1. Enunciado del problema: Laberinto.

El problema consiste en encontrar la salida de un laberinto. Más concretamente, supondremos que el laberinto se representa mediante una matriz bidimensional de tamaño n x n.

Cada posición almacena un valor 0 si la casilla es transitable y cualquier otro valor si la casilla no es transitable. Los movimientos permitidos son a casillas adyacentes de la misma fila o la misma columna. Podemos suponer que las casillas de entrada y salida del laberinto son (0,0) y (n-1, n-1) respectivamente. Por tanto el problema consiste en, dada una matriz que representa el laberinto, encontrar si existe un camino para ir desde la entrada hasta la salida.

Para ello deberemos implementar un algoritmo backtracking para resolver el problema. Después modificar el algoritmo para que encuentre el camino más corto. Realizaremos también un estudio empírico de la eficiencia del algoritmo.

2. Introducción general al algoritmo Backtracking.

El backtracking es una estrategia usada para encontrar soluciones a problemas que tienen una solución completa, en los que el orden de los elementos no importa, y en los que existen una serie de variables a cada una darle un valor teniendo en cuenta unas restricciones dadas.

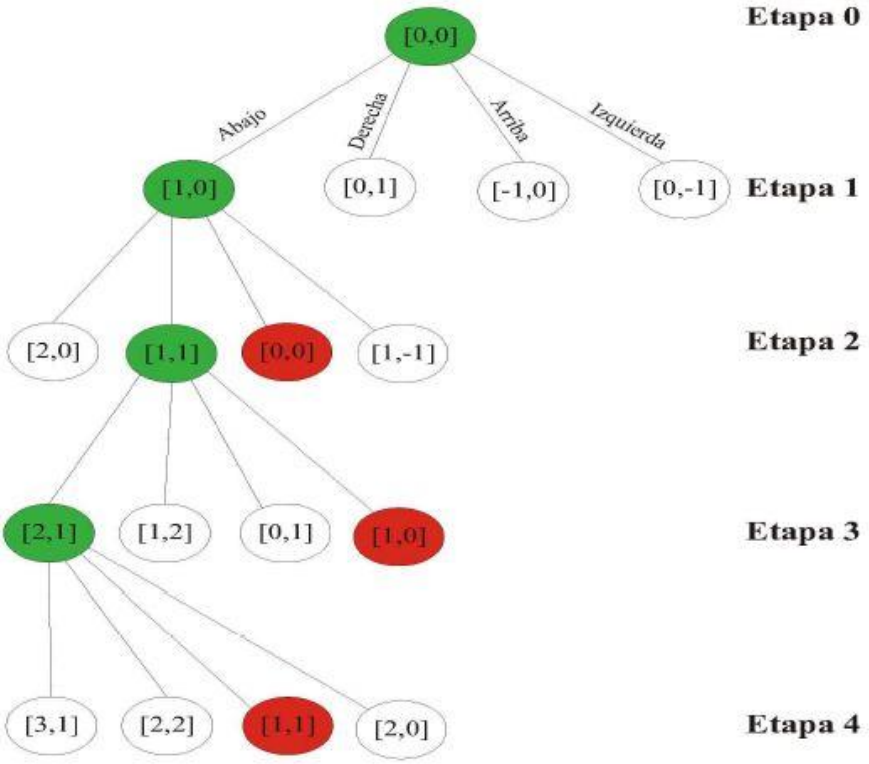
Esta estrategia se asemeja a un recorrido en profundidad dentro de un grafo dirigido. El grafo en cuestión suele ser un árbol, pero esta estructura existe sólo implícitamente.

De forma general, el método del backtracking, genera todas las secuencias de forma sistemática y organizada, de manera que prueba todas las posibles combinaciones construyendo soluciones parciales hasta llegar a la correcta.

La forma de actuar consiste en elegir una alternativa del conjunto de opciones de cada etapa del proceso de resolución, y si esta elección no funciona (no nos lleva a ninguna solución), la búsqueda vuelve al punto donde realizó dicha elección, e intenta con otro valor. Cuando se han agotado todos los posibles valores en ese punto, la búsqueda vuelve a la anterior fase en la que se hizo otra elección entre valores, Si no hay más puntos de elección, la búsqueda finaliza.

