

Programació Avançada



Tècniques de disseny d'algorismes – Backtracking

Aplicació.

Filosofia de funcionament.

Avantatges e Inconvenients.

Esquema/es (3).

Exemples.



Tècnica del backtracking

- Habitualment es tracta de problemes **d'optimització amb ó sense restriccions. En els que:**
- La **solució** pot ésser expressada com una seqüència de decisions.
- Ha de ser possible, mitjançant algun mètode determinar si la seqüència de decisions és o no **factible** tenint en compte les restriccions del problema, és a dir, **poder determinar si la solució és completable.**
- També ha de ser possible determinar si una seqüència de decisions factible **és o no solució** al problema plantejat.
- **També** s'aplica la tècnica per resoldre problemes dels que volem trobar **TOTES** les solucions.



Tècnica del backtracking

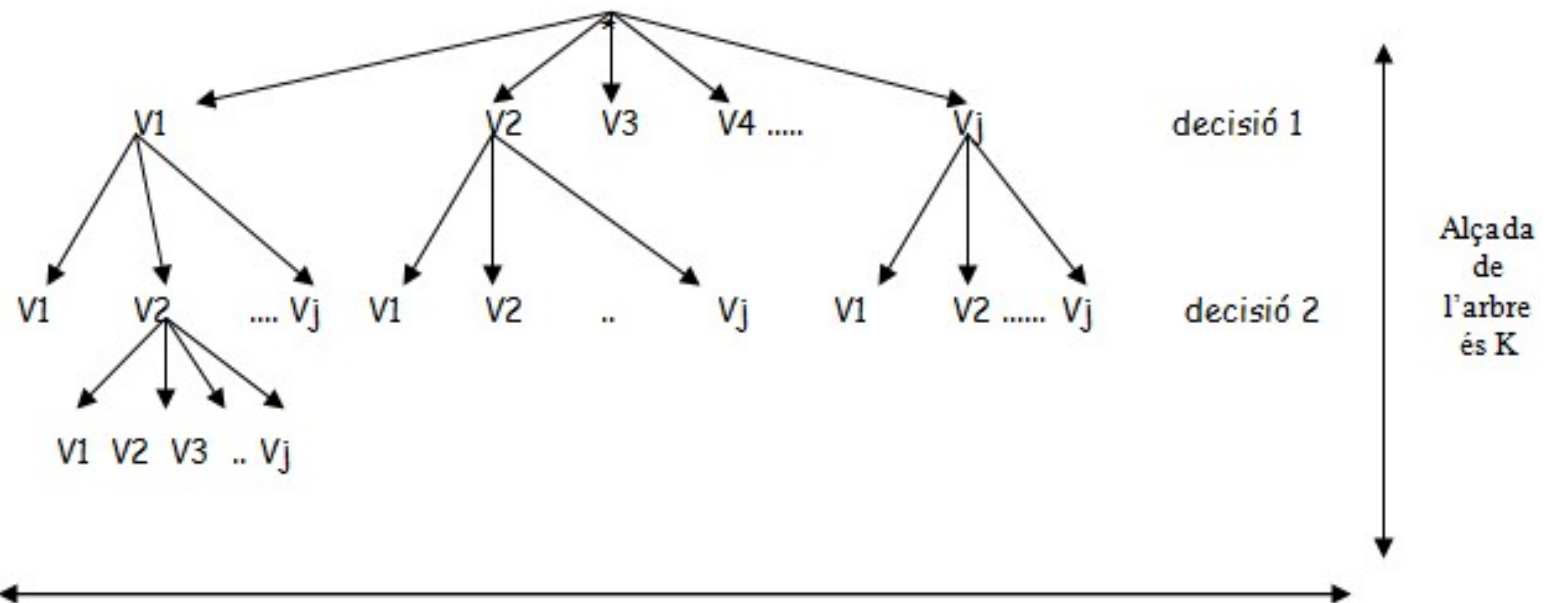
Filosofia

- La diferència entre aquesta tècnica i la voraç ve donada per la filosofia que usen.
- Aquesta tècnica genera **totes les seqüències possibles de decisions, d'una manera sistemàtica i organitzada**. Aquesta col·lecció de seqüències de decisions constitueix l'anomenat **espai de cerca** del problema.
- L'aplicació d'aquesta filosofia implica que si el problema **és resoluble**, és a dir té solució, segur que la troba.



Tècnica del backtracking

- Suposeu un problema que té associat K decisions a prendre, i en el que cada decisió té un domini format per j valors diferents. L'espai de cerca del problema és un arbre (en alguns casos podria ser un graf), amb els següents elements:





Tècnica del backtracking

- Si es vol aplicar la tècnica del backtracking a un problema, cal començar per **determinar l'espai de cerca** que té associat.
- **L'espai de cerca no necessàriament ha de ser únic** per cada problema. Cal escollir el més petit ó el menys costós.
- El temps que es tarda en recórrer (**generar**) tot l'arbre de cerca és sempre **exponencial**.
- L'algorisme pot:
 - cercar **una solució** qualsevol, normalment la primera que troba (problema de cerca).
 - cercar **totes** les solucions.
 - cercar la **millor solució**, problema d'optimització.



