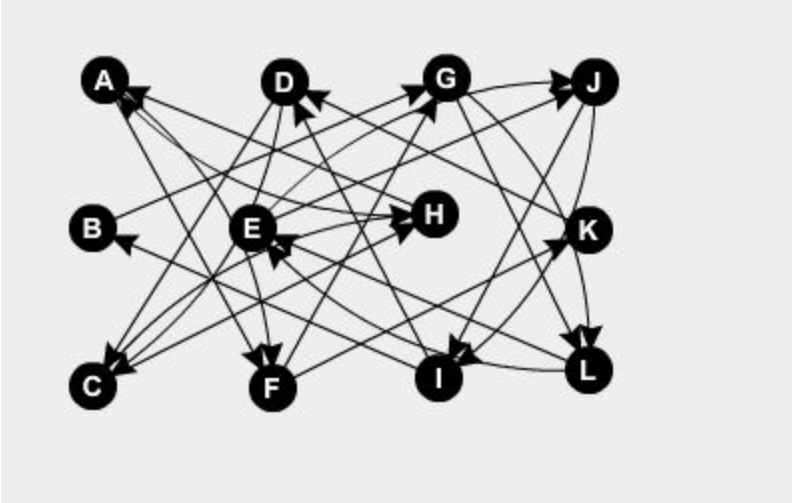
Carson Sklare

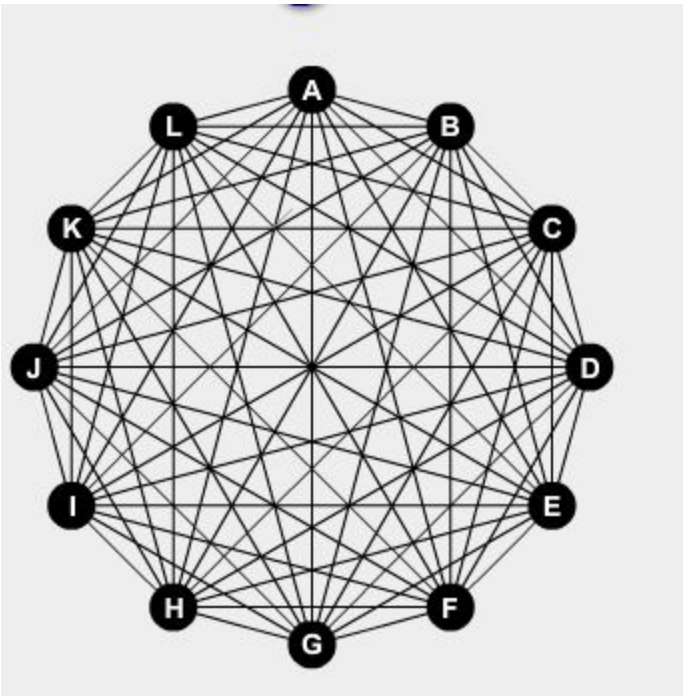
2a.
Board:

A	D	G	J
B	E	H	K
C	F	I	L

Graph directed:



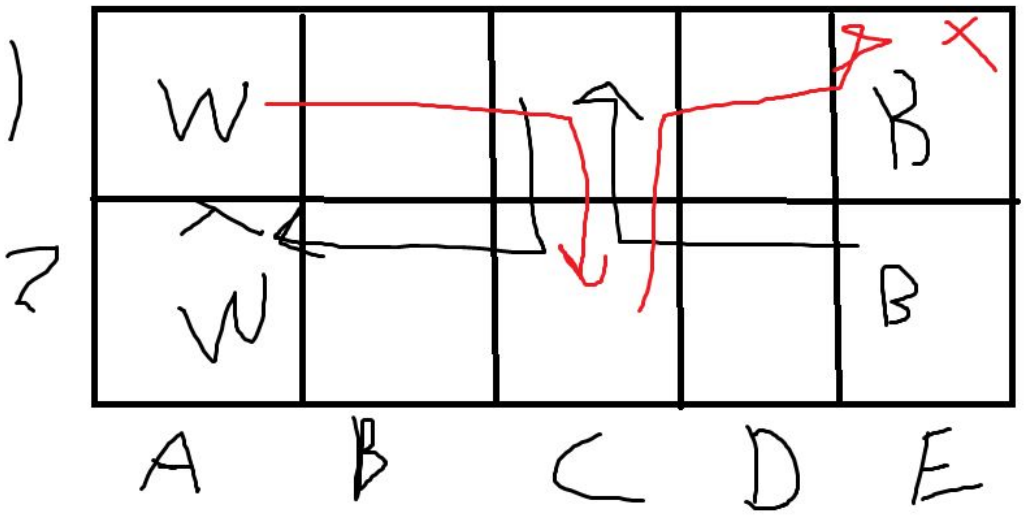
2b. Graph complete:



