AI prog 3

1.程式架構：

control module:
根據boardsizex和boardsizey建立一個二維array(arr)，並且填上炸彈，以x表示炸彈，接著用迴圈在所有非炸彈的點填上hint

player module:
(1)根據boardsizex和boardsizey建立一個二維array(player_arr)，將所有點填上-1（代表還沒填上答案）
(2)隨機獲取safe point，在這些點上填上0（一定沒炸彈），並將這些點依負號減一的模式存入KB（為了避免+0,-0的情況）
ex: 4 -> -5
(3)開始迴圈，只有兩種情況停止，一種是KB完全沒有東西了，另一種是KB無論如何都無法再生出新的clause了
(4)進入迴圈後，開始尋找長度為一的clause(single clause)
(5)如果成功找到single clause，依據正負號（正號代表有炸彈，負號代表沒炸彈），將player_arr中的那一點標示為有無炸彈，並將此clause移至KB0，接著判斷KB中各點是否有clause可以和此single clause合併，如果有的話就加入KB，另外，如果此點是無炸彈的話，我們可以繼續向control module(arr)獲取hint，藉由這個hint我們可以在和出新的clause來加入KB
(6)如果找不到single clause(KB此時不存在任合single clause)，則將所有KB互相配對，如果成功合成就繼續迴圈，失敗即跳出迴圈
(7)如果此時KB0包含所有點（KB空了），代表遊戲成功，找到所有答案了，如果KB0並未包含所有點（KB還有東西），代表遊戲失敗（stucked）

2.使用的函式：
(1)printarray
將array以二維方式印出
(2)minesweeperfinish
確認是否所有點都已被解決

(3)negative
將正數變負數
(4)comb
對兩數做combination，同時將回傳值type設為list
(5)deleteLoose(2d array, new list)
用迴圈檢查每個在2d array中的clause，如果clause當中有element和new list的element相同，就暫時拿掉new clause的此element（之後會恢復），如果new clause成為空list，代表new list is stricter than clause，因此就可以刪除這個clause
(6)isStricterThan(2d array, new list)
類似deleteLoose，但是這次是判斷new list有沒有比任何2d array中的clause還要loose，如果有return False(不需要插入此new list)，如果沒有return True
(7)countComplement(arrA, arrB)
檢查兩個clause之間complement的數量
(8)generateNewClause(arrA, arrB)
將兩個clause的complement消掉後，將兩個clause合併，但是忽略掉重複的element
(9)insertKB(KB, KB0, insertList)
這是用來插入KB的函式，可以分為兩個步驟，第一個步驟，對insertList和所有clause in KB0做合併（如果可以的話），完成後檢查此clause是否和KB中的clause相同或更loose(isStricterThan檢查)，如果都沒有，代表可以插入，然後把比這個clause更loose的clause刪掉(deleteLoose)
(10)GenerateClauseFromHint(KB0, KB, arr, player_arr, x, y, position, boardsizex, boardsizey)
這是在拿到hint後，要生成新的clause，此時找此點附近剩餘的所有炸彈，也就是player_arr[i][j] - 所有找到的炸彈，當作N，附近所有還沒被填上的點，當作M，如果N＝M，所有點都是炸彈，如果N＝0，所有點都不是炸彈，其餘的做combination(comb)

