

Backtracking

Diseño y Análisis de Algoritmos



Universidad
Rey Juan Carlos

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Contenidos

- 1 Introducción
- 2 Árboles de búsqueda
- 3 N reinas
- 4 Otros problemas
- 5 Ramificación y poda
- 6 Poda alfa-beta

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Introducción

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

Backtracking - Vuelta atrás

- Estrategia para encontrar soluciones a problemas con restricciones definidos sobre espacios discretos (de elevado tamaño)
- Construye soluciones parciales progresivamente, las cuales deben cumplir las restricciones del problema
- El recorrido (en profundidad) tiene éxito si, procediendo de esta forma, se puede definir por completo una solución (en una hoja de un árbol de recursión)
 - Puede detenerse al encontrar una solución o seguir hasta encontrar todas
- Si en alguna etapa la solución parcial construida hasta el momento no se puede completar, se **vuelve atrás** deshaciendo la solución parcial.

Cartagena99

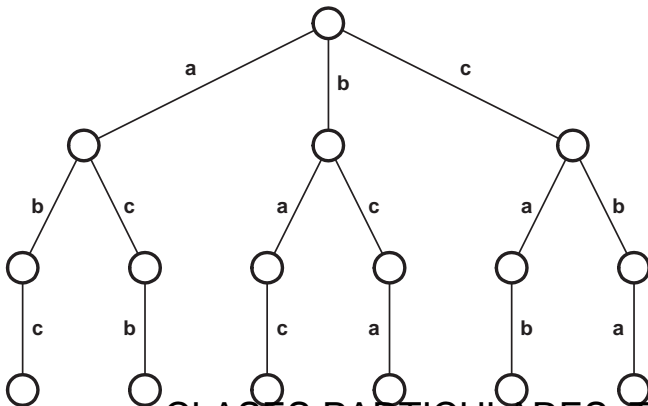
CLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Árboles de búsqueda

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

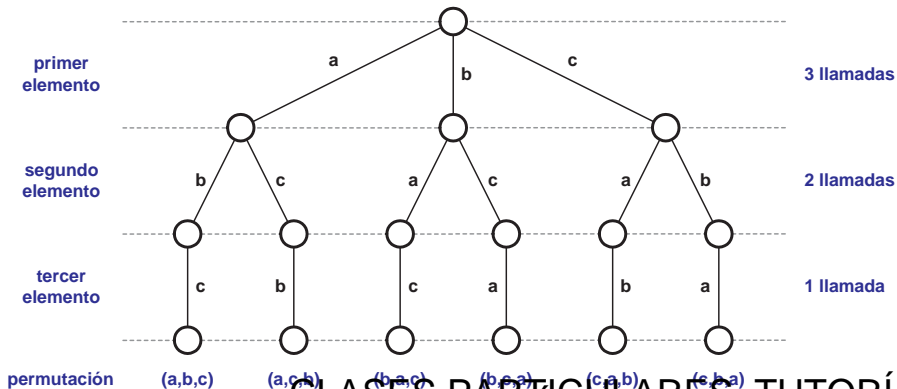
Permutaciones de $\{a, b, c\}$



Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

Permutaciones de $\{a, b, c\}$



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Permutaciones de $\{a, b, c\}$

- ¿Cuántas llamadas recursivas se hacen en cada nivel?
 - 3, 2, 1 (se podrían controlar con bucles)
 - Pero lo más fácil es generar siempre 3 posibles llamadas, y usar un vector de valores booleanos para ver si realmente se debe realizar la llamada recursiva
 - El vector de booleanos se puede pasar por valor o referencia
 - Referencia: habrá que deshacer los cambios al retroceder (C o Java)
 - Valor: no será necesario deshacer cambios
- El nivel de la llamada indica la posición del nuevo elemento a añadir
- Al llegar al último nivel (a las hojas) tenemos completada la

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 void permutaciones(int n){
2     int[] perm = new int[n];
3     boolean[] libres = new boolean[n];
4
5     for(int i=0; i<n; i++)
6         libres[i] = true;
7
8     perms(n, 0, perm, libres);
9 }
10
11 void imprimir(int[] v){
12     for(int i=0; i<v.length; i++)
13         System.out.print(v[i]+" ");
14 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```