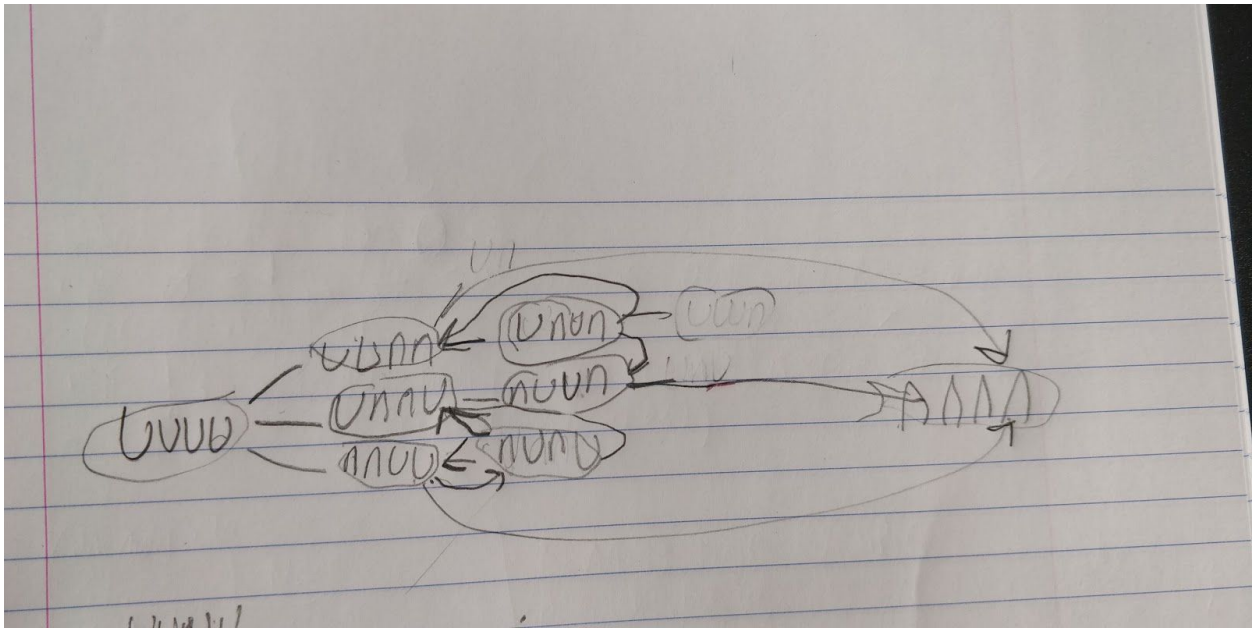
2c. Yes you can complete the puzzle, it will take 22 steps listed here:

```
C - H
L - E
B - G
K - D
A - F
J - I
H - A
E - J
G - L
D - C
F - G
I - D
A - F
J - I
L - E
C - H
G - L
D - C
E - J
H - A
F - K
I - B
```

