探索遊戲的人工智慧：
從tic tac toe到AlphaGo

研究者:吳宇倫
指導老師:鄭綺瑩老師

緒論

文獻探討

研究方法及步驟

製作歷程

製作結果

心得與建議

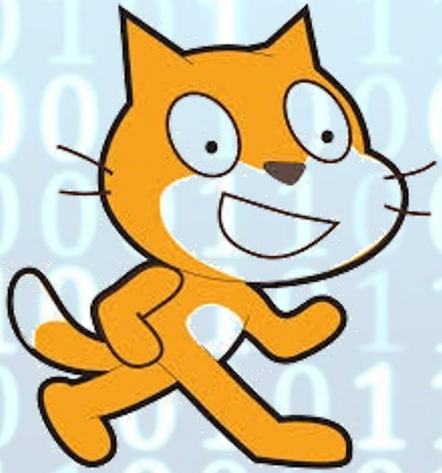
參考資料

壹、緒論

一、研究動機

圍棋規則簡單、組合複雜，AI如何戰勝世界棋王的呢?

我決定利用我學過的Scratch和Python做出打不贏的AI。



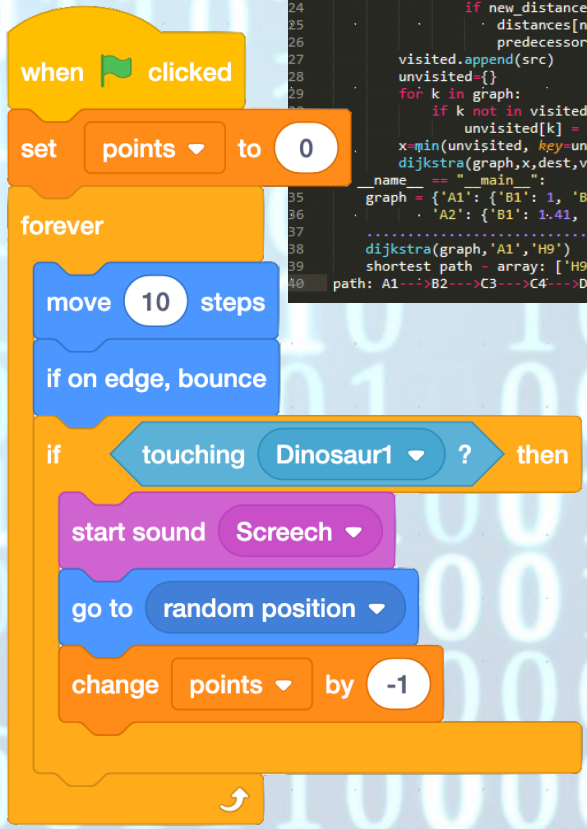
二、研究目的

1.了解遊戲的AI的邏輯和如何製作。

2.用Python做出可以在電腦上玩的兩人遊戲

3.用Scratch做出可以在電腦上玩的兩人遊戲

```python
def dijkstra(graph,src,dest,visited=[],distances={},predecessors={}):
    """ calculates a shortest path tree routed in src
    """
    if src not in graph:
        raise TypeError('The root of the shortest path tree cannot be found')
    if dest not in graph:
        raise TypeError('The target of the shortest path cannot be found')
    if src == dest:
        path=[]
        pred=dest
        while pred != None:
            path.append(pred)
            pred=predecessors.get(pred,None)
            readable=path[0]
        for index in range(1,len(path)): readable = path[index]+'--->'+readable
        print('shortest path - array: '+str(path))
        print("path: "+readable+",   cost="+str(distances[dest]))
    else :
        if not visited:
            distances[src]=0
        for neighbor in graph[src] :
            if neighbor not in visited:
                new_distance = distances[src] + graph[src][neighbor]
                if new_distance < distances.get(neighbor,float('inf')):
                    distances[neighbor] = new_distance
                    predecessors[neighbor] = src
        visited.append(src)
        unvisited={}
        for k in graph:
            if k not in visited:
                unvisited[k] = distances.get(k,float('inf'))
        x=min(unvisited, key=unvisited.get)
        dijkstra(graph,x,dest,visited,distances,predecessors)
__name__ == "__main__":
    graph = {'A1': {'B1': 1, 'B2': 1.41, 'A2': 1},
             'A2': {'B1': 1.41, 'B2': 1, 'B3': 1.41, 'A1': 1, 'A3': 1},
    ..........................................................
    dijkstra(graph,'A1','H9')
    shortest path - array: ['H9', 'G8', 'F8', 'E7', 'D6', 'D5', 'C4', 'C3', 'B2', 'A1']
path: A1--->B2--->C3--->C4--->D5--->D6--->E7--->F8--->G8--->H9,   cost=11.459999999999999
```

