Teoría de Algoritmos

Capitulo 5: Algoritmos para la Exploración de Grafos.

Tema 14: Backtracking y Branch and Bound

- Método general backtracking.
- Árboles. Espacio de estados
- Problema de las ocho reinas.
- Problema de la suma de subconjuntos.

El método General

- El backtracking es una de las técnicas mas generales para la exploración de Grafos.
- Muchos problemas que buscan en un conjunto de soluciones o que buscan una solución optimal que satisfaga algunas restricciones, pueden resolverse con backtracking.
- El nombre backtrack lo acuñó D.H. Lehmer en los 50.
- R.J. Walker, dio una versión algorítmica (1960)
- Golomb y Baumert presentaron una descripción general del backtracking asociada a una gran variedad de aplicaciones.

El método General

- Para aplicar el método backtracking, la solución deseada debe ser expresable como una n-tupla $(x_1, ..., x_n)$ en la que x_i se elige de algún conjunto finito S_i.
- A menudo el problema a resolver trata de encontrar un vector que maximiza (o minimiza) una función criterio $P(x_1, ...,x_n)$.
- A veces, también se trata de encontrar todos los vectores que satisfagan P.
- ¿No se parece esto a las técnicas Greedy o basadas en Programación Dinámica?

Un ejemplo

- Ordenar los enteros en $A(1..n)$ es un problema cuya solución es expresable mediante una n-tupla en la que x_i es el índice en A del i-esimo menor elemento.
- La función de criterio P es la desigualdad

$$A(x_i) \leq A(x_{i+1}),\ 1 \leq i \leq n.$$

- El conjunto S_i es finito e incluye a todos los enteros entre 1 y n.
- La ordenación no es uno de los problemas que habitualmente se resuelven con backtracking

El método general

- Sea m_i el tamaño del conjunto S_i.
- Hay $m = m_1 \cdot m_2 \cdot \ldots \cdot m_n$ n-tuplas posibles candidatos a satisfacer la función P.
- El enfoque de la fuerza bruta propondría formar todas esas tuplas y evaluar cada una de ellas con P, escogiendo la que diera un mejor valor.
- Backtracking proporciona la misma solución pero en mucho menos de m intentos.

El método general

- La idea básica es construir el mismo vector escogiendo una componente cada vez, y usando funciones de criterio modificadas $P_i(x_1, ..., x_n)$, que a veces se llaman funciones de acotación, para testear si el vector que se esta formando tiene posibilidad de éxito.
- La principal ventaja de este método es que si a partir del vector parcial $(x_1, x_2, ..., x_i)$ se deduce que no se podrá construir una solución, entonces pueden ignorarse por completo $m_{i+1} \cdot ... \cdot m_n$ posibles test de vectores.

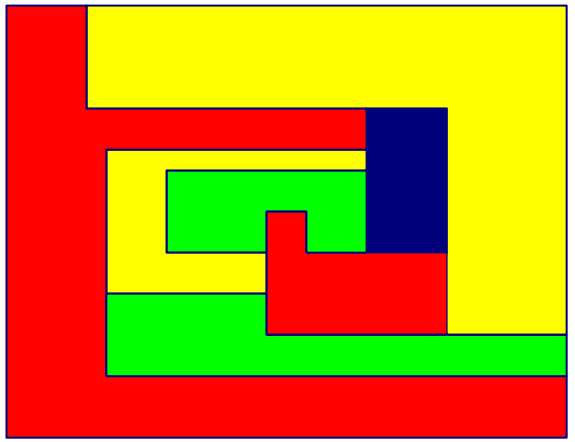
Diferencias con otras técnicas

- En los algoritmos greedy se construye la solución buscada, aprovechando la posibilidad de calcularla a trozos, pero con backtracking la elección de un sucesor en una etapa no implica su elección definitiva
- En Programación Dinámica, la solución se construye por etapas a partir del principio de optimalidad, y los resultados se almacenan para no tener que recalcular, lo que no es posible en Backtracking

