Algoritmos y Estructuras de Datos II
Clase 16

Carreras:

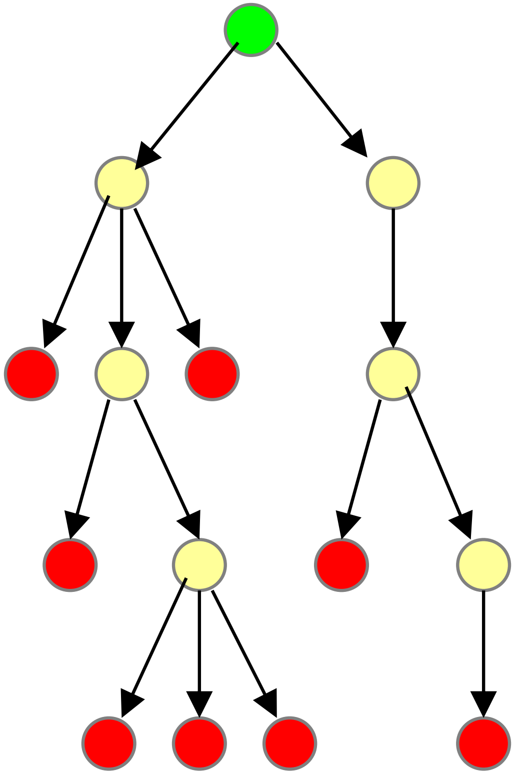
Licenciatura en Informática

Ingeniería en Informática

2024

Unidad III

Técnicas de diseño de algoritmos



Vuelta Atrás (2) (en inglés: ***Backtracking***)

Problemas de las n reinas

Datos:

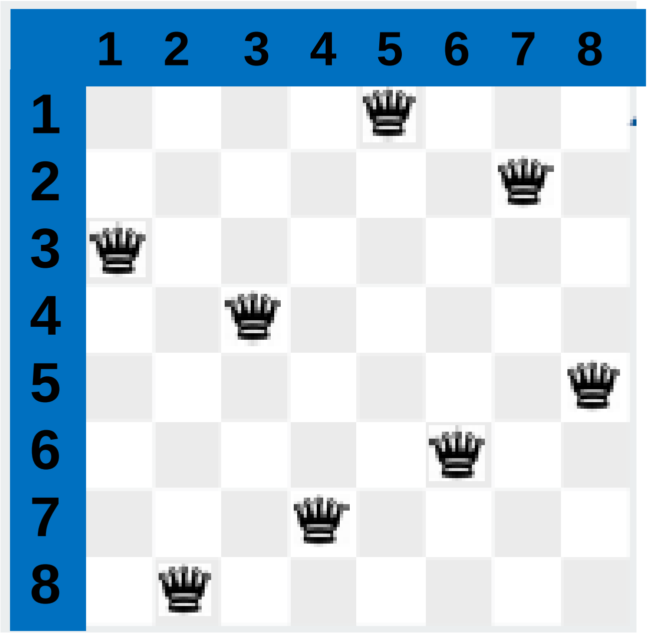
- Se tiene un tablero de ajedrez de *n*x*n* posiciones donde se van a ubicar *n* reinas.

Objetivo:

- Se quiere ubicar las ***n*** reinas en el tablero de ajedrez de tal modo que ninguna de ellas ***amenace*** a ninguna de las demás.

- Una reina ***amenaza*** a las que se encuentran situadas en los cuadrados de la misma fila, en la misma columna o en las mismas diagonales.

Ej. clásico: 8 reinas



Problemas de las n reinas

Solución por la fuerza bruta:

- Este problema tiene una solucion para n=1
- No tiene solucion para n=2 ni para n=3.
- Tiene 2 soluciones para n=4
- Para otros n se muestra la cantidad de soluciones Q(n) en la tabla.

n	$Q(n)$
4	2
5	10
6	4
7	40
8	92
9	352
10	724
11	2.680
12	14.200
13	73.712
14	365.596
15	2.279.184
16	14.772.512
17	95.815.104
18	666.090.624
19	4.968.057.848
20	39.029.188.884

Problemas de las n reinas

Para encontrar la ***solución con algoritmo vuelta atrás*** :

- Se numeran las reinas de 1 a n.
- La solución se almacena en un vector X (1..n) de n componentes.
- El algoritmo trabaja por pasos y se realizan n pasos.
- En el paso k se ubica la reina en el la fila k. Ya quedan así asignadas las componentes del vector solución X (1..k).
- A cada componente del vector X: X(k), k=1..n se le asigna un numero de 1 a n que indica el numero de la columna en que se almacena la reina k .

Problemas de las n reinas

Restricciones de la ***solución con algoritmo <u>vuelta atrás</u>*** :

- Cada X(k), k=1..n, almacena el valor de la columna en que se ubica la reina de la fila k, de modo que se garantiza una sola reina por fila.

- Dos reinas no pueden estar en la misma columna de modo que 2 componentes del vector solución X no pueden tener el mismo valor.

- Dos reinas no pueden estar en la misma diagonal.

Problemas de las n reinas

De esta manera, y aplicando las restricciones, en cada etapa *k* se va generando sólo las *k*-tuplas con posibilidad de solución.

A los prefijos de longitud *k* de la *n*tupla solución que se va construyendo y que verifiquen las restricciones expuestas se denominan *k-prometedores*, porque en principio pueden llevar a la solución buscada.

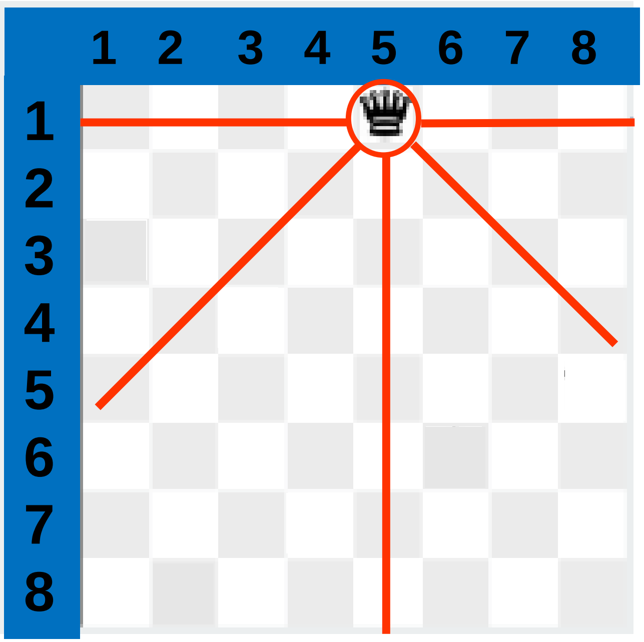
Cada nodo generado es de algunos de estos 2 tipos:

- fracaso
- *k*-prometedor.