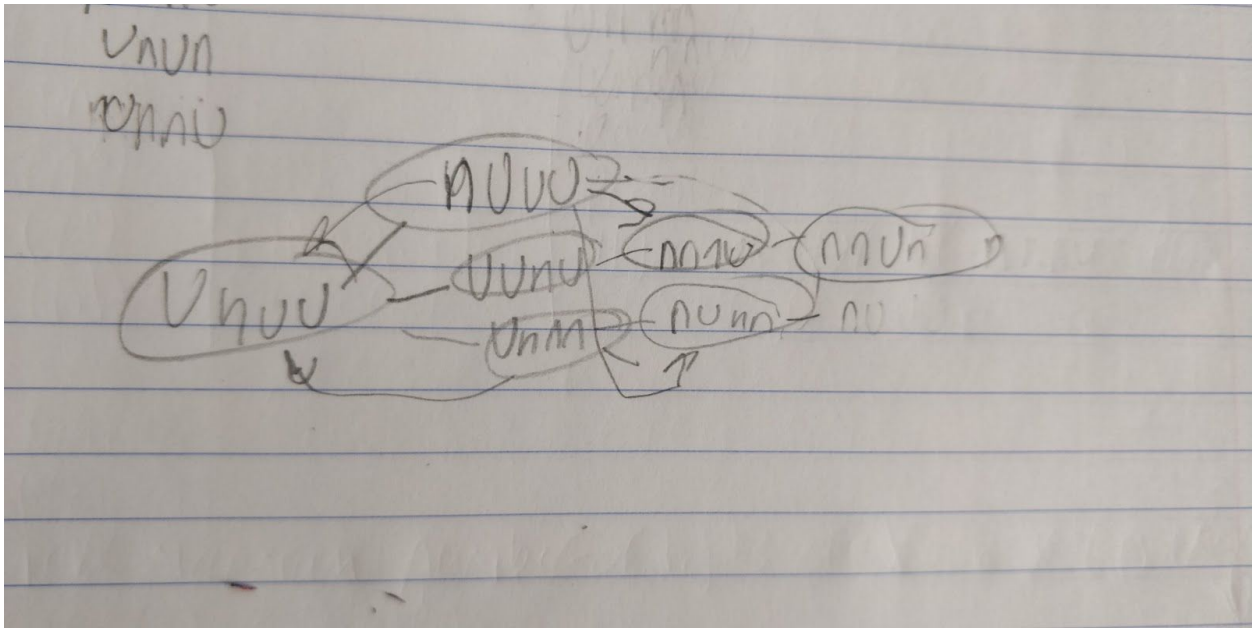
2d. No it is not possible because as shown in the picture example below, once you move either black or white piece from its starting position, it can only move back to the starting position since it would land on a taken space.



3.

a. 

B and c.



What I charted out showed that starting at UΠUU would not get to a result of UΠUΠ.
I did find that in 4 steps I would be able to get ΠΠUΠ.

4.
4a.Code:

mode.py - C:/Users/Carson/Desktop/mode.py (3.8.2)

File Edit Format Run Options Window Help

```
def mode(numlist):
    numbers = {}
    modes = []
    maxval = 0
    for num in numlist:
        if(num in numbers):
            numbers[num]+=1
        else:
            numbers[num] = 1
        if(numbers[num] >= maxval):
            maxval=numbers[num]
    for i in numbers:
        if(numbers[i] == maxval):
            modes.append(i)
    print(modes)
```

Tests:

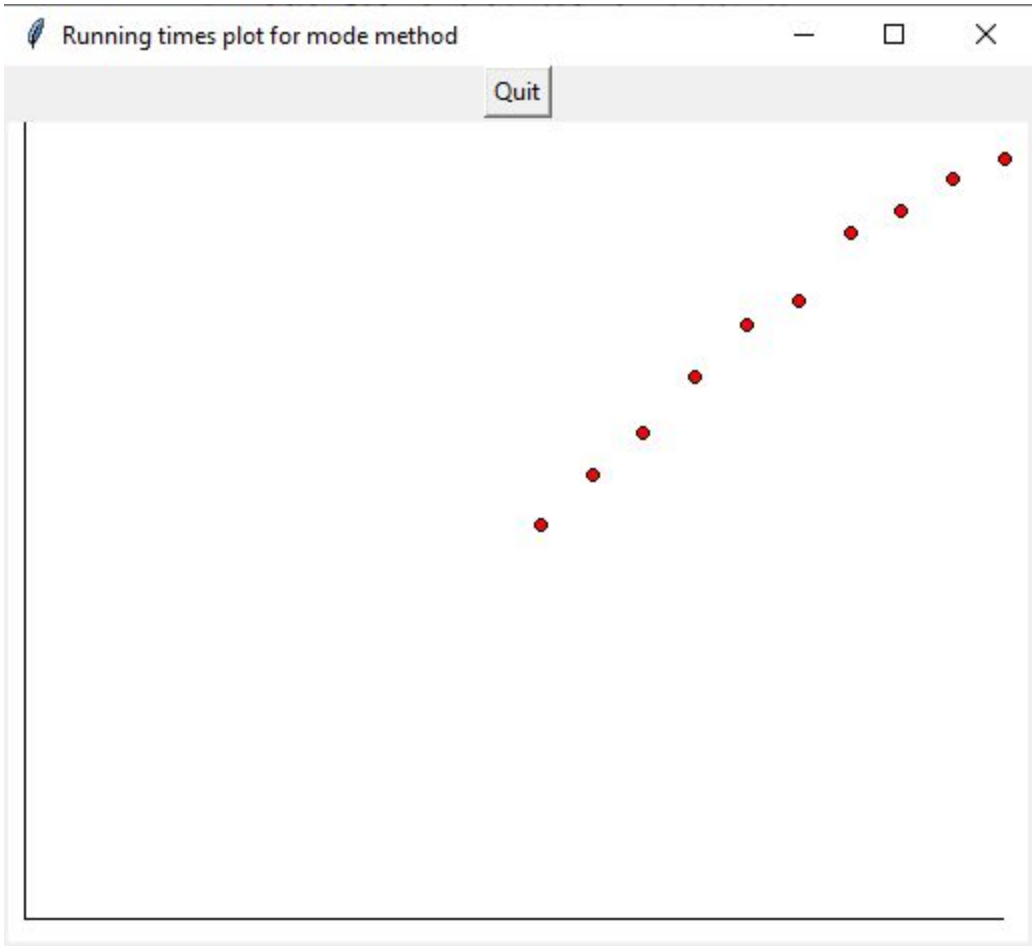
```
>>> mode([5,5,2,6,8,2,9,6,2,5,3,1])
[5, 2]
>>> mode([1,1,2,3,3,3,1,8,8,9,10])
[1, 3]
>>> mode([1,1,2,3,3,3,1,8,8,9,10,10,10,10,10])
[10]
>>>
```

4b. My running time would be 2n because it goes through the list of numbers twice, once when adding numbers to the dictionary and modifying their value, then again when looking for the mode.

4c.
Code:

```
#built from sample code on d2l
def runmode(mode,n,runs):
    t = 0
    for r in range(0,runs):
        randlist = [random() for i in range(n)]
        t1 = time.time()
        mode(randlist)
        t2 = time.time()
        t += t2-t1
    return(n,t/runs)

def plot_runtimes(data):
    T = Tk()
    T.title('Running times plot for mode method')
    Button(T, text='Quit', command=T.quit).pack()

    cv = Canvas(T,width=510, height=410, bg='white')
    cv.pack()
    cv.create_line(10,400,500,400, width=1)
    cv.create_line(10,400,10,0, width=1)

    maxn = 0
    maxt = 0
    for(n,t) in data:
        if(n >= maxn):
            maxn = n
        if(t >= maxt):
            maxt = t
    for(n,t) in data:
        x,y = 10+(n/maxn)*490,400-(t/maxt)*380
        cv.create_oval(x-3,y-3,x+3,y+3,width=1, outline='black', fill='red')
    T.mainloop()

def run_analysis():
    data = []
    for i in range(1000,2000,100):
        data.append(runmode(mode,i,1000))
    plot_runtimes(data)
```

Result:


Running times plot for mode method
Quit