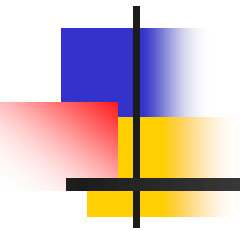


Programació Avançada

Tècnica Backtracking

Cartes set i mig

Tècnica: Backtracking

Enunciat. Exercici 1

- Al joc del set i mig s'hi juga amb les cartes de la baralla espanyola: 1,2,3,4,5,6,7,10,11 i 12 dels quatre pals (Monedes, Copes, Espases, Bastons). Es vol escriure un programa per obtenir 7.5, fent valer cada carta pel seu número de carta, excepte els 10, 11 i 12 que tenen un valor de 0.5. L'algorisme ha de trobar i visualitzar per **pantalla totes les combinacions possibles que facin 7,5**.

- Per cada carta de cada solució cal visualitzar: nom i pal. Per exemple:

  7 de Bastos – Rei de Copes

  /*és una solució al problema */

Tècnica: Backtracking

Estructura

```
public class Carta{
   private String nom; /*as, rei, cavall, ….*/
   private String pal; /*Monedes, Copes, Espases,
                                  Bastons */
   private float valor;/*valor que té la carta
                                  en el joc */
   public Carta(String n, String pa, int val){
       nom=n; pal=pa; valor=val;
   }
   public String getPal(){return pal;}
   public String getNom(){ return nom;}
   public float getValor(){ return valor;}
}
```

Tècnica: Backtracking

Estructura

```
public class Solucio{
   private Carta []cartes;
   //contidor de  totes les cartes de la baralla
   private Carta []solucio; //magatzem per construir la
                            // solució
   public Solucio(){
   //sentències que creen i emplenen el magatzem amb
   // totes les cartes i es crea i inicialitza  a nuls
   // el magatzem solució
   }
   public static void main(String args[]){
       //sentències
   }
   public  ??? Backtracking(??????????){
   //denoteu que aquest procediment NO és estàtic
       //sentències
   } //fi procediment
} // fi classe
```

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar, només, **cinc solucions** al problema. L'algorisme seguirà el mateix espai de cerca?
- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar la millor solució. La millor solució és aquella que **té més cartes**.
- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que el primer exercici però les solucions **no han de tenir cartes del mateix pal**.