Con estas condiciones queda definida la estructura del árbol de expansión.

Ejemplo:
Problemas de las 8 reinas

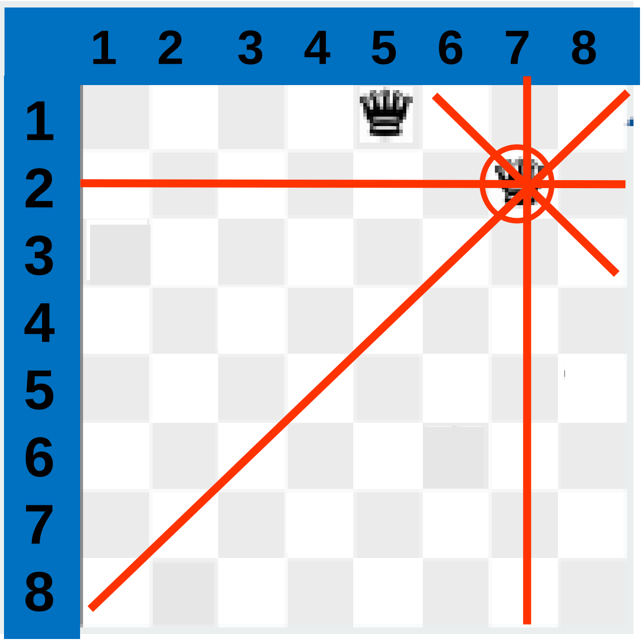
- **Paso 1:** ubica la reina de la primera fila



X=(5,-,-,-,-,-,-,-)

Ejemplo:
Problemas de las 8 reinas

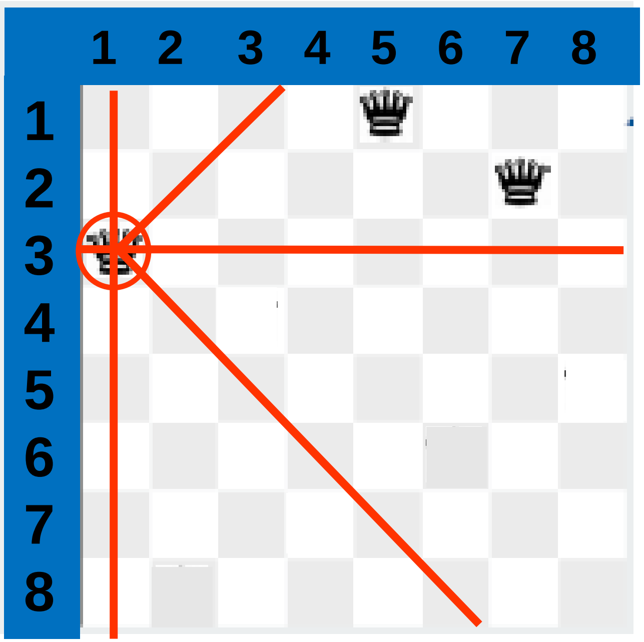
- **Paso 2:** ubica la reina de la segunda fila



X=(5,7,-,-,-,-,-,-)

Ejemplo:
Problemas de las 8 reinas

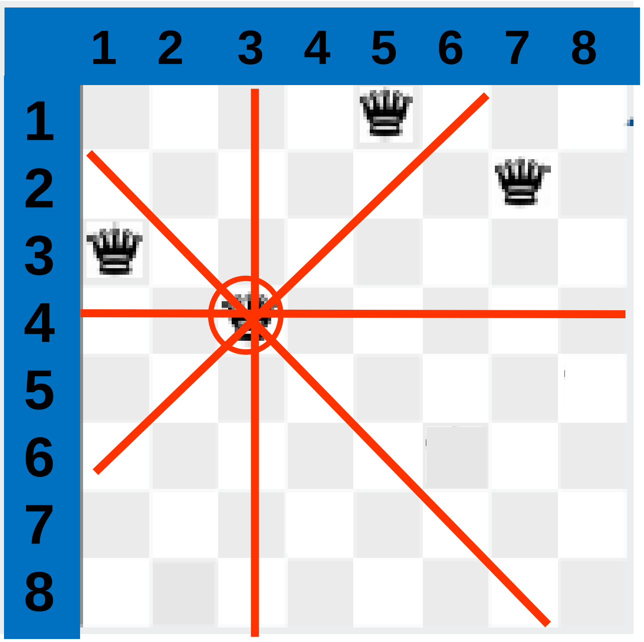
- **Paso 3:** ubica la reina de la tercera fila



X=(5,7,1,-,-,-,-,-)

Ejemplo:
Problemas de las 8 reinas

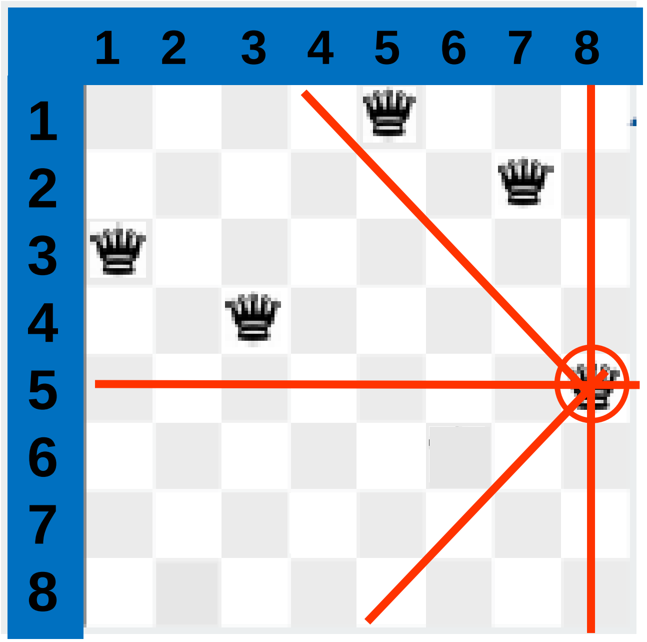
- **Paso 4:** ubica la reina de la cuarta fila