1 void perms(int n, int i, int[] solucion, boolean[] xs){
2   for(int k=0; k<n; k++){
3     if(xs[k]){
4       solucion[i] = k;
5       xs[k] = false;
6
7       if(i==n-1)
8         imprimir(solucion);
9       else
10        perms(n, i+1, solucion, xs);
11
12      xs[k] = true;
13    }
14  }

```

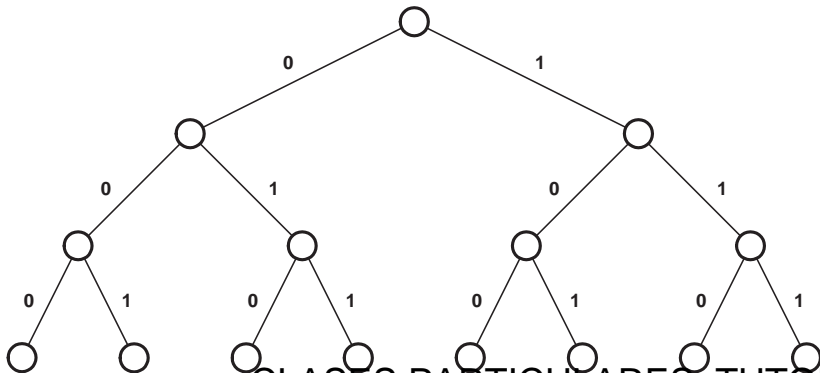
- n: número de elementos (profundidad del árbol)

Cartagena99

CLASAS PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Partes de $\{a, b, c\}$

Solución - árbol binario

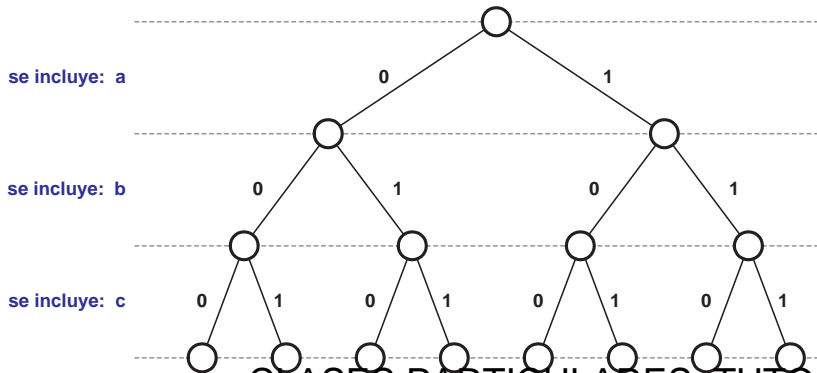


Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Partes de $\{a, b, c\}$

Solución - árbol binario



Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Partes de $\{a, b, c\}$

Solución - árbol binario

- Árbol binario
- En cada nodo decides si un elemento estará presente o no
- Simplemente crea un vector un vector de valores booleanos
 - $[1, 0, 1] = \{a, c\}$
- No es necesario deshacer los cambios

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - I

Solución - árbol binario

```
1 void partesConj1(int[] c){
2     boolean[] subc = new boolean[c.length];
3
4     partes1(c.length, 0, c, subc);
5 }
6
7 void imprimir(int[] c, boolean[] v){
8     for(int i=0; i<v.length; i++){
9         if(v[i])
10            System.out.print(c[i]+" ");
11
12     System.out.println();
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

Solución - árbol binario

```

1 void partes1(int n, int i, int[] c, boolean[] subc){
2   for(int k=0; k<=1; k++){
3     if(k==0)
4       subc[i] = false;
5     else
6       subc[i] = true;
7
8     if(i==n-1)
9       imprimir(c,subc);
10    else
11      partes1(n, i+1, c, subc);
12  }
13 }

```

- n: número de elementos (profundidad del árbol)

