Programming Assignment #2

0611031 資工 10 謝至恆

**目標:**

寫出一個類似踩地雷的程式，給定一個棋盤大小，地雷數，已點開hint的格子，求出一個可能地雷出現的表

使用backtracking Search(類似DFS)

用domain{0,1}代表有沒有地雷

給個給的hint可得1 constraint

Ex: 有1 hint =3 ，代表九宮格內除了那一格以外有的地雷總數=3

n(0,0)+n(1,0)+…..=3

解題思路:

使用stack(先進去的後出來 像是疊盤子) 存一個node

每個node內存list of variable + variable 的 值(0 or 1) 代表地雷

Node:

1. 存所有已assign過的紀錄
2. 存所有variable剩下的domain (沒有assign的)

(可用於forward checking)

Constraint:

1. 先計算variable的Lower bound 和 Upper bound
2. 若lower bound > hint , upper bound < hint 則失敗，需要backtrack
3. 若lower bound = hint , 則所有domain 須選最小值
4. 若upper bound = hint , 則所有domain 須選最大值
5. 有一constraint失敗就失敗

Heuristic:

1. MRV : 選domain小的variable 優先
2. Degree heuristic : 多少constraint 包含此variable (可以事先算)
3. LCV : 2個都能選的時候才能用，都試試看，觀察剩餘的domain狀況，選影響較小的優先

沒有做MRV 因為感覺每次都要全部檢查太浪費時間…

**實作:**

我的作法是這樣

Node裡面存 r\_domain

(剩下的domain 預設沒有hint的格子是2 有hint是9若給予放地雷則改為1反之0)

Boom\_board 原始給的棋盤

Choose\_board 跟演算法改變的棋盤

Num\_boom (總共地雷)

Now\_boom(現在有多少地雷)

Num\_constraint(總共有幾個hint)

Parent(上一個node)

Deep(tree的深度)

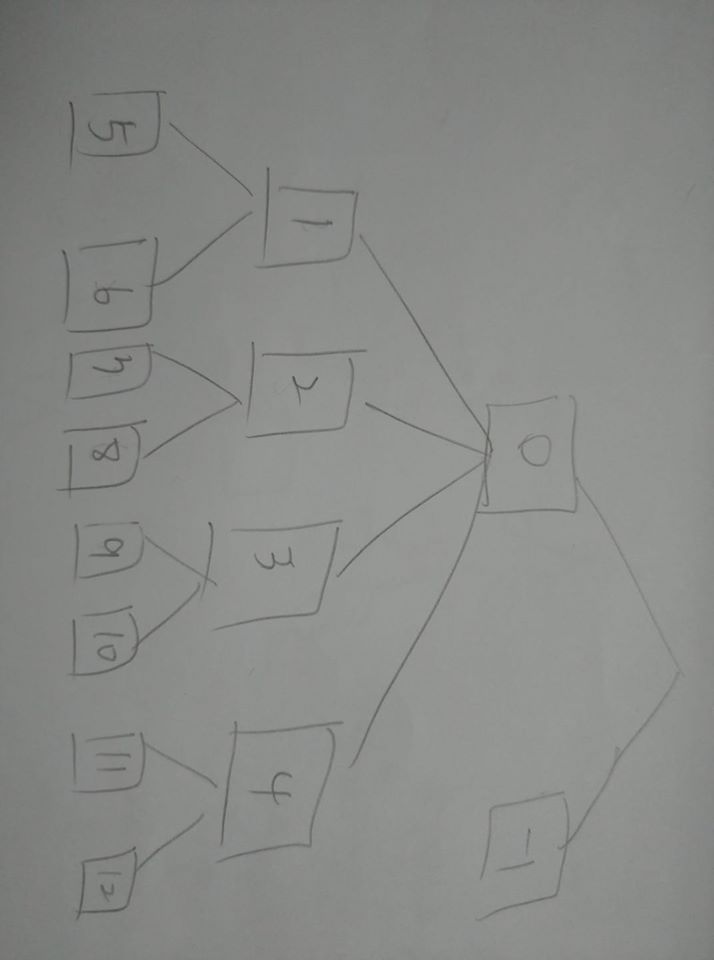
Right(是否為最右邊的node)