X=(5,7,1,3,-,-,-,-)

Ejemplo:
Problemas de las 8 reinas

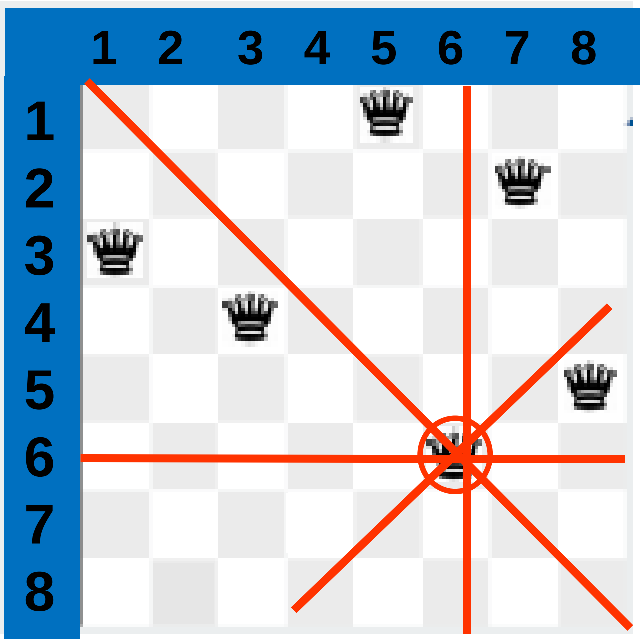
- **Paso 5:** ubica la reina de la quinta fila



X=(5,7,1,3,8,-,-,-)

Ejemplo:
Problemas de las 8 reinas

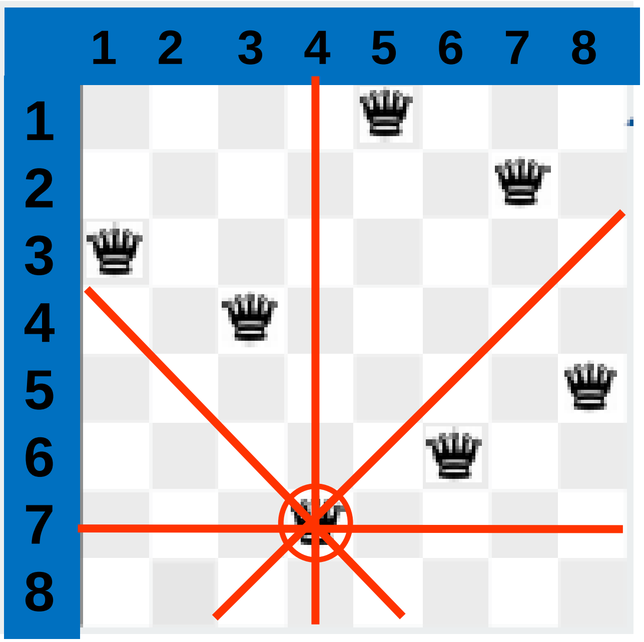
- **Paso 6:** ubica la reina de la sexta fila



X=(5,7,1,3,8,6,-,-)

Ejemplo:
Problemas de las 8 reinas

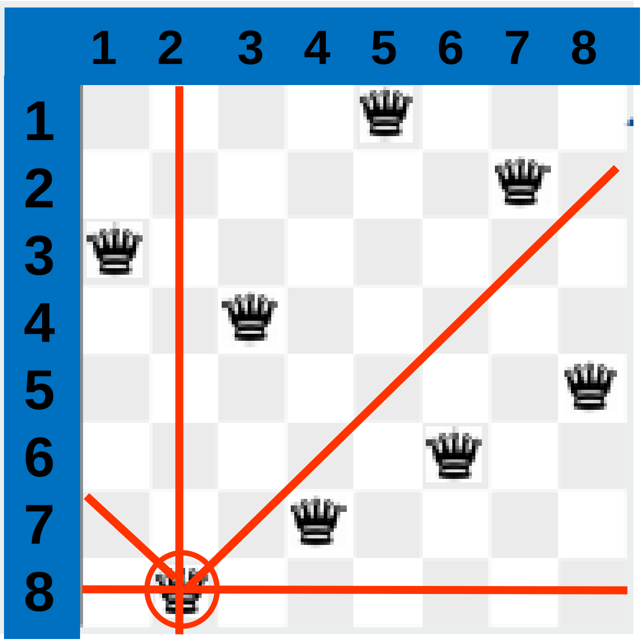
- **Paso 7:** ubica la reina de la séptima fila



X=(5,7,1,3,8,6,4,-)

Ejemplo:
Problemas de las 8 reinas

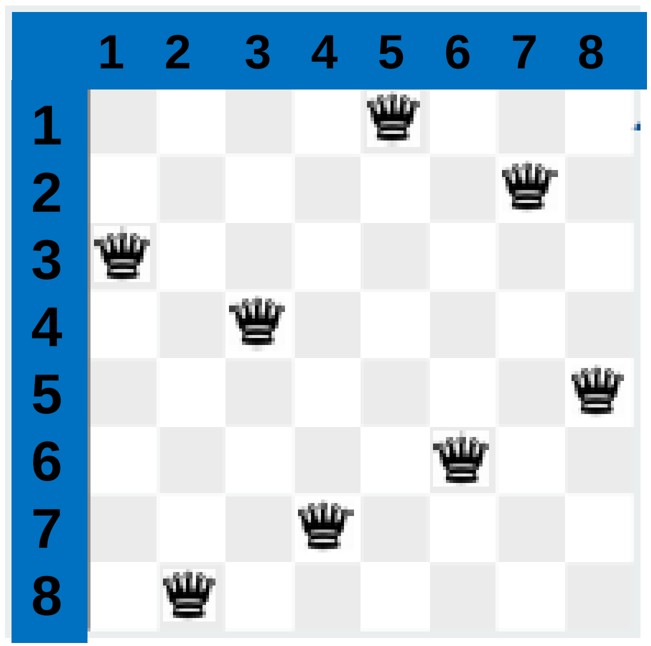
- **Paso 8:** ubica la reina de la octava fila



X=(5,7,1,3,8,6,4,2)

Ejemplo:
Problemas de las 8 reinas

- **Se ha conseguido una solución en 8 pasos**



Una solución: X=(5,7,1,3,8,6,4,2)

Problemas de las n reinas

Un algoritmo *vuelta atrás* :

```
Función nreinas (n,k,X): entero x entero x vector → bool
éxito ← false ; X(k)←0
Repetir
    X(k)←X(k)+1
    Si NOT amenaza(X,k) entonces
        Si k ≠ n entonces
            éxito ← nreinas(n,k+1,X)
        Sino
            éxito ← true
Hasta que (X(k) = n) OR  éxito
Retorna éxito
Fin
```

Llamada inicial: **nreinas(n,1,X)**

Problemas de las n reinas

La función amenaza controla las posiciones de las reinas ya ubicadas para detectar si existe una amenaza.