Tècnica del backtracking-**Anàlisi**

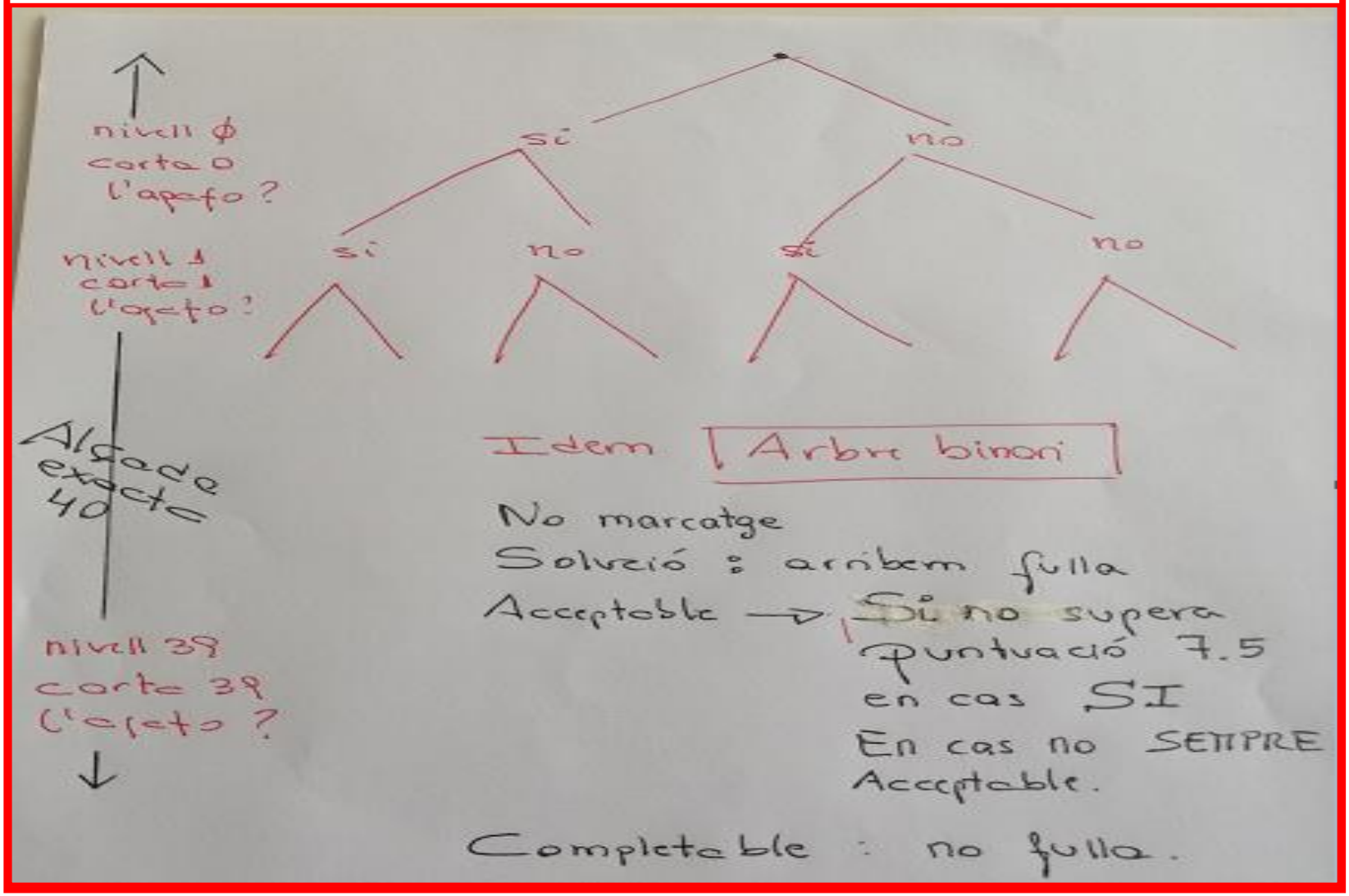
- Perquè podem aplicar backtracking
- Decisió:
- Quantes decisions:
- Acceptable:
- Solució:
- Completable:
- Espai de cerca:
- Esquema a aplicar:

```
public static void BackTotesSolucions ( TaulaSolucio  TS, int k){
        inicialitzem_valors_domini_nivell_k
        agafar_el_primer_valor_decisio_K
        while (quedin_valors_domini){
                if (valor_acceptable){ //no viola les restriccions
                    anotem_valor_a_la_solucio
                    if (solucio_final) escriure_solucio
                    else if (solucio_completable)
                        BackTotesSolucions(TS, k+1);
                    desanotem_valor
                } //fi if
                agafar_seguent_valor
                 // passem al següent germà a la dreta
        } // fi while
} // fi procediment
```