\*代表地雷

/代表不放地雷

\_代表尚未assign

使用Degree heuristic 來決定先試哪個測資 如果相同constraint數則一起加進去



往左都是有地雷往右是沒有地雷(從左邊數2個為一組) (2個2個為一組)

進stack的順序是4->3->2->1 (4我會標註是parent最右邊的node)

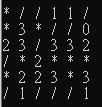
接下來會對1做處理產生5.6的node也是5優先做，假設一直做做到5做不了了就會pop掉到node 6 假設node 6 也做不下去 代表放滿版面但地雷還不夠10顆於是就pop掉他此時會往上一層跑，就要檢查是不是最右邊的node了，如果是就一直往上pop直到不是最右邊的node再pop一次，假設現在做node12做不了了，1234上面的node為0且0還有一個右邊的node為-1好了

12會一直pop到0 發現他不是最右邊的node了則再pop掉0進到-1繼續做，這樣下去照理來說應該可以把所有可能性做完。除非沒有解答，可是我照這樣寫出來程式就會一直無限迴圈一直跑下去，本來想要debug但是層數太多，從19層tracing到15層就看到頭昏眼花了，所以也沒有找到為甚麼會卡住的原因，加上時間管理沒做好，所以沒有辦法在時間內寫出這個程式。

最後只有第三組測資

看起來比較簡單的能跑出正確答案

第一組測資



最上面的1附近沒有炸彈

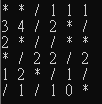
第二組測資



第三組測資

(把判斷是否所有條件都符合的條件拿掉)

(不然他會無限迴圈)



心得；

感覺這次作業好像蠻難的，不知道多花點時間可不可以做出來，適逢期中考周，花了1個周末沒有做出來最後只能放棄，先交尚未完成的檔案，跟我修課的同學也沒有做出來，但認識的人不多，不知道是不是大部分，但實際寫程式的時候其實覺得蠻好玩的，一直在想到底哪裡會有問題，哪裡不對，但可能還是不夠細心沒有注意到某些條件吧，每次覺得好像想通了寫出來卻又不對….，g甚至一度以為寫對了只是要跑比較久，但跑了很久後還是沒有跑出答案，還蠻崩潰的。

感覺我不太熟悉寫演算法類型的程式

程式碼(code)

#include <iostream>

#include <queue>

#include <math.h>

#include <algorithm>

#include <vector>

#include <stack>

using namespace std;

struct node {

int r\_domain[36]; //每個格子剩下的domain 如果有一格不是原來的0.1 則表示已assign

bool map[36]; //36格中已assign的值

int boom\_board[36];

int b\_degree[36];

int choose\_board[36];

int num\_boom;

int now\_boom;

int num\_constraint;

node \*parent;

int deep;

//node \*r\_kid;

int right;

};

void boomalogorithm(node \*start){

//進到函式

cout << endl << "boom!" << endl;

//建立stack

stack<node> s;

cout <<endl<< "stack top: "<<endl;

s.push(\*start);

char result[36];

int end\_condition=0;

while(end\_condition==0){

int correct=0;

node \*daddy = new node;

daddy->num\_boom=s.top().num\_boom;

daddy->now\_boom=s.top().now\_boom;

daddy->deep=s.top().deep;

daddy->right=s.top().right;

daddy->num\_constraint=s.top().num\_constraint;

for(int i=0;i<36;i++){

daddy->b\_degree[i]=s.top().b\_degree[i];

daddy->r\_domain[i]=s.top().r\_domain[i];

daddy->boom\_board[i]=s.top().boom\_board[i];

daddy->choose\_board[i]=s.top().choose\_board[i];

if(s.top().r\_domain[i]==9){

start->map[i]=true;

}

else{

start->map[i]=false;

}

}

int pop\_already=0;

cout << endl;

cout << "-----------------------------now board-----------------------------";

cout << endl << "deep : " << s.top().deep << endl;

cout <<endl << endl<< "domain :";

for(int i=0;i<36;i++){

if(i%6==0){

cout << endl;

}

cout <<s.top().r\_domain[i]<<" ";

