UAEM | Universidad Autónoma del Estado de México

Universidad Autónoma del Estado de México

Centro Universitario UAEM Atlacomulco

“Recorrido del tablero de ajedrez”

Presenta:

Elias Edgardo Segundo Antonio

Revisa:

Ingeniero José Luis García Morales

Unidad de Aprendizaje:

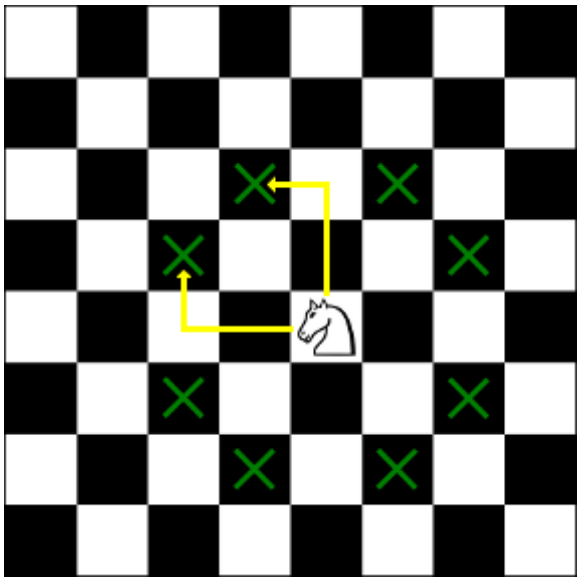
Programación avanzada

Fecha de entrega:

30 de abril de 2017

Planteamiento

El caballo en el ajedrez es una pieza que puede tener los siguientes 8 movimientos:



Estos movimientos pueden representarse mediante la siguiente tabla:

Horizontal	**Vertical**
1	2
2	1
-1	2
-2	1
-1	-2
-2	-1
1	-2
2	-1

Con este set de movimientos, se debe de escribir un programa que partiendo de una posición trivial pueda encontrar el camino para recorrer todo el tablero usando movimientos de caballo solamente o en su caso encuentre el camino para cubrir la mayor cantidad de tablero posible para un tablero cuadrado de lado n donde n>2.

Método de solución

Para este problema se hizo uso de comparaciones entre el número de movimientos posibles para cada posición y sus posiciones subsecuentes, es decir, si me encuentro en la posición 0,0 dentro de la matriz que representa el tablero entonces a partir de esta posición debo de encontrar la nueva posición dentro de las posibles opciones que tenga menor número de movimientos posibles.

Para esto se implementó un código que representase el tablero como una matriz, entonces en cada iteración se obtenía un arreglo de opciones, para encontrar la opción optima se buscaba para cada una de esas opciones la que tuviese menos posibles movimientos a realizar, entonces se establecía esa nueva posición y se continuaba con la búsqueda.