3.結果：

easy(9X9, 16mines)

```
KB: []
KB0: [[-74], [-9], [-65], [-48], [-75], [-79], [-33], [-1], [-24], [-21], [-29], [-37], [-60], [-71], [-54], [-17],
[-27], [-22], [-64], [-66], [-73], [-8], [-18], [-55], [-56], [-57], [-38], [-39], [-40], [-47], [-49], [-58], [-69
], [-70], [-78], [-80], [-2], [-10], [-11], [-14], [-15], [-16], [-23], [-25], [-32], [-34], [-7], [-26], [-46], [6
7], [-68], [-77], [61], [-72], [-81], [-3], [-12], [6], [35], [28], [-76], [59], [-4], [-13], [-5], [-36], [-42], [
-30], [-51], [-50], [31], [20], [-44], [-45], [41], [-19], [-43], [-52], [-53], [62], [-63]]
81
```

```
0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0

1 0 0 1 0 0 0 1 0

0 0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 1 0

0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0
```

```
0 0 0 0 1 x 1 0 0

1 1 1 0 1 1 1 0 0

2 x 2 1 1 0 1 1 1

x 2 2 x 2 1 1 x 1

1 1 1 2 x 1 1 1 1

0 0 0 2 2 3 2 2 1

0 0 1 2 x 2 x x 1

0 0 1 x 2 2 2 2 1

0 0 1 1 1 0 0 0 0
```

player_arr:
0->no bombs
1->bombs

ans(arr):
numbers->hint
x->bombs

medium(16X16, 25mines)

```
KB: []
KB0: [[-198], [-81], [-124], [-147], [-33], [-77], [-27], [-72], [-67], [-165], [-117], [-56], [-170], [-12], [-37]
, [-246], [-107], [-97], [-177], [-95], [-45], [-101], [-159], [-255], [-114], [-194], [-156], [-223], [-195], [-13
9], [-4], [-90], [-181], [-182], [-183], [-197], [-199], [-213], [-214], [-215], [-17], [-18], [-34], [-49], [-50],
[-10], [-11], [-26], [-28], [-42], [-43], [-44], [-148], [-149], [-150], [-164], [-166], [-180], [-100], [-102], [-
116], [-118], [-132], [-133], [-134], [-153], [-154], [-155], [-169], [-171], [-185], [-186], [-187], [-13], [-29],
[-91], [-92], [-106], [-108], [-122], [-123], [-84], [-85], [-86], [-98], [-99], [-113], [-115], [-129], [-130], [-
131], [-178], [-179], [-193], [-209], [-210], [-211], [-206], [-207], [-208], [-222], [-224], [-238], [-239], [-240
], [-196], [-212], [-3], [-5], [-19], [-20], [-21], [-73], [-74], [-75], [-89], [-105], [-167], [-184], [-200], [-2
16], [-230], [-231], [-232], [-1], [-2], [-35], [-51], [-65], [-66], [-9], [-25], [-41], [-58], [-59], [-60], [-61]
, [-163], [-135], [-151], [-83], [-137], [-138], [-172], [-188], [-201], [-202], [-203], [-204], [-14], [-30], [-46
], [-76], [-93], [-109], [-68], [-69], [-70], [-71], [-87], [-103], [82], [-145], [-146], [-225], [-226], [-227], [
-228], [-189], [-190], [-191], [-205], [-221], [-237], [-256], [-88], [-104], [121], [168], [152], [-217], [-233],
[229], [36], [-8], [-24], [-40], [57], [162], [119], [140], [-157], [-173], [-218], [-219], [-220], [-15], [-31], [
-47], [62], [125], [-52], [-53], [-54], [-55], [-241], [-242], [-243], [-244], [-174], [192], [-236], [253], [254],
[-120], [-136], [-234], [-248], [-249], [-250], [-245], [-7], [-23], [-39], [-161], [-141], [-142], [-158], [-235],
[-16], [-32], [-48], [-78], [-64], [-38], [175], [252], [247], [143], [251], [-63], [94], [-79], [-80], [22], [-176
], [-96], [-111], [-6], [-160], [-112], [144], [126], [-127], [128], [-110]]
256
```