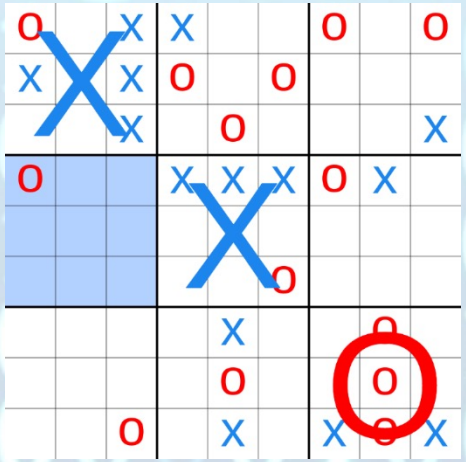


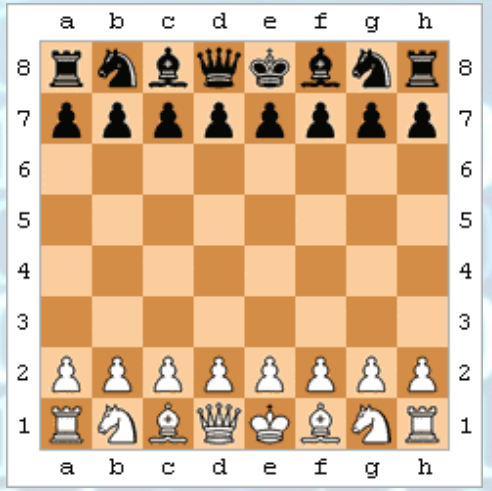
二、研究目的

4.用Python做出會玩桌遊的AI

(1) 用Python做出做出會玩tic tac toe的AI

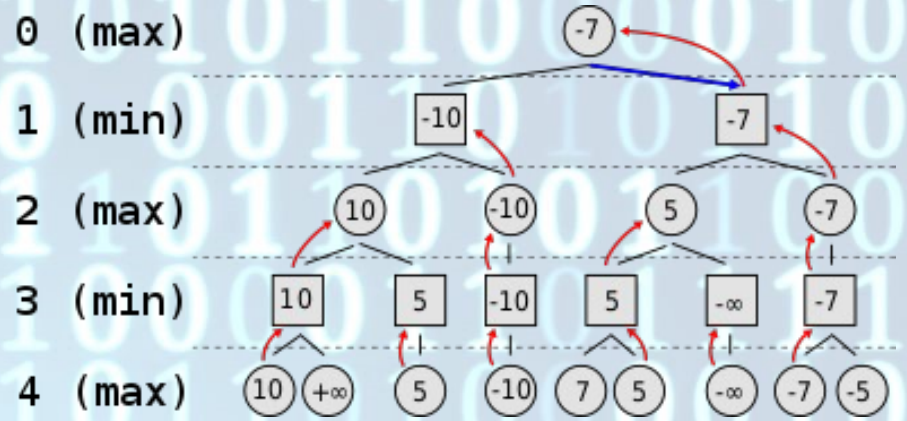
(2) 進階挑戰：用Python做出會玩西洋棋的AI





貳、文獻探討

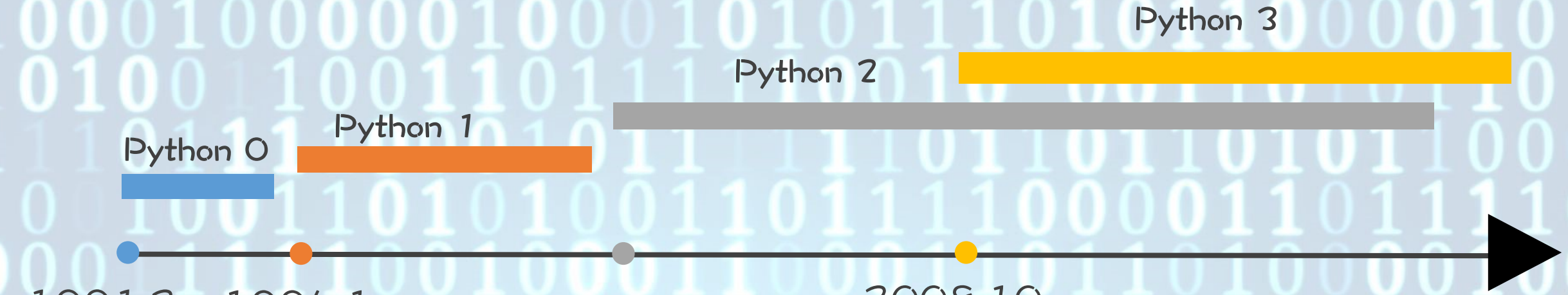
一、Minimax演算法介紹



- 用於兩人互相較量的遊戲中
