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)

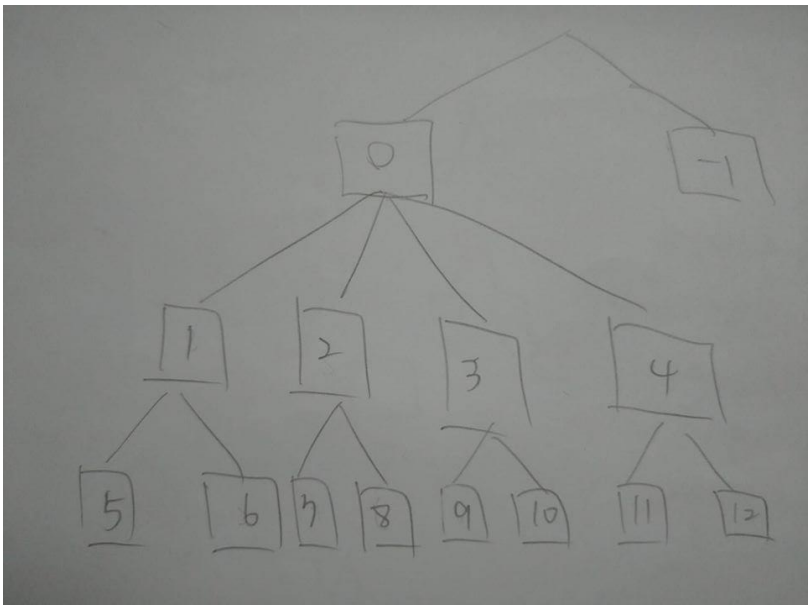
Right(是否為最右邊的 node)

*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;
}
```