Ejemplo: Salir de un laberinto

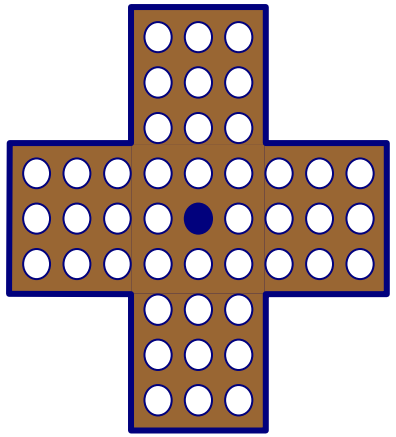
- En un laberinto, encontrar el camino desde la entrada hasta el final
- En cada intersección hay que decidir entre varias alternativas:
  - Seguir recto
  - Girar a la izquierda
  - Girar a la derecha
- No tenemos suficiente información para decidir bien
- Cada elección nos lleva a otro conjunto de elecciones
- Una sucesión de elecciones pueden (o no) llevarnos a una solución
- Muchos recorridos de laberintos pueden resolverse con backtracking

El coloreo de un mapa



- Queremos colorear un mapa con 4 colores a lo sumo
  - rojo, amarillo, azul, verde
- Los paises adyacentes deben tener colores diferentes
- No tenemos suficiente información para elegir los colores
- Cada elección nos conduce a otro conjunto de elecciones
- Una sucesión de elecciones puede (o no) llevarnos a una solución
- Muchos problemas de coloreo pueden resolverse con backtracking

Resolver un puzzle



- En este puzzle, todos las piedras menos una son blancas
- Podemos saltar de una piedra a otra
- Las piedras saltadas se eliminan
- Queremos eliminar todas las piedras menos la ultima
- No tenemos suficiente información para saltar correctamente
- Cada elección lleva a otro conjunto de elecciones
- Una sucesión de elecciones puede (o no) llevarnos a una solución
- Muchos problemas de puzzles pueden resolverse con backtracking

Tipos de restricciones en backtrack

- Muchos de los problemas que resolveremos usando backtracking requieren que todas las soluciones satisfagan un complejo conjunto de restricciones.
- En cualquier problema que consideremos, estas restricciones podrán dividirse en dos categorías:
  - explícitas e implícitas.

Tipos de restricciones en backtrack

- Las **restricciones explícitas** son reglas que restringen cada x_i a tomar valores solo en un conjunto dado. Ejemplos comunes de restricciones explícitas son,
- $x_i \geq 0$ S_i = {nºs reales no negativos}
- $x_i = 0,1$ $S_i = \{0,1\}$
- $l_i \leq x_i \leq u_i$ $S_i = \{a: l_i \leq a \leq u_i\}$
- Las restricciones explícitas pueden depender o no del caso particular del problema a resolver.

Tipos de restricciones en backtrack

- Todas las tuplas que satisfacen las restricciones explícitas definen un espacio de soluciones del caso que estamos resolviendo.
- Las **restricciones implícitas** determinan cual de las tuplas en ese espacio solución satisface la función de criterio.
- Las restricciones implícitas describen la forma en la que las x_i deben relacionarse entre si.