Tècnica del backtracking

- La tècnica del backtracking fa un recorregut en fondària de l'arbre de cerca partint sempre de l'arrel de l'arbre. Esquema general té diferències en funció de si volem 1, la millor ó totes les solucions però en tots els casos hi ha un cos comú. Aquest serà:
 - **Recursiu** (per baixar un nivell a l'arbre de cerca, es pot pensar una versió iterativa), i
 - l'estructura és una **iteració**. (per moure's dins d'un nivell)
- Recordeu les implemetacions dels recorreguts en fondària dels arbres **preordre**



Tècnica de

- La tècnica del b
fondària de l'ar
l'arrel de l'arbre
en funció de si vo
però en tots els ca

```
preorde (Arbre A) {  
  if (A no és buit) {  
    visitar arrel d'A  
    for (ai ∈ fills de l'arrel d'A) {  
      preorde (ai)  
    }  
  }  
}
```

- **Recursiu** (per baixar un nivell a l'arbre de cerca, es pot pensar una versió iterativa), i
- l'estructura és una **iteració**. (per moure's dins d'un nivell)
- Recordeu les implemetacions dels recorreguts en fondària dels arbres **preordre**



Tècnica del backtracking

Consideracions

- **Avantatges:**

- Sempre troba la solució al problema

- **Inconvenients**

- Ineficiència
- Temps exponencial

Per això és important
usar estratègies que la
millorin.

Podar l'arbre.



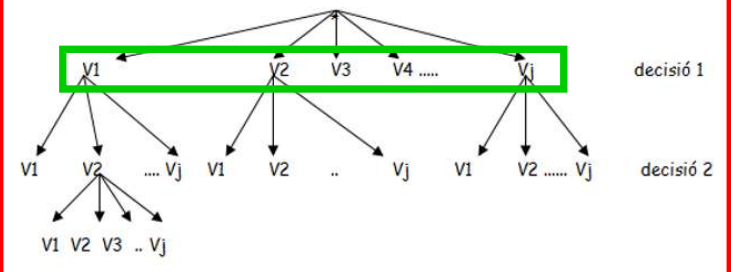
Tècnica del backtracking

Esquema

Esquema Backtracking

```
// ens trobem en un determinat nivell de l'arbre de
// cerca, generem el domini de valors del nivell i
// prenem un primer valor del domini
while (quedin_valors_domini){ //recorrem dins d'un nivell
    anotem_el_valor
    if (es_solucio) ..... //depèn del que volem
    else if (es_factible)
        Esquema Backtracking
        //crida recursiva, baixem un nivell
    desanotem_el_valor
    pendre_el_següent_valor
} //fi while
```

fiEsquema





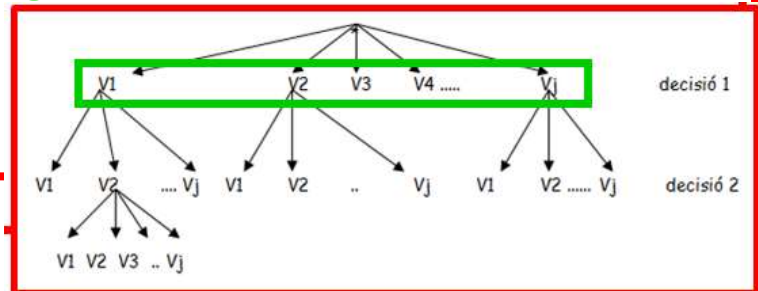
Tècnica de backtracking

Esquema TOTES

```
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

Quan tornem de la recursivitat la k manté el valor.
Incorrecte $k++$





Tècnica de backtracking

Exemple 1

- **Enunciat:** Trobar **totes** les combinacions, amb o sense repeticions, **de fins a 4 enters** de la col·lecció {1,2,3,5,7,8,9,11,13} que sumin 29. *restricció*
- **Consideracions:**
 - Una combinació és solució quan sumi 29 amb màx 4 números.
 - Una combinació és completable si suma menys de 29 i no s'ha superat el nivell 4, si és superior a 29 no cal continuar.