}

cout << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(s.top().choose\_board[i]==100){

cout <<"\* ";

}

else if(s.top().choose\_board[i]==98){

cout <<"/ ";

}

else if(s.top().choose\_board[i]==99){

cout <<"\_ ";

}

else{

cout << s.top().choose\_board[i]<<" ";

}

}

cout << endl;

//check local constraint

for(int i=0; i<36 ; i++){

int check=0;

int find\_constraint=0;

//cout << s.top().boom\_board[i] << endl;

if(s.top().boom\_board[i]!=-1){

find\_constraint=1;

if(i%6!=0 && i-6>0){

if(s.top().boom\_board[i-7]==-1 && s.top().r\_domain[i-7]!=2)

check+=s.top().r\_domain[i-7];

}

if(i-6>0){

if(s.top().boom\_board[i-6]==-1 && s.top().r\_domain[i-6]!=2)

check+=s.top().r\_domain[i-6];

}

if(i%6!=5 && i-6>0){

if(s.top().boom\_board[i-5]==-1 && s.top().r\_domain[i-5]!=2)

check+=s.top().r\_domain[i-5];

}

if(i%6!=0){

if(s.top().boom\_board[i-1]==-1 && s.top().r\_domain[i-1]!=2)

check+=s.top().r\_domain[i-1];

}

if(i%6!=5){

if(s.top().boom\_board[i+1]==-1 && s.top().r\_domain[i+1]!=2)

check+=s.top().r\_domain[i+1];

}

if(i%6!=0 && i<30){

if(s.top().boom\_board[i+5]==-1 && s.top().r\_domain[i+5]!=2)

check+=s.top().r\_domain[i+5];

}

if(i<30){

if(s.top().boom\_board[i+6]==-1 && s.top().r\_domain[i+6]!=2)

check+=s.top().r\_domain[i+6];

}

if(i%6!=5 && i<30){

if(s.top().boom\_board[i+7]==-1 && s.top().r\_domain[i+7]!=2)

check+=s.top().r\_domain[i+7];

}

}

if(check==s.top().boom\_board[i]&&find\_constraint==1){

correct++;

}

if(check>s.top().boom\_board[i] && find\_constraint==1){

s.pop();

cout << endl << "-------------------pop--------------------" <<endl;

pop\_already=1;

break;

}

//cout << endl << "---------check-----------" << check <<endl;

}

//check global constraint

int zero=0;

if(correct==s.top().num\_constraint){

cout << "correct"<<endl;

end\_condition=1;

pop\_already=1;

break;

}

if(s.top().now\_boom==s.top().num\_boom){

//if(zero==36){

if(correct==s.top().num\_constraint){

cout << endl;

cout << s.top().now\_boom << endl;

cout << s.top().num\_boom <<endl;

cout << "booms ok" <<endl;

end\_condition=1;

break;

}

else{

s.pop();

pop\_already=1;

}

//}

/\*else{

s.pop();

cout << endl <<"not yet done pop" <<endl;

}\*/

}

cout << endl;

cout << endl;

cout <<endl<< "degree : ";

zero=0;

/\*for(int i=0;i<36;i++){

if(s.top().b\_degree[i]==0){

zero++;

}

if(i%6==0){

cout <<endl;

}

cout << s.top().b\_degree[i] << " ";

}

if(zero==36 && pop\_already==0){

while(s.top().right==1){

cout << endl << " right "<<endl;

s.pop();

}

s.pop();

pop\_already=1;

cout <<endl<< "--/////////////////////////////////////////////--------zero\_ pop----------/////////////////////////////////////////////" <<endl;

}\*/

cout << endl;

if(pop\_already==0){

//找出Degree heuristic 中包含最多constraint的

int max\_constraint=0;

int locate\_tail=0;

//清空locate

int locate[36];

for(int i=0;i<36;i++){

locate[i]=-1;

}

//找出最先出現的最大值

for(int i=0;i<36;i++){

if(max\_constraint<s.top().b\_degree[i]){

max\_constraint=s.top().b\_degree[i];

locate[0]=i;

