

Backtracking: Introducción

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski

✉ vpodberezski@fi.uba.ar

Jerarquía de espacio de soluciones

En un problema combinatorio

tenemos “n” elementos,

Podemos conformar diferentes posibles soluciones “combinando” los elementos.

Podemos expresar una posible solución

Como una tupla de como mucho $t \leq n$ elementos $(x_1, x_2, \dots, x_{t-1}, x_t)$

Existen un subconjunto de posibles soluciones

que comienzan con los mismos “t-1” elementos iniciales

A su vez estos forman parte de un conjunto de soluciones que inician con “t-2” mismo elementos.

Esto nos permite establecer una jerarquía en el espacio de soluciones.

Que podemos representar mediante un árbol de decisiones

Ejemplo: 8-reinas

Contamos con

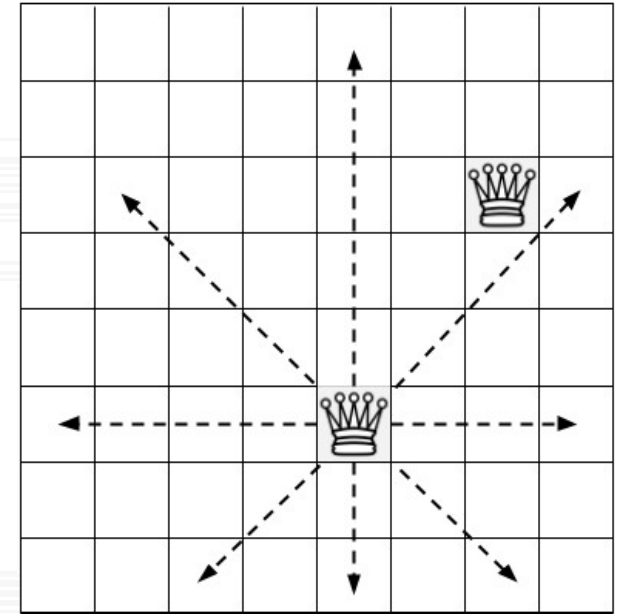
un tablero de ajedrez de 8 filas por 8 columnas.

Sabemos que

la pieza conocida como “reina” ubicada en un celda del tablero ataca a toda pieza que se encuentra en su misma fila, columna o diagonal.

Queremos

ubicar 8 reinas en el tablero de forma tal que ninguna de ellas ataque a otra.



Ejemplo: 8-reinas – Espacio de soluciones

Podemos considerar

a cada celda como un elemento

a la elección de 8 celdas como una posible solución

buscamos combinaciones de 64 elementos tomados de a 8.
(4.426.165.368 posibles soluciones)









Lo representamos como un vector donde cada elemento es mayor a los anteriores.

La posible solución (2,12,24,27,37,47,49,62)

Cumple las restricciones explícitas (8 celdas diferentes)

No cumple las restricciones implícitas (La reina en la celda 2 ataca a la reina en la celda 47)

(2,12,25,27,37,47,49,62) comparte con la solución anterior las primeras dos elecciones de celdas

1		3	4	5	6	7	8
9	10	11		13	14	15	16
17	18	19	20	21	22	23	
25	26		28	29	30	31	32
33	34	35	36		38	39	40
41	42	43	44	45	46		48
	50	51	52	53	54	55	56
57	58	59	60	61		63	64