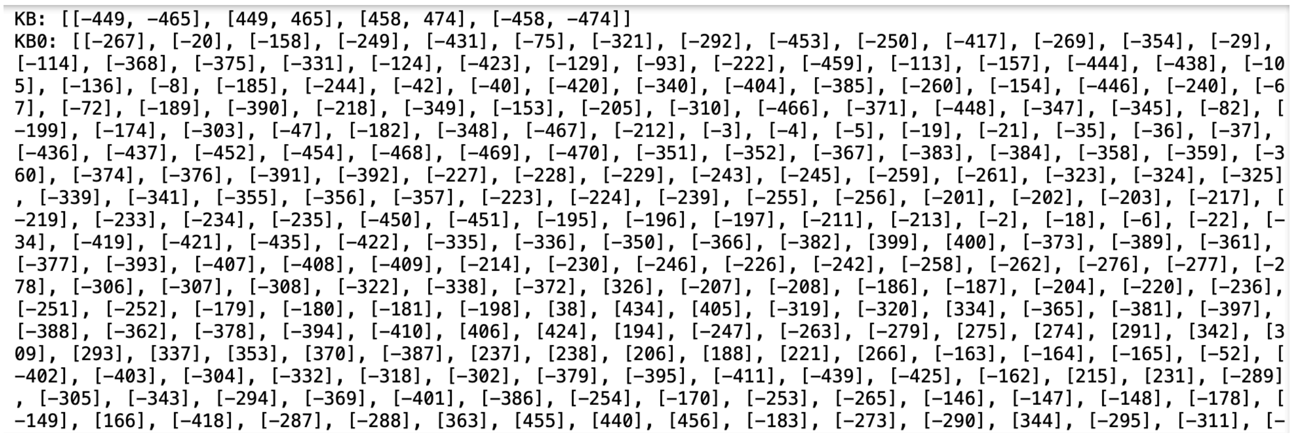
```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 1 0 0 0 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 1
0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 1 0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 1 0 0 0 1 1 1 1 0 0
```

```
0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0
0 0 1 1 2 x 1 0 0 0 0 0 0 0 0 0
0 0 1 x 2 1 1 1 1 1 0 0 1 1 1 0
0 0 1 1 1 0 0 1 x 1 0 0 1 x 1 0
1 1 1 0 0 0 0 1 1 1 0 0 2 2 2 0
1 x 1 0 0 0 0 0 0 0 0 0 1 x 1 0
1 1 1 0 0 1 1 2 1 1 0 1 3 3 3 1
0 0 0 0 0 1 x 2 x 1 1 2 x x 4 x
0 0 0 0 0 1 2 3 2 1 1 x 3 3 x x
1 1 1 0 0 0 2 x 2 0 1 1 1 2 3 3
1 x 1 0 0 0 2 x 2 0 0 0 0 1 x 2
1 1 1 0 0 0 1 1 1 0 0 0 0 1 2 x
0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1
0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0
0 0 0 1 x 2 1 1 0 1 2 3 3 2 1 0
0 0 0 1 1 2 x 1 0 1 x x x x 1 0
```

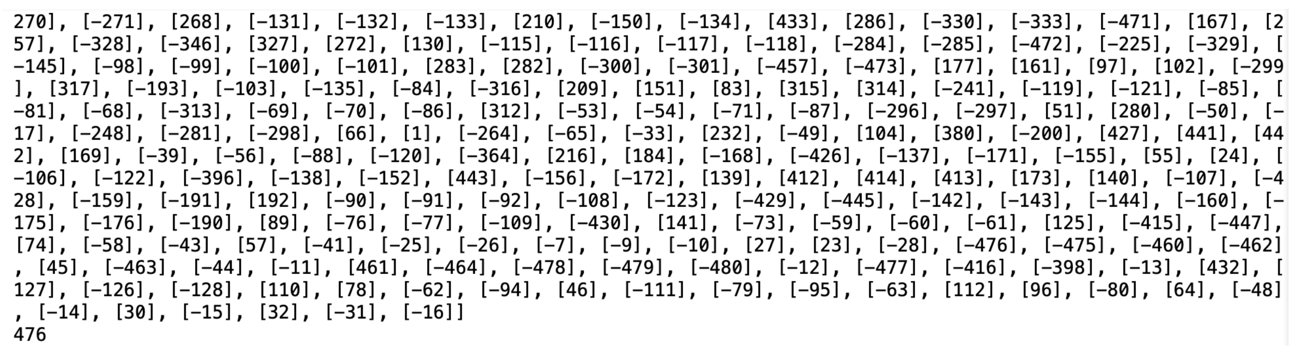
player_arr

ans(arr)

hard(30X16, 99mines)



