UNIVERSIDAD JOSÉ ANTONIO PÁEZ

FACULTAD DE INGENIERIA

ESCUELA DE COMPUTACIÓN

CÁTEDRA DE REDES DE COMPUTADORAS

**BACKTRACKING Y METODOS DE BUSQUEDA**

**Autor:**

* Manuel Antonio Reyes 17079726
* Carlos Silva 15951742

San Diego, 03 de Septiembre de 2007

**Backtracking**

* **Técnica general de resolución de problemas, que suele aplicarse sobre todo a juegos y problemas de óptimización.**
* **Realiza una búsqueda exhaustiva y sistemática en el espacio de soluciones del problema.**
* **La solución de un problema de backtracking se puede expresar como una tupla (x1,x2,…,xn), que satisface una restricciones R(x1,x2,…,xn) y a veces optimizando una función objetivo.**
* **En cada momento el algoritmo se encontrará en un cierto nivel k, con una solución parcial (x1,x2,…,xk) (con k<=n).**
  + **Si puede añadirse un elemento xk+1 a la solución parcial se avanza**

**al nivel k+1.**

* + **Si no se prueban otros valores válidos para xk.**
  + **Si no existe ningún valor que sea válido por probar, se retrocede al nivel anterior k-1.**
  + **Se continua con este proceso hasta que la solución parcial sea una solución del problema o hasta que no queden más posibilidades por probar (en el caso de que no se encuentre ninguna solución o se busquen todas las soluciones del problema).**

**Complejidad**

* **Por realizar una búsqueda exhaustiva en el espacio de soluciones del problema, los algoritmos de backtracking son bastante ineficientes.**
* **En general, se tienen tiempos con órdenes de complejidad factoriales o exponenciales.**
* **Por esto, los algoritmos de backtracking se utilizan en problemas para los que no existen un algoritmo eficiente**

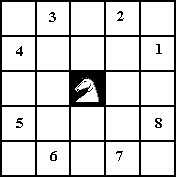
**que los resuelva.**

La Vuelta del Caballo

Se dispone de un tablero rectangular, por ejemplo el tablero de ajedrez, y de un caballo, que se mueve según las reglas de este juego. El objetivo es encontrar una manera de recorrer todo el tablero partiendo de una casilla determinada, de tal forma que el caballo pase una sola vez por cada casilla. Una variante es obligar al caballo a volver a la posición de partida en el último movimiento.  
Por último se estudiará como encontrar todas las soluciones posibles partiendo de una misma casilla.

Para resolver el problema hay que realizar todos los movimientos posibles hasta que ya no se pueda avanzar, en cuyo caso hay que dar marcha atrás, o bien hasta que se cubra el tablero. Además, es necesario determinar la organización de los datos para implementar el algoritmo.

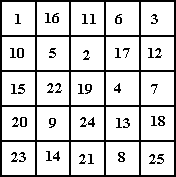
- ¿Cómo se mueve un caballo?. Para aquellos que no sepan jugar al ajedrez se muestra un gráfico con los ocho movimientos que puede realizar. Estos movimientos serán los ocho candidatos.



Con las coordenadas en las que se encuentre el caballo y las ocho coordenadas relativas se determina el siguiente movimiento. Las coordenas relativas se guardan en dos arrays:

ejex = [2, 1, -1, -2, -2, -1,   1,  2]  
ejey = [1, 2,  2,  1,  -1, -2, -2, -1]

El tablero, del tamaño que sea, se representará mediante un array bidimensional de números enteros. A continuación se muestra un gráfico con un tablero de tamaño 5x5 con todo el recorrido partiendo de la esquina superior izquierda.



Cuando se encuentra una solución, una variable que se pasa por referencia es puesta a 1 (cierto). Puede hacerse una variable de alcance global y simplificar un poco el código, pero esto no siempre es recomendable.

Para codificar el programa, es necesario considerar algunos aspectos más, entre otras cosas no salirse de los límites del tablero y no pisar una casilla ya cubierta (selección del candidato). Se determina que hay solución cuando ya no hay más casillas que recorrer.

A continuación se expone un código completo en C, que recubre un tablero cuadrado de lado N partiendo de la posición (0,0).

#include <stdio.h>

#define N 5

#define ncuad N\*N

void mover(int tablero[][N], int i, int pos\_x, int pos\_y, int \*q);

const int ejex[8] = { -1,-2,-2,-1, 1, 2, 2, 1 },

ejey[8] = { -2,-1, 1, 2, 2, 1,-1,-2 };

int main(void)

{

int tablero[N][N]; /\* tablero del caballo. \*/

int i,j,q;

/\* inicializa el tablero a cero \*/

for (i = 0; i < N; i++)

for (j = 0; j < N; j++)

tablero[i][j] = 0;

/\* pone el primer movimiento \*/

tablero[0][0] = 1;

mover(tablero,2,0,0,&q);

if (q) { /\* hay solucion: la muestra. \*/

for (i = 0; i < N; i++) {

for (j = 0; j < N; j++)

printf("%3d ", tablero[i][j]);

putchar('\n');

}

}

else

printf("\nNo existe solucion\n");

return 0;

}

void mover(int tablero[][N],int i, int pos\_x, int pos\_y, int \*q)

{

int k, u, v;

k = 0;

\*q = 0;

do {

u = pos\_x + ejex[k]; v = pos\_y + ejey[k]; /\* seleccionar candidato \*/

if (u >= 0 && u < N && v >= 0 && v < N) { /\* esta dentro de los limites? \*/

if (tablero[u][v] == 0) { /\* es valido? \*/

tablero[u][v] = i; /\* anota el candidato \*/

if (i < ncuad) { /\* llega al final del recorrido? \*/

mover(tablero,i+1,u,v,q);

if (!\*q) tablero[u][v] = 0; /\* borra el candidato \*/

}

else \*q = 1; /\* hay solucion \*/

}

}

k++;

} while (!\*q && k < 8);

}

Cambiando el valor de *N* puede obtenerse una solución para un tablero cuadrado de tamaño N.

A continuación, se muestra una adaptación del procedimiento que muestra todas las soluciones. Si se ejecuta para N = 5 se encuentra que hay 304 soluciones partiendo de la esquina superior izquierda.

Cuando se encuentra una solución se llama a un procedimiento (no se ha codificado aquí) que imprime todo el tablero.

void mover(int tablero[][N],int i, int pos\_x, int pos\_y)  
{  
  int k, u, v;

  for (k = 0; k < 8; k++) {  
    u = pos\_x + ejex[k]; v = pos\_y + ejey[k];   
    if (u >= 0 && u < N && v >= 0 && v < N) { /\* esta dentro de los limites \*/  
      if (tablero[u][v] == 0) {   
        tablero[u][v] = i;    
        if (i < ncuad)     
          mover(tablero,i+1,u,v);  
        else imprimir\_solucion(tablero);  
        tablero[u][v] = 0;    
      }  
    }  
  }  
}

**BIBLIOGRAFÍA**

**Algoritmos de Búsqueda**  
**“Backtracking y Branch and Bound”**

**Antonio Paredes Risueño**

<http://www.cablemodem.com/specifications/specifications30.html>

http://es.wikipedia.org/wiki/**Backtraking**

[www.bandaancha.st/weblogart.php?artid=4124](http://www.bandaancha.st/weblogart.php?artid=4124)