```
public class Exemple1{  
    public static void main(String args[]){  
        int []x={0,0,0,0};  
        BackTotesSolucions(x, 1);  
    }  
}
```

Indica el nivell de l'arbre



Tècnica del backtracking-**Anàlisi**

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



Tècnica del backtracking-Anàlisi

- Perquè podem aplicar backtracking

Hem de prendre decisions i la solució la podem expressar com una seqüència de decisions.

- Decisió:

Quin número trio?

- Quantes decisions:

Màxim 4

- Acceptable:

Si tinc menys de 4 números triats

- Solució:

Els números triats són màxim 4 i sumen 29

- Completable:

Els números triats són menys de 4 i sumen menys de 29

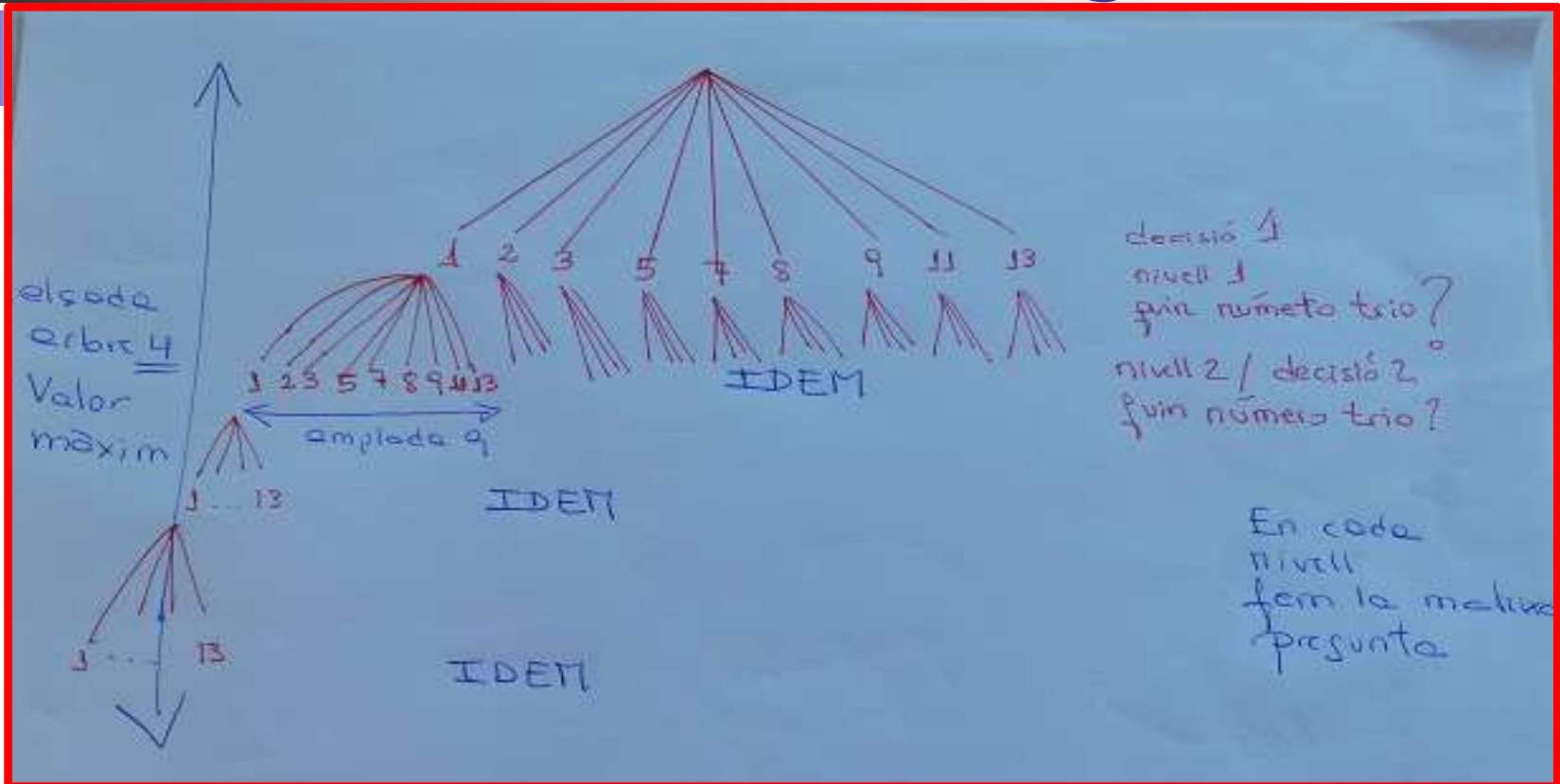
- Espai de cerca:

- Esquema a aplicar:

Trobar totes les solucions



Tècnica del backtracking-**Anàlisi**



Espai de cerca

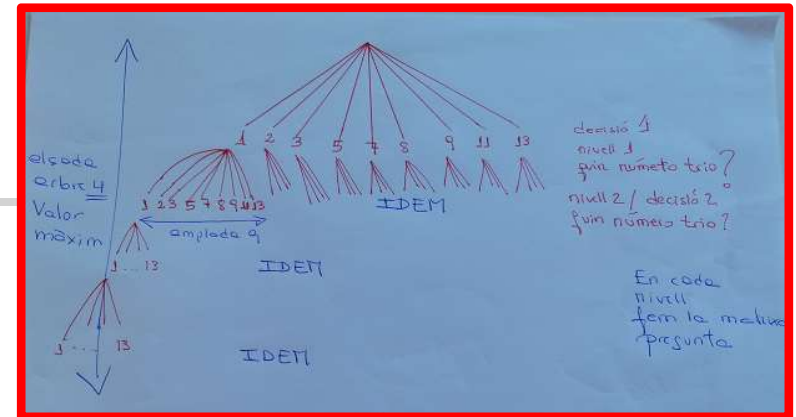
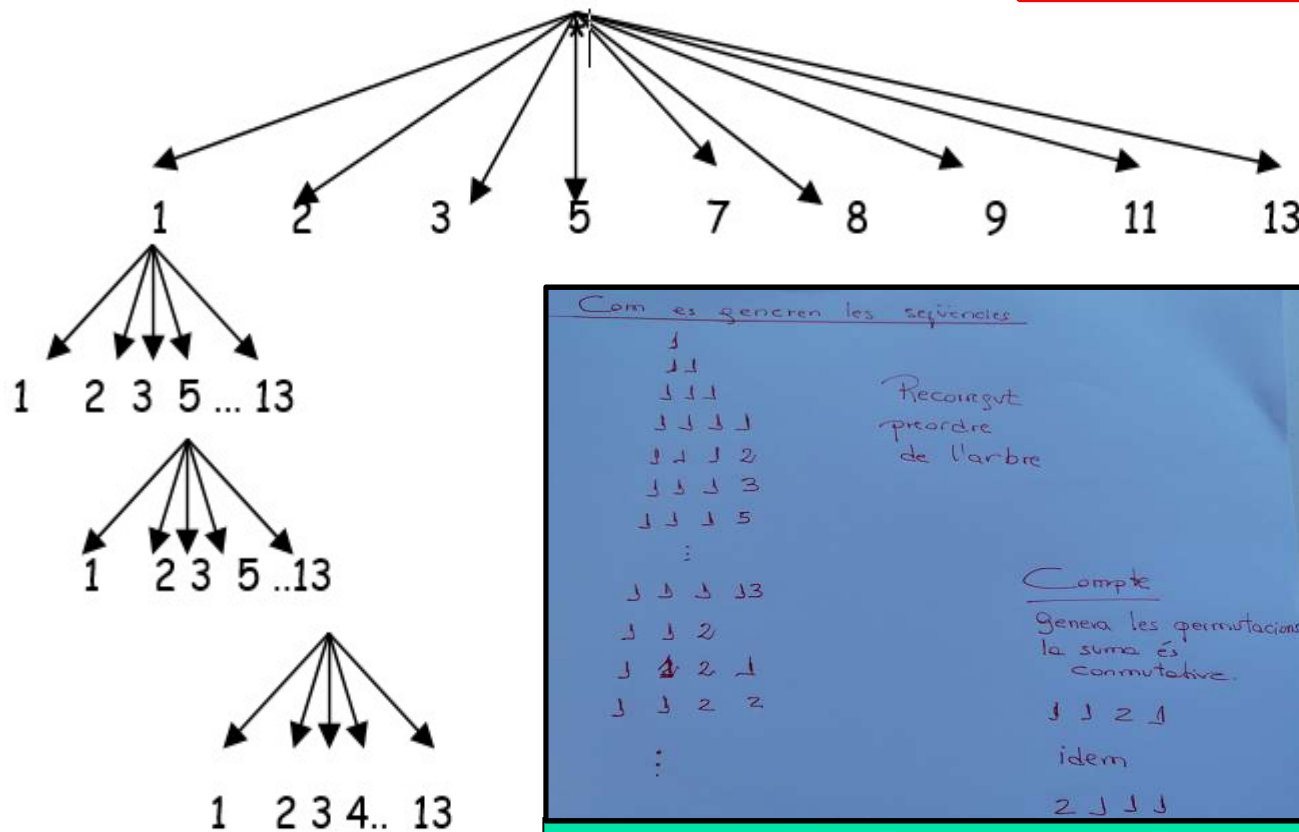
- Esquema a aplicar:

Trobar totes les solucions



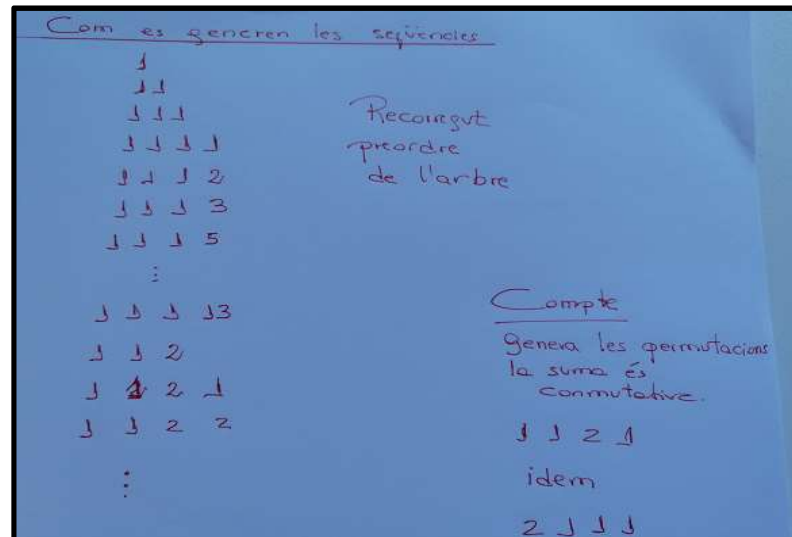
Espai de Cerca

Arbre homogeni



Quin número escullo?

4 nivells
màxim



En cada nivell tenim els mateixos valors per estriar



Tècnica de backtracking