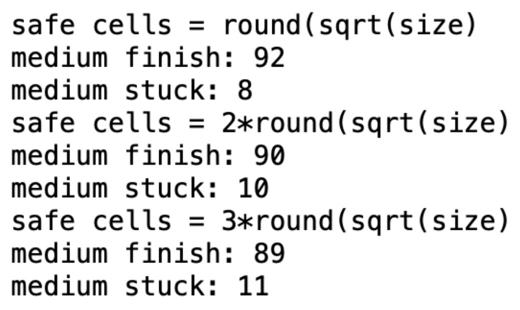
```
KB: [[-449, -465], [449, 465], [458, 474], [-458, -474]]
KB0: [[-267], [-20], [-158], [-249], [-431], [-75], [-321], [-292], [-453], [-250], [-417], [-269], [-354], [-29],
[-114], [-368], [-375], [-331], [-124], [-423], [-129], [-93], [-222], [-459], [-113], [-157], [-444], [-438], [-10
5], [-136], [-8], [-185], [-244], [-42], [-40], [-420], [-340], [-404], [-385], [-260], [-154], [-446], [-240], [-6
7], [-72], [-189], [-390], [-218], [-349], [-153], [-205], [-310], [-466], [-371], [-448], [-347], [-345], [-82], [
-199], [-174], [-303], [-47], [-182], [-348], [-467], [-212], [-3], [-4], [-5], [-19], [-21], [-35], [-36], [-37],
[-436], [-437], [-452], [-454], [-468], [-469], [-470], [-351], [-352], [-367], [-383], [-384], [-358], [-359], [-3
60], [-374], [-376], [-391], [-392], [-227], [-228], [-229], [-243], [-245], [-259], [-261], [-323], [-324], [-325]
, [-339], [-341], [-355], [-356], [-357], [-223], [-224], [-239], [-255], [-256], [-201], [-202], [-203], [-217], [
-219], [-233], [-234], [-235], [-450], [-451], [-195], [-196], [-197], [-211], [-213], [-2], [-18], [-6], [-22], [-
34], [-419], [-421], [-435], [-422], [-335], [-336], [-350], [-366], [-382], [399], [400], [-373], [-389], [-361],
[-377], [-393], [-407], [-408], [-409], [-214], [-230], [-246], [-226], [-242], [-258], [-262], [-276], [-277], [-2
78], [-306], [-307], [-308], [-322], [-338], [-372], [326], [-207], [-208], [-186], [-187], [-204], [-220], [-236],
[-251], [-252], [-179], [-180], [-181], [-198], [38], [434], [405], [-319], [-320], [334], [-365], [-381], [-397],
[-388], [-362], [-378], [-394], [-410], [406], [424], [194], [-247], [-263], [-279], [275], [274], [291], [342], [3
09], [293], [337], [353], [370], [-387], [237], [238], [206], [188], [221], [266], [-163], [-164], [-165], [-52], [
-402], [-403], [-304], [-332], [-318], [-302], [-379], [-395], [-411], [-439], [-425], [-162], [215], [231], [-289]
, [-305], [-343], [-294], [-369], [-401], [-386], [-254], [-170], [-253], [-265], [-146], [-147], [-148], [-178], [
-149], [166], [-418], [-287], [-288], [363], [455], [440], [456], [-183], [-273], [-290], [344], [-295], [-311], [-
```