Tipos de restricciones: ejemplo

- Para este problema,

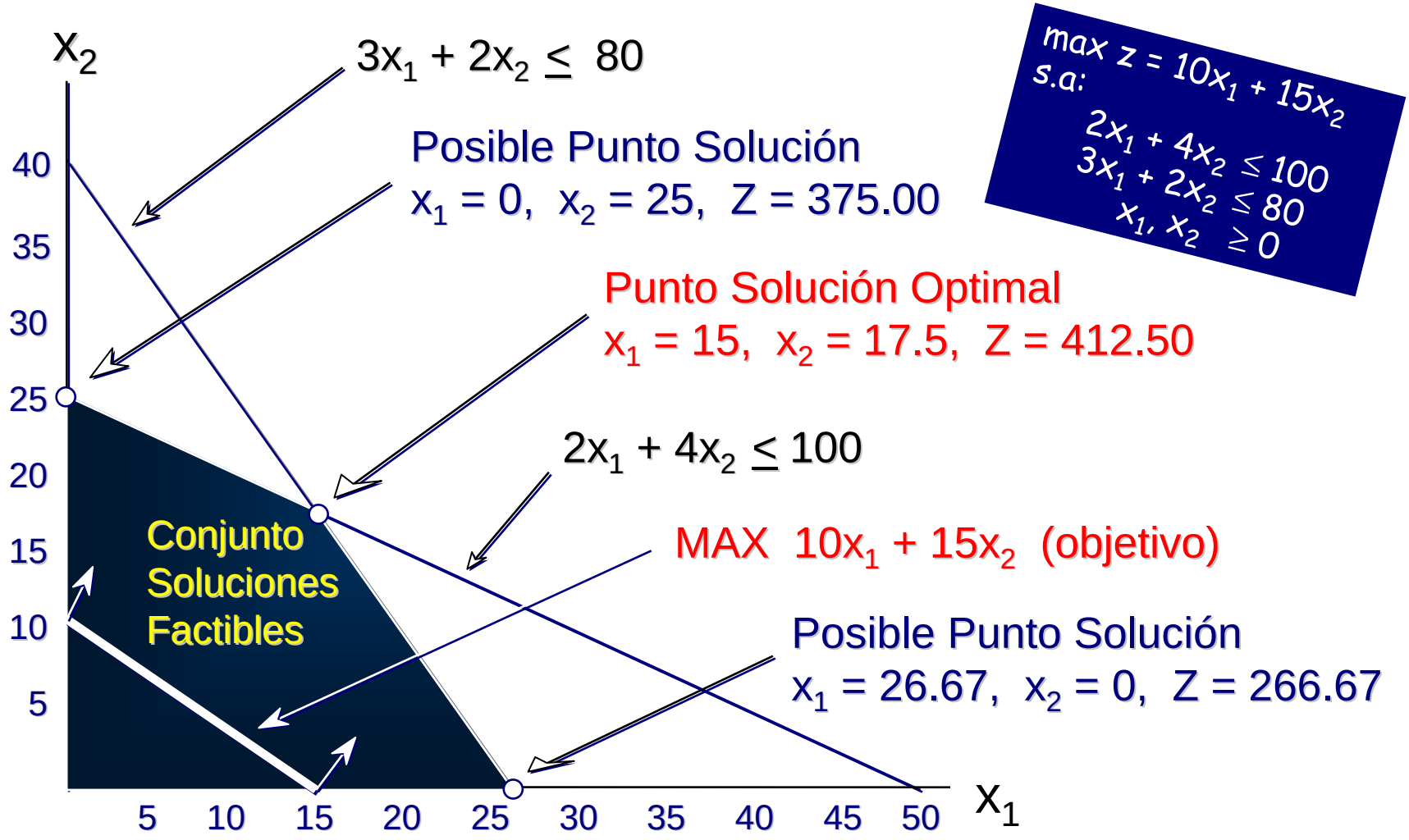
$$\max z = 10x_1 + 15x_2$$

s.a:

$$\begin{aligned} 2x_1 + 4x_2 &\leq 100 \\ 3x_1 + 2x_2 &\leq 80 \\ x_1, x_2 &\geq 0 \end{aligned}$$

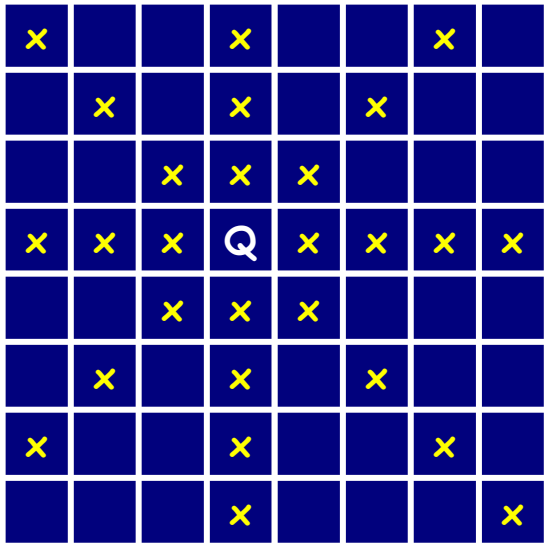
- Las restricciones explicitas las describe el conjunto de soluciones factibles (el conjunto donde toman valores las variables)
- Las restricciones implicitas describen los puntos que pueden ser solucion del problema (los puntos extremos del poliedro)

Tipos de restricciones: Interpretación



El problema de las ocho reinas

- Un clásico problema combinatorio es el de colocar ocho reinas en una tablero de ajedrez de modo que no haya dos que se ataquen, es decir, que estén en la misma fila, columna o diagonal.
- Las filas y columnas se numeran del 1 al 8.
- Las reinas se numeran del 1 al 8.