```
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
            anatem_valor_a_la_solucio
            if (solucio_final) escriure_solucio
            else if (solucio_completable)
                BackTotesSolucions(TS, k+1);
            desanatem_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

Exercici TOTES

```
private static void BackTotesSolucions(int [] x, int k){
    int d[]={1,2,3,5,7,8,9,11,13};
    int j=0; //seleccionem el primer valor del domini
    while (j<9){ Quedin valors
        if (k<=4){ és acceptable?
            x[k-1]=d[j]; Anotem solució
            if (x[0]+x[1]+x[2]+x[3]==29) //solucio
                System.out.println("Solucio:
                                     "+x[0]+x[1]+x[2]+x[3]);
            else if (és completable? (x[0]+x[1]+x[2]+x[3]<29) //anem bé
                BackTotesSolucions(x, k+1);
            x[k-1]=0; //treiem valor Desanotem decisió
        } // fi if
        j++; //seleccionem al següent valor Agafem següent valor
    } //fi while
} // fi procediment
```


Millorem implementació --- ens allunyem de l'esquema!!!



Tècnica de backtracking-Millores

Exercici TOTES

```
private static void BackTotesSolucions(int [] x, int
k){ //Invocació principal → K=0
int d[]={1,2,3,5,7,8,9,11,13}; Millor un atribut static de
la classe

int j=0; //seleccionem el primer valor del domini
while (j<9){
if (k<=4){ //FORA
    x[k]=d[j];
    if (x[0]+x[1]+x[2]+x[3]==29) //solució
        System.out.println("Solucio:
                                "+x[0]+x[1]+x[2]+x[3]);

    else if (x[0]+x[1]+x[2]+x[3]<29 && k<3) //anem bé
        BackTotesSolucions(x,k+1);

    x[k]=0; //treiem valor
} // fi if FORA
    j++; //seleccionem al següent valor
} //fi while
} // fi procediment
```

Millorem implementació --- ens allunyem de l'esquema!!!



Tècnica de backtracking

Exercici TOTES

```
private static void BackTotesSolucions(int [] x, int
k){ //Invocació principal → K=0
int j=0; //seleccionem el primer valor del domini
while (j<9){
    x[k]=d[j];
    if (x[0]+x[1]+x[2]+x[3]==29) //solució
        System.out.println("Solucio:
                               "+x[0]+x[1]+x[2]+x[3]);
    else if (x[0]+x[1]+x[2]+x[3]<29 && k<3)
        BackTotesSolucions(x,k+1);
    x[k]=0; //treiem valor
    j++; //seleccionem al següent valor
} //fi while
} // fi procediment
```

Exercici

Totes les solucions han de
tenir exactament 4 elements

Millorem implementació --- ens allunyem de l'esquema!!!



Tècnica de backtracking

Exercici TOTES

```
private static void BackTotesSolucions(int [] x, int
k){ //Invocació principal → K=0
int j=0; //seleccionem el primer valor del domini
while (j<9){
    x[k]=d[j];
    if (x[0]+x[1]+x[2]+x[3]==29 && k==3) //solució
        System.out.println("Solucio:
                                "+x[0]+x[1]+x[2]+x[3]);
    else if (x[0]+x[1]+x[2]+x[3]<29 && k<3)
        BackTotesSolucions(x,k+1);
    x[k]=0; //no cal treure el darrer valor
    j++; //seleccionem al següent valor
} //fi while
} // fi procediment
```

Exercici

Totes les solucions han de
tenir exactament 4 elements



Tècnica de backtracking

Esquema UNA

```
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 el
tractament
abans d'acabar
el procediment



Exercici

Tècnica de backtracking

- **Enunciat:** Trobar **una** combinació, amb ó sense repeticions, de fins a 4 enters de la col·lecció {1,2,3,5,7,8,9,11,13} que sumi 29.
- Consideracions:
 - Una combinació és solució quan sumi 29 amb màxim 4 números.
 - Una combinació és completable si suma menys de 29 i no hem superat el nivell 4, si es superior a 29 no cal continuar.

Anàlisi del problema no varia

```
public class Exemple2{  
    public static void main(String args[]){  
        int [] x={0,0,0,0};boolean trobada;  
        trobada=Back1Solucio(x, 1);  
        if (!trobada)  
            System.out.println("No existeix solució");  
        } //fi main  
    }
```



Tècnica de backtracking

Exercici UNA

```
private static boolean Back1Solucio(int []x, int k){
    boolean trobada=false; int d[]={1,2,3,5,7,8,9,11,13};
    int j=0; //seleccionem el primer valor del domini
    while (j<9 && !trobada){
        if (k<=4){
            x[k-1]=d[j];
            if (x[0]+x[1]+x[2]+x[3]==29){ //solucio
                trobada=true;
                for (int i=0; i<k; i++)
                    System.out.println(x[i]);
            }
            else if (x[0]+x[1]+x[2]+x[3]<29) //anem bé
                trobada=Back1Solucio(x,k+1);
            if (!trobada) x[k-1]=0; //treiem valor
        } /* fi if*/ j++; //passem al següent valor
    } //fi while
    return trobada;
} // fi procediment
```

**Imprimir
la solución**



Tècnica de backtracking

Exercici UNA