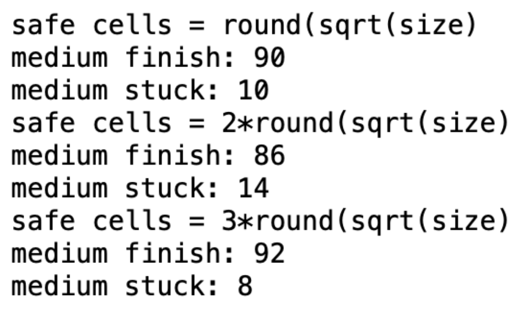
```
270], [-271], [268], [-131], [-132], [-133], [210], [-150], [-134], [433], [286], [-330], [-333], [-471], [167], [2
57], [-328], [-346], [327], [272], [130], [-115], [-116], [-117], [-118], [-284], [-285], [-472], [-225], [-329], [
-145], [-98], [-99], [-100], [-101], [283], [282], [-300], [-301], [-457], [-473], [177], [161], [97], [102], [-299
], [317], [-193], [-103], [-135], [-84], [-316], [209], [151], [83], [315], [314], [-241], [-119], [-121], [-85], [
-81], [-68], [-313], [-69], [-70], [-86], [312], [-53], [-54], [-71], [-87], [-296], [-297], [51], [280], [-50], [-
17], [-248], [-281], [-298], [66], [1], [-264], [-65], [-33], [232], [-49], [104], [380], [-200], [427], [441], [44
2], [169], [-39], [-56], [-88], [-120], [-364], [216], [184], [-168], [-426], [-137], [-171], [-155], [55], [24], [
-106], [-122], [-396], [-138], [-152], [443], [-156], [-172], [139], [412], [414], [413], [173], [140], [-107], [-4
28], [-159], [-191], [192], [-90], [-91], [-92], [-108], [-123], [-429], [-445], [-142], [-143], [-144], [-160], [-
175], [-176], [-190], [89], [-76], [-77], [-109], [-430], [141], [-73], [-59], [-60], [-61], [125], [-415], [-447],
[74], [-58], [-43], [57], [-41], [-25], [-26], [-7], [-9], [-10], [27], [23], [-28], [-476], [-475], [-460], [-462]
, [45], [-463], [-44], [-11], [461], [-464], [-478], [-479], [-480], [-12], [-477], [-416], [-398], [-13], [432], [
127], [-126], [-128], [110], [78], [-62], [-94], [46], [-111], [-79], [-95], [-63], [112], [96], [-80], [64], [-48]
, [-14], [30], [-15], [32], [-31], [-16]]
476
```

4.比較

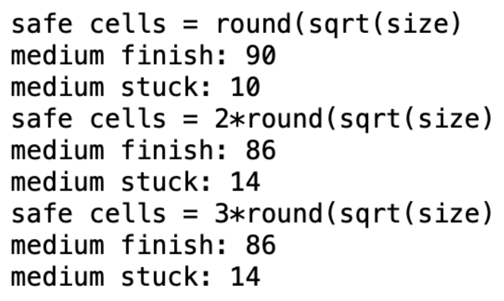
(1)不同大小的safe points 影響成功率的比較（執行100次，做四次測試，以medium難度）：



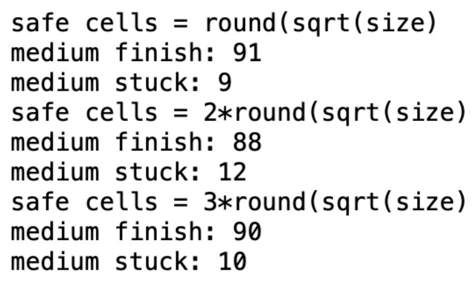
```
safe cells = round(sqrt(size)
medium finish: 92
medium stuck: 8
safe cells = 2*round(sqrt(size)
medium finish: 90
medium stuck: 10
safe cells = 3*round(sqrt(size)
medium finish: 89
medium stuck: 11
```



```
safe cells = round(sqrt(size)
medium finish: 90
medium stuck: 10
safe cells = 2*round(sqrt(size)
medium finish: 86
medium stuck: 14
safe cells = 3*round(sqrt(size)
medium finish: 92
medium stuck: 8
```



```
safe cells = round(sqrt(size)
medium finish: 90
medium stuck: 10
safe cells = 2*round(sqrt(size)
medium finish: 86
medium stuck: 14
safe cells = 3*round(sqrt(size)
medium finish: 86
medium stuck: 14
```



```
safe cells = round(sqrt(size)
medium finish: 91
medium stuck: 9
safe cells = 2*round(sqrt(size)
medium finish: 88
medium stuck: 12
safe cells = 3*round(sqrt(size)
medium finish: 90
medium stuck: 10
```

