**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**

**KHOA CÔNG NGHỆ THÔNG TIN**

MÔN PHÂN TÍCH THIẾT KẾ THUẬT TOÁN

---------o0o---------

****

BÁO CÁO BÀI TẬP LỚN

**BÀI TOÁN MÃ ĐI TUẦN**

**THUẬT TOÁN QUAY LUI**

**Sinh viên thực hiện**:

Đoàn Minh Giang (MSV: 181201630)

**Giáo viên hướng dẫn :**

Giảng viên *:Phạm Xuân Tích*

Hà Nội, ngày 30 tháng 05 năm 2020

1. ***Đề bài và ý tưởng***
2. **Đề bài**: Mã đi tuần (Hành trình của quân mã) là phải di chuyển quân mã trên bàn cờ vua n\*n cho sẵn (trống). Quân mã phải đi theo quy tắc của bàn cờ vua và không đi được đi qua 1 ô 2 lần sao cho đi đươc hết các ô của bàn cờ
3. **Ý tưởng:**

* Có 2 khả năng xảy ra:

Quân mã đi hết các ô của bàn cờ hoặc quân mã không đi được hết các ô của bàn cờ

- Sử dụng mảng lưu lại tọa độ có thể đi được của quân mã trên bàn cờ (8 tọa độ đi được theo trục Oxy)

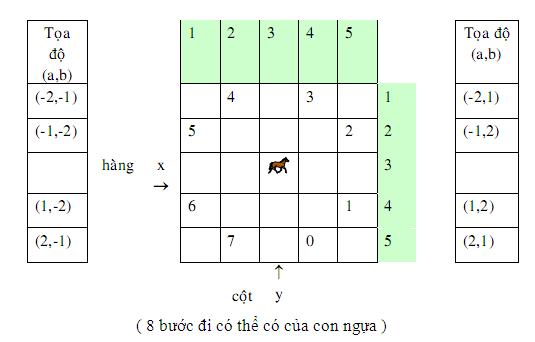
***\*) Tính chất***

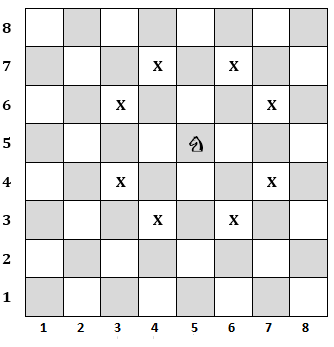
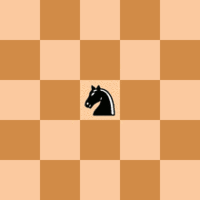
- Bước đi hợp lệ x>0 & i<=n và y>0 & y<=n

- Ô đó đã đi qua 1 lần hay chưa :