```
private static boolean Back1Solucio(int []x, int k){
    //Crida principal → k=0
    boolean trobada=false;
    int j=0; //seleccionem el primer valor del domini
    while (j<9 && !trobada){
        x[k]=d[j];
        if (x[0]+x[1]+x[2]+x[3]==29){ //solucio
            trobada=true;
            for (int i=0; i<k; i++)
                System.out.println(x[i]);
        }
        else if (x[0]+x[1]+x[2]+x[3]<29 && k<3) //anem bé
            trobada=Back1Solucio(x,k+1);
        if (!trobada) x[k]=0; //treiem valor
        j++; //passem al següent valor
    } //fi while
    return trobada;
} // fi procediment
```

**Imprimir
la solució**

Podem fer les mateixes
millores aplicades a
l'exemple precedent



Tècnica de backtracking

Esquema MILLOR

```
public static void BackMillorSolucio( TaulaSolucio TS, int k,
TaulaSolucio Millor){
    inicialitzem_valors_domini_decisio_nivell_k
    agafar_el_primer_valor
    while (quedin_valors){ //Recorregut de tot l'
        if (valor_acceptable){ //no viola les re
            anotem_el_valor_a_la_solucio
            if (solucio_final)
                if (millor_solucio) Millor=TS; //else res
            else if (solucio_completable)
                BackMillorSolucio(TS, k+1, Millor);
            desanotem_el_valor
        } //fi if
        agafar_següent_valor
        //passem al següent germà a la dreta
    } // fi while
} // fi procediment
```

Sovint s'han d'afegir paràmetres per poder determinar si una solució és millor



Tècnica de backtracking

Exercici

- **Enunciat:** Trobar una combinació, amb ó sense repeticions, de fins a 4 enters de la col·lecció {1,2,3,5,7,8,9,11,13} que sumi 29 agafant el **mínim** nombre d'enters.

```
public class Exemple3 {  
    public static void main(String []args){  
        int x[]={0,0,0,0};  
        int millor[]={0,0,0,0};  
        int []minim=new int[1];minim[0]=5;  
        BackMillorSolucio(x, 1, millor, minim);  
        if (minim[0]==5)  
            System.out.println("No hi ha cap solució");  
        else for (int i=0; i<minim[0]; i++)  
            System.out.println(millor[i]);  
    }  
}
```

Quants nombres té la millor solució

Visualitzar solució



Tècnica de backtracking

Exercici

- **Enunciat:** Trobar una combinació, amb ó sense repeticions, de fins a 4 enters de la col·lecció {1,2,3,5,7,8,9,11,13} que sumi 29 agafant el **mínim** nombre d'enters.

```
public class Exemple3 {  
    public static void main(String []args){  
        int x[]={0,0,0,0};  
        int millor[]={0,0,0,0};  
        int []minim=new int[1];minim[0]=5;  
        BackMillorSolucio(x, 1, millor, minim);  
        if (minim[0]==5)  
            System.out.println("No hi ha solució");  
        else for (int i=0; i<4; i++)  
            System.out.print(millor[i] + " ");  
    }  
}
```

Quants nombres té
la millor solució

Perquè uso una taula i no
un `int`?

Perquè l'inicialitzo a 5?



Tècnica de backtracking

Exercici MILLOR

```
private static void BackMillorSolucio(int [] x, int k,
int[]millor, int []minim){
    int d[]={1,2,3,5,7,8,9,11,13};int j=0; //seleccionem el primer
    while (j<9){
        if (k<=4){
            x[k-1]=d[j];
            if (x[0]+x[1]+x[2]+x[3]==29){ //solucio
                if (minim[0]>k) {
                    for (int i=0; i<4; i++) millor[i]=x[i];
                    minim[0]=k;
                }
            }
            else if (x[0]+x[1]+x[2]+x[3]<29)//anem bé
                BackMillorSolucio(x,k+1,millor,minim);
            x[k-1]=0; //treiem valor
        } // fi if
        j++; //passem al següent valor
    } //fi while
} // fi procediment
```

**important
dígit a dígit**



Tècnica de backtracking

Exercici MILLOR

```
private static void BackMillorSolucio(int [] x, int k,
int[]millor, int []minim){ //Crida principal k=0
    int j=0; //seleccionem el primer
    while (j<9){
        x[k]=d[j];
        if (x[0]+x[1]+x[2]+x[3]==29){ //solucio
            if (minim[0]>k) {
                for (int i=0; i<4; i++) millor[i]=x[i];
                minim[0]=k;
            }
        }
        else if (x[0]+x[1]+x[2]+x[3]<29 && k<3) //anem bé
            BackMillorSolucio(x,k+1,millor,minim);
        x[k]=0; //treiem valor
        j++; //passem al següent valor
    } //fi while
} // fi procediment
```

**important
dígit a dígit**

Aplicades les millores
com a l'exemple
precedent



Tècnica de backtracking **Millora**

Exercici MILLOR