}

}

//找出所有跟最大值一樣的地方

for(int i=1;i<36;i++){

int fin=0;

for(int j=locate[i-1]+1;j<36;j++){

if(max\_constraint==s.top().b\_degree[j]){

locate[i]=j;

fin=1;

break;

}

}

if(fin==0){

locate\_tail=i-1;

break;

}

}

cout << "tail :" << locate\_tail <<endl;

/\*

for(int i=0;i<36;i++){

cout << "locate : " << i << "-" << locate[i]<<endl;

}\*/

//沒有被pop過的話

int k=locate\_tail;

node \*next = new node;

//最後一個node

next->parent=daddy;

//next->parent->r\_kid=next;

next->right=1;

next->deep=daddy->deep+1;

next->num\_boom=daddy->num\_boom;

next->now\_boom=daddy->now\_boom;

next->num\_constraint=daddy->num\_constraint;

for(int i=0;i<36;i++){

next->b\_degree[i]=daddy->b\_degree[i];

next->r\_domain[i]=daddy->r\_domain[i];

next->boom\_board[i]=daddy->boom\_board[i];

next->choose\_board[i]=daddy->choose\_board[i];

next->map[i]=daddy->map[i];

}

next->b\_degree[locate[k]]=0;

next->r\_domain[locate[k]]=0;

next->map[locate[k]]=true;

next->choose\_board[locate[k]]=98;

s.push(\*next);

cout << endl << "the " << k << " node";

cout << endl << "deep : " << next->deep << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(next->choose\_board[i]==100){

cout <<"\* ";

}

else if(next->choose\_board[i]==98){

cout <<"/ ";

}

else if(next->choose\_board[i]==99){

cout <<"\_ ";

}

else{

cout << next->choose\_board[i]<<" ";

}

}

cout << endl;

//是炸彈

next->parent=daddy;

//next->parent->r\_kid=next;

next->right=0;

next->deep=daddy->deep+1;

next->num\_boom=daddy->num\_boom;

next->now\_boom=daddy->now\_boom+1;

next->num\_constraint=daddy->num\_constraint;

for(int i=0;i<36;i++){

next->b\_degree[i]=daddy->b\_degree[i];

next->r\_domain[i]=daddy->r\_domain[i];

next->boom\_board[i]=daddy->boom\_board[i];

next->choose\_board[i]=daddy->choose\_board[i];

next->map[i]=daddy->map[i];

}

next->b\_degree[locate[k]]=0;

next->r\_domain[locate[k]]=1;

next->map[locate[k]]=true;

next->choose\_board[locate[k]]=100;

s.push(\*next);

cout << endl << "the " << k << " node";

cout << endl << "deep : " << next->deep << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(next->choose\_board[i]==100){

cout <<"\* ";

}

else if(next->choose\_board[i]==99){

cout <<"\_ ";

}

else if(next->choose\_board[i]==98){

cout <<"/ ";

}

else{

cout << next->choose\_board[i]<<" ";

}

}

cout << endl;

k--;

while(k>=0){

//不是炸彈

next->parent=daddy;

next->deep=daddy->deep+1;

next->right=0;

next->num\_boom=daddy->num\_boom;

next->now\_boom=daddy->now\_boom;

next->num\_constraint=daddy->num\_constraint;

for(int i=0;i<36;i++){

next->b\_degree[i]=daddy->b\_degree[i];

next->r\_domain[i]=daddy->r\_domain[i];

next->boom\_board[i]=daddy->boom\_board[i];

next->choose\_board[i]=daddy->choose\_board[i];

next->map[i]=daddy->map[i];

}

next->b\_degree[locate[k]]=0;

next->r\_domain[locate[k]]=0;

next->map[locate[k]]=true;

next->choose\_board[locate[k]]=98;

s.push(\*next);

cout << endl << "the " << k << " node";

cout << endl << "deep : " << next->deep << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(next->choose\_board[i]==100){

cout <<"\* ";

}

else if(next->choose\_board[i]==98){

cout <<"/ ";

}

else if(next->choose\_board[i]==99){

cout <<"\_ ";

}

else{

cout << next->choose\_board[i]<<" ";

}

}

cout << endl;