Estem al nivell **k** de l'arbre de cerca

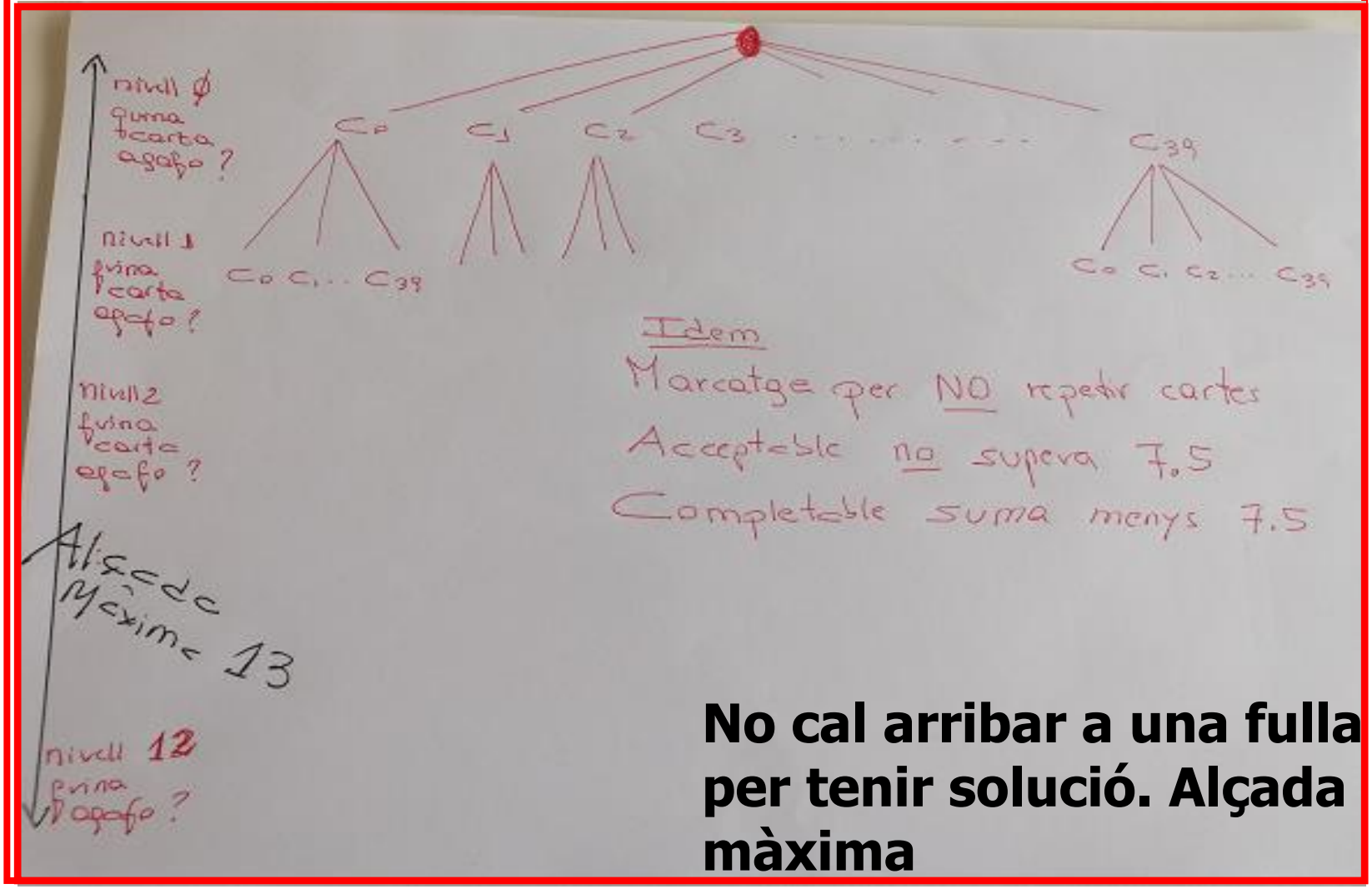
Tècnica del backtracking-**Anàlisi**

Proposta 1



Tècnica del backtracking-**Anàlisi**

Proposta 2



No cal arribar a una fulla per tenir solució. Alçada màxima

Tècnica: Backtracking. Proposta 2

Anàlisi

- Perquè podem aplicar backtracking
- La solució la podem expressar com una seqüència de decisions.
- Quina decisió? quina carta.
- Quantes decisions? màxim 13.
- Conjunt de decisions és acceptable? si sumen menys o igual de 7,5 (menys de 14).
- Completable? quan sumen menys de 7,5 i menys de 12 cartes seleccionades
- Solució? suma exactament 7,5 (no cal arribar fulla).
- Amplada arbre: 40 + **Tècnica del marcatge**.

Tècnica: Backtracking. Proposta 2

Exercici 1

```
public Solucio(){
   //sentències que creen i emplenen el magatzem amb
   //totes les cartes. Emplena l'atribut cartes
    solucio= new Carta[13];
    for (int i=0; i<13; i++) solucio[i]=null;
    marcats= new boolean [40];
    for (int i=0; i<40; i++) marcats[i]=false;
    suma=0;  //acumulador
}
public static void main(String args[]){
    Solucio sol = new Solucio();
    sol.Backtracking(0);
}
```

Atributs

Afegit!