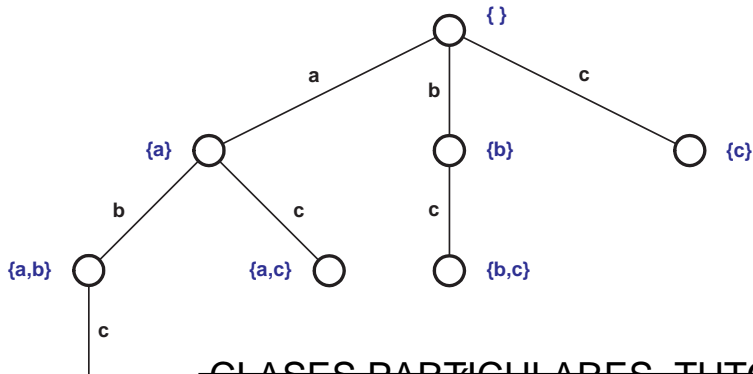
Cartagena99

CLASAS PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

- subc: valores booleanos que definen los subconjuntos

Partes de $\{a, b, c\}$

Solución - subconjunto en cada nodo

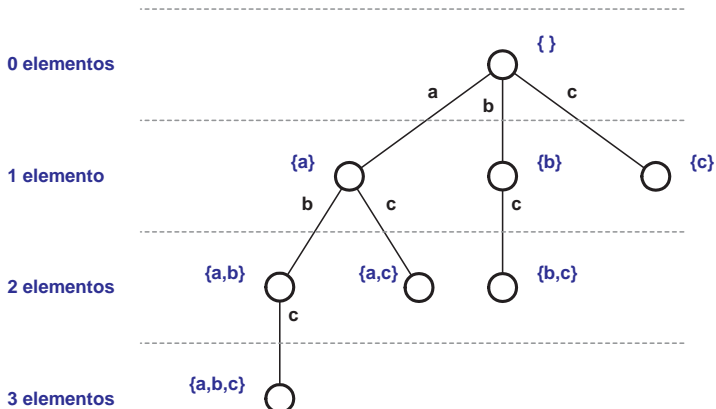


Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Partes de $\{a, b, c\}$

Solución - subconjunto en cada nodo



CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

La marca que acompaña a insertar

Partes de $\{a, b, c\}$

Solución - subconjunto en cada nodo

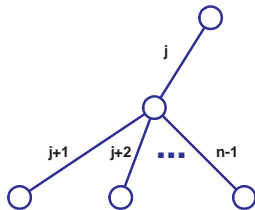
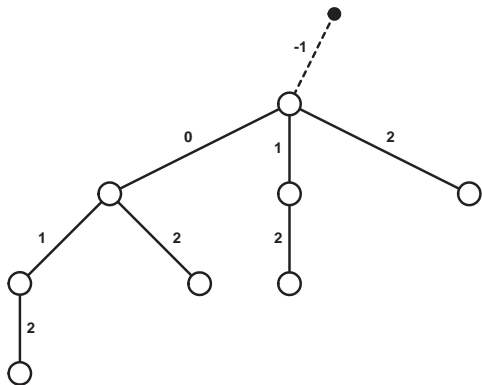
- El árbol de recursión no es binario
- En el nivel i las soluciones tienen i elementos
- Crea un vector con los índices de los elementos que se van incluyendo
 - $[0, 2] = \{a, c\}$
- Tiene la mitad de nodos (factor constante) que el árbol binario del primer algoritmo

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Partes de $\{a, b, c\}$

Solución - subconjunto en cada nodo



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

Solución - subconjunto en cada nodo

```
1 void partesConj2(int[] c){
2     int[] subc = new int[c.length];
3
4     partes2(c.length, 0, -1, c, subc);
5 }
6
7 void imprimir(int[] c, int[] subc, int fin){
8     for(int i=0; i<=fin; i++)
9         System.out.print (c[subc[i]]+" ");
10
11     System.out.println();
12 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

Solución - subconjunto en cada nodo