//是炸彈

next->parent=daddy;

next->right=0;

next->deep=daddy->deep+1;

next->num\_boom=daddy->num\_boom;

next->now\_boom=daddy->now\_boom+1;

next->num\_constraint=daddy->num\_constraint;

for(int i=0;i<36;i++){

next->b\_degree[i]=daddy->b\_degree[i];

next->r\_domain[i]=daddy->r\_domain[i];

next->boom\_board[i]=daddy->boom\_board[i];

next->choose\_board[i]=daddy->choose\_board[i];

next->map[i]=daddy->map[i];

}

next->b\_degree[locate[k]]=0;

next->r\_domain[locate[k]]=1;

next->map[locate[k]]=true;

next->choose\_board[locate[k]]=100;

s.push(\*next);

cout << endl << "the " << k << " node";

cout << endl << "deep : " << next->deep << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(next->choose\_board[i]==100){

cout <<"\* ";

}

else if(next->choose\_board[i]==99){

cout <<"\_ ";

}

else if(next->choose\_board[i]==98){

cout <<"/ ";

}

else{

cout << next->choose\_board[i]<<" ";

}

}

cout << endl;

k--;

}

}

}

//while 結束

for(int i=0;i<36;i++){

if(s.top().r\_domain[i]==2){

s.top().choose\_board[i]=98;

}

}

cout <<endl << endl<< "domain :";

for(int i=0;i<36;i++){

if(i%6==0){

cout << endl;

}

cout <<s.top().r\_domain[i]<<" ";

}

cout << endl;

for(int i=0;i<36;i++){

if(i%6==0){

cout <<endl;

}

if(s.top().choose\_board[i]==100){

cout <<"\* ";

}

else if(s.top().choose\_board[i]==99){

cout <<"\_ ";

}

else if(s.top().choose\_board[i]==98){

cout <<"/ ";

}

else{

cout << s.top().choose\_board[i]<<" ";

}

}

cout << endl;

}

int main(){

int width;

int height;

int booms;

int board[36];

cin >> width >> height >> booms;

for(int i=0;i<width \* height;i++){

cin >> board[i];

}

for(int i=0;i<width \* height;i++){

if(i%6==0){

cout <<endl;

}

cout << board[i]<<" ";

}

cout << endl;

//全域限制

//區域限制

int degree[36];

int domain[36];

//初始化

for(int i=0;i<width \* height;i++){

degree[i]=0;

domain[i]=0;

}

//算出degree 和 初始化domain

for(int i=0;i<width\*height;i++){

if(board[i]!=-1){

domain[i]=9;

if(i%6!=0 && i-6>0){

if(board[i-7]==-1)

degree[i-7]+=1;

}

if(i-6>0){

if(board[i-6]==-1)

degree[i-6]+=1;

}

if(i%6!=5 && i-6>0){

if(board[i-5]==-1)

degree[i-5]+=1;

}

if(i%6!=0){

if(board[i-1]==-1)

degree[i-1]+=1;

}

if(i%6!=5){

if(board[i+1]==-1)

degree[i+1]+=1;

}

if(i%6!=0 && i<30){

if(board[i+5]==-1)

degree[i+5]+=1;

}

if(i<30){

if(board[i+6]==-1)

degree[i+6]+=1;

}

if(i%6!=5 && i<30){

if(board[i+7]==-1)

degree[i+7]+=1;

}

}

else{

domain[i]=2;

}

}

//

//印出degree

for(int i=0;i<width\*height;i++){

if(i%6==0){

cout <<endl;

}

cout << degree[i]<<" ";

}

cout << endl;

//初始化開始節點

node \*start=new node;

start->num\_boom=booms;

start->now\_boom=0;

start->right=0;

start->deep=0;

for(int i=0;i<36;i++){

start->b\_degree[i]=degree[i];

start->r\_domain[i]=domain[i];

start->boom\_board[i]=board[i];

if(board[i]==-1){

start->choose\_board[i]=99;

}

else{

start->choose\_board[i]=board[i];

}

if(domain[i]==9){

start->num\_constraint+=1;

start->map[i]=true;

}

else{

start->map[i]=false;

}

}

boomalogorithm(start);

return 0;

}