- 假設對方會選最好的一步，來告訴AI要下哪裡。
- 利用recursion(遞迴)做出來的，所以跑很慢。
- alpha-beta-pruning，加快速度。

二、Python

- Python的創始人是Guido Van Rossum
- 用C語言寫出來的
- 一種高階語言
- 優點-便於閱讀，簡潔有力
- 缺點-跑得比較慢



1991.2
Python 0.9

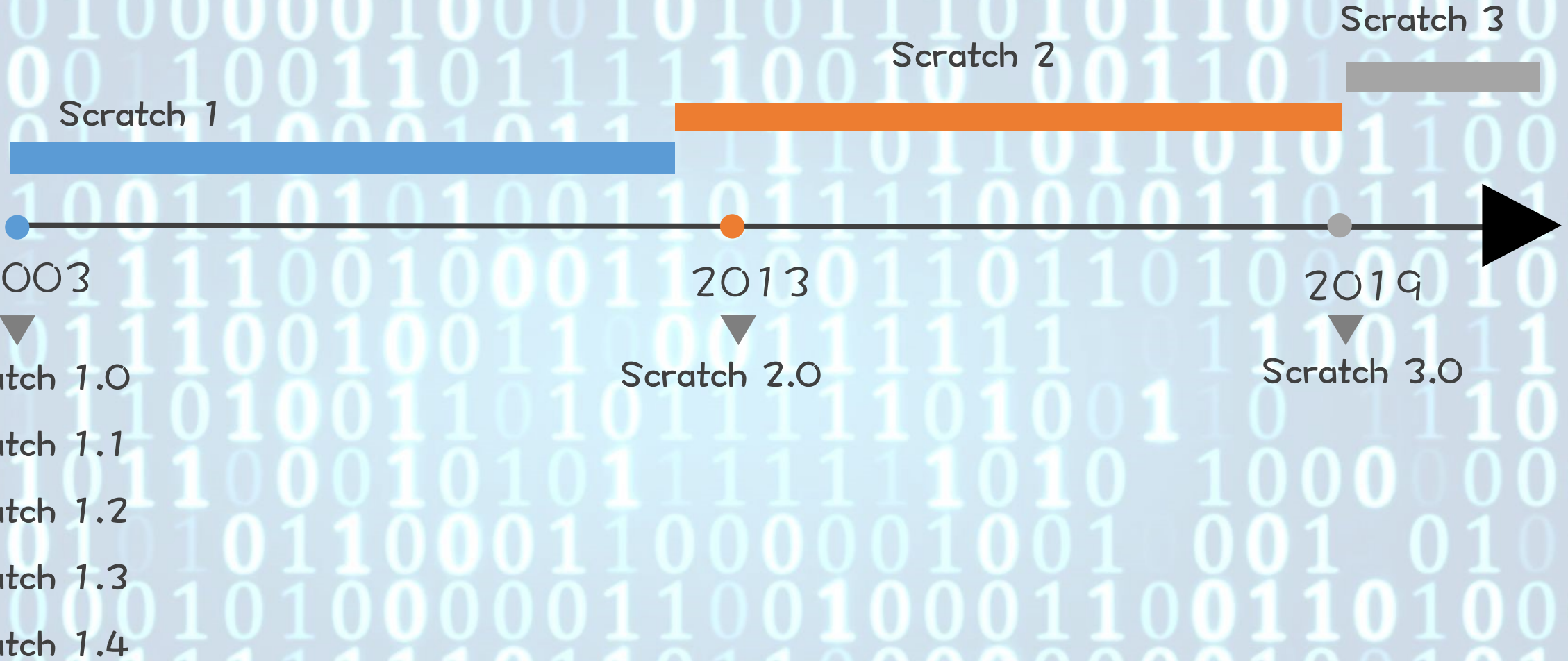
1994.1
Python 1.0
Python 1.1
Python 1.2
Python 1.3
Python 1.4
Python 1.5
Python 1.6