結論：safe points似乎不太會影響成功率？

(2)不同難度執行時間（各取50筆資料）
easy: 0.0515972375869751
medium: 1.3812112283706666
hard: 737.6631927490234

結論：隨著難度增加，時間也隨之暴增

code:

```
#!/usr/bin/env python
# coding: utf-8

# In[35]:


import numpy as np
import math
#boardsizex = 16
#boardsizey = 16
#mines = 25
safe_cells = round(math.sqrt(boardsizex*boardsizey))


# In[36]:


from random import sample
position_list = [i for i in range(0,boardsizex*boardsizey)]
mines_position = sample(position_list, mines)
arr = []
for i in range(0, boardsizex):
    col = []
    for j in range(0, boardsizey):
        col.append(0)
    arr.append(col)
for i in range(0, len(mines_position)):
    bomb = mines_position[i]
    m = (int)(bomb/boardsizey)
    n = bomb%boardsizey
    arr[m][n] = 'x'
for i in range(len(arr)):
    for j in range(len(arr[i])):
        if(arr[i][j] == 'x'):
            if(i-1 >= 0 and j-1 >= 0 and arr[i-1][j-1] != 'x'):
                arr[i-1][j-1]+=1
            if(i-1 >= 0 and arr[i-1][j] != 'x'):
                arr[i-1][j]+=1
            if(i-1 >= 0 and j+1 < boardsizey and arr[i-1][j+1] != 
'x'):
                arr[i-1][j+1]+=1
            if(j-1 >= 0 and arr[i][j-1] != 'x'):
                arr[i][j-1]+=1
            if(j+1 < boardsizey and arr[i][j+1] != 'x'):
                arr[i][j+1]+=1
            if(i+1 < boardsizex and j-1 >= 0 and arr[i+1][j-1] != 
'x'):
                arr[i+1][j-1]+=1
            if(i+1 < boardsizex and arr[i+1][j] != 'x'):
                arr[i+1][j]+=1
            if(i+1 < boardsizex and j+1 < boardsizey and arr[i+1]
[j+1] != 'x'):
                arr[i+1][j+1]+=1


# In[37]:


def printarray(array):
    for i in range(len(array)):
        for j in range(len(array[i])):
            print(array[i][j], end = ' ')
        print('\n')


# In[38]:


# printarray(arr)


# In[39]:


player_arr = []
for i in range(0, boardsizex):
    col = []
    for j in range(0, boardsizey):
        col.append(-1)
    player_arr.append(col)


# In[40]:


from itertools import combinations

def minesweeperFinish(array):
    flag = True
    for i in range(len(array)):
        for j in range(len(array[i])):
            if(array[i][j] == -1):
                flag = False
                break
        if(flag == False):
            break
    return flag

def negative(num):
    ans = num - 2*num
    return ans
```

```
def combi(arr, r):
    C = list(combinations(arr, r))
    for i in range(len(C)):
        C[i] = list(C[i])
    return C


# In[41]:


KB = []
safepoint_list = [i for i in range(0,boardsizex*boardsizey)]
temp_position = sample(position_list, safe_cells+mines)
safepoint_position = []
for i in range(len(temp_position)):
    position = temp_position[i]
    m = (int)(position/boardsizey)
    n = position%boardsizey
    if(arr[m][n] != 'x' and len(safepoint_position) < safe_cells):
        safepoint_position.append(position)
        player_arr[m][n] = 0
KB0 = []


# In[42]:


pre_list = []
for i in range(len(safepoint_position)):
    temp_list = []
    position = safepoint_position[i]
    temp_list.append(negative(position)-1)
    KB.append(temp_list)

#print(KB)
#print(KB0)


# In[43]:


def deleteLoose(twoDArr, li):
    delete_arr = []
    for i in range(len(twoDArr)):
        deleteItem = []
        arr = li.copy()
        for j in range(len(twoDArr[i])):
            for k in range(len(li)):
                if(twoDArr[i][j] == li[k] and (li[k] in