Ejemplo: 8-reinas – Jerarquía en el espacio de soluciones

Si la primera elección corresponde a la celda 1

Existen $\binom{63}{7}$ cantidad de posibles soluciones que se pueden construir

Si la primera elección corresponde a la celda 5

Existen $\binom{59}{7}$ cantidad de posibles soluciones que se pueden construir

Si las primeras dos elecciones corresponden a la celda a las celdas 2 y 5

Existen $\binom{59}{6}$ cantidad de posibles soluciones que se pueden construir

Si las primeras dos elecciones corresponden a la celda a las celdas 4 y 20

Existen $\binom{44}{6}$ cantidad de posibles soluciones que se pueden construir

Ejemplo: 8-reinas – Jerarquía en el espacio de soluciones

Podemos representar las elecciones (y los estados del problema) como un árbol

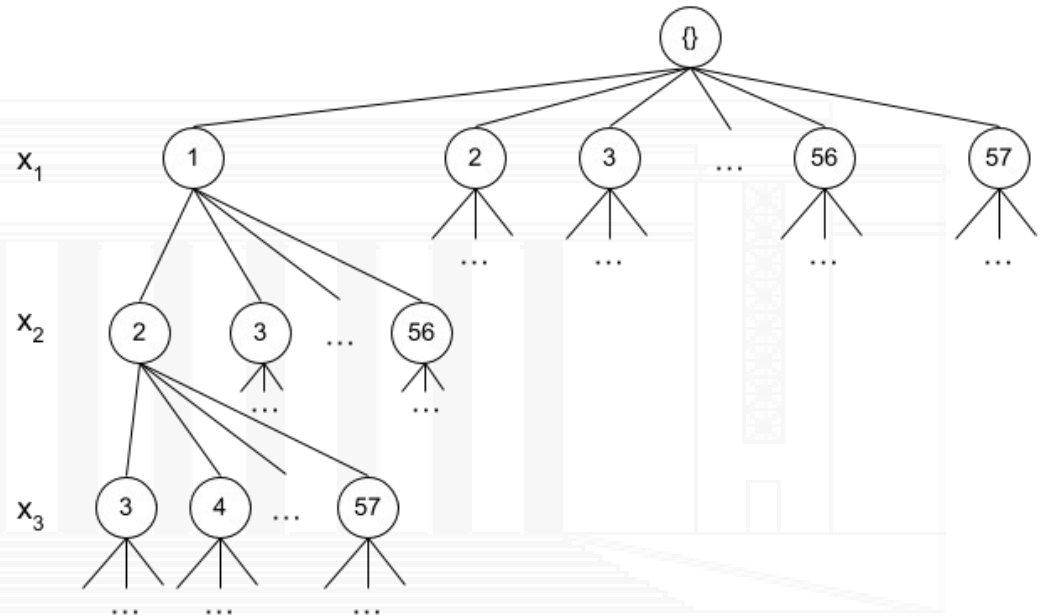
La raíz corresponde al tablero vacío

Los nodos en el nivel “i” en el árbol a seleccionar una celda en la i-esima posición del vector de soluciones

La profundidad máxima del árbol será 8

Existirán $\binom{64}{8}$ hojas en el árbol

Recorrer el árbol de forma exhaustiva nos permite encontrar todas las posibles soluciones



Clasificación de estados en árbol de estados

Estados del problema

Corresponden a todos los nodos del árbol de estados

Estados solución

Subconjunto de estados del problema

Corresponde a aquellos que cumplen con las restricciones explícitas del problema

Estados respuesta

Subconjunto de estados solución

Corresponde a aquellos que cumplen con las restricciones implícitas del problema

Propiedad de corte

Tamaño del árbol de estados

La cantidad de estados del problema es igual o mayor que la cantidad de estados solución

Muchas veces no es necesario recorrer todo el árbol

En ocasiones al realizar las primeras j decisiones podemos saber si existe una posible respuesta al problema que las incluya

Llamaremos propiedad de corte

A la posibilidad de evaluar en un estado del problema la ausencia de algún estado respuesta descendiente del mismo

Aplicaremos una función límite a un estado del problema para evaluar la propiedad de corte

En caso de aplicarse la propiedad de corte, desistiremos la exploración de los descendientes de ese nodo del árbol

Diremos que “PODAMOS” el árbol.

Ejemplo: 8-reinas – Propiedad de corte

Considerar el estado del problema (1,2)