Función **amenaza** (X,k): vector x entero → bool

Para i=1 hasta k-1 hacer

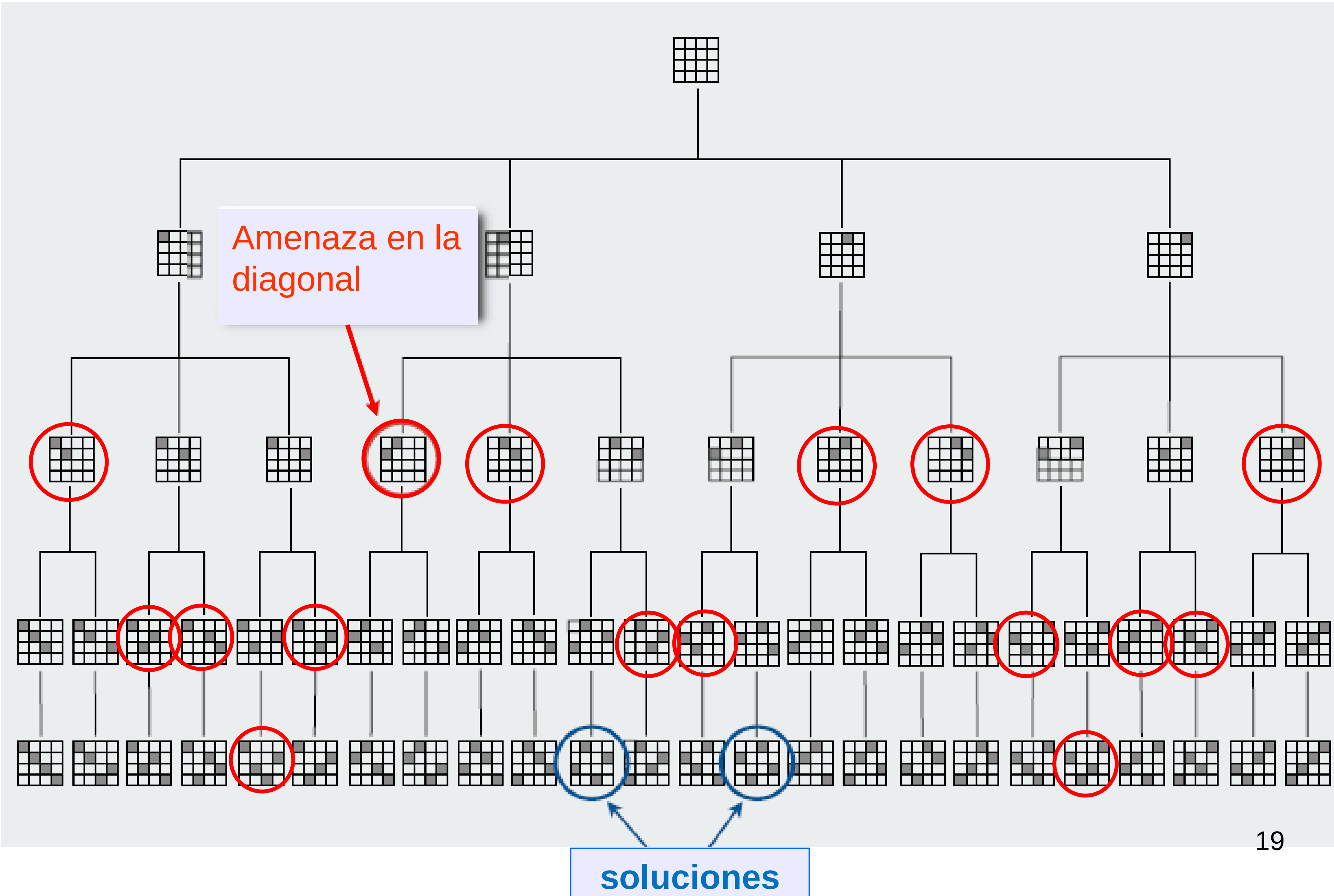
    Si X(i) = X(k) OR $|X(i)-X(k)| = |i-k|$ entonces