```
1 void partes2(int n, int i, int j, int[] c, int[] subc){
2     imprimir(c, subc, i-1);
3
4     for(int k=j+1; k<n; k++){
5         subc[i] = c[k];
6
7         partes2(n, i+1, k, c, subc);
8     }
9 }
```

- n: número de elementos (profundidad máxima del árbol)
- i: posición en la solución del elemento a incluir.

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

N reinas

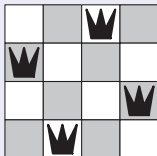
Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

N reinas

Problema de las N reinas

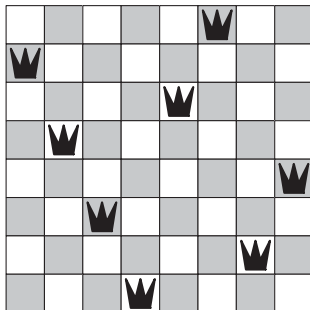
- Dado un “tablero ajedrez” de $n \times n$ celdas, se pide ubicar n reinas de modo que no se amenacen
 - No pueden estar en la misma fila
 - No pueden estar en la misma columna
 - No pueden estar en la misma diagonal (principal o secundaria)



Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

8 reinas

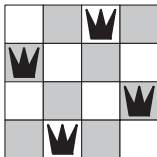


- Para $n = 8$ hay 92 soluciones posibles
- Aunque 12 únicas

Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

N reinas

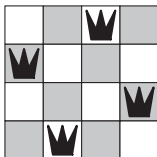


- Solución obvia pero absurda:
 - Probar las 2^{n^2} formas de colocar reinas en el tablero
 - $1,84 \cdot 10^{19}$ para $n = 8$
 - 65536 para $n = 4$
 - Pero los conjuntos solución solo deben contener n elementos

Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

N reinas

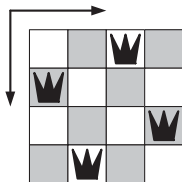


- Además solo puede haber una reina por cada columna:
 - Esto reduce las posibilidades a n^n (hay n formas de colocar una reina en una columna, y hay n columnas)
 - 16777216 para $n = 8$
 - 256 para $n = 4$
- Pero además, no puede haber dos reinas en la misma fila
 - Esto convierte nuestro problema en la búsqueda de una permutación

Cartagena99

CLASES EN VÍDEO PARA LOS TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

N reinas



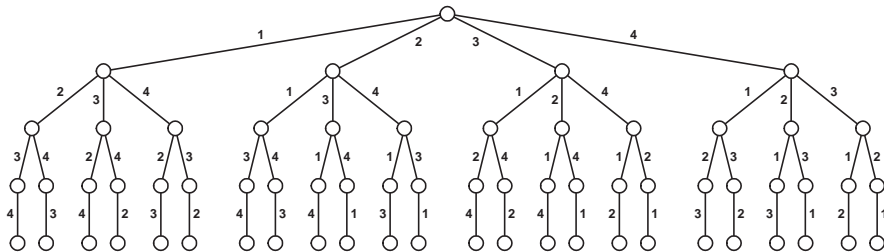
- Formato de la solución:

- (fila de la columna 1, fila de la columna 2, ..., fila de la columna n)
- (2, 4, 1, 3) en la figura
- Todas las filas son diferentes y ninguna está representada

Cartagena99

CLASOS PARTICULARES Y TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Árbol de búsqueda para $N = 4$

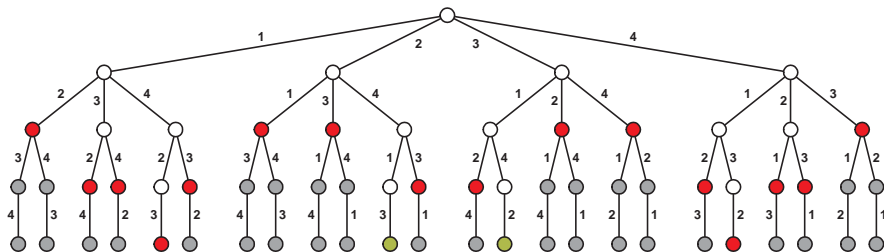


- Podemos emplear un algoritmo para buscar permutaciones
 - Al llegar a una hoja se “han colocado” las cuatro reinas y podemos probar si la solución es válida

Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SCIENCE
 CALL OR WHATSAPP. 689 45 44 70

Árbol de búsqueda podado para $N = 4$



○ solución parcial válida

● solución completa válida

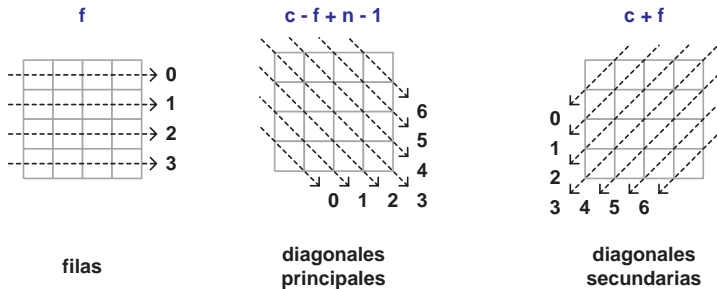
● solución no válida

● nodo no explorado

Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación



- Para verificar que una solución parcial es válida usamos:

- f : filas libres
- dn : diagonales

Cartagena99

CLASES PARTICULARES) TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 void n_reinas(int n){
2     int[] c = new int[n];
3
4     boolean[] f = new boolean[n];
5     for(int i=0; i<n; i++)
6         f[i] = true;
7
8     boolean[] dp = new boolean[2*n-1];
9     for(int i=0; i<2*n-1; i++)
10        dp[i] = true;
11
12    boolean[] ds = new boolean[2*n-1];
13    for(int i=0; i<2*n-1; i++)
14        ds[i] = true;
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```
1 void buscarReinas(int n, int i, int[] solucion,
2                 boolean[] f, boolean[] dp, boolean[] ds){
3     for(int j=0; j<n; j++){
4         if(f[j] && dp[i-j+n-1] && ds[i+j]){
5             solucion[i] = j;
6
7             f[j] = false;
8             dp[i-j+n-1] = false;
9             ds[i+j] = false;
10
11            if(i==n-1)
12                imprimir(solucion);
13            else
14                buscarReinas(n, i+1, solucion, f, dp, ds);
15
16            f[j] = true;
17        }
18    }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - III

- Línea 3: se generan los candidatos
- Línea 4: se comprueba la validez del candidato
- Líneas 5 – 9: se incluye el candidato en la solución, y se actualizan las estructuras de datos
- Línea 12: si se ha llegado a una solución válida se imprime
- Línea 14: en caso contrario se sigue buscando
- Líneas 16 – 18: se borra el candidato de la solución, y se actualizan las estructuras de datos

Cartagena99

CLASES PARTICULARES, TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

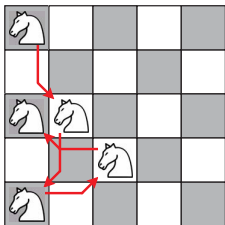
Otros problemas

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

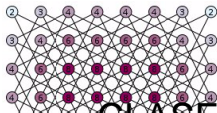
Salto de caballo

- No es requisito que pueda volver al punto de partida



1	14	9	20	23
10	19	22	15	8
5	2	13	24	21
18	11	4	7	16
3	6	17	12	25

- Variante: **ciclo Hamiltoniano** (camino cerrado en un grafo)

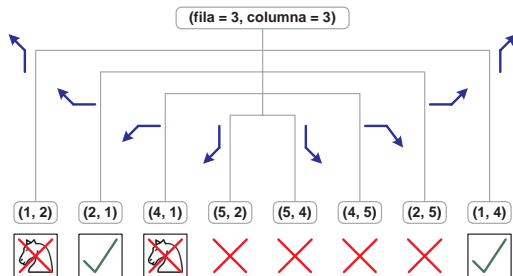
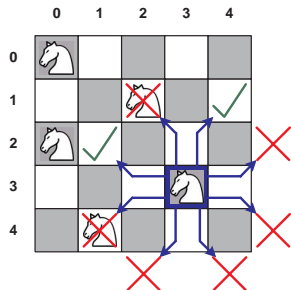


Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Salto de caballo

Árbol de búsqueda



casilla origen



casilla ocupada



casilla libre



fuera del tablero

Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 boolean saltoCaballo(int f, int c, int lado){
2     int[] [] tablero = new int[lado][lado];
3     int[] incrX = new int[] {-1, 1, 2, 2, 1, -1, -2, -2};
4     int[] incrY = new int[] {-2, -2, -1, 1, 2, 2, 1, -1};
5
6     tablero[f][c] = 1;
7     boolean hay = buscar(tablero.length*tablero.length,
8                           tablero.length, 2, f, c, tablero, incrX, incrY);
9     if(hay)
10        imprimir(tablero);
11    return hay;
12 }
13
14 void imprimir(int[] [] tablero){
15     for(int i=0; i<tablero.length; i++){
16         for(int j=0; j<tablero.length; j++){
17             System.out.print(((tablero[i][j]>0?"1":"") + tablero[i][j] + " "));
```