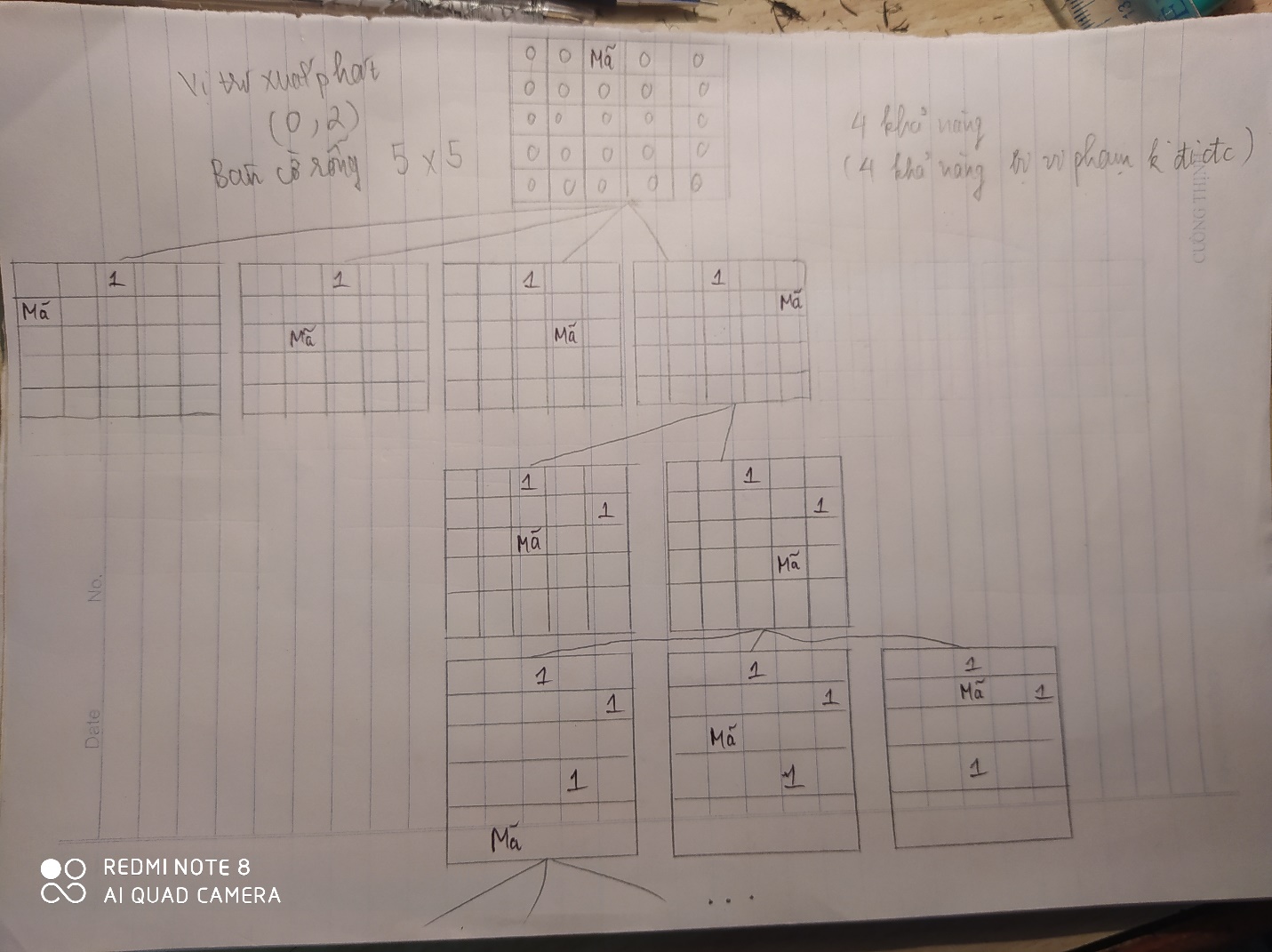
visited[nextX][nextY] = false

- Đi đến ô có số thứ tự bằng kích thước bàn cờ thì dừng step = n\*n





Minh họa sơ lược với bàn cờ 5 x 5



2. Code

\*) Hiển thị 1 cách đi

#include<bits/stdc++.h>

**using** **namespace** std**;**

**class** KnightTour

**{**

**private:**

**int** n**;**

**int** **\*\***visited**;**

**int** xMoves**[**8**]** **=** **{**2**,** 1**,** **-**1**,** **-**2**,** **-**2**,** **-**1**,** 1**,** 2**};**

**int** yMoves**[**8**]** **=** **{**1**,** 2**,** 2**,** 1**,** **-**1**,** **-**2**,** **-**2**,** **-**1**};**

**public:**

KnightTour**(int** n**)**

**{**

**this->**n **=** n**;**

init**();**

**}**

**void** init**()**

**{**

visited **=** **new** **int\*[**n**+**1**];**

**for(int** i**=**0**;**i**<**n**;**i**++)**

visited**[**i**]=new** **int[**n**+**1**];**

**for(int** i**=**0**;**i**<** n**;**i**++)**

**for(int** j**=**0**;**j**<**n**;**j**++)**

visited**[**i**][**j**]** **=**std**::**numeric\_limits**<int>::**min**();**

**}**

**void** print**()**

**{**

**for(int** i**=**0**;**i**<** n**;**i**++){**

**for(int** j**=**0**;**j**<**n**;**j**++)**

**{**

cout**<<**visited**[**i**][**j**]<<" ";**

**}**

cout**<<**endl**;**

**}**

**}**

**bool** TRY**(int** step**,** **int** x**,** **int** y**)**

**{**

**if(**step **==** n **\*** n**)** // di duoc het ban` co

**return** **true;**

**for(int** i**=**0**;** i **<** 8**;** i**++)**

**{**

**int** nextX **=** x **+** xMoves**[**i**];**

**int** nextY **=** y **+** yMoves**[**i**];**

**if(**check**(**nextX**,** nextY**)** **&&** visited**[**nextX**][**nextY**]** **==** std**::**numeric\_limits**<int>::**min**())**

