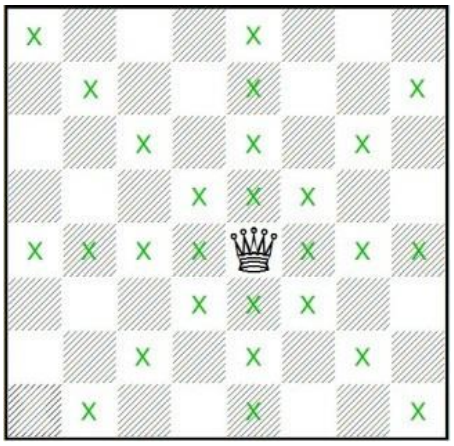
Backtracking
El problema de las 8 reinas

Problema combinatorio clásico:

Colocar ocho reinas en un tablero de ajedrez de modo que no haya dos que se ataquen; (que estén en la misma fila, columna o diagonal).



- Como cada reina debe estar en una fila diferente, sin pérdida de generalidad podemos suponer que la reina i se coloca en la fila i.
- Todas las soluciones para este problema pueden representarse como 8-tuplas $(x_1,...,x_8)$ en las que x_i indica la columna en la que se coloca la reina i.

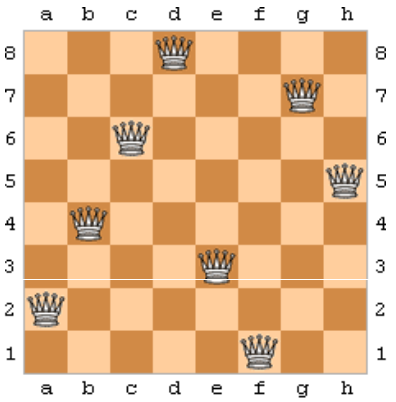
Backtracking
El problema de las 8 reinas

Restricciones explícitas:

$S_i = \{1,2,3,4,5,6,7,8\}$, $1 \leq i \leq 8$

Espacio de soluciones:

Tamaño $|S_i|^8 = 8^8 = 2^{24} = 16M$



Restricciones implícitas:

- Ningún par (x_i, x_j) con $x_i=x_j$ (todas las reinas deben estar en columnas diferentes).
- Ningún par (x_i, x_j) con $|j-i|=|x_j-x_i|$ (todas las reinas deben estar en diagonales diferentes).

NOTA:
La primera de las restricciones implícitas implica que todas las soluciones son permutaciones de {1,2,3,4,5,6,7,8}, lo que reduce el espacio de soluciones de 8^8 tuplas a 8! = 40320.

Backtracking
El problema de las N reinas

Espacio de soluciones

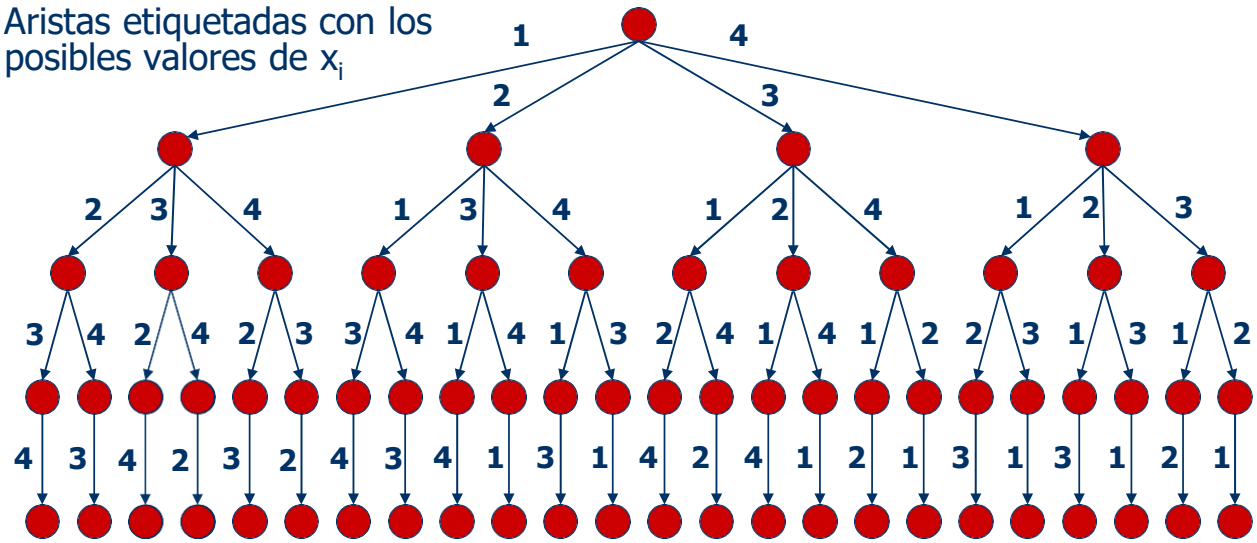
Generalización del problema de las 8 reinas:

Colocar N reinas en un tablero NxN
de modo se ataquen entre ellas.

El espacio de soluciones consiste en las N!
permutaciones de la N-tupla (1,2,...,N)

La generalización nos sirve, a efectos didácticos, para poder hablar del problema de las 4 reinas, que se puede representar en un árbol de permutaciones...

Backtracking
El problema de las 4 reinas

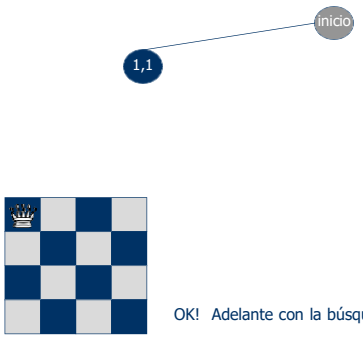


Las aristas desde los nodos del nivel i al i+1 están etiquetadas con los valores de x_i
p.ej. La rama más a la izquierda representa la solución $x_1=1$, $x_2=2$, $x_3=3$ y $x_4=4$.

El espacio de soluciones viene definido por todos los caminos desde el nodo raíz a un nodo hoja (4! permutaciones ⇒ 4! nodos hoja).

Backtracking
El problema de las 4 reinas

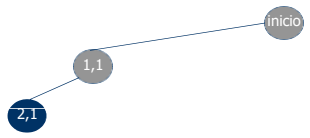
Generación de estados usando backtracking



OK! Adelante con la búsqueda…

Backtracking
El problema de las 4 reinas

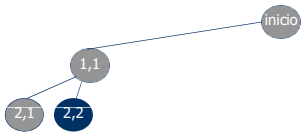
Generación de estados usando backtracking



No se cumplen las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

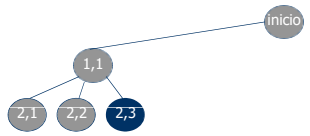
Generación de estados usando backtracking

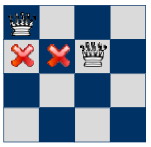


No se cumplen las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

Generación de estados usando backtracking

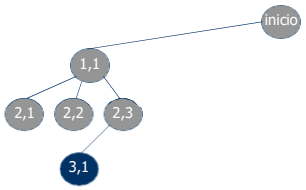




OK! Adelante con la búsqueda…

Backtracking
El problema de las 4 reinas

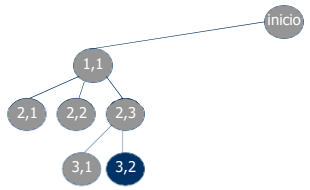
Generación de estados usando backtracking



No se cumplen las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

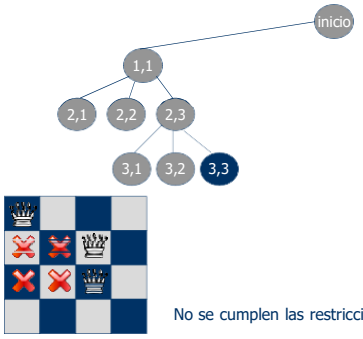
Generación de estados usando backtracking



No se cumplen las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

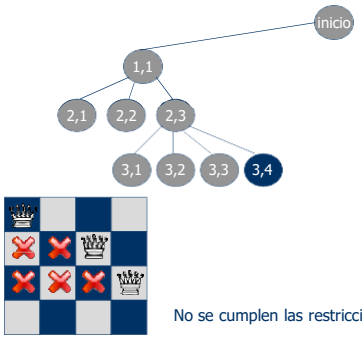
Generación de estados usando backtracking