Afegit!

nivell

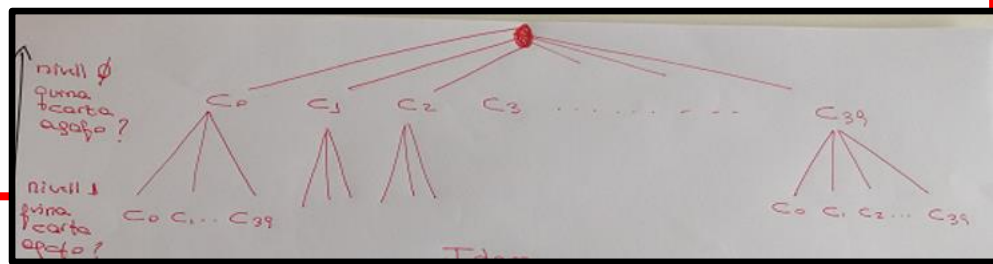
No calen tants paràmetres perquè hem afegit atributs propis

Tècnica: Backtracking

Exercici 1

```
public void Backtracking(int k){
  for (int i=0; i<40 ; i++){                    Amplada de l'espai de cerca
   if (!marcats[i] && suma+cartes[i].getValor()<=7.5F){   acceptable
        marcats[i]=true;
        suma+=cartes[i].getValor();
        solucio[k]=cartes[i];
        if (suma==7.5F){ /*és una solució*/              És solució
            for (int j=0; j<=k; j++){
                System.out.println(solucio[j].getNom()+" de "+
                        solucio[j].getPal());
            }
        } // if solució
        else if (k<12) Backtracking(k+1);   Alçada de l'espai de cerca
         /*queden cartes*/
        marcats[i]=false; suma-=cartes[i].getValor();
        solucio[k]=null;                                 desfer
   }
  } // fi for
} //fi procediment
```

Millor redefinir `toString` a la classe Carta



Tècnica: Backtracking

```
public void Backtracking(int k){
  for (int i=0; i<40 ; i++){
    if (!marcats[i] && suma+cartes[i].getValor()<=7.5F){
        marcats[i]=true;
        suma+=cartes[i].getValor();
        solucio[k]=cartes[i];
        if (suma==7.5F){ /*és una solució*/
            for (int j=0; j<=k; j++){
                System.out.println(solucio[j].getNom()+" de "+
                        solucio[j].getPal());
            }
        } // if solució
        else if (k<12) Backtracking(k+1);
          /*queden cartes*/
        marcats[i]=false; suma-=cartes[i].getValor();
        solucio[k]=null;
    }
  } // fi for
} //fi procediment
```

Amplada de l'espai de cerca

acceptable

És solució

Alçada de l'espai de cerca

És necessari?

desfer

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar, només, **cinc solucions** al problema. L'algorisme seguirà el mateix espai de cerca?

- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar la millor solució. La millor solució és aquella que té més cartes.

- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que el primer exercici però les solucions no han de tenir cartes del mateix pal.

Tècnica: Backtracking

Exercici 2

```
public Solucio(){
   //sentències que creen i emplenen el magatzem amb
   //totes les cartes
   // es crea i inicialitza a nuls el magatzem solució
   solucio= new Carta[13];
   for (int i=0; i<13; i++) solucio[i]=null;
   marcats= new boolean [40];
   for (int i=0; i<40; i++) marcats[i]=false;
   suma=0;  //acumulador
    cont = 0;
}
public static void main(String args[]){
   Solucio sol = new Solucio();
   sol.Backtracking(0);
}
```

Tècnica: Backtracking

Exercici 2

```
public void Backtracking(int k){
  for (int i=0; i<40 && cont<5; i++){   Cerca
   if (!marcats[i] && suma+cartes[i].getValor()<=7.5F){
        marcats[i]=true;
        suma+=cartes[i].getValor();
        solucio[k]=cartes[i];
        if (suma==7.5F){ /*és una solució*/
            for (int j=0; j<=k; j++){
                System.out.println(solucio[j].getNom()+" de "+
                        solucio[j].getPal());
            }
            cont++;
        } // if solució
        else if (k<12) Backtracking(k+1);
         /*queden cartes*/
        marcats[i]=false; suma-=cartes[i].getValor();
        solucio[k]=null;
      }
   } // fi for
} //fi procediment
```

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar, només, cinc solucions al problema. L'algorisme seguirà el mateix espai de cerca?

- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar **la millor solució.** La millor solució és aquella que **té més cartes**.

- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que el primer exercici però les solucions no han de tenir cartes del mateix pal.

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

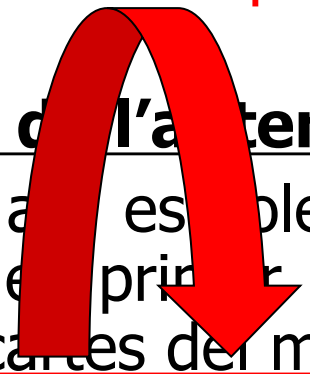
  Repetir l'exercici anterior, ara es volen trobar, només, cinc solucions al problema. L'algorisme seguirà el mateix espai de cerca?

- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar **la millor solució.** La millor solució és aquella que **té més cartes**.

- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que en el primer exercici però les solucions no han de tenir cartes del mateix pal.



La seqüència més llarga que suma 7,5 té 13 cartes, **trobem la primera de 13 cartes que suma 7,5**

Tècnica: Backtrackin

Exercici 3

```
public static boolean Back1Solucio( TaulaSolucio  TS, int k){
   boolean trobada=false;
   inicialitzem_valors_domini_decisio_k
   agafar_el_primer_valor
   while (quedin_valors && !trobada){
        if (valor_acceptable){  //no viola les restriccions
                anotem_el_valor_a_la_solucio
                if (solucio_final) trobada=true;
                else if (solucio_completable)
                        trobada=Back1Solucio(TS, k+1);
                if (!trobada) desanotem_el_valor
        } //fi if
        agafar_seguent_valor
         //passem al següent germà a la dreta
   } // fi while
   return trobada;
} // fi procediment
```

Esquema de cerca

Important

Opcionalment es pot fer tractament abans d'acab el procedimen

```
public Solucio(){
   //sentències que creen i emplenen el magatzem amb
   //totes les cartes
   // es crea i inicialitza a nuls el magatzem solució
   // Exercici 5
   solucio= new Carta[13];
   for (int i=0; i<13; i++) solucio[i]=null;
   marcats= new boolean [40];
   for (int i=0; i<40; i++) marcats[i]=false;
   suma=0;  //acumulador
   trobat=false;

}
public static void main(String args[]){
   Solucio sol = new Solucio();
   sol.Backtracking(0);
}
```

Afegit!

La primera solució de 13 cartes

Tècnica: Backtracking

Exercici 3

```
public void Backtracking(int k){
   for (int i=0; i<40 && !trobat ; i++){
    if (!marcats[i] && suma+cartes[i].getValor()<=7.5F){
        marcats[i]=true;
        suma+=cartes[i].getValor();
        solucio[k]=cartes[i];
        if (suma==7.5F && k==12){ /*és una solució*/
            for (int j=0; j<=k; j++){
                System.out.println(solucio[j].getNom()+" de "+
                        solucio[j].getPal());
            }
            trobat=true;
        } // if solució
        else if (k<12) Backtracking(k+1);
         /*queden cartes*/
        marcats[i]=false; suma-=cartes[i].getValor();
        solucio[k]=null;
      }
   } // fi for
} //fi procediment
```

No retona cap trobat!

Com ja he tractat la solució puc desfer sempre

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar, només, cinc solucions al problema. L'algorisme seguirà el mateix espai de cerca?.

- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar la millor solució. La millor solució és aquella que té més cartes.

- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que el primer exercici **però les solucions no han de tenir cartes del mateix pal**.

  Mateix espai de cerca?

Tècnica: Backtracking

Variants

- **Exercici 2 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar, només, cinc solucions al problema. L'algorisme seguirà el mateix espai de cerca?.

- **Exercici 3 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es vol trobar la millor solució. La millor solució és aquella que té més cartes.

- **Exercici 4 – Modificació de l'anterior.**

  Repetir l'exercici anterior, ara es volen trobar totes les solucions, a l'igual que el primer exercici **però les solucions no han de tenir cartes del mateix pal**.

  Mateix espai de cerca?