2.1. Backtracking en el problema del laberinto.

La representación que se hace en este problema es de un árbol de tal manera que se representa en cada nodo una etapa y los hijos de dichos nodos serán las alternativas que se nos ofrecen en cada etapa. Como decisión tomaremos un hijo por el que seguir explorando y expandiendo el árbol, siempre en profundidad. A continuación se muestra el ejemplo de un árbol de búsqueda para el problema del laberinto:



Partimos de la posición [0,0] y por ello es la raíz del árbol, a partir de este salen las distintas alternativas que podemos tomar, para ir avanzando descartamos las posibilidades que nos saquen del mapa del laberinto y por la que ya hemos pasado, tomando una de las posibilidades que nos queden, este paso se repite hasta encontrar una salida (o no).
Como caso particular, en el problema del laberinto el árbol tendría una altura infinita, si no se controlaran con la poda de las ramas.

3. Solución para el problema del laberinto y la mejor solución del problema.

Para su resolución utilizamos la técnica Backtracking, lo primero que hace es crear un laberinto con los valores, como ya hemos dicho antes, de 0 en las casillas que son transitables y uno las que no. Primero se toma la primera decisión que será moverse desde la casilla [0, 0] hacia el este o si no puede va hacia el sur, el movimiento lo va alternando el movimiento en las agujas del reloj (es decir, sur, oeste, norte y después volviendo al este, si en este caso el primer paso es al este, si fuese al sur seria oeste, norte y este volviendo después al sur).

De esta manera va avanzando el camino a la salida, si en algún momento la rama en la que nos encontramos no llega a una solución vuelve atrás por la recursividad y sigue explorando el espacio de soluciones hasta dar con la solución adecuada.
Para la mejor solución va guardando el número de recorridos y cogiendo el menor.
Alguno de los ejemplos son estos:

```
Tama: 10
Entrada: (0 , 0)
Salida: (9 , 9)

0 1 1 0 0 0 1 1 0 0
0 0 0 0 0 0 1 0 0 1
0 1 0 1 1 1 1 0 1 1
0 0 0 0 1 1 0 0 0 0
1 1 1 1 1 1 1 0 1 1
1 1 0 1 1 1 0 0 1 0
0 1 0 0 1 1 1 1 0 0
0 1 1 1 0 0 0 1 0 0
0 1 1 0 0 0 1 1 1 0
1 1 1 1 1 1 1 1 0 0

Solucion:

No hay solucion

MEJOR Solucion:

No hay solucion
```

```
Tama: 6
Entrada: (0 , 0)
Salida: (5 , 5)

0 0 0 0 0 0
0 1 1 1 1 0
0 0 0 0 0 0
0 1 1 1 1 1
0 0 0 1 1 1
0 0 0 0 0 0

Solucion:
E --> E --> E --> E --> E --> S --> S --> 0 --> 0 --> 0 --> 0 --> 0 --> S --> S --> E --> E --> S --> E --> E --> E -->
MEJOR Solucion:
S --> S --> S --> S --> E --> E --> S --> E --> E --> E -->
```

```
Tama: 10
Entrada: (0 , 0)
Salida: (9 , 9)

0 1 1 1 0 1 1 1 1 0
0 0 1 1 0 0 0 1 0 0
1 0 0 0 0 0 0 0 0 0
1 0 0 0 1 0 0 0 0 1
1 0 1 0 1 1 0 0 1 0
0 0 0 0 1 1 1 1 1 1
1 0 1 1 1 0 0 1 1 0
0 0 0 1 0 0 0 0 0 1
0 0 0 0 0 1 1 1 0 0
0 1 1 1 1 0 0 1 1 0

Solucion:
S --> E --> S --> E --> E --> S --> S --> S --> O --> O --> S --> S --> E --> S --> E --> E --> N --> E --> E --> E --> E --> S --> E --> S -->
MEJOR Solucion:
S --> E --> S --> S --> S --> S --> S --> S --> E --> S --> E --> E --> N --> E --> E --> E --> E --> S --> E --> S -->
```