Corresponde a incluir una reina en la celda 1 y otra en la 2

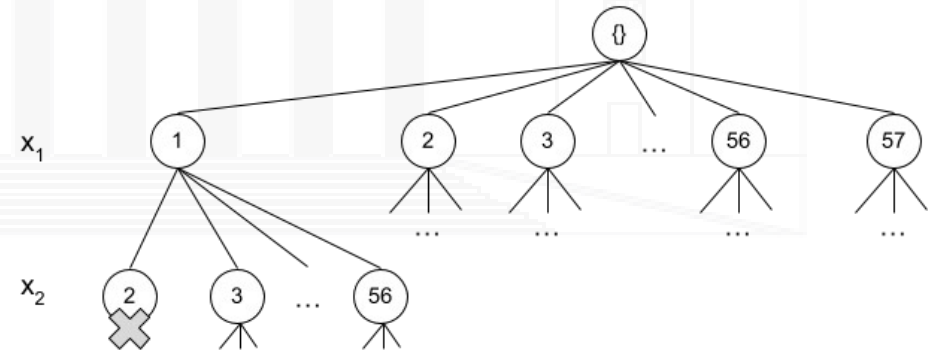
Estas se atacan entre sí

Cualquier estado solución (1,2,x₃,x₄,...,x₈)

No sera un estado respuesta (no cumple con las restricciones implícitas)

Por lo tanto, por propiedad de corte, podemos evitar la exploración de los estados del problema descendientes de (1,2)

		3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64



Recorrido y creación del árbol de estados del problema

Generaremos el árbol de estados de forma dinámica

Partiremos de la raíz

Por cada posible estado de problema descendiente se generarán sus descendientes

Existen diferentes maneras de recorrer el árbol

Utilizaremos Depth-First Search

Se adentrará lo más que puede en la profundidad del árbol

Se evalúa si se hay encontrado una solución.

Se retrocede (Backtrack) cuando no quedan caminos por recorrer en la rama actual

Este retroceso ocurre cuando todos los descendientes ya fueron explorados o podados

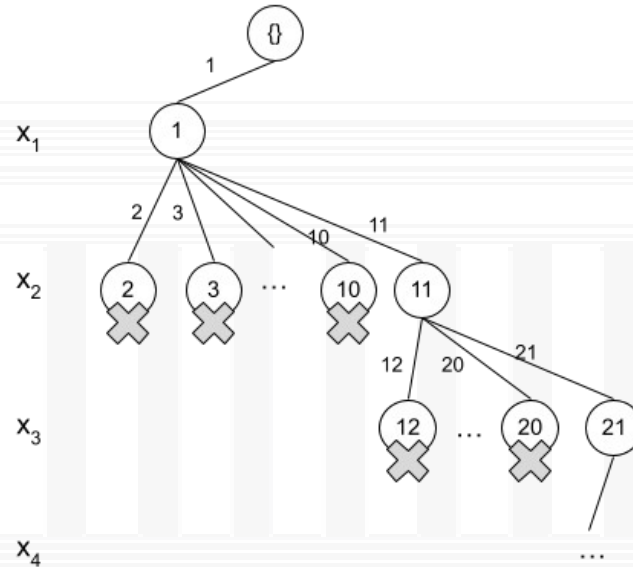
Ejemplo: 8-reinas – Recorrido del árbol




Se presentan primeros pasos en el recorrido del árbol

Al incluir la reina en la celda 1 se podan todos los nodos del problema descendentes que tienen a la segunda en las celdas 2 a 10.

Al incluir la segunda reina en la celda 11 se podan aquellos descendentes que tiene a la tercera en las celdas todas las soluciones que tienen

El orden de exploración se muestra como una etiqueta en el eje del árbol



	2	3	4	5	6	7	8
9	10		12	13	14	15	16
17	18	19	20		22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Backtracking

Se conoce como Backtracking

Al mecanismo de crear dinámicamente el árbol de estados del problema

Recorrerlo mediante Depth-First Search

Podar la exploración utilizando la propiedad de corte

El algoritmo fue trabajado y explorado inicialmente entre la década del 50 y 60

Se reconoce a D. H. Lehmer en los 50 como quien acuñó esta denominación.

A R. J. Walker en los 60 por darle forma algorítmica.

A S. Golomb y L. Baumert por demostrar su aplicabilidad en una gran variedad de situaciones

Backtracking – Pseudocódigo recursivo

Backtrack (estadoActual):

Si estadoActual es un estado resultado
retornar estadoActual

Si estadoActual supera la propiedad de corte
Por cada posible estadoSucesor de estadoActual
resultado = Backtrack(estadoSucesor)

si resultado es un estado resultado
retornar resultado

retornar vacio

Sea estadoInicial la raiz del arbol de estados
Backtrack(estadoInicial)