Varia l'alçada. Com a màxim les solucions han de tenir 4 cartes

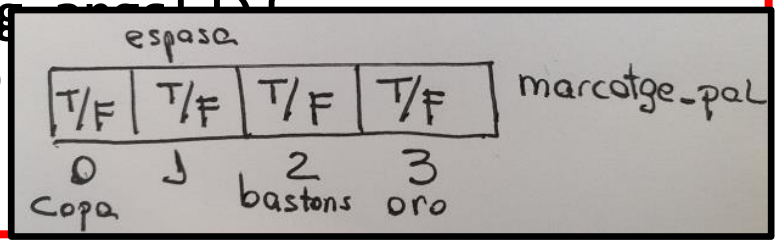
Tècnica: Backtracking

Exercici 4

```
public Solucio(){
   //sentències que creen i emplenen el magatzem amb
   //totes les cartes
   // es crea i inicialitza a nuls el magatzem solució
   // Exercici 5
   solucio= new Carta[13];
   for (int i=0; i<13; i++) solucio[i]=null;
   marcats= new boolean [40];
   for (int i=0; i<40; i++) marcats[i]=false;
   suma=0;   //acumulador
   boolean[] aux={false,false,false,false}
   marcatge_pal=aux;
}
public static void main(String
   Solucio sol = new Solucio()
   sol.Backtracking(0);
}
```

Atribut

Afegit!



Marcatge per cada pal

Tècnica: Backtracking

Exercici 4

```
public void Backtracking(int k){
  for (int i=0; i<40 ; i++){
    if (!marcats[i] && suma+cartes[i].getValor()<=7.5F &&
                                          ok_pal(cartes[i].getPal())){
          marcats[i]=true;
          marcar_pal(cartes[i].getPal(), true);
          suma+=cartes[i].getValor();
          solucio[k]=cartes[i];
          if (suma==7.5F){ /*és una solució*/
              for (int j=0; j<=k; j++){
                    System.out.println(solucio[j].getNom()+" de "+
                              solucio[j].getPal());
                    }

          } // if solució
          else if (k<3) Backtracking(k+1); //poda de l'arbre, 4 cartes màxim
           /*queden cartes*/
          marcats[i]=false; suma-=cartes[i].getValor(); solucio[k]=null;
          marcar_pal(cartes[i].getPal(), false);

       }
    } // fi for
} //fi procediment
```

Mètodes privats ajudants

Tècnica: Backtracking

Exercici 4

```
private boolean ok_pal(String pal){
   if (pal.equals("copa") && marcatge_pal[0])
            return false;
   if (pal.equals("espasa") && marcatge_pal[1])
            return false;
  if (pal.equals("bastons") && marcatge_pal[2])
            return false;
   if (pal.equals("oros") && marcatge_pal[3])
            return false;
   return true;
}
```

0: copa
1: espasa
2: bastons
3: oros

0: copa
1: espasa
2: bastons
3: oros

Tècnica: Backtracking

Exercici 4

```
private void marcar_pal(String pal
boolean valor){
  if (pal.equals("copa"))
                marcatge_pal[0]=valor;
  else if (pal.equals("espasa"))
                marcatge_pal[1] = valor;
  else if (pal.equals("bastons"))
                marcatge_pal[2] = valor;
  else if (pal.equals("oros"))
                marcatge_pal[3]= valor;
}
```

Tècnica: Backtracking

Exercici 4

```
public void Backtracking(int k){
  for (int i=0; i<40 ; i++){
    if (!marcats[i] && suma+cartes[i].getValor()<=7.5F &&
                                          ok_pal(cartes[i].getPal())){
          marcats[i]=true;
          marcar_pal(cartes[i].getPal(), true);
          suma+=cartes[i].getValor();
          solucio[k]=cartes[i];
          if (suma==7.5F){ /*és una solució*/
              for (int j=0; j<=k; j++){
                    System.out.println(solucio[j].getNom()+" de "+
                              solucio[j].getPal());
                    }

          } // if solució
          else if (k<3) Backtracking(k+1); //poda de l'arbre, 4 cartes màxim
           /*queden cartes*/
          marcats[i]=false; suma-=cartes[i].getValor(); solucio[k]=null;
          marcar_pal(cartes[i].getPal(), false);

       }
    } // fi for
} //fi procediment
```

Podem estalviar el marcats?