deleteItem) == False):
                    deleteItem.append(li[k])
        for t in range(len(deleteItem)):
                arr.remove(deleteItem[t])
        if(arr == []):
            delete_arr.append(i)
    for j in range(len(delete_arr)-1, -1, -1):
        del twoDArr[delete_arr[j]]

def isStricterThan(twoDArr, li):
    deleteItem = []
    for i in range(len(twoDArr)):
        arr = twoDArr[i].copy()
        for j in range(len(twoDArr[i])):
            for k in range(len(li)):
                if(twoDArr[i][j] == li[k] and (li[k] in arr) == 
True):
                    arr.remove(li[k])
        if(arr == []):
            return False
    return True

def countComplement(arrA, arrB):
    count = 0
    for i in range(len(arrA)):
        for j in range(len(arrB)):
            if(arrA[i]+arrB[j] == 0):
                count+=1
    return count

def GenerateNewClause(arrA, arrB):
    newArrA = arrA.copy()
    newArrB = arrB.copy()
    for i in range(len(arrA)):
        for j in range(len(arrB)):
            if(arrA[i]+arrB[j] == 0):
                del newArrA[i]
                del newArrB[j]

    nnewArrA = newArrA.copy()
    nnewArrB = newArrB.copy()
    for i in range(len(newArrA)):
        for j in range(len(newArrB)):
            if(newArrA[i] == newArrB[j]):
                nnewArrB.remove(newArrB[j])
    returnList = []
    returnList = nnewArrA + nnewArrB
    return returnList

def insertKB(KB, KB0, insertList):
    for i in range(len(KB0)):
        if(countComplement(KB0[i], insertList) == 1):
            insertList = GenerateNewClause(KB0[i], insertList)
    if((insertList in KB0) == False):
```

```python
        if((insertList in KB) == False and isStricterThan(KB, 
insertList) == True):
            deleteLoose(KB, insertList)
#             print(insertList)
            KB.append(insertList)

def GenerateClauseFromHint(KB0, KB, arr, player_arr, x, y, 
position, boardsizex, boardsizey):
        N = arr[x][y]
        M = 0
        t_list = []
        if(x-1 >= 0 and y-1 >= 0 and player_arr[x-1][y-1] == -1):
            M+=1
            t_list.append(position-boardsizey-1+1)
        if(x-1 >= 0 and player_arr[x-1][y] == -1):
            M+=1
            t_list.append(position-boardsizey+1)
        if(x-1 >= 0 and y+1 < boardsizey and player_arr[x-1][y+1] 
== -1):
            M+=1
            t_list.append(position-boardsizey+1+1)
        if(y-1 >= 0 and player_arr[x][y-1] == -1):
            M+=1
            t_list.append(position-1+1)
        if(y+1 < boardsizey and player_arr[x][y+1] == -1):
            M+=1
            t_list.append(position+1+1)
        if(x+1 < boardsizex and y-1 >= 0 and player_arr[x+1][y-1] 
== -1):
            M+=1
            t_list.append(position+boardsizey-1+1)
        if(x+1 < boardsizex and player_arr[x+1][y] == -1):
            M+=1
            t_list.append(position+boardsizey+1)
        if(x+1 < boardsizex and y+1 < boardsizey and 
player_arr[x+1][y+1] == -1):
            M+=1
            t_list.append(position+boardsizey+1+1)
        if(x-1 >= 0 and y-1 >= 0 and player_arr[x-1][y-1] == 1):
            N-=1
        if(x-1 >= 0 and player_arr[x-1][y] == 1):
            N-=1
        if(x-1 >= 0 and y+1 < boardsizey and player_arr[x-1][y+1] 
== 1):
            N-=1
        if(y-1 >= 0 and player_arr[x][y-1] == 1):
            N-=1
        if(y+1 < boardsizey and player_arr[x][y+1] == 1):