Cartagena99

CLASOS PARTICULARES Y TUTORÍAS
CLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

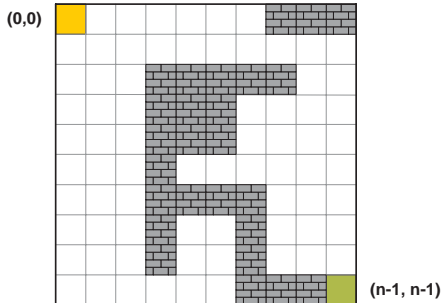
```
1 boolean buscar(int n2, int n, int i, int f, int c,  
2               int[] [] recorrido, int[] incrX, int[] incrY){  
3     boolean exito = false;  
4     for(int k=0; k<8 && !exito; k++){  
5         int nuevaF = f + incrY[k];  
6         int nuevaC = c + incrX[k];  
7         if(nuevaF>=0 && nuevaF<n && nuevaC>=0 && nuevaC<n)  
8             if(recorrido[nuevaC][nuevaF] == 0){  
9                 recorrido[nuevaC][nuevaF] = i;  
10                if(i==n2)  
11                    exito = true;  
12                else{  
13                    exito = buscar (n2, n, i+1,  
14                                   nuevaF, nuevaC, recorrido, incrX, incrY);  
15                }  
16                if(!exito)  
17                    recorrido[nuevaC][nuevaF] = 0;  
18            }  
19    }
```

Cartagena99

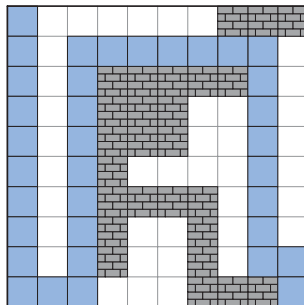
CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Laberinto