        Retorna true

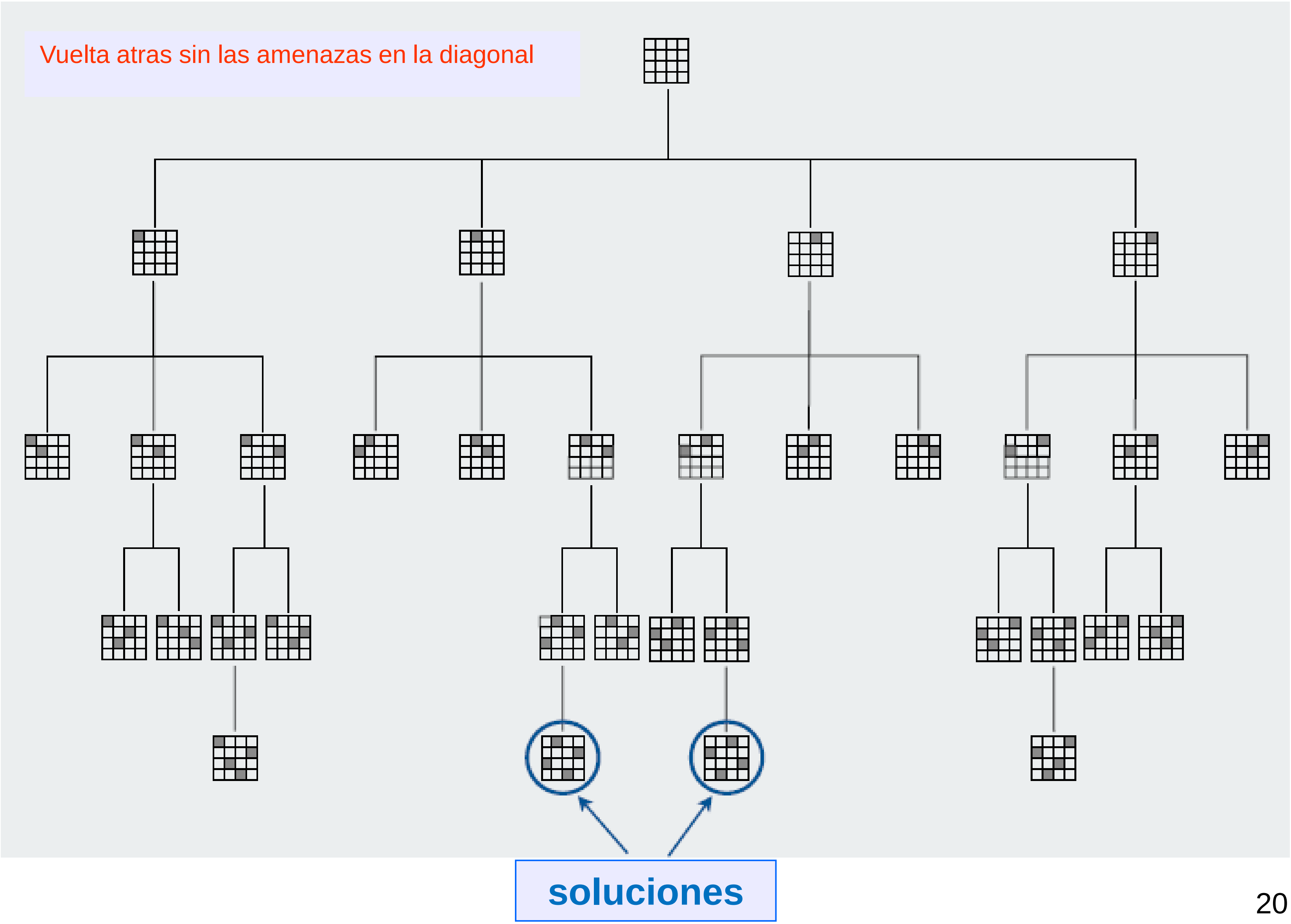
Retorna false

Fin

Problema de las 4 reinas, árbol con todas las posiciones posibles:

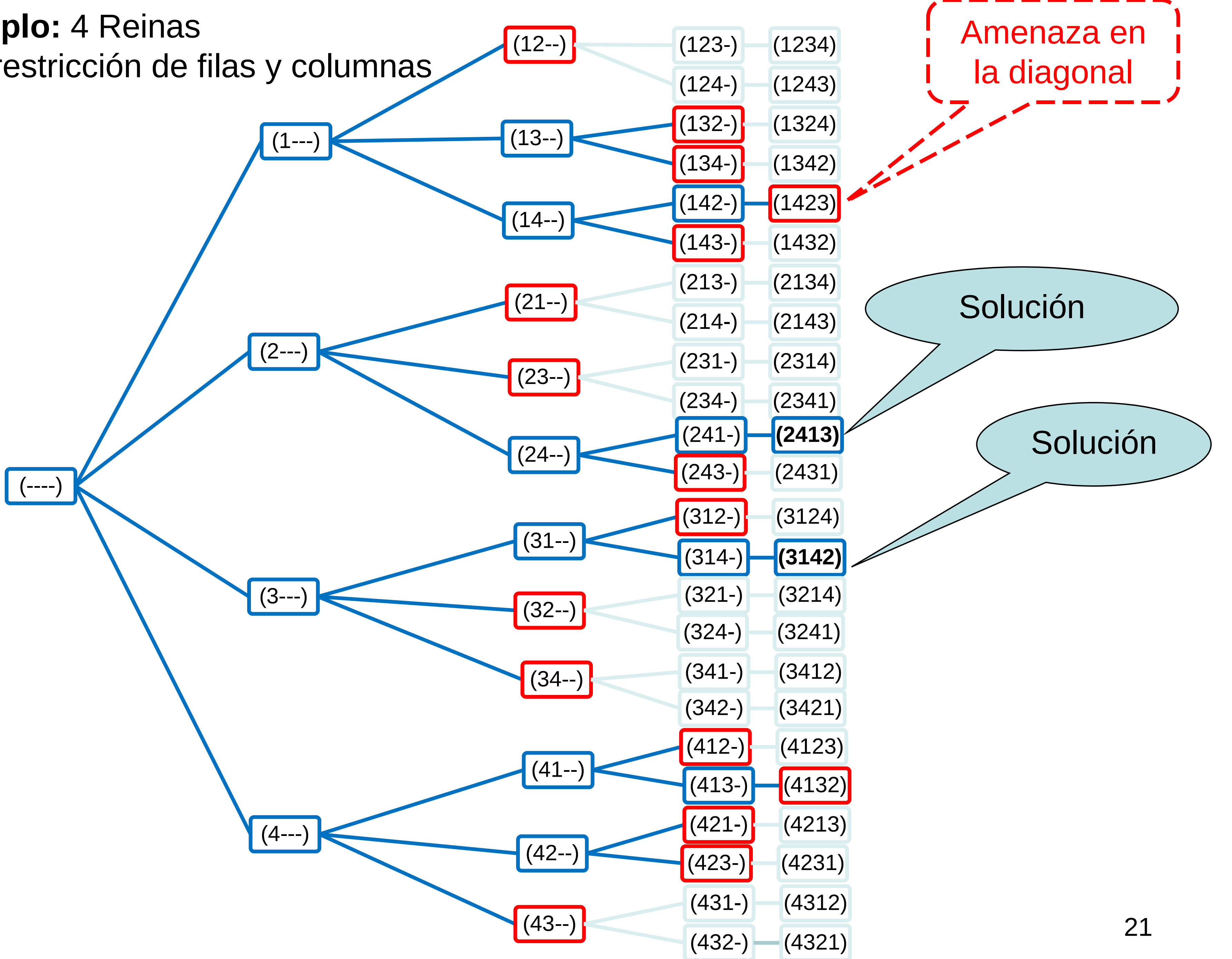


Problema de las 4 reinas, árbol podado:

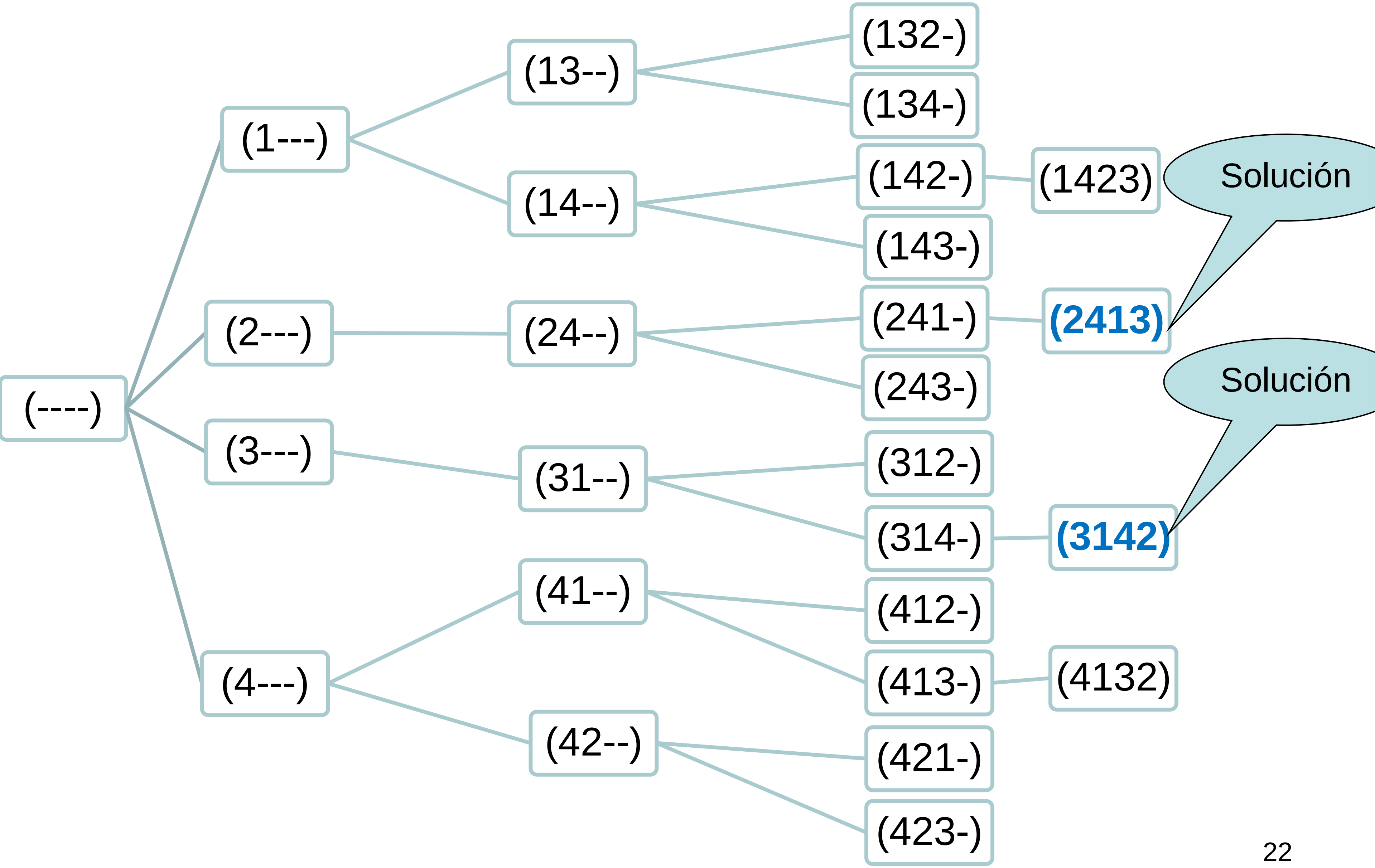


Ejemplo: 4 Reinas

Con restricción de filas y columnas



Ejemplo: 4 Reinas Con restricción de filas, columnas y diagonal



Problemas de las n reinas

El problema de las n reinas se considera una solución para un modelo de máxima cobertura, ya que la solución garantiza que cada objeto puede ser alcanzado desde cualquiera de sus 8 direcciones vecinas sin que tenga conflicto con otros objetos:

Algunas aplicaciones:

- Control de tráfico aéreo.
- Sistemas de comunicaciones.
- Planificación de tareas.
- Procesamiento paralelo.
- Compresión de datos.
- Ruteo de datos.
- Mayor ancho de banda en comunicaciones, distribución de transmisores y receptores.
- Y muchos más.

Problema de los matrimonios estables

Knuth, D. E. *Marriages stables*. Montreal: Les Presses de I'Universite de Montreal. (1976).

Problema de los matrimonios estables

En el problema de los matrimonios estables se busca encontrar una correspondencia entre hombres y mujeres que cumpla con la propiedad de *estabilidad.*

Se dice que una *correspondencia es estable* si no existe ninguna pareja que hubieran preferido estar juntos, antes que con su matrimonio actual.

Si existiera dicho par, el mismo se denomina **"*par inestable*"**

***La correspondencia es estable si no existe ningún par inestable*.**

Problema de los matrimonios estables

Una **matrimonio (*h*,*m*)** ***es estable*** si **<u>no se da</u>** ninguna de estas dos circunstancias:

1) Existe otro **matrimonio (*h’*,*m’*)** tal que:
el **hombre *h*** prefiere a la **mujer *m’*** sobre la **mujer *m***
y además la **mujer *m’*** también prefiere a ***h*** sobre ***h*’.**

2) Existe otro **matrimonio (*h*”,*m*”)** tal que:
la **mujer *m*** prefiere al **hombre *h*”** sobre el hombre ***h***
y además el **hombre *h*”** también prefiere a ***m*** sobre la **mujer *m*”.**

Definiciones

Definición: Dados dos conjuntos H y M, una ***correspondencia*** **S** es un conjunto de pares ordenados (**h**,**m**) donde **h** ∈ H y **m** ∈ M tal que:

- Cada hombre **h** ∈ H aparece en al menos un par de S.
- Cada mujer **m** ∈ M aparece en al menos un par de S.