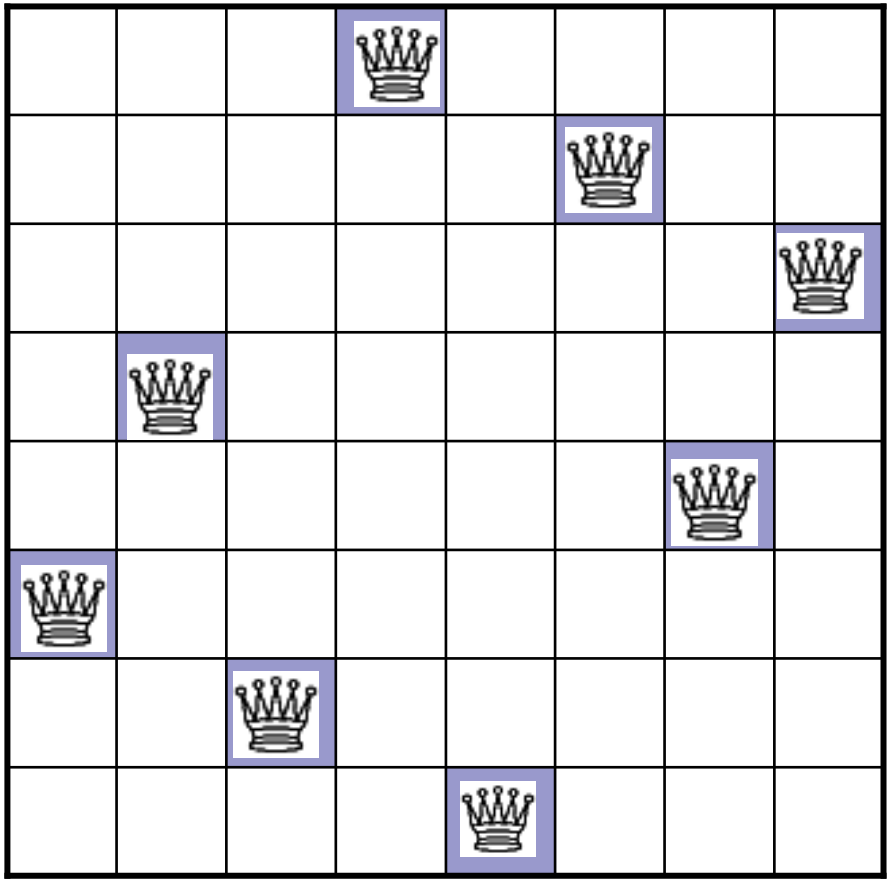


Como cada reina debe estar en una fila diferente, sin perdida de generalidad podemos suponer que la reina i se coloca en la fila i.

Todas las soluciones para este problema, pueden representarse como 8 tuplas $(x_1,\ldots,x_n)$ en las que x_i es la columna en la que se coloca la reina i.

El problema de las ocho reinas

- Las restricciones explícitas son Si = {1,2,3,4,5,6,7,8}, $1 \leq i \leq n$.
- Espacio solución con $8^8 = 2^{24}$ = 16M tuplas.
- Restricciones implícitas: ningún par de x_i puedan ser iguales (todas las reinas deben estar en columnas diferentes). Ningún par de reinas pueden estar en la misma diagonal.
- La primera de estas dos restricciones implica que todas las soluciones son permutaciones de (1,2,3,4,5,6,7,8).
- Esto lleva a reducir el tamaño del espacio solución de 8^8 tuplas a 8! = 40,320
- Una posible solución del problema es la (4,6,8,2,7,1,3,5).



Problema de la suma de subconjuntos

- Dados n+1 números positivos:

  w_i, $1 \le i \le n$, y uno mas M,

- se trata de encontrar todos los sub-conjuntos de números w_i cuya suma valga M.
- Por ejemplo, si n = 4, (w_1, w_2, w_3, w_4) = (11, 13, 24, 7) y M = 31, entonces los subconjuntos buscados son (11, 13, 7) y (24, 7).

Problema de la suma de subconjuntos

- Para representar la solución podríamos notar el vector solución con los índices de los correspondientes w_i.
- Las dos soluciones se describen por los vectores (1,2,4) y (3,4).
- Todas las soluciones son k-tuplas $(x_1, x_2,\ldots,x_n)$, $1 \leq k \leq n$, y soluciones diferentes pueden tener tamaños de tupla diferentes.
- Restricciones explícitas: $x_i \in \{j: j \text{ es entero y } 1 \leq j \leq n\}$
- Restricciones implícitas: que no haya dos iguales y que la suma de los correspondientes w_i sea M.
- Como, por ejemplo (1,2,4) y (1,4,2) representan el mismo subconjunto, otra restricción implícita que hay que imponer es que $x_i < x_{i+1}$, para $1 \leq i < n$.