No se cumplen las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

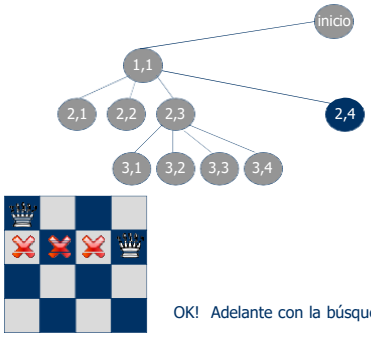
Generación de estados usando backtracking



No se cumplen las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

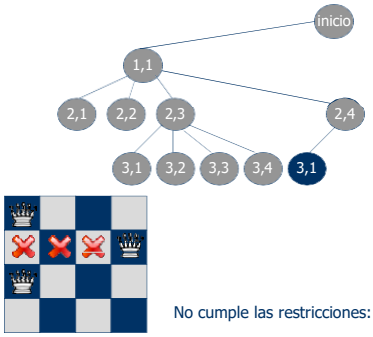
Generación de estados usando backtracking



OK! Adelante con la búsqueda…

Backtracking
El problema de las 4 reinas

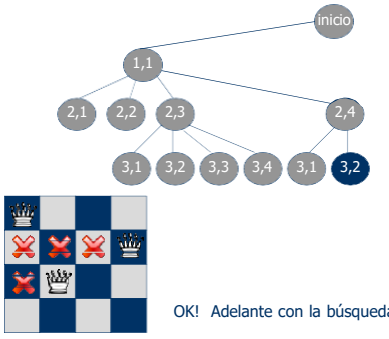
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

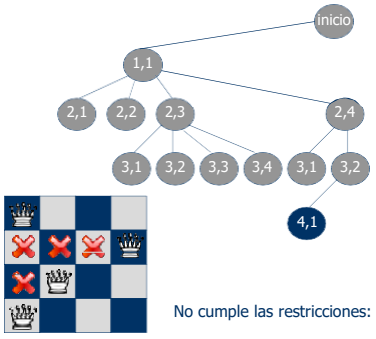
Generación de estados usando backtracking



OK! Adelante con la búsqueda…

Backtracking
El problema de las 4 reinas

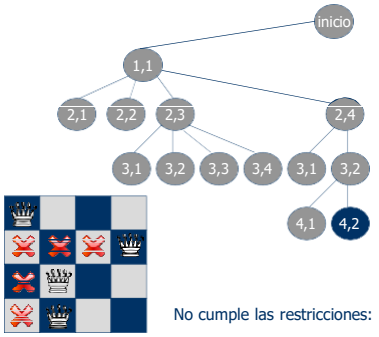
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

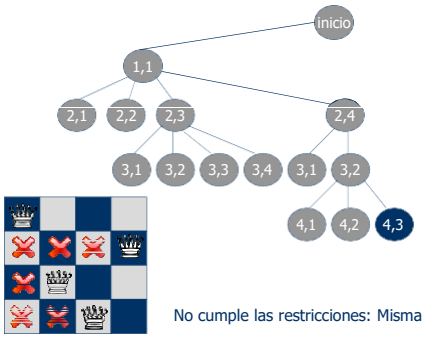
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

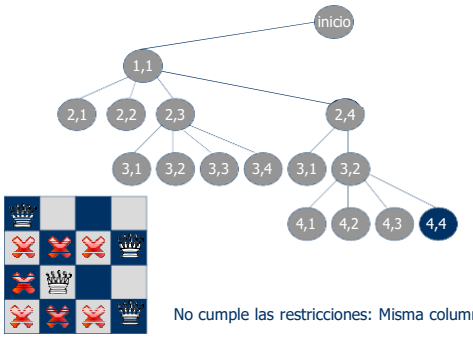
Generación de estados usando backtracking



No cumple las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

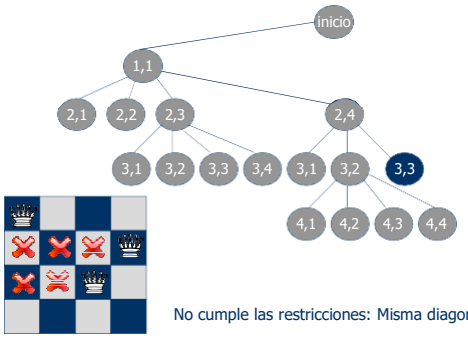
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