Definición: una correspondencia **S** es ***perfecta*** si:

$$| S | = | H | = | M | = n.$$

Definiciones

Definición: dada una correspondencia perfecta S es ***no estable*** si contiene pares (**h**,**m**) y (**h’**,**m’**) tales que se da alguna de estas situaciones:

- El hombre **h** prefiere a **m’** más que a su pareja **m** y a su vez **m’** lo prefiere antes que a **h’**.
- La mujer **m** prefiere a **h’** más que a su pareja **h** y a su vez **h’** la prefiere antes que a **m’**.

Si no se da ninguna de estas situaciones la correspondencia S es estable.

Problema de los matrimonios estables

Ejemplo:

Datos: H={ h0,h1,h2,h3}, M={m0,m1,m2,m3} y sus preferencias:

h0: m0 m1 m2 m3	m0: h0 h2 h3 h1
h1: m2 m1 m3 m0	m1: h0 h1 h2 h3
h2: m2 m1 m3 m0	m2: h1 h2 h0 h3
h3: m1 m2 m3 m0	m3: h3 h0 h2 h1

La Solución {(h0, m0),(h1, m2),(h2, m1),(h3, m3)} es estable.

No hay pares inestables para esta solución.

Problema de los matrimonios estables

Ejemplo:

Datos: H={ h0,h1,h2,h3}, M={m0,m1,m2,m3} y sus preferencias:

h0: m0 m1 m2 m3	m0: h0 h2 h3 h1
h1: m2 m1 m3 m0	m1: h0 h1 h2 h3
h2: m2 m1 m3 m0	m2: h1 h2 h0 h3
h3: m1 m2 m3 m0	m3: h3 h0 h2 h1

La solución {(h0,m1),(h1,m2),(h2,m0),(h3,m3)} no es estable

El par (h0,m1) es inestable, porque el hombre h0 prefiere estar con la mujer m0 antes que con su pareja actual (m1), y m0 prefiere estar con el hombre h0 antes que con su pareja actual (h2).

Problema de los matrimonios estables

Ejemplo:

Datos: H={ h0,h1,h2,h3}, M={m0,m1,m2,m3} y sus preferencias:

h0: m0 m1 m2 m3	m0: h0 h2 h3 h1
h1: m2 m1 m3 m0	m1: h0 h1 h2 h3
h2: m2 m1 m3 m0	m2: h1 h2 h0 h3
h3: m1 m2 m3 m0	m3: h3 h0 h2 h1

La solución {(h0, m0),(h1, m3),(h2, m2),(h3, m1)} no es estable

Tiene 2 pares inestables:

- (h2, m2): m2 prefiere a h1 antes que a h2 y h1 la prefiere primero.
- (h1,m3): h1 prefiere a m2 antes que a m3 y m2 lo prefiere primero.

Problema: correspondencia estable

Problema de correspondencia estable: dada las listas de preferencias de n hombres y n mujeres, encontrar una correspondencia estable (si es que existe.)

Observación: la correspondencia estable puede no existir para un dado conjunto de datos.

El algoritmo de Gale-Shapley(*) es un método muy intuitivo que garantiza encontrar una correspondencia estable.

(*) D.Gale and L.S. Shapley ,
College admissions and the stability of marriage ,
American Mathematical Monthly, 69, pp9-15. , 1962.

```
ALGORITMO GALE-SHAPLEY // encuentra una correspondencia estable
Entrada:  Listas de Preferencias de hombres y mujeres
Salida:  S          // conjunto de parejas estables
S ← ∅

MIENTRAS (algún hombre h no tenga pareja AND
                    no propuso matrimonio a todas las mujeres) HACER
    m← primera mujer en la lista de preferencias de h a la que no se  propuso todavía
    SI m no está en pareja ENTONCES
                    Agregar la pareja (h,m) a S
    SINO    // m esta en pareja
        SI m prefiere a h sobre su actual pareja h’ ENTONCES
                    Sacar la pareja (h’,m) de S
                    Agregar la pareja (h,m) a S
        SINO
                    m rechaza a h
        FINSI
    FINSI
FIN MIENTRAS
Retorna S
Fin
```

Algoritmo GALE-SHAPLEY

- Teorema. [Gale-Shapley -1962] El algoritmo GS garantiza una correspondencia estable para cualquier instancia del problema.
- En la solución del algoritmo de GS no hay pares inestables.
- Todos los hombres encuentran su correspondencia.
- Los hombres proponen a las mujeres en orden decreciente de preferencia.
- El algoritmo es optimo desde el punto de vista de los hombres.
- Una vez que una mujer encontró pareja, ya no vuelve a estar sola, en todo caso mejora su situación.

- Tiempo. El algoritmo termina después de al menos n^2 *iteraciones de la iteración condicional mientras,* en cada iteración un hombre se propone a una nueva mujer, así, hay solo n^2 posibles propuestas.

Algoritmo para matrimonios estables

Diseñar un algoritmo que encuentre, si es que existe, un casamiento de hombres y mujeres tal que todos las matrimonios formados sean *matrimonios estables*.

Mas detalles del algoritmo con vuelta atrás:

<u>Entrada:</u>

- Se tiene ***n*** hombres y ***n*** mujeres y dos matrices ***M*** y ***H*** que contienen las preferencias de los unos por los otros.
- La fila $M(i,j),j=1..n$ es una ordenación (de mayor a menor) de las mujeres según las preferencias del i-ésimo hombre.
- La fila $H(i,j),j=1..n$ es una ordenación (de mayor a menor) de los hombres según las preferencias de la i-ésima mujer.

Algoritmo para matrimonios estables

El algoritmo vuelta atrás que resuelve el problema trabaja por etapas, completando dos vectores X e Y de n componentes que serán la solución del problema:

<u>Salida:</u>

vector *X* que contiene las mujeres asignadas a cada uno de los hombres.

X(i): indica el número de la mujer asignada al *i*-ésimo hombre

vector *Y* que contiene los hombres asignados a cada mujer.

Y(i): indica el número del hombre asignado a la *i*-ésima mujer

En cada etapa *k* se elige la mujer que se casa con el hombre *k*.