Problema de la suma de subconjuntos

- Puede haber diferentes formas de formular un problema de modo que todas las soluciones sean tuplas satisfaciendo algunas restricciones.
- Otra formulación del problema:
  - Cada subconjunto solución se representa por una n-tupla $(x_1,\ldots,x_n)$ tal que $x_i \in \{0,1\}$, $1 \leq i < n$, y $x_i = 0$ si w_i no se elige y $x_i = 1$ si se elige w_i.
  - Las soluciones del anterior caso son (1,1,0,1) y (0,0,1,1).
  - Esta formulación expresa todas las soluciones usando un tamaño de tupla fijo.
- Se puede comprobar que para estas dos formulaciones, el espacio solución consiste en ambos casos de 2^4 tuplas distintas.

Espacios de soluciones

- Los algoritmos backtracking determinan las soluciones del problema buscando en el espacio de soluciones del caso considerado sistemáticamente.
- Esta búsqueda se facilita usando una organización en árbol para el espacio solución.
- Para un espacio solución dado, pueden caber muchas organizaciones en árbol.
- Los siguientes ejemplos examinan algunas de las formas posibles para estas organizaciones.

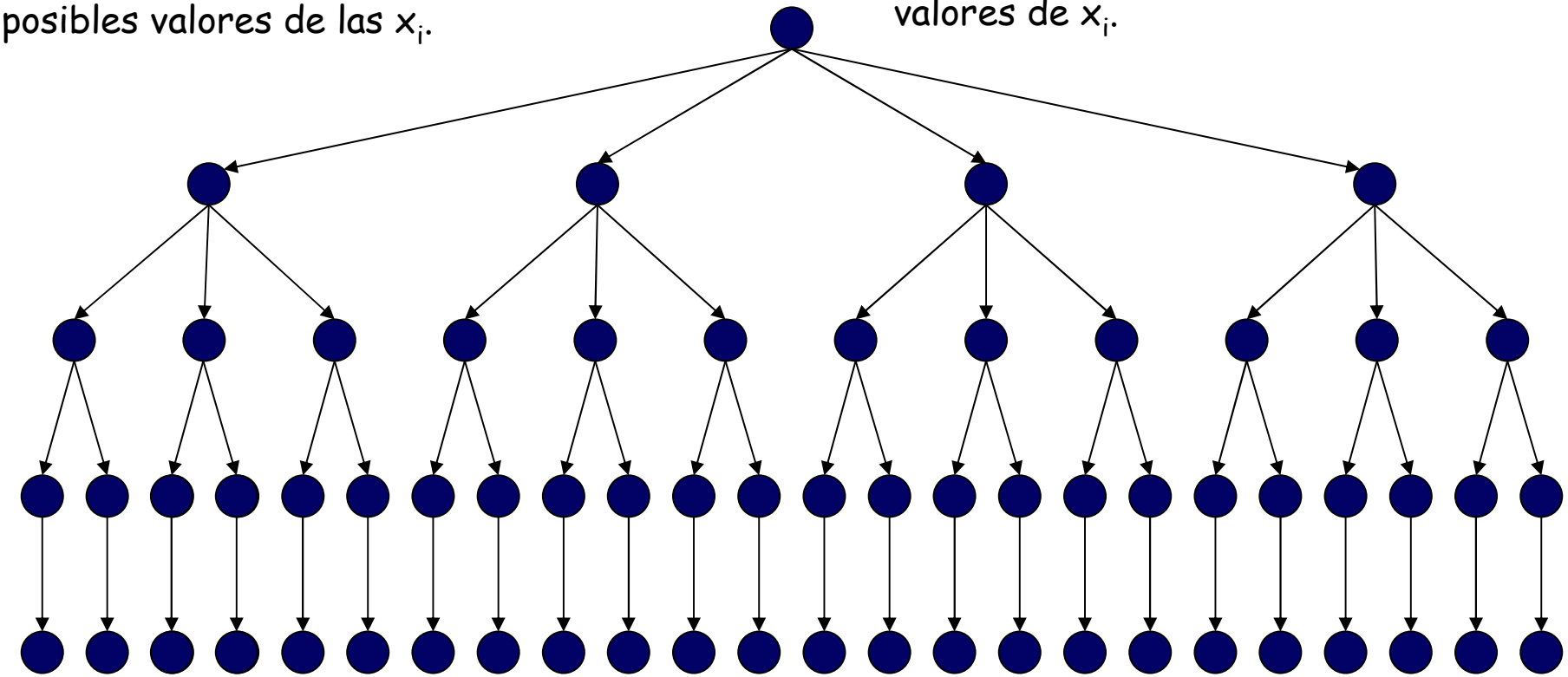
Ejemplo de espacio de soluciones

- La generalización del problema de las 8 reinas es el de las N reinas: Colocar N reinas en un tablero NxN de modo que no haya dos que se ataquen
- Ahora 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
- La siguiente figura muestra una posible organización de las soluciones del problema de las 4 reinas en forma de árbol.
- A un árbol como ese se le llama **Árbol de** (búsqueda de soluciones) **Permutación.**

4-Reinas: Arbol de permutación

Las aristas se etiquetan con los posibles valores de las x_i.