```
private static void BackMillorSolucio(int [] x, int k,
int[]millor, int []minim){ //Crida principal k=0
    int j=0; //seleccionem el primer
    while (j<9){
        x[k]=d[j];
        if (x[0]+x[1]+x[2]+x[3]==29){ //solucio
            if (minim[0]>k) {
                for (int i=0; i<4; i++) millor[i]=x[i];
                minim[0]=k;
            }
        }
        else if (x[0]+x[1]+x[2]+x[3]<29 && k<3 && K<minim[0]-1)
            BackMillorSolucio(x,k+1,millor,minim);
        x[k]=0; //treiem valor
        j++; //passem al següent valor
    } //fi while
} // fi procediment
```

Podem l'arbre

En la condició de completable podem afegir que si k és menor que minim[0] cal continuar cridant recursivament, en cas contrari la possible solució a trobar no serà mai millor

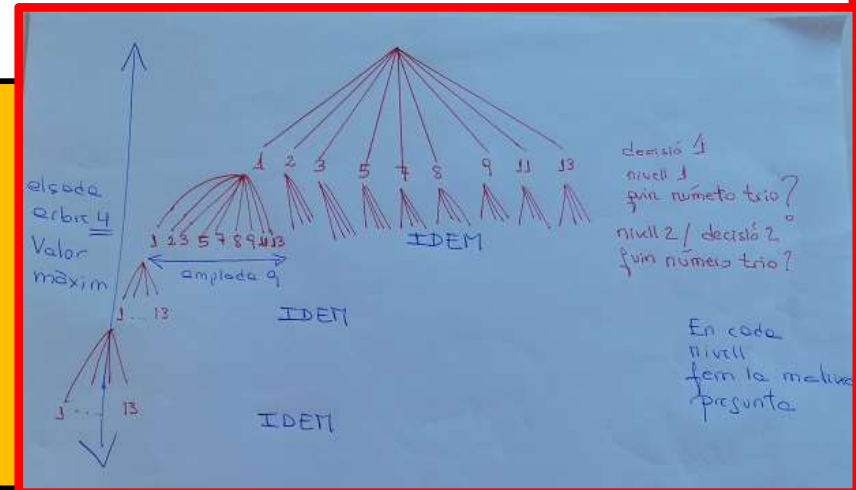


Tècnica de backtracking

Exercici

- **Enunciat:** Trobar la millor combinació, **sense** repeticions, de fins a 4 enters de la col·lecció $\{1,2,3,5,7,8,9,11,13\}$ que sumi 29 **agafant el mínim nombre d'enters**.
- **Solució 1** → Mateix espai de cerca, abans d'acceptar la decisió mirar si ja hi és o no al magatzem solució.

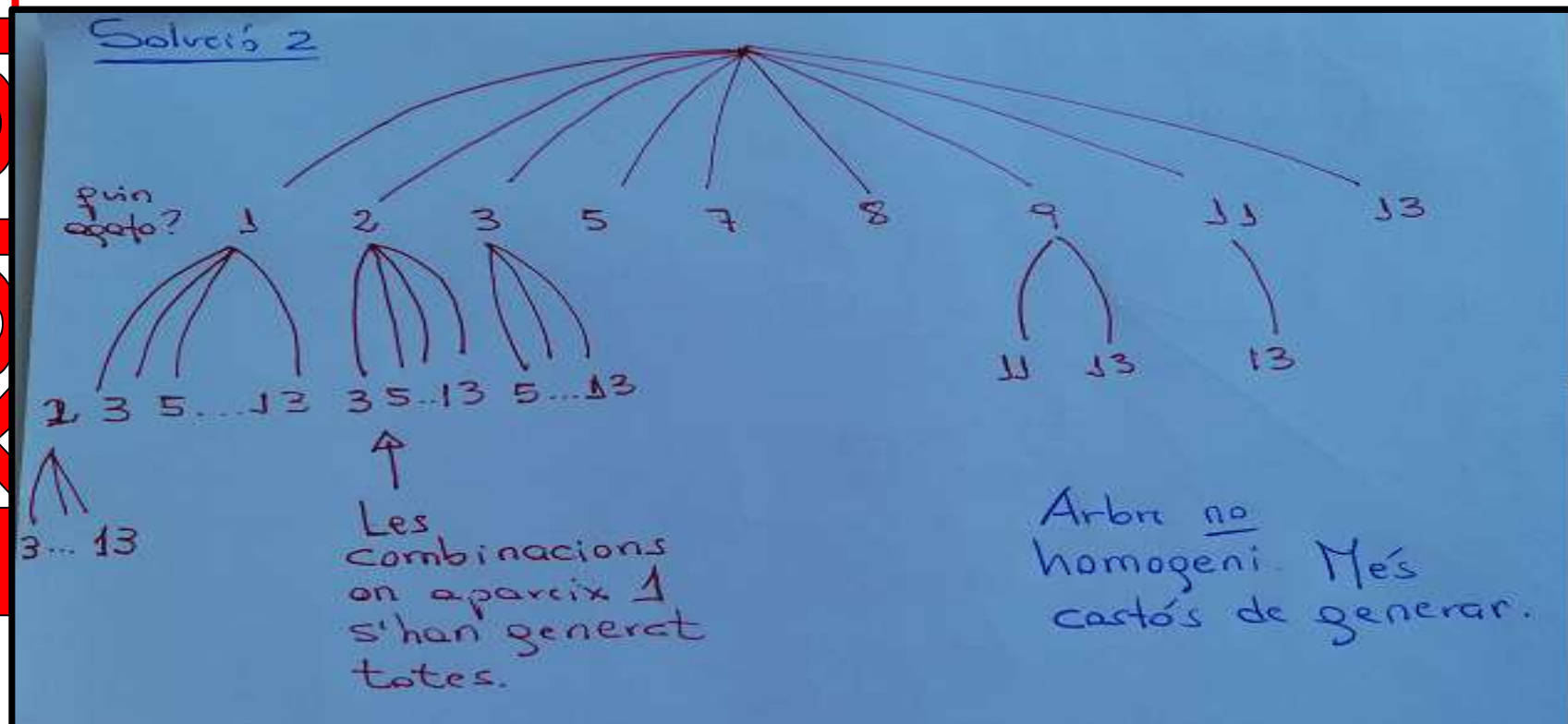
Està solucionat al dossier.
Mateix espai de cerca.
Ineficient cerca cada cop que s'ha de prendre una decisió per evitar repeticions





Tècnica de backtracking

- **Enunciat:** Trobar la millor combinació, **sense** repeticions, de fins a 4 enters de la col·lecció $\{1,2,3,5,7,8,9,11,13\}$ que sumi 29 **agafant el mínim nombre d'enters.**
- **Solució 2** → Espai cerca diferent.





Tècnica de backtracking

- **Enunciat:** Trobar la millor combinació, **sense** repeticions, de fins a 4 enters de la col·lecció {1,2,3,5,7,8,9,11,13} que sumi 29 **agafant el mínim nombre d'enters.**

Solució 3. Més espai memòria + eficient

1	2	3	5	7	8	9	11	13
0	1	2	3	4	5	6	7	8
T/F	T/F	T/F	T/F	T/F	T/F	T/F	T/F	T/F
0	1	2	3	4	5	6	7	8

marcats

↑ true ja s'escullit el número
false no s'ha escullit

taules relacionades per index

- **Solució 3 → Tècnica del marcatge.**

- Solució 1 + usar una taula de **booleans** per controlar els valors que ja formen part de la solució.
- La taula també pot ser de **int** si es permet repetir un número determinat de vegades.




Tècnica de backtracking


- **Enunciat:** Trobar **millor combinació**, **sense** repeticions, de fins a 4 enters de la col·lecció {1,2,3,5,7,8,9,11,13} que sumi 29 **agafant el mínim nombre d'enters**.

Solució 3 Tècnica del marcatge