laberinto



solución

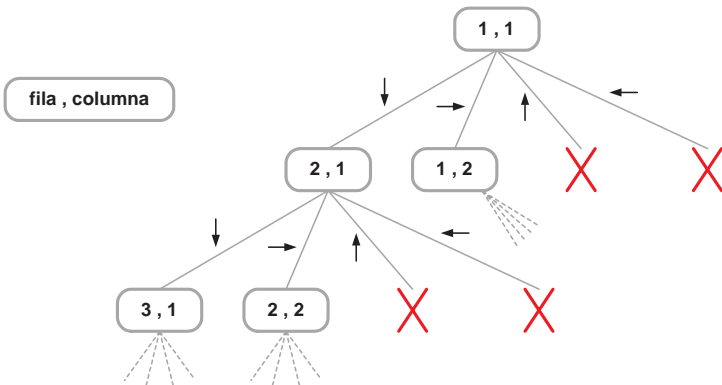


Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Laberinto

Árbol de búsqueda



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 boolean hayCamino(char[] [] laberinto){
2     int[] incrX = new int[] {1, 0, -1, 0};
3     int[] incrY = new int[] {0, 1, 0, -1};
4
5     laberinto[0][0] = 'C';
6     boolean exito =
7         buscar(laberinto.length, 0, 0, laberinto, incrX, incrY);
8
9     imprimir(laberinto);
10    return exito;
11 }
12
13 void imprimir(char[] [] laberinto){
14     for(int i=0; i<laberinto.length; i++){
15         for(int j=0; j<laberinto.length; j++)
16             System.out.print (laberinto[i][j]+" ");
17 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```
1 boolean buscar(int n, int x, int y, char[][] laberinto,
2               int[] incrX, int[] incrY){
3     boolean exito = false;
4     for(int k=0; k<4 && !exito; k++){
5         int coordX = x + incrX[k];
6         int coordY = y + incrY[k];
7         if(coordX>=0 && coordX<n && coordY>=0 && coordY<n)
8             if(laberinto[coordY][coordX] == ' '){
9                 laberinto[coordY][coordX] = 'C';
10                if (coordX==n-1 && coordY==n-1)
11                    exito = true;
12                else{
13                    exito = buscar(n, coordX, coordY, laberinto, incrX, incrY);
14                    if(!exito)
15                        laberinto[coordY][coordX] = ' ';
16                }
17            }
```

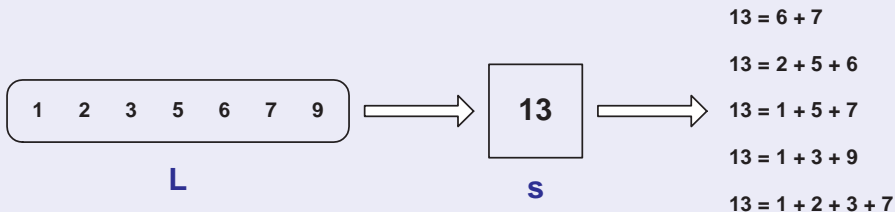
Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Suma de subconjuntos

Problema de la suma de subconjuntos

Dada una lista L de números no negativos y otro número s , determinar los subconjuntos de L que suman s

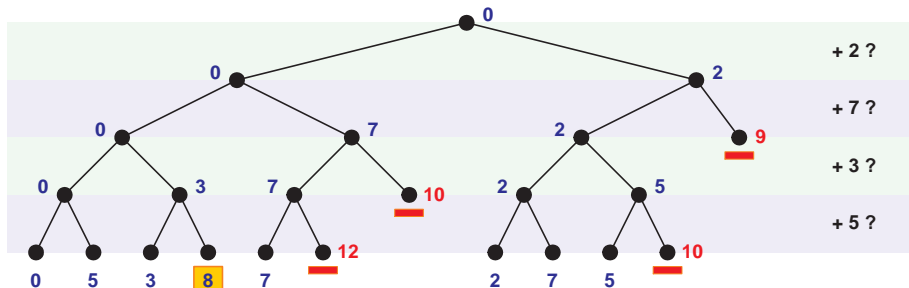


Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Suma de subconjuntos

- $L = \langle 2, 7, 3, 5 \rangle$, $s = 8$
- Se añade una condición para podar el árbol si la suma parcial es mayor que s



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 void main(String[] args){
2     int[] L = new int[] {1, 2, 3, 5, 6, 7, 9};
3
4     sumaSubconjuntos(L, 13);
5 }
6
7 void sumaSubconjuntos (int[] L, int s){
8     buscarSubconjs(L.length, 0, 0, L, s, new boolean[L.length]);
9 }
10
11 void imprimir(int[] L, boolean[] v){
12     for(int i=0; i<v.length; i++)
13         if(v[i])
14             System.out.print(L[i]+" ");
15
16     System.out.println();
17 }
```

Cartagena99

CLASES PARTICULARES, TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```
1 void buscarSubconjs (int n, int i, int p,  
2                       int[] L, int s, boolean[] subconj){  
3     for (int k=0; k<=1; k++){  
4       int nuevoP = p + k*L[i];  
5       if (nuevoP<=s){  
6         if(k==0)  
7           subconj[i]=false;  
8         else  
9           subconj[i]=true;  
10  
11        if(i==n-1){  
12          if(nuevoP==s)  
13            imprimir (L, subconj);  
14        }  
15        else  
16          buscarSubconjs(n, i+1, nuevoP, L, s, subconj);  
17      }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Problema de la mochila 0-1

Problema de la mochila 0-1

Dado un conjunto de n objetos, cada uno con un peso p_i y un valor v_i , $i = 1, \dots, n$, y una mochila con capacidad C . Maximizar la suma de los valores asociados a los objetos que se introducen en la mochila, sin sobrepasar la capacidad C , sabiendo que los objetos **NO** pueden partirse en fracciones más pequeñas:

$$\underset{x}{\text{maximizar}} \quad \sum_{i=1}^n x_i v_i \quad \text{sujeto a} \quad x_i \in \{0, 1\} \quad i = 1, \dots, n$$

$$\sum_{i=1}^n x_i p_i \leq C$$



p_1



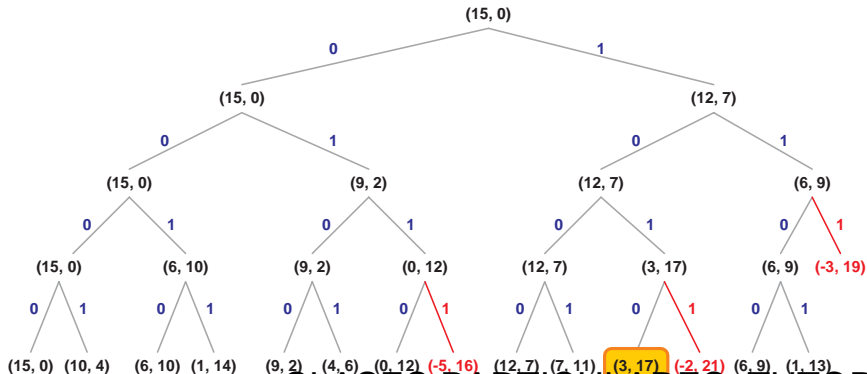
Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVÍA WHATSAPP. 689 45 44 70
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Mochila 0-1