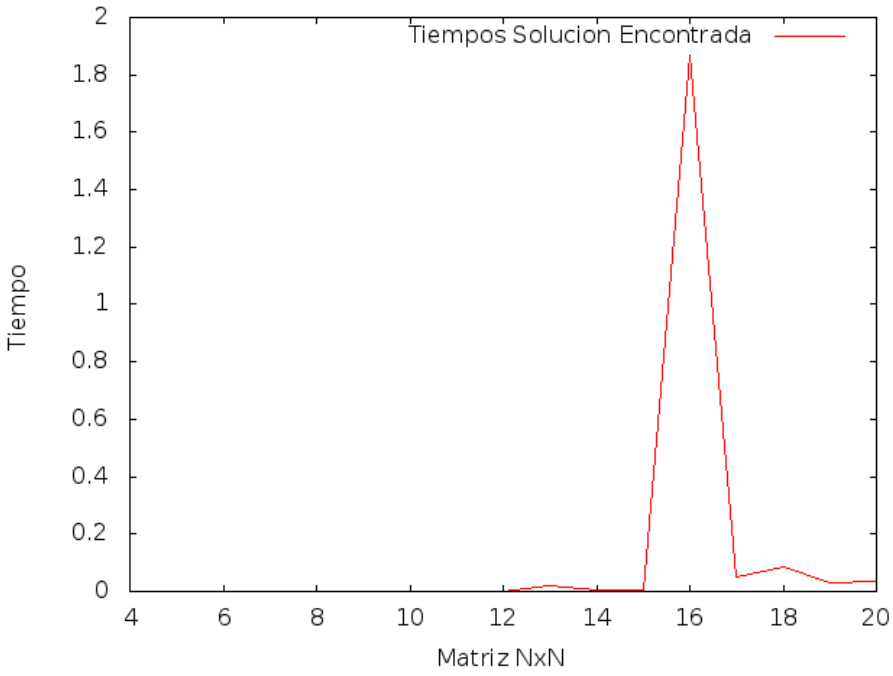
4. Cálculo de la eficiencia empírica del problema del laberinto.

El cálculo de la eficiencia se ha llevado a cabo utilizando chrono.

Para la representación de las gráficas se ha tenido en cuenta la media de tiempos de varios laberintos realizados al alzar por el programa.

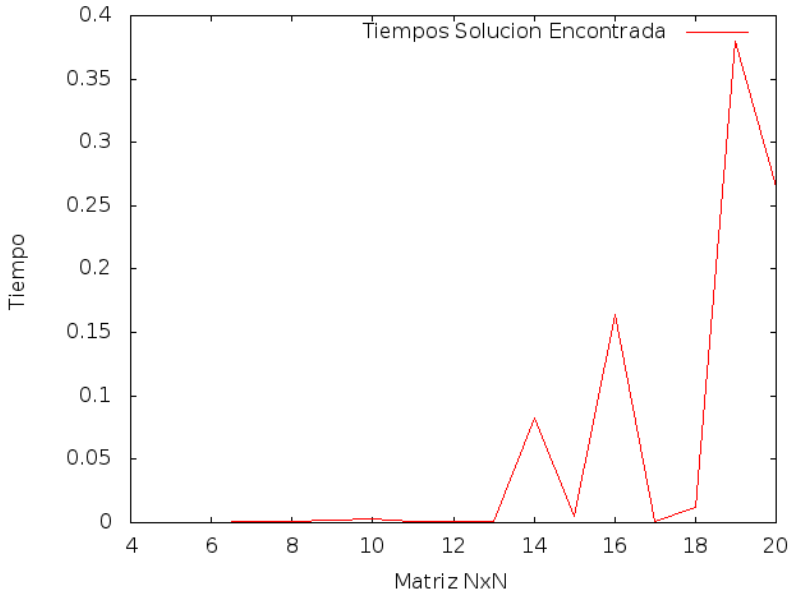
4.1. Primera solución.

Elementos matriz	Tiempo(s)	Elementos matriz	Tiempo(s)
4	3,43E-03	13	0.0199105
5	2,96E-02	14	0.00431616
6	9,83E-03	15	0.00699563
7	3,19E-03	16	1.87063
8	5,40E-01	17	0.0521821
9	4,19E-01	18	0.0874728
10	0.00114296	19	0.031764
11	0.00164059	20	0.0330296
12	0.000128574		



4.2. Algoritmo mejorado

Elementos matriz	Tiempo(s)	Elementos matriz	Tiempo(s)
4	1,80E+00	13	0.00139031
5	9,50E+00	14	0.0825609
6	0.000140302	15	0.00549129
7	0.000645815	16	0.164093
8	0.000541086	17	0.00131956
9	0.00197997	18	0.0122801
10	0.00351746	19	0.379662
11	0.000585826	20	0.264537
12	0.000698876		



Como vemos para hallar la mejor solución del laberinto el tiempo utilizado es más que para cualquiera de las soluciones que pudiera tener este.

5. Conclusión.

La conclusión de este programa es que no necesariamente, la mejor solución conlleva un tiempo menor a cualquier otra solución posible, por ello el óptimo de la solución es el que menos casillas recorre pero no necesariamente el que menos tiempo tarda.