Las aristas desde los nodos del nivel nivel i al i+1 están etiquetadas con los valores de x_i.



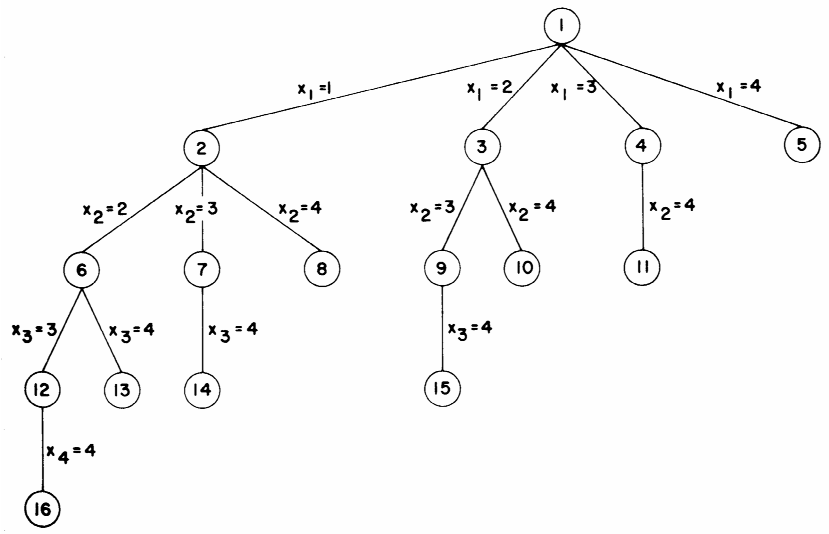
El subárbol de la izquierda contiene todas las soluciones con $x_1 = 1$ y $x_2 = 2$, etc. El espacio de soluciones esta definido por todos los caminos desde el nodo raíz a un nodo hoja. Hay 4! = 24 nodos hoja en la figura.

Suma de subconjuntos: árbol

- Vimos dos posibles formulaciones del espacio solución del problema de la suma de subconjuntos.
  - La primera corresponde a la formulación por el tamaño de la tupla
  - La segunda considera un tamaño de tupla fijo
- Con ambas formulaciones, tanto en este problema como en cualquier otro, el numero de soluciones tiene que ser el mismo

Suma de subconjuntos: árbol 1

- Las aristas se etiquetan de modo que una desde el nivel de nodos i hasta el i+1 representa un valor para x_i.
- En cada nodo, el espacio solución se particiona en espacios subsolución.
- Los posibles caminos son (), que corresponde al camino vacío desde la raíz a si misma, (1), (12), (123), (1234), (124), (134), (2), (23), etc.
- Así, el subarbol de la izquierda define todos los subconjuntos conteniendo w_1, el siguiente todos los que contienen w_2 pero no w_1, etc.



Suma de subconjuntos: árbol 2

- Una arista del nivel i al i+1 se etiqueta con el valor de x_i (0 o 1).
- Todos los caminos desde la raíz a las hojas definen el espacio solución.
- El subarbol de la izquierda define todos los sub-conjuntos conteniendo w_1, mientras que el de la derecha define todos los subconjuntos que no contienen w_1, etc.
- Consideramos el caso de n = 4
- Hay 2^4 nodos hoja, que representan 16 posibles tuplas.

Suma de subconjuntos: árbol 2