Árbol de búsqueda

- pesos = (3, 6, 9, 5), beneficios = (7, 2, 10, 4), capacidad = 15



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 void main(String[] args){
2     int[] ps= new int[] {3, 6, 9, 5};
3     int[] bs= new int[] {7, 2, 10, 4};
4     mochila_0_1(ps,bs, 15);
5 }
6
7 int mochila_0_1(int[] ps, int[] bs, int c){
8     int[] solParcial = new int[ps.length];
9     int[] solOptima = new int[ps.length];
10    int bOpt = buscar01(ps.length, 0, c, 0, solParcial,
11                        solOptima, -1, ps, bs, c);
12    imprimir(solOptima, bOpt);
13    return bOpt;
14 }
15
16 void imprimir(int[] v, int bOpt){
17    for (int i=0; i<v.length; i++)
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```
1 int buscar01(int n, int i, int p, int b, int[] solParc,  
2             int[] sol0pt, int b0pt, int[] ps, int[] bs, int c){  
3     for(int k=0; k<=1; k++){  
4         if(k*ps[i]<=p){  
5             solParc[i] = k;  
6             int np = p - k*ps[i];    int nb = b + k*bs[i];  
7             if(i==n-1){  
8                 if(nb>b0pt){  
9                     b0pt = nb;  
10                    for(int j=0; j<ps.length; j++){  
11                        sol0pt[j] = solParc[j];  
12                    }  
13                }  
14                else  
15                    b0pt = buscar01(n, i+1, np, nb, solParc, sol0pt, b0pt, ps, bs, c);  
16                //int np = p + k*ps[i]; innecesaria  
17                //int nb = b - k*bs[i]; innecesaria
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Ramificación y poda

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

Ramificación y poda

- Hasta ahora podábamos el árbol de recursión si una solución parcial no cumplía los requisitos del problema para llegar a ser una solución completa
- En **problemas de optimización** se pueden realizar podas adicionales:

Habiendo conseguido hallar una solución válida con valor óptimo v , podemos descartar soluciones que nunca puedan alcanzar dicho valor óptimo

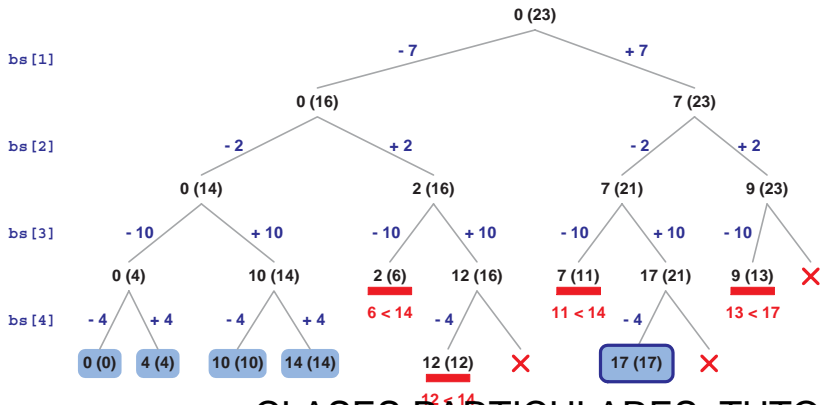
- Es un método exacto de resolución de problemas de optimización

Cartagena99

CLASES PARTICULARES, TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Ramificación y poda: Mochila 0-1

- pesos = (3, 6, 9, 5), beneficios = (7, 2, 10, 4), capacidad = 15



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Implementación - I

```
1 void main(String[] args){
2     int[] ps= new int[] {3, 6, 9, 5};
3     int[] bs= new int[] {7, 2, 10, 4};
4
5     mochila_0_1(ps,bs,15);
6 }
7
8 void mochila_0_1(int[] ps, int[] bs, int c){
9     int[] solParcial = new int[ps.length];
10    int[] solOptima = new int[ps.length];
11
12    int suma=0;
13    for(int i=0; i<bs.length; i++)
14        suma += bs[i];
15
16    int bOpt = buscar01poda(ps.length, 0, c, 0, suma, solParcial, solOptima, -1, ps, bs, c);
17    imprimir(solOptima, bOpt);
18 }
19
20 void imprimir(int[] v, int bOpt){
21     for(int i=0; i<v.length; i++)
22         System.out.print(v[i] + " ");
23 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Implementación - II

```
1 int buscar01poda(int n, int i, int p, int b, int bmax, int[] solParc,  
2     int[] solOpt, int bOpt, int[] ps, int[] bs, int c){  
3  
4     for(int k=0; k<=1; k++){  
5         if(k*ps[i]<=p){  
6             solParc[i] = k;  
7             int np = p - k*ps[i];  
8             int nb = b + k*bs[i];  
9             int nbmax = bmax - (1-k)*bs[i];  
10  
11             if(i==n-1){  
12                 if(nb>bOpt){  
13                     bOpt = nb;  
14                     for(int j=0; j<ps.length; j++){  
15                         solOpt[j] = solParc[j];  
16                     }  
17                 }  
18                 else  
19                     if(nbmax>bOpt)  
20                     bOpt = buscar01poda(n, i+1, np, nb, nbmax, solParc, solOpt, bOpt, ps, bs, c);  
21             }  
22         }  
23     }  
24 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Poda alfa-beta