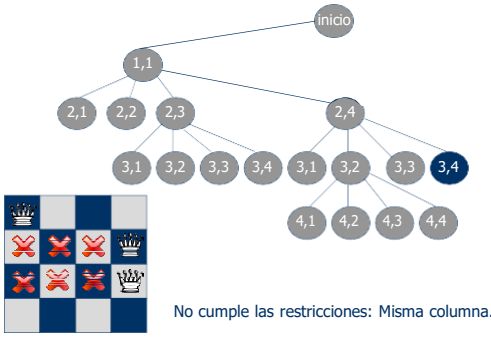
Generación de estados usando backtracking



No cumple las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

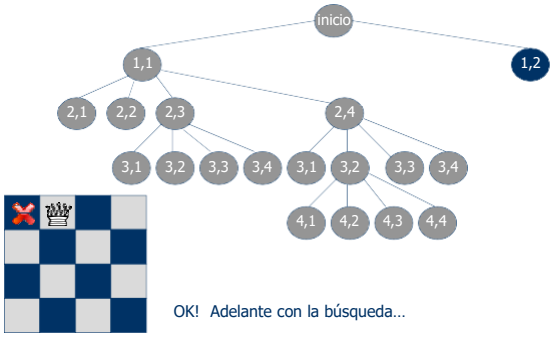
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

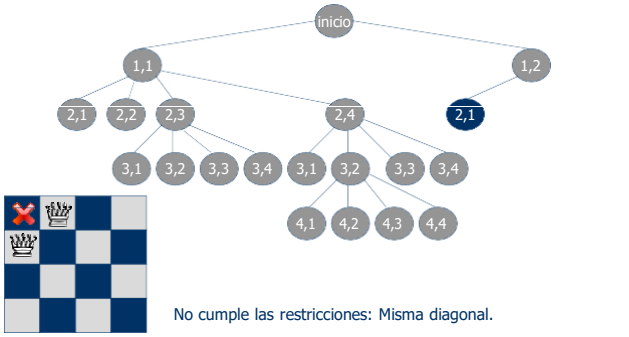
Generación de estados usando backtracking



OK! Adelante con la búsqueda…

Backtracking
El problema de las 4 reinas

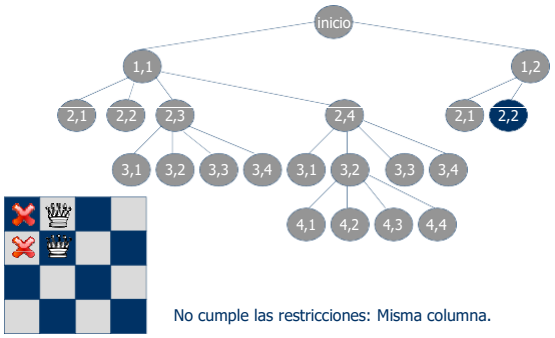
Generación de estados usando backtracking



No cumple las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

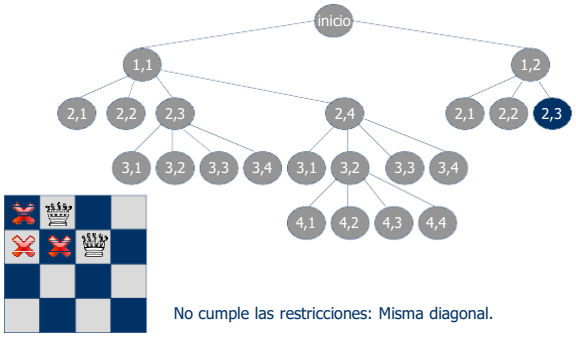
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

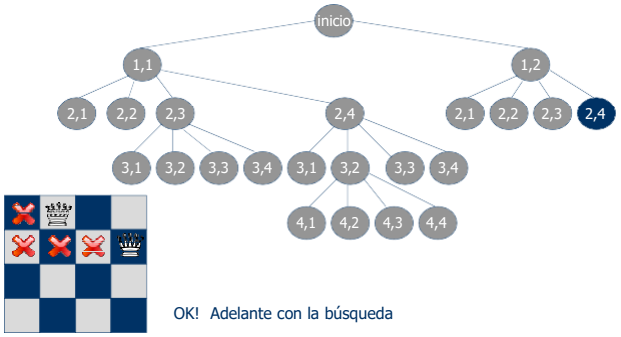
Generación de estados usando backtracking



No cumple las restricciones: Misma diagonal.

Backtracking
El problema de las 4 reinas

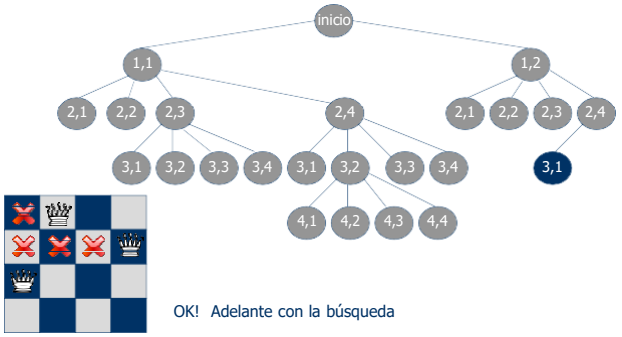
Generación de estados usando backtracking



OK! Adelante con la búsqueda

Backtracking
El problema de las 4 reinas

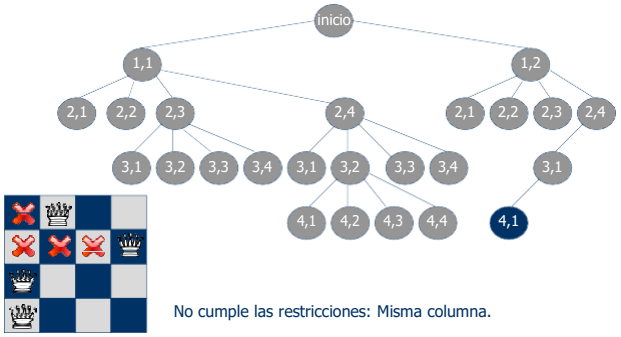
Generación de estados usando backtracking



OK! Adelante con la búsqueda

Backtracking
El problema de las 4 reinas

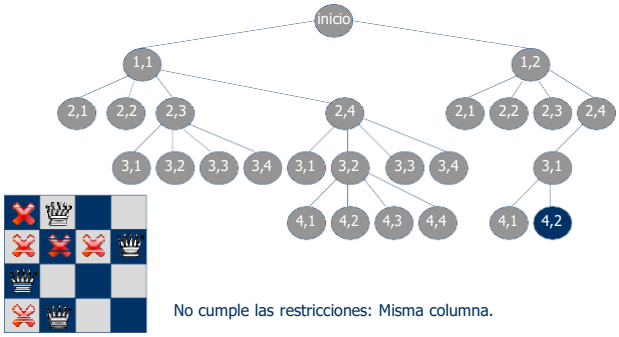
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

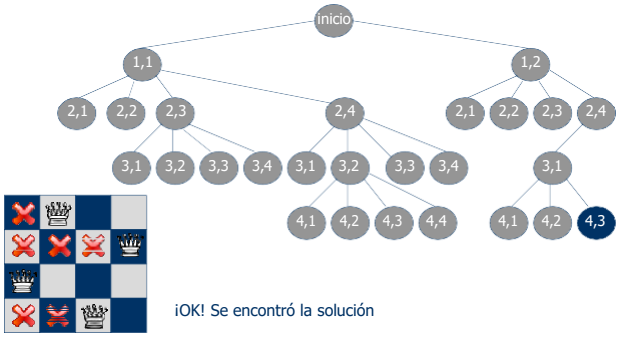
Generación de estados usando backtracking



No cumple las restricciones: Misma columna.

Backtracking
El problema de las 4 reinas

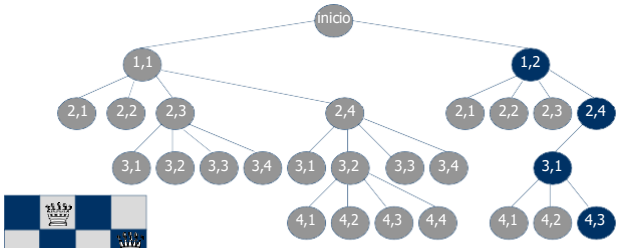
Generación de estados usando backtracking



¡OK! Se encontró la solución

Backtracking
El problema de las 4 reinas

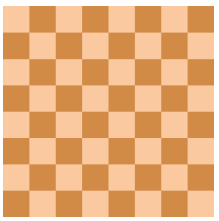
Generación de estados usando backtracking



Se comprobaron 27 nodos en total…
Sin backtracking se habrían comprobado 155 nodos.

Backtracking
El problema de las 8 reinas

Generación de estados usando backtracking



Backtracking
El problema de las N reinas

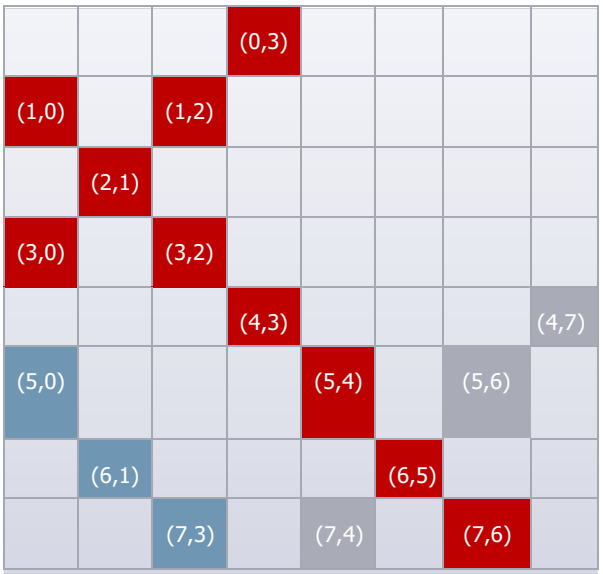
Implementación

Tablero NxN en el que colocar N reinas que no se ataquen.

- **Solución**: $(x_0,x_2,x_3,\ldots,x_{n-1})$, donde x_i es la columna de la i-ésima fila en la que se coloca la reina i.
- **Restricciones implícitas**: $x_i \in \{0..n-1\}$.
- **Restricciones explícitas**: No puede haber dos reinas en la misma columna ni en la misma diagonal.

Backtracking
El problema de las N reinas

Implementación



- Distinta columna:
  Todos los x_i diferentes.
- Distinta diagonal:
  Las reinas (i,j) y (k,l) están en la misma diagonal si **i-j=k-l** o bien **i+j=k+l**, lo que se puede resumir en |j-l| = |k-i|.

  En términos de los x_i, tendremos $|x_k-x_i|=|k-i|$.

Backtracking
El problema de las N reinas

Implementación

```
// Comprobar si la reina de la fila k está bien colocada
// (si no está en la misma columna ni en la misma diagonal
// que cualquiera de las reinas de las filas anteriores)
// Eficiencia: O(k-1).

bool comprobar (int reinas[], int n, int k)
{
  int i;

  for (i=0; i<k; i++)
      if (  ( reinas[i]==reinas[k] )
         || ( abs(k-i) == abs(reinas[k]-reinas[i]) ) )
         return false;

  return true;
}
```

Backtracking
El problema de las N reinas

Implementación recursiva mostrando todas las soluciones

```
// Inicialmente, k=0 y reinas[i]=-1.

void NReinas (int reinas[], int n, int k)
{
  if (k==n) {            // Solución (no quedan reinas por colocar)
     print(reinas,n);
  } else {                  // Aún quedan reinas por colocar (k<n)
     for (reinas[k]=0; reinas[k]<n; reinas[k]++)
         if (comprobar(reinas,n,k))
            NReinas (reinas, n, k+1);
  }
}
```

Backtracking
El problema de las N reinas

Implementación recursiva mostrando sólo la primera solución

```
bool NReinas (int reinas[], int n, int k)
{
  bool ok = false;
  if (k==n) {            // Caso base: No quedan reinas por colocar
     ok = true;
  } else {                  // Aún quedan reinas por colocar (k<n)
     while ((reinas[k]<n-1) && !ok) {
           reinas[k]++;
           if (comprobar(reinas,n,k))
              ok = NReinas (reinas, n, k+1);
     }
  }
  return ok;  // La solución está en reinas[] cuando ok==true
}
```

Backtracking
El problema de las N reinas

Implementación iterativa mostrando todas las soluciones

```
void NReinas (int reinas[], int n)
{
  int k=0;                                  // Fila actual = k
  for (int i=0; i<n; i++) reinas[i]=-1;  // Configuración inicial

  while (k>=0) {
        reinas[k]++; // Colocar reina k en la siguiente columna…
        while ((reinas[k]<n) && !comprobar(reinas,n,k))
              reinas[k]++;
        if (k<n) {                              // Reina colocada
           if (k==n-1) { print(reinas,n);    // Solución
           } else { k++; reinas[k] = -1; }  // Siguiente reina
        } else { k--; }
  }
}
```