```
public class Exemple4{  
    private static final int d[]={1,2,3,5,7,8,9,11,13};  
    public static void main(String args[]){  
        int [] x={0,0,0,0}; int Xmillor[]={0,0,0,0};  
        int []minim =new int[1]; minim[0]=5;  
        boolean marcats[] = new boolean[d.length];  
        for (int i=0; i<marcats.length; i++)marcats[i]=false;  
        BackMillorSolucio(x,Xmillor,0,minim, marcats);  
        //Imprimir solució  
    }  
}
```



darrera



marcatge

Exercici



Tècnica de backtracking

Exercici

```
// Detall imprimir solució
//després de la crida al backtracking
if (minim[0]==5)
    System.out.println("NO existeix cap solucio");
else{
    System.out.println("la millor solució té
                        ↓ :"+(minim[0]+1)+" xifres");
    for (int i=0; i<=minim[0]; i++)
        System.out.println (Xmillor[i]);
} //fi else
```



Tècnica de backtracking

Exercici SENSE REPES MILLOR

```
private static void BackMillorSolucio(int []x, int
[]Xmillor, int k, int[] minim, boolean []marcats){
    int j=0;
    while (j<d.length){    //d és atribut estàtic
        if (marcats[j]==false){// acceptable ←
            x[k]=d[j];
            marcats[j]=true; ←
            if (x[0]+x[1]+x[2]+x[3]==29){
                if (minim[0]>k){
                    for (int i=0; i<4; i++)
                        Xmillor[i]=x[i];
                    minim[0]=k;
                } //fi es millor
            } //fi és solució
            else    // no es solucio
```



Tècnica de backtracking

Exercici SENSE REPES MILLOR

```
else // no es solucio
    if ((x[0]+x[1]+x[2]+x[3]<29) && (k<3) &&
        k<minim[0]-1)
        //evitem crides recursives amb les
        // darreres condicions. PODEM !
        BackMillorSolucio(x, Xmillor, k+1,
            minim, marcats);
        x[k]=0;
        marcats[j]=false; //desfer valor
    } //fi if acceptable
    j++; //següent valor
} // fi while
} // fi procediment
```



Tècnica de backtracking

Exercici SENSE REPES MILLOR

Variants de l'Exercici

- 1.- Indica els canvis a fer si es vol que els valors es puguin repetir un màxim de dues vegades.
- 2.- Canvis a fer si només els números parells es poden repetir, tantes vegades com es vulgui.
- 3.- Totes les combinacions per ser solució, a més de sumar 29 han de contenir un número senar.