Código fuente

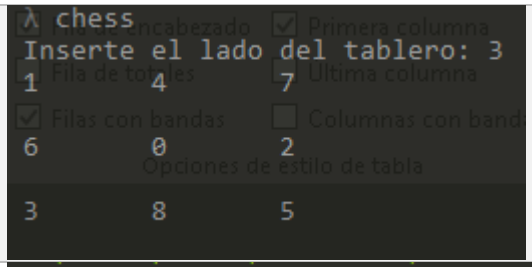
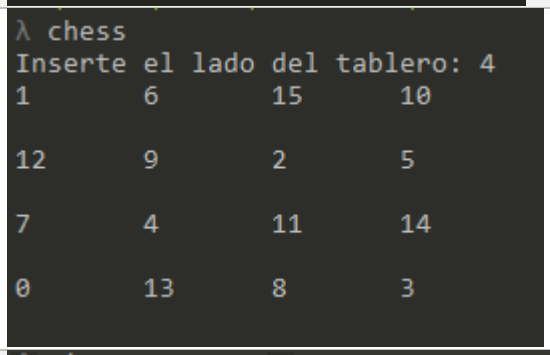
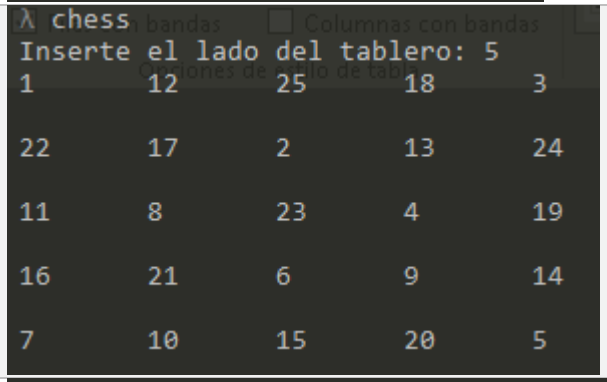
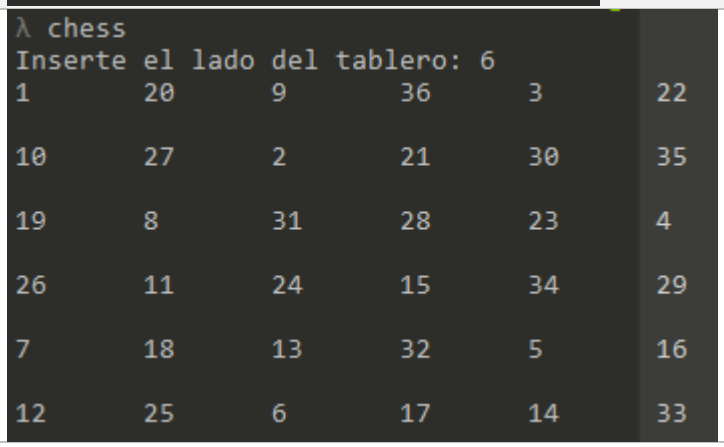
chess.cpp

```
#include <bits/stdc++.h>
using namespace std;


vector<vector<int>> horseMoves;

bool isValidPath(vector<vector<int>> board, int x, int y)
{
                        return x>=0 && x<board.size() && y>=0 && y<board.size() &&
!board[x][y];
}

vector<vector<int>> posibleMoves(vector<vector<int>> board, vector<int> pos)
{
                        vector< vector<int>> newPositions;
                        int newX, newY;
                        for( vector<int> move: horseMoves)
                        {
                           newX=pos[0]+move[0];
                           newY=pos[1]+move[1];
                           if(isValidPath(board,newX,newY))
                           {
                                    vector<int> newPos;
                                    newPos.push_back(newX);
```

```
                                        newPos.push_back(newY);
                                        newPositions.push_back(newPos);
                                    }
                                }
                                return newPositions;
}

vector<int> bestOption(vector<vector<int>> board, vector<vector<int>> candidates )
{
                                vector<int> weights(candidates.size(),0);
                                for(int i=0;i<candidates.size();i++)
                                {
                                    weights[i]= posibleMoves(board,candidates[i]).size();
                                }
                                int minIndex=-1;
                                int min=1e9;
                                if (candidates.size()==1)
                                {
                                    min=weights[0];
                                    minIndex=0;
                                    return candidates[minIndex];
                                }
                                for(int i=0;i<candidates.size();i++)
                                {
                                    if(weights[i]<min and weights[i]!=0)
                                    {
                                        min=weights[i];
                                        minIndex=i;
                                    }
                                }
                                vector<int>a;
                                if (minIndex!=-1)
                                {
                                    return candidates[minIndex];
                                }
                                else return a;
}

int main()
{
                                horseMoves={ {1,2} ,{2,1}, {-1,-2}, {-2,-1}, {-1,2}, {1,-2}, {-2,1}, {2,-1}
};
                                vector<vector<int>> board;
                                int size=0;
                                cout<<"Inserte el lado del tablero: ";
```

```
                        cin>>size;
                        for(int i=0;i<size;i++)
                        {
                           vector<int> row;
                           for(int j=0;j<size;j++)
                                   row.push_back(0);
                           board.push_back(row);
                        }

                        vector<int> pos(2,0);
                        int step=1;
                        vector<vector<int>> candidates;
                        bool isOver=false;
                        while(step-1!=(board.size()*board.size()) and !isOver)
                        {
                           board[pos[0]][pos[1]]=step;
                           step++;
                           candidates=posibleMoves(board,pos);
                           if(bestOption(board,candidates).size())
                           {
                                   pos=bestOption(board,candidates);
                           }
                           else isOver=true;
                        }

                        for(int i=0;i<board.size();i++)
                        {
                           for(int j=0;j<board.size();j++)
                           {
                                   cout<<board[i][j]<<"\t";
                           }
                           cout<<"\n\n";
                        }
        return 0;
}
```

Capturas

n (lado del tablero)	Captura de solución
3	
4	
5	
6	

7	λ chess Inserte el lado del tablero: 7 1 12 33 38 3 14 17 32 37 2 13 16 39 4 11 24 45 34 41 18 15 46 31 36 25 44 5 40 23 10 47 42 35 26 19 30 49 8 21 28 43 6 9 22 29 48 7 20 27
8	λ chess Inserte el lado del tablero: 8 1 16 63 34 3 18 21 36 52 33 2 17 64 35 4 19 15 62 53 48 43 20 37 22 32 51 46 61 54 49 42 5 59 14 55 50 47 44 23 38 28 31 60 45 56 41 6 9 13 58 29 26 11 8 39 24 30 27 12 57 40 25 10 7

En conclusión, de acuerdo al algoritmo planteado solo se encontraron soluciones de recorrido para los tableros mayores a 4 de longitud, siendo que los tableros de 5,6,7,8 si obtuvieron solución mientras que para el de 3,4 no existió solución para recorrer todas las posiciones pero si un estimado aceptable.