Download (right-click, save target as ...) this page as a jupyterlab notebook from: Lab8

Laboratory 8: Matrices a Red Pill Approach

Medrano, Giovanni

R11521018

ENGR 1330 Laboratory 8 - In-Lab

In [1]:
```
# Preamble script block to identify host, user, and kernel
import sys
! hostname
! whoami
print(sys.executable)
print(sys.version)
print(sys.version_info)
```

```
DESKTOP-6HAS1BN
desktop-6has1bn\medra
C:\Users\medra\anaconda3\python.exe
3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]
sys.version_info(major=3, minor=8, micro=5, releaselevel='final', serial=0)
```

Reading Files, Working with 2D Lists

Is the file A-Inverse.txt indeed the inverse of A-Matrix.txt ?

Example - Using the Treasure Map

The treasure map problem was already presented, and we replaced the explicitly defined map lists with a file, allowing for the use of multiple maps. Starting with our original map, but contained in a text file named http://54.243.252.9/engr-1330-webroot/8-Labs/Lab07/treasure1.txt we can read the map using file manipulation methods.

Here is what the file looks like

```
   c1,c2,c3,c4,c5
r1,34,21,32,41,25
r2,14,42,43,14,31
r3,54,45,52,42,23
r4,33,15,51,31,35
r5,21,52,33,13,23
```

The upper left hand corner appears to be 3 spaces, then the remainder of the first row is column headings, which we dont need. Similarily the second row and beyond, has a column of row labels, then the actual data contents.

Our reading exercise will need to get just the data and ignore (or discard) the rest of the information.

However our search method visited all cells in the grid, and did not use the clues explicitly in the map. Modify the search method to use the clues in the individual cells.

In [2]:

```
treasuremap = [] # empty list to the map information
treasurefile = open("treasure1.txt","r") # open a read connection
for line in treasurefile:
    treasuremap.append([str(n) for n in line.strip().split(",")])
treasurefile.close()
```

Now we have the map, we can use list delete and slicing to remove un-necessary data

In [3]:

```
del treasuremap[0] #remove entire first row
for irow in range(len(treasuremap)): #step through remaining rows
    del treasuremap[irow][0] #kill leading column each row
```

Now we can use our treasure map search to complete the example

In [4]:

```
#####################################
for i in range(0,5,1):
    what_to_print =','.join(map(repr, treasuremap[i][:]))
    print(what_to_print) # print the map by row
howMany = 25 # set how many moves before we quit
#### Clue Directed Search ####
found = False
# start at (1,1)
rowNow=1
colNow=1
tryCount = 0
while not found:
# get row and column from rowNow and colNow values
    row = rowNow
    column = colNow
# get maprowval and mapcolval
    maprowval = str(treasuremap[row-1][column-1])[0]
    mapcolval = str(treasuremap[row-1][column-1])[1]
# test if cell is a treasure cell or not
    if int(maprowval) == row and int(mapcolval) == column :
        print('Cell ',treasuremap[row-1][column-1], ' contains TREASURE ') # print the
        print('Treasure found after ',tryCount,' cells visited')
        found = True
        break
        pass #comment this line out when have message
    else:
        print('Cell ',row,column, ' contains no treasure, move to Cell ',treasuremap[ro
        rowNow = int(maprowval)
        colNow = int(mapcolval)
        found = False
        pass #comment this line out when have message
    tryCount+=1
    if tryCount > howMany :
```

```
        print('No treasure after ',tryCount,' cells visited')
        break
```

```
'34','21','32','41','25'
'14','42','43','14','31'
'54','45','52','42','23'
'33','15','51','31','35'
'21','52','33','13','23'
Cell  1 1  contains no treasure, move to Cell  34
Cell  3 4  contains no treasure, move to Cell  42
Cell  4 2  contains no treasure, move to Cell  15
Cell  1 5  contains no treasure, move to Cell  25
Cell  2 5  contains no treasure, move to Cell  31
Cell  3 1  contains no treasure, move to Cell  54
Cell  5 4  contains no treasure, move to Cell  13
Cell  1 3  contains no treasure, move to Cell  32
Cell  3 2  contains no treasure, move to Cell  45
Cell  4 5  contains no treasure, move to Cell  35
Cell  3 5  contains no treasