]=-1; //sentinella
            else{ afinitats[i][j]=(int)(6*Math.random());
                  afinitats[j][i]=afinitats[i][j];}
    }
}
```

afegit atribut!!

Aniversari: Mètode main

Solució

```
public static void main(String args[]){
   int quants=Keyboard.readInt();
   Aniversari a=new Aniversari(quants);
   int guany=a.Voraç();
   System.out.println("L'afinitat global és de:" +guany);
   System.out.println(a);
}

public String toString(){
   String r="Llista començant per l'amfitrió i seguint
   el sentit de les agulles del rellotge";
   for (int i=0; i<sol.length; i++)
        r+=convidats[sol[i]]+" ";
   return r;
}
```

Aniversari: Mètode voraç

Observeu que la iteració només té una condició

solució

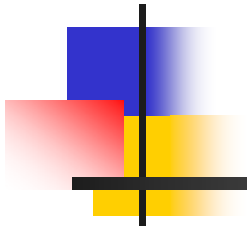
```
public int Voraç(){
   int afinitat=0;int i=1; //índex a emplenar
   //sempre hi ha solució- mentre quedin candidats
   while (quants>1){ //l'amfitrió no l'hem de col·locar
       sol[i]=FuncioSeleccio(sol[i-1]);
        quants--;
        afinitat+=afinitat[sol[i-1]][sol[i]];i++;
        //per no repetir candidat!!!! - Marcatge
        for (int u=0; u<afinitats.lenght; u++)
              afinitats[u][sol[i-1]]=-1;
   }
   return afinitat+=afinitats[0][sol[i-1]];???
}
```

Aniversari: Funció Selecció

Solució

```
public int FuncioSeleccio(int veí){
   // Determinar el valor major de la fila veí
   int quin=-1; //l'amfitrió no el podem escollir
   for (int i=1; i<afinitats.lenght; i++){
       if (quin==-1 ||
          afinitats[veí][i]>afinitats[veí][quin])
             quin=i;
   }
   return quin;
}
```

Busquem quin és més afí a veí. Recorregut per la fila veí



Programació Avançada

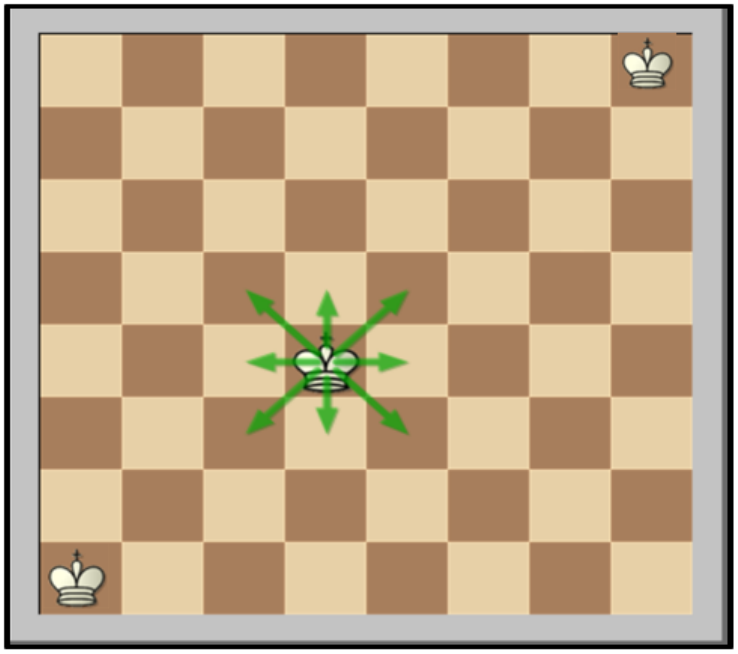
Exercici 18 pàgina 67

El rei coix

Tècnica Voraç

Tècnica Voraç: El rei coix

Enunciat



Posició final

Posició inicial

Quantitat d'euros en cada casella.

Cal maximitzar la quantitat final acumulada

- **Moviment 1**: la seva posició canviarà a la posició (x-1, y) **Amunt**
- **Moviment 2**: canviarà a la posició (x, y+1) **Dreta**
- **Moviment 3**: canviarà a la posició (x-1, y+1) **Diagonal amunt**

Tècnica Voraç: El rei coix

Enunciat

```
public class Taulell{
   private class Coordenada{
        private int x,y;
        public Coordenada(int x, int y){
               this.x=x; this.y=y;}
        public String toString(){
               return new String(x+""+y);}
        public int getX(){return x;}
        public int getY(){return y;}
   } // fi classe interna
   public static final int N=8; // taulell quadrat de 8x8
   private double [][]taulell; Atribut
   public Taulell(){ //constructor
   // sentències per la creació i omplenat del taulell
}
```

Tècnica Voraç: El rei coix

Enunciat

```
public static void main(String args[]){
  //creació objectes i variables
  int quantes = ??????;//crida procediment voraç
  for (int i=0; i<quantes; i++)
            System.out.println(?????);
  // solució a pantalla
}
```

Tècnica Voraç: El rei coix

Enunciat

```
public int voraç(?????){
   int x=7,y=0; //posició inicial del rei
   // emmagatzematge en el magatzem solució de la
   // casella de sortida
   // sentències esquema voraç obligatòriament s'ha de
   // cridar al mètode que aplica la funció de
   // selecció
   System.out.println(???????); //guany del rei amb la
   // solució trobada
   return ?????? //nombre de caselles que formen la
   // solució
}
```

Tècnica Voraç: El rei coix

Enunciat

```
private Coordenada FuncioSeleccio(int x, int y){
   // els paràmetres determinen la posició actual
   // del rei, determinar i retornar quin
   // moviment aplicar, coordenada on ubicar el
   // rei
   // Sentències
   }
}
```

Tècnica Voraç: El rei coix

Anàlisi

Anàlisi:

- El rei sempre podrà arribar a la casella destí. **Sempre** hi haurà solució! El nombre màxim de moviments a realitzar per arribar des de la sortida a la casella de destí són 14.
- **Candidats: els tres moviments permesos**. No sempre seran factibles els tres.
- **Funció Selecció:** s'escollirà aquell moviment que comporti a la casella que emmagatzemi més diners.
- No és una funció òptima.

Tècnica Voraç: El rei coix

Solució

```
public class Taulell{
   private class Coordenada{
      ........
   } // fi classe interna
   public static final int N=8;
   private double [][]taulell;
   public Taulell(){ //constructor
   // sentències per la creació i omplenat del taulell
       taulell=new float[N][N];
       for (int i=0; i<N; i++)
              for (int i=0; i<N; i++)
                 taulell[i][j]=Keyboard.readDouble();
   }
```

Tècnica Voraç: El rei coix

Solució

```
public static void main(String ars[]){
  //creació objectes i variables
  Taulell t=new Taulell();
  Coordenades []s = new Coordenades[15];
  // 14+1 per emmagatzemar també la casella de
  // sortida
  int quantes = t.voraç(s);
  for (int i=0; i<quantes; i++)
            System.out.println(s[i]);
  // solució a pantalla
}
```

Tècnica Voraç: El rei coix

Solució

```
public int voraç(Coordenades []s ){
   int x=7,y=0; //posició inicial del rei
   double guany=0.0;
   int quantes=1; s[0]=new Coordenada(7,0);
   while (x!=0 || y!=7){ //sempre hi ha solució
        Coordenada on=FuncioSeleccio(x,y);
        s[quantes]=on; quantes++;
        x=on.getX(); y=on.getY();
        guany+= taulell[x][y];
   }
   System.out.println(guany); //guany del rei
   return quantes;
}
```

Tècnica Voraç: El rei coix

Solució

```
private Coordenada FuncioSeleccio(int x, int y){
   if (x==0) return new Coordenada(x,y+1);
   if   (y==7)   return   new   Coordenada(x-1,y);
   // tenim els 3 moviements per escollir
   if (taulell[x-1][y+1]) > taulell[x][y+1] &&
            taulell[x-1][y+1] >taulell[x-1][y])
             return new Coordenada(x-1,y+1);
   if (taulell[x][y+1] > taulel[x-1][y])
            return new Coordenada(x,y+1);
   return new Coordenada(x-1,y);
}
} // fi classe
```

Tècnica Voraç: El rei coix

Millora!

Anàlisi

Anàlisi:

- El rei sempre podrà arribar a la casella destí. Sempre hi haurà solució!!!. El nombre màxim de moviments a realitzar per arribar des de la sortida a la casella de destí són 14.
- **Candidats:** els tres moviments permesos. No sempre seran factibles els tres.
- **Funció Selecció:** s'escollirà aquell moviment que comporti a la casella que emmagatzemi més diners. Només considerant la dreta o amunt ja que si hi ha més diners a la diagonal podríem accedir-hi posteriorment.
- No és una funció òptima.

Refer funció selecció