// ktra buoc di hop le va chua di qua

**{**

visited**[**nextX**][**nextY**]** **=** step**;** // danh dau vi tri

**if(**TRY**(**step **+** 1**,**nextX**,** nextY**))** **return** **true;**

visited**[**nextX**][**nextY**]** **=**std**::**numeric\_limits**<int>::**min**();** // lui

**}**

**}**

**}**

**bool** check**(int** x**,int** y**)**

**{**

**if(**x **<** 0 **||** x**>=** n **||** y **<** 0 **||** y**>=** n**)** **return** **false;**

**return** **true;**

**}**

**void** sol**()**

**{**

**int** m**,**n**;**

cin**>>**m**>>**n**;**

visited**[**m**][**n**]** **=** 0**;**

**if(**TRY**(**1**,**m**,**n**)==false)** cout**<<"Khong co loi giai";**

**else** print**();**

**}**

**};**

**int** main**()**

**{**

**int** n**;**

cin**>>**n**;**

KnightTour **\***T **=** **new** KnightTour**(**n**);**

T**->**sol**();**

**}**

\*) Hiển thị nhiều cách đi

#include<bits/stdc++.h>

**using** **namespace** std**;**

**class** KnightTour

**{**

**private:**

**int** cachDi**; // cac cach di**

**int** n**;**

**int** **\*\***visited**;**

**int** xMoves**[**8**]** **=** **{**2**,** 1**,** **-**1**,** **-**2**,** **-**2**,** **-**1**,** 1**,** 2**};**

**int** yMoves**[**8**]** **=** **{**1**,** 2**,** 2**,** 1**,** **-**1**,** **-**2**,** **-**2**,** **-**1**};**

**public:**

KnightTour**(int** n**)**

**{**

**this->**n **=** n**;**

init**();**

**}**

**void** init**()**

**{**

visited **=** **new** **int\*[**n**+**1**];**

**for(int** i**=**0**;**i**<**n**;**i**++)**

visited**[**i**]=new** **int[**n**+**1**];**

**for(int** i**=**0**;**i**<** n**;**i**++)**

**for(int** j**=**0**;**j**<**n**;**j**++)**

visited**[**i**][**j**]** **=** std**::**numeric\_limits**<int>::**min**();**

**}**

**void** print**()**

**{**

cout**<<"----";**

cout**<<"Cach di "<<**cachDi**++<<"------ "** **<<**endl**;**

**for(int** i**=**0**;**i**<** n**;**i**++){**

**for(int** j**=**0**;**j**<**n**;**j**++)**

**{**

cout**<<**visited**[**i**][**j**]<<" ";**

**}**

cout**<<**endl**;**

**}**

cout**<<**endl**;**

**}**

**bool** TRY**(int** step**,** **int** x**,** **int** y**)**

**{**

**if(**step **==** n **\*** n**)** // di duoc het ban` co

**{**

print**();**

**return** **true;**

**}**

**bool** res **=** **false;** // su dung bien res de hien thi them cac truong hop di duoc cua ma

**for(int** i**=**0**;** i **<** 8**;** i**++)**

**{**

**int** nextX **=** x **+** xMoves**[**i**];**

**int** nextY **=** y **+** yMoves**[**i**];**

**if(**check**(**nextX**,** nextY**)** **&&** visited**[**nextX**][**nextY**]** **==** std**::**numeric\_limits**<int>::**min**())** // ktra buoc di hop le va chua di

**{**

visited**[**nextX**][**nextY**]** **=** step**;** // vi tri da di qua

**if(**TRY**(**step **+** 1**,**nextX**,** nextY**)** **||** res**)** res **=** **true;** // tien' neu di duoc het ban co thi hien thi

**else** res **=** **false;** // k di duoc het ban co

visited**[**nextX**][**nextY**]** **=**std**::**numeric\_limits**<int>::**min**();** // lui

**}**

**}**

**return** res**;**

**}**

**bool** check**(int** x**,int** y**)**

**{**

**if(**x **<** 0 **||** x**>=** n **||** y **<** 0 **||** y**>=** n**)** **return** **false;**

**return** **true;**

**}**

**void** sol**()**

**{**

**int** m**,**n**;**

cin**>>**m**>>**n**;**

visited**[**m**][**n**]** **=** 0**;**

**if(**TRY**(**1**,**m**,**n**)==false)** cout**<<"Khong co loi giai";**

**}**

**};**

**int** main**()**

**{**

**int** n**;**

cin**>>**n**;**

KnightTour **\***T **=** **new** KnightTour**(**n**);**

T**->**sol**();**

**}**

3. Đánh giá độ phức tạp

- Độ phức tạp bài toán O(n^2)