            N-=1
        if(x+1 < boardsizex and y-1 >= 0 and player_arr[x+1][y-1] 
== 1):
            N-=1
        if(x+1 < boardsizex and player_arr[x+1][y] == 1):
            N-=1
        if(x+1 < boardsizex and y+1 < boardsizey and 
player_arr[x+1][y+1] == 1):
            N-=1
        if(N == M):
            for j in range(len(t_list)):
                temp_list = []
                temp_list.append(t_list[j])
                insertKB(KB, KB0, temp_list)
        elif(N == 0):
            for j in range(len(t_list)):
                temp_list = []
                temp_list.append(negative(t_list[j]))
                insertKB(KB, KB0, temp_list)
        else:
            comb = combi(t_list, M-N+1)
            for j in range(len(comb)):
                temp_list = []
                for k in range(len(comb[j])):
                    temp_list.append(comb[j][k])
                insertKB(KB, KB0, temp_list)
            temp_list = []
            comb = combi(t_list, N+1)
            for j in range(len(comb)):
                temp_list = []
                for k in range(len(comb[j])):
                    temp_list.append(negative(comb[j][k]))
                insertKB(KB, KB0, temp_list)


# In[44]:


noInsert = False
while((len(KB) != 0 and noInsert == False)):
    i = 0
    while(len(KB[i]) != 1):
        if(i == len(KB)-1):
            break
        i+=1
    if(len(KB[i]) == 1):
        singleClause = KB[i]
        cell = KB[i][0]
        del KB[i]
        position = 0
        if(cell>0):
            position = cell - 1
            x = (int)(position/boardsizey)
            y = position%boardsizey
            player_arr[x][y] = 1
            KB0.append(singleClause)
            j = 0
            while(j < len(KB)):
                if(countComplement(singleClause, KB[j]) == 1):
                    newClause = GenerateNewClause(singleClause, 
KB[j])
                    insertKB(KB, KB0, newClause)
                j+=1
        else:
            position = cell - 2*cell -1
            x = (int)(position/boardsizey)
            y = position%boardsizey
            player_arr[x][y] = 0
            KB0.append(singleClause)
            j = 0
            while(j < len(KB)):
                if(countComplement(singleClause, KB[j]) == 1):
                    newClause = GenerateNewClause(singleClause, 
KB[j])
                    insertKB(KB, KB0, newClause)
                j+=1
            GenerateClauseFromHint(KB0, KB, arr, player_arr, x, y, 
position, boardsizex, boardsizey)
    else:
        noInsert = True
        insertList = []
        for j in range(0, len(KB)):
            for k in range(j+1, len(KB)):
                if(countComplement(KB[k], KB[j]) == 1):
                    newClause = GenerateNewClause(KB[k], KB[j])
                    if((newClause in KB) == False and 
isStricterThan(KB, newClause) == True):
                        insertList.append(newClause)

        if(len(insertList) != 0):
            noInsert = False
            for t in range(len(insertList)):
                insertKB(KB, KB0, insertList[t])


# In[45]:


#print("KB: ",end='')
#print(KB)
#print("KB0: ",end='')
#print(KB0)
#print(len(KB0))
#printarray(player_arr)
#printarray(arr)

#flag = True
```

```python
if(minesweeperFinish(player_arr) == True):
    print("game finished!")
    isFinish = True
else:
    print("game stucked!")
    isFinish = False


# In[ ]:
```