Para ello, en la etapa *k*, el *k*-ésimo hombre elegirá la mujer que prefiere en primer lugar, siempre y cuando esta mujer aún esté libre y el matrimonio resulte estable.

Algoritmo para matrimonios estables

Para saber las mujeres aún libres se usa un vector auxiliar denominado *libre*.

Auxiliar:

Se necesitan dos tablas auxiliares, *ordenM* y *ordenH* :

***ordenM*(*i,j*)** *:* almacena el orden de preferencia de la mujer *i* por el hombre *j*,

***ordenH* (*i,j*)** : almacena el orden de preferencia del hombre *i* por la mujer *j*.

El problema se resuelve mediante la inicialización apropiada de las matrices de preferencias: *M* y *H* que contienen las preferencias de los unos por los otros.

Se invoca a: ***Matrimonio*(1,*exito*)**

Cuando termina su ejecución, las variables *X* e *Y* contendrán la asignaciones respectivas siempre que la variable *exito* lo indique.

Función Matrimonio(hombre,exito) entero x bool → vector x vector

Entrada: hombre

Salida: éxito:bool, X,Y:vectores

```
prefiere←0  // recorre las posibles elecciones del hombre
Repetir
        prefiere← prefiere+1
        mujer←M(hombre,prefiere)
        Si libre(mujer) AND Estable(hombre,mujer,prefiere) entonces
                X(hombre) ←mujer
                Y(mujer) ←hombre
                libre(mujer) ←FALSE
                Si hombre<n entonces
                        Matrimonio(hombre+1,exito)
                        Si NOT exito entonces
                                libre(mujer) ←TRUE
                Sino
                        exito←TRUE
Hasta que  (prefiere=n) OR éxito
Retorna (X,Y)
Fin
```

```
Función Estable(h,m,p):hombre x mujer x prefiere→BOOL

si←TRUE          // alamacena si es estable
i←1
Mientras (i<p) AND si hacer        // es estable respecto al hombre?
        mejormujer←M(h,i)
        i←i+1
        Si NOT(libre(mejormujer))entonces
             si←ordenM(mejormujer,h)>ordenM(mejormujer,Y(mejormujer))
i←1
limite←ordenM(m,h)                 // es estable respecto a la mujer?
Mientras(i<limite) AND si hacer
        mejorhombre←H(m,i)
        i←i+1
        Si mejorhombre<h entonces
             si ←ordenH(mejorhombre,m)>ordenH(mejorhombre,X(mejorhombre))

Retorna si
Fin
```

Ejemplo para matrimonios estables

Encontrar una correspondencia estable en el siguiente caso:
Preferencia de los hombres:

$$M = \begin{pmatrix} m1 & m2 & m3 & m4 \\ m3 & m2 & m4 & m1 \\ m3 & m2 & m4 & m1 \\ m2 & m3 & m4 & m1 \end{pmatrix} \qquad ordenH = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \\ 4 & 2 & 1 & 3 \\ 4 & 1 & 2 & 3 \end{pmatrix}$$

Preferencia de las mujeres:

$$H = \begin{pmatrix} h1 & h3 & h4 & h2 \\ h1 & h2 & h3 & h4 \\ h2 & h3 & h1 & h4 \\ h4 & h1 & h3 & h2 \end{pmatrix} \qquad ordenM = \begin{pmatrix} 1 & 4 & 2 & 3 \\ 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \\ 2 & 4 & 3 & 1 \end{pmatrix}$$

Solución: { (h1, m1), (h3,m3), (h2,m2), (h4,m4) }
Salida: exito=true ***X=(1,3,2,4) ,*** ***Y=(1,3,2,4)***

Problema de Correspondencia

Otra aplicación:

Elección de residentes en los hospitales.

Datos:

- un conjunto de estudiantes aspirantes a ser residentes en los hospitales
- un conjunto de plazas vacantes en hospitales (cupos)
- una lista de preferencias de los participantes sobre las vacantes
- una lista de preferencias de los hospitales sobre los estudiantes.

Problema de Correspondencia

Algunas Variantes del problema de correspondencia pueden ser:

- Algunos participantes declaran a otros inaceptables
(por ej. Un residente no quiere trabajar en un determinado hospital)

- Distinto cantidad de residentes y de hospitales.

- Poligamia limitada
(Un hospital quiere tomar mas de un residente, por ej. 4 residentes)

- Matrimonios de residentes que quieran corresponder en el mismo hospital.

Problema de Correspondencia

Un conjunto de asignaciones aceptables es una correspondencia *estable*, si ningún estudiante y hospital se prefieren antes que sus asignaciones actuales.

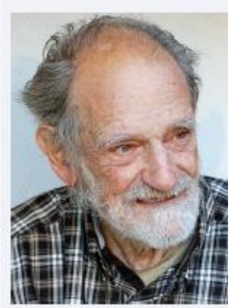
Gale y Shapley mostraron que siempre existe una correspondencia estable para cualquier conjunto de listas de preferencias, siempre que la lista de preferencias de cada agente incluya a todos los agentes del conjunto opuesto.

También está demostrado que existe una ***correspondencia óptima*** para uno de los dos conjuntos en el sentido que cada agente está en su mejor asignación posible, y que si hubiera otro correspondencia estable, estaría igual o menos conforme según sus preferencias.

Premio Nobel 2012

Premio Nobel en Economía: por contribuciones a la teoría de la asignación de recursos en mercados bilaterales y por las mejoras al funcionamiento de varios mercados centralizados, para 2 científicos:

- **Lloyd Shapley.** (1923-2016) Matemático y economista. Catedrático en la UCLA**.** Teoría de "Stable matching" y algoritmo de Gale-Shapley.

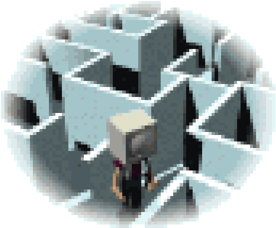


Lloyd Shapley

- **Alvin Roth**. (1951) (Economista, Universidad de Stanford) Aplicó el algoritmo de Gale-Shapley para hacer la correspondencia de médicos con hospitales, dadores de órganos con pacientes en lista de espera, estudiantes con escuelas y plasmó las ideas de Shapley en diferentes análisis de mercados que funcionan en forma centralizada.

Alvin Roth

Vuelta atrás (Backtracking)



Trabajo Práctico no. 7