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Poda alfa-beta

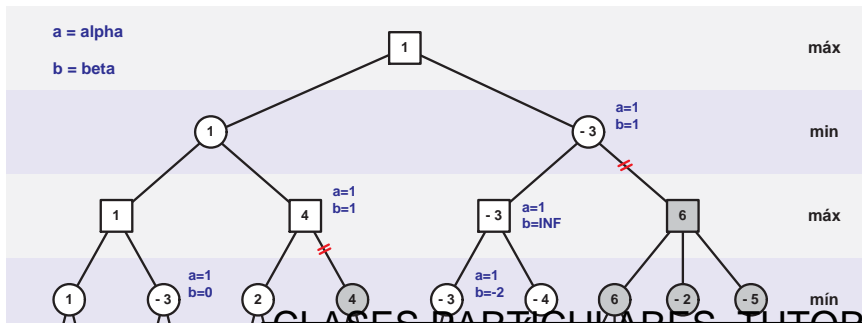
- Algoritmo minimax
 - Se utiliza en juegos de estrategia entre adversarios (p.e., ajedrez)
 - Cada jugador toma turnos
 - Se evalúa una función numérica acerca de la situación
 - Jaque-mate a favor (∞), en contra ($-\infty$), comer reina contraria (valor muy alto), etc.
 - Un jugador trata de maximizar la función, el otro de minimizarla
- Poda alfa-beta
 - Estrategia de búsqueda que reduce el número de nodos evaluados por

Cartagena99

CLASAS O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Poda alfa-beta

- α = cota inferior de los máximos
- β = cota superior de los mínimos
- Se poda si $\beta \leq \alpha$
- Se conocen valores heurísticos de los nodos hoja



Cartagena99

CLASES PARTICULARES TUTORÍAS
 LLAMA O ENVIA WHATSAPP. 689 45
 ONLINE PRIVATE LESSONS FOR SC
 CALL OR WHATSAPP. 689 45 44 70

Pseudocódigo de la poda alfa-beta

```
1 alfabeta(nodo, n, a, b, jugador)
2   si (n=0) o si el nodo es una hoja
3     devuelve el valor heurístico del nodo
4   si jugador = jugadorMax
5     para cada hijo del nodo
6       a := max(a, alfabeta(hijo, n-1, a, b, not(jugador) ))
7       si b>a
8         break                                     (* Poda Beta *)
9     devuelve a
10  en caso contrario
11  para cada hijo del nodo
12    b := min(b, alfabeta(hijo, n-1, a, b, not(jugador) ))
13    si b<a
14      break                                     (* Poda Alfa *)
15  devuelve b
16
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVIA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Esquemas generales

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SCIENCE
CALL OR WHATSAPP. 689 45 44 70

Pasos a seguir para diseñar algoritmos

- Diseño de la representación de la solución y del árbol de búsqueda
- Deducción de las comprobaciones de validez a realizar en cada nodo a partir de las restricciones globales de la solución
- Diseño de estructuras de datos auxiliares para la generación de candidatos o las comprobaciones de validez
- Codificación con ayuda de los **esquemas generales** de código

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Esquema para todas las soluciones - versión 1

```
1 void buscarTodas (int n, int i, Valor[ ] solucion){
2   for(int k=0; k<n; k++){
3     <<generar candidato k-ésimo>>;
4
5     if(<<candidato válido>>){
6       <<incluirlo en solucion >>;
7
8       if(i==n-1)
9         imprimirSolucion(solucion);
10      else
11        buscarTodas(n, i+1, solucion);
12
13      <<borrarlo de solucion >>;
14    }
15  }
16 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Esquema para todas las soluciones - versión 2

```
1 void buscarTodas(int n, int i, Valor[ ] solucion){
2   if(i==n)
3     imprimirSolucion(solucion);
4   else
5     for(int k=0; k<n; k++){
6       <<generar candidato k-ésimo>>;
7
8       if(<<candidato válido>>){
9         <<incluirlo en solucion>>;
10
11         buscarTodas(n, i+1, solucion);
12
13         <<borrarlo de solucion>>;
14       }
15     }
16 }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Esquema para una solución

```
1 boolean buscarUna(int n, int i, Valor[ ] solucion){
2     boolean exito=false;
3     for(int k=0; k<n && !exito; k++){
4         <<generar candidato k-ésimo>>;
5         if (<<candidato válido>>){
6             <<incluirlo en solucion>>;
7
8             if(i==n-1)
9                 exito = true;
10            else{
11                exito = buscarUna(n, i+1, solucion);
12                if (!exito)
13                    <<borrarlo de solucion >>;
14            }
15        }
```

Cartagena99

CLASES PARTICULARES TUTORÍAS
LLAMA O ENVÍA WHATSAPP. 689 45 44 70
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70

Esquema para la solución óptima

```
1 int buscarOptima(int n, int i,  
2                 Valor[ ] solActual, int valorActual,  
3                 Valor[ ] solOptima, int valorOptimo){  
4     int valor;  
5     for(int k=0; k<n; k++){  
6         <<generar candidato k-ésimo>>;  
7         if(<<candidato válido>>){  
8             <<incluirlo en solActual y actualizar valorActual>>;  
9             if(i==n-1)  
10                if(valorActual>valorOptimo){  
11                    <<solOptima = solActual>>  
12                    valorOptimo = valorActual;  
13                }  
14                else  
15                    valorOptimo = buscarOptima(n, i+1, solActual,  
16                                                valorActual, solOptima, valorOptimo);  
17                <<borrarlo de solActual y restaurar valorActual>>;
```

Cartagena99

CLASES PARTICULARES: TUTORÍAS
ONLINE PRIVATE LESSONS FOR SC
CALL OR WHATSAPP. 689 45 44 70