2000.10
Python 2.0
Python 2.1
Python 2.2
Python 2.3
Python 2.4
Python 2.5
Python 2.6
Python 2.7

2008.10
Python 3.0
Python 3.1
Python 3.2
Python 3.3
Python 3.4
Python 3.5
Python 3.6
Python 3.7
Python 3.8
Python 3.9
Python 3.10

三、Scratch

- Scratch的創始公司是MIT
- 用Javascript寫出來的
- 用拖拉積木的方式寫程式
- 優點-適合沒學過寫程式的人，容易看懂、好學
- 限制-有一些事情不能做，也不像Python會告訴你哪裡有問題。


Scratch 3
Scratch 2
Scratch 1
2003
2013
2019
Scratch 1.0
Scratch 1.1
Scratch 1.2
Scratch 1.3
Scratch 1.4
Scratch 2.0
Scratch 3.0

參、研究方法及步驟

一、研究方法

(一)Python篇:

利用PyGame做出遊戲畫面，利用PyCharm和Replit寫程式並測試AI，並利用minimax演算法。

一、研究方法

(二)Scratch篇:

利用Scratch做出遊戲畫面、寫程式

二、研究步驟

肆、製作歷程

- 問題
- 自己解決->尋求幫助
- 放棄Scratch AI->製作遊戲畫面

伍、製作結果

連結

- https://replit.com/@JamesWu13/Tic-Tac-Toe-AI-alpha-beta#main.py
- https://replit.com/@JamesWu13/Ultimate-Tic-Tac-Toe#main.py
- https://replit.com/@JamesWu13/Chess#Chess/ChessMain.py
- https://scratch.mit.edu/projects/562342230/
- https://scratch.mit.edu/projects/564905886/
- https://scratch.mit.edu/projects/659252993/

陸、心得與建議

- 挫折->過程最重要
- 思考、解決->進步
- 成就感
- 依照興趣選題目
- Pygame、Replit、PyCharm
- 未來繼續製作

柒、參考資料

- Ultimate tic-tac-toe:https://reurl.cc/41b5lj
- Pygame Tutorial for Beginners - Python Game Development Course: https://reurl.cc/p1Kn8x
- Tic-tac-toe-minimax:https://reurl.cc/M0M1Dv
- Minimax Algorithm in Game Theory | Set 3 (Tic-Tac-Toe AI – Finding optimal move):https://reurl.cc/KbOKDn
- Minimax Algorithm in Game Theory | Set 4 (Alpha-Beta Pruning):https://reurl.cc/ZAlazl
- History of Python:https://reurl.cc/Wr8aMD
- Scratch (programming language):https://reurl.cc/3obr9j
- Creating a Chess Engine in Python:https://reurl.cc/XjaDge

非常感謝