# 2022人工智慧project demo

Author:B0829060黃至祥

### project要求

**作業分組**

**修課共76人，每1~4人自由報隊。**

**3/25之前在 E-Learning 系統的課程討論區回覆組隊名單。**

**作業驗收**

**助教劉智傑** [**jack.eric95@gmail.com**](mailto:jack.eric95@gmail.com)

**期中驗收時間：5/10 (課堂時間)**

**期末報告上傳：6/13 (早於期末報告)**

**期末驗收時間：6/14 (或另由助教公告)**

**期中驗收部份**

**第1題為基本題(30%), 期中demo從簡，現場展示即可**

**第1題期中demo時，須以debug模式或另存log檔的形式展示 OPEN-lise/ Closed-list之內容變化，以證實解題進行中**

**文字說明寫入期末報告，於期末繳交**

**期末驗收部份**

**第2~4題為進階題(30%,20%,10%), 期末報告為主, 期末demo為輔**

**第2~4題期末demo時，助教以書面報告內容質詢應答，並查驗程式內容與執行結果。**

**期末報告內容**

**組員分工與貢獻百分比**

**各題結果與討論，佐以圖表說明**

**每組由組長代表上傳即可**

**其他10%**

**介面設計有效、美觀**

**執行過程記錄週詳、確實**

**團隊合作表現好、組長盡心負責**

### 組員分工

為個人一組，全部獨力完成

貢獻百分比B0829060黃至祥100%

組長 黃至祥

# 第1題 (30%, demo@5/10)

1. **有食人族3名、傳教士3名、小船2艘皆處於河的右岸**
2. **在河兩岸以及船上，傳教士人數若少於食人族，就會被殺害**
3. **傳教士人數為 0, 或者大於等於食人族人數**
4. **boat\_A: 每次搭載1~2人，單趟投幣3元**
5. **Boat\_B: 每次搭載1~3人，單趟投幣25元**
6. **必須至少一個人操作船隻，可以是傳教士或食人族**
7. **假設二艘船速度相等，如果起動則同時離開某岸又同時停靠對岸**
8. **Action: boat\_A(停泊/左向右/右向左/人數組合) x boat\_B()**

**假設渡河除了航行時間之外，無其他工作佔用時間**

1. **試回答以最少費用渡河的方法，其花費多少？ (total cost=?)**
2. **試回答以最少時間渡河的方法，共需時幾趟？ (#action=?)**
3. **除了Uniform Cost Search 之外，可否進行A\* Search? Heuristic 如何設計？**

回答：

我先展示Uniform Cost Search(dfs)與A\*的function code設計

**dfs:**

def dfs(s,good,bad):

    s = tuple(s)     #字典的值是列表里面包含元组，原因是字典的键值不能是列表

    if s in route:   #已经在路徑里面

        route.append(s)

        return

    # 到達終點，紀錄路径

    if s == (0, 0, good, bad, -1):

        route.append(s)

        way.append(route[:])

        return

    route.append(s)

    for i in range(len(g[s])):

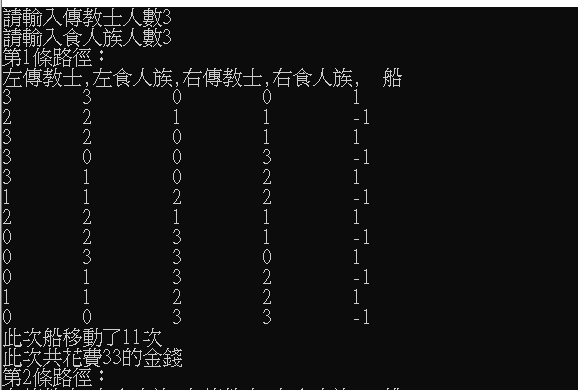
        dfs(g[s][i],good,bad)   #搜索直到到達終點或者重複

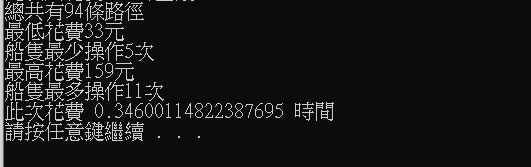
        route.pop()

**-執行結果**

**-條列過程**

**-總計**





A\*:

#判斷是否在closed表中

def isclosed(node):

    global closed

    flag=0

    for n in closed:

        if operator.eq(n,node):

            flag=1

            break

    if flag==1:

        return True

    else:

        return False

# In[6]:

#判斷是否在opened表中

def isopen(node):

    global opened

    flag=0

    for n in opened:

        if operator.eq(n,node):

            flag=1

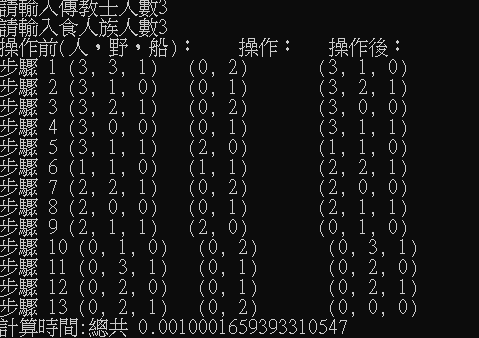
            break

    if flag==1:

        return True

    else:

        return False



答:最少花費33元

最少5趟

可以

設計

#合法的且不在closed表中的子節點添加到opened表中

#使用的insert方法在opened表的頭存入節點

#使得opened列表=stack的功能深度優先搜索

def Toopen(tnode):

    global opened

    for m in tnode:

        if islegal(m) and isclosed(m)==False:

            opened.insert(0,m)

#判斷closed表中的節點的子節點是否在opened表中或在closed表中

#即該節點已被訪問，不是目標狀態，且不存在子節點或子節點下不存在有效路徑

#將從closed表中删除

#因為最後closed表中存放的是一條有效路徑，所以需要將無效節點删除

def cino():

    global closed

    for k in closed:

        flag1=0

        for x in Expand(k):

            if isopen(x) or isclosed(x):

                flag1=1

        if flag1==0 :

            closed.remove(k)

open closed變化：

open新增了 (3, 2, 0)

open新增了 (2, 2, 0)

open新增了 (3, 1, 0)

closed新增了 (0, 1)

closed新增了 (1, 1)

closed新增了 (0, 2)

open新增了 (3, 2, 1)

closed新增了 (0, 1)

open新增了 (2, 2, 0)

open新增了 (2, 1, 0)

open新增了 (3, 0, 0)

closed新增了 (1, 0)

closed新增了 (1, 1)

closed新增了 (0, 2)

open新增了 (3, 1, 1)

closed新增了 (0, 1)

open新增了 (2, 1, 0)

open新增了 (2, 0, 0)

open新增了 (1, 1, 0)

closed新增了 (1, 0)

closed新增了 (1, 1)

closed新增了 (2, 0)

open新增了 (2, 1, 1)

open新增了 (2, 2, 1)

closed新增了 (1, 0)

closed新增了 (1, 1)

open新增了 (2, 1, 0)

open新增了 (0, 2, 0)

open新增了 (2, 0, 0)

closed新增了 (0, 1)

closed新增了 (2, 0)

closed新增了 (0, 2)

open新增了 (3, 0, 1)

open新增了 (2, 1, 1)

closed新增了 (1, 0)

closed新增了 (0, 1)

open新增了 (1, 0, 0)

open新增了 (0, 1, 0)

closed新增了 (1, 1)

closed新增了 (2, 0)

open新增了 (1, 1, 1)

open新增了 (0, 2, 1)

open新增了 (0, 3, 1)

closed新增了 (1, 0)

closed新增了 (0, 1)

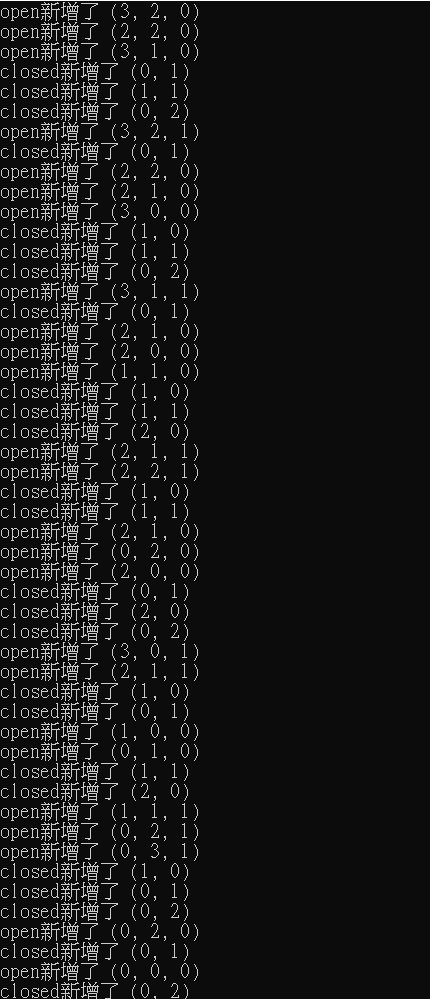
closed新增了 (0, 2)

open新增了 (0, 2, 0)

closed新增了 (0, 1)

open新增了 (0, 0, 0)

closed新增了 (0, 2)



### 第2題 (30%)

**假設有食人族 N 名、傳教士 N+M 名，其餘條件不變**

**N= 3,4..,10; M=0,1,2**

**以表一列出各狀況之下渡河所需最少船資**

**以表二比較上述各狀況下 Uniform Cost Search 與 A\* Search**

**所需之計算時間與所需展開節點數(#OPEN + #CLOSED)**

**以表三列出各狀況之下渡河所需最少時間**

**以表四比較上述各狀況下 Uniform Cost Search 與 A\* Search**

**所需之計算時間與所需展開節點數(#OPEN + #CLOSED)**

#99999表示沒有找到答案

空白為計算時間過於長久難以計算，不予填上  
可能是演算法的效率並不高而導致，或是我電腦的運算速度不夠好

表一(最少船資)：

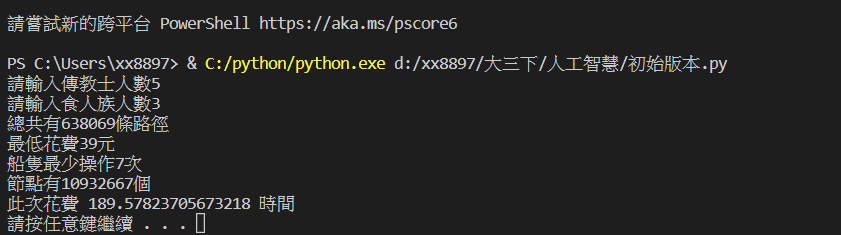
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m\n | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 33 | 61 | 89 | 99999 | 99999 | 99999 | 99999 | 99999 |
| 1 | 33 | 45 | 57 | 69 | 81 | 93 |  | 117 |
| 2 | 39 |  |  |  |  |  |  |  |

表二(計算時間)(dfs)：四捨五入取到小數第二位

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m\n | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 0.29 | 0.77 | 1.43 | 0.01 | 0.01 | 0.02 | 0.03 | 0.07 |
| 1 | 0.60 | 10.56 | 45.82 | 257.64 | 889.72 | 3122.88 | 10632.76 | 38293.65 |
| 2 | 253.51 |  |  |  |  |  |  |  |

表二(節點數)(dfs)：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m\n | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 734 | 2280 | 5035 | x | x | x | x | x |
| 1 | 28127 | 548488 | 2835560 |  |  |  |  |  |
| 2 | 10932667 |  |  |  |  |  |  |  |



表三(最少時間(操作次數))(dfs)：

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| m\n | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 0 | 5 | 9 | 11 | x | x | x | x | x |
| 1 | 5 | 7 | 9 | 11 | 13 | 15 |  | 19 |
| 2 | 7 |  |  |  |  |  |  |  |

在做的過程中疑惑了一下

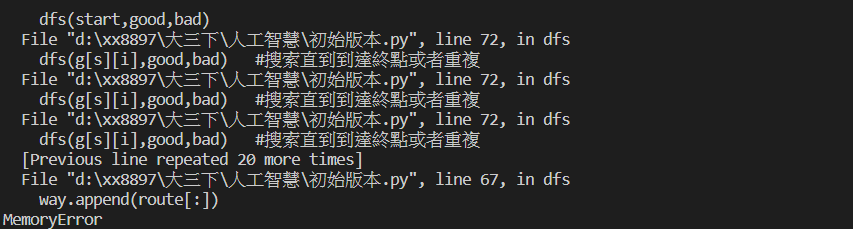
覺得我寫的程式沒有問題

後來爬文後發現

|  |
| --- |
| 如果船可以同時乘坐 2 位乘客時，傳教士和惡魔都為 2 時，也有 4 種不同解法。但傳教士和惡魔數都為 4 時沒有解。 如果船可以同時乘坐 3 位乘客時，傳教士和惡魔數都為 2，3，4，5時都有解法，其唯一解數分別為 12，54，72，216 種。 如果船可以同時乘坐 4 位乘客時，傳教士和惡魔數為任意相同的數都有解法。比如 5 和 6 時，其唯一解數分別為 1200 和 3600 種。  --維基百科 |

故有三人船且人數相等且大於5的情況無解

後來出現了記憶體不足的情況，故放棄繼續運算(我記憶體好歹加起來也有24GB誒…)



### 第3題 (20%)

1. **假設限制船資花費不大於P，如何找到最快速過河的方法？並舉一實例試作**

設定limitprice=p

得出每一個price時比對是否小於p

若小於，則:

更新limitmintime

更新limitmaxtime

更新limitminprice

更新limitmaxprice

else:

甚麼都不做

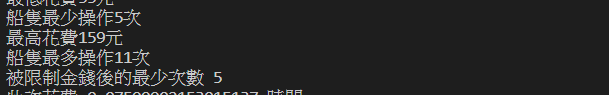
接著透過mintime尋找方法即可

demo展示

一般情況



得出一般結果



得知最少五次

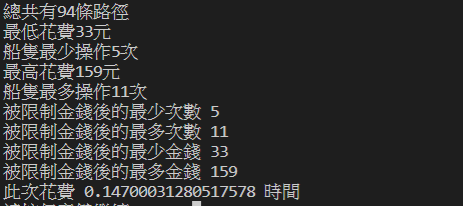
看看五次時最少花費



接著可以改成限制成限制50元

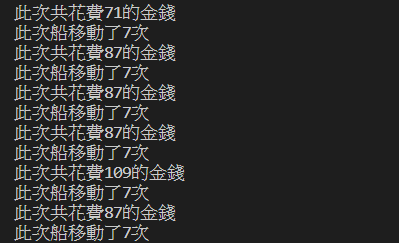


得出結果



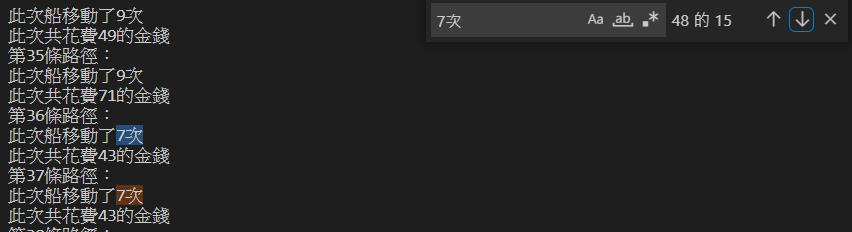
得出被限制金錢後的最少次數(只看次數),輸出太多東西有時候容易重點失焦

再從結果中尋找最少次數7次即可

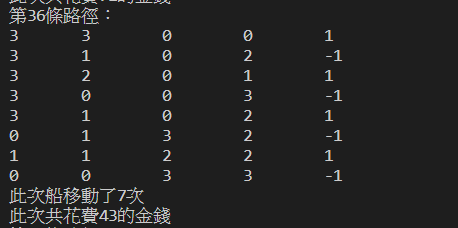


把第幾次的輸出打開

且尋找移動7次的路徑



再把過程的輸出打開



若要找到所有符合條件的路徑則搜尋即可

**2.假設限制過河時間不大於Q，如何找到最經濟過河的方法？並舉一實例試作**

設定limittime=q

得出每一個time時比對是否小於q

若小於，則:

更新limitmintime

更新limitmaxtime

更新limitminprice

更新limitmaxprice

else:

甚麼都不做

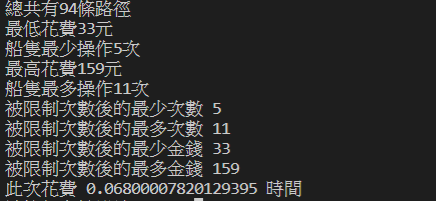
接著透過minprice尋找方法即可

demo展示

先不設限制



得出結果

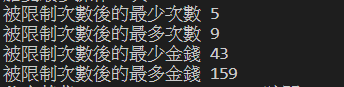


沒限制下最經濟33元 11次

限制成10次



得出限制後的結果

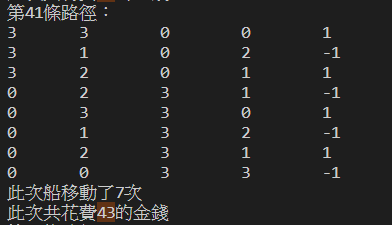


最少金錢為43(只看金錢)

尋找一下是第幾條路徑



開啟路徑的輸出 尋找第41條



尋找到結果

若要找到所有符合條件的路徑則搜尋即可

### 第4題 (10%)

**你覺得可以如何修改這次的作業題目，讓它變得更有趣或更有意義？請舉實作結果加以說明。**

**再加上**

**可針對題目中的各項設定加以變化，使修改後的題目可以觀察到你想要的特定結果**

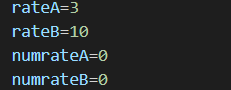
**例如船資與數量、人數與種類、船的速度或其他…**

船資減少，船有速率

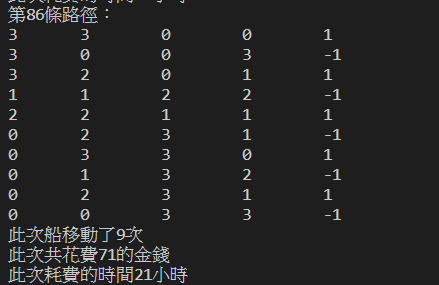
我將三人船的船資改為15

新增耗時

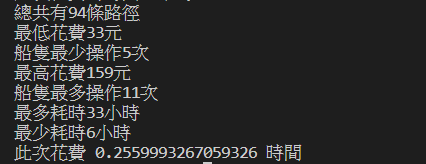
1人船和3人船的耗時



可以得出每一次的耗時



得出最久及最短的耗時



此做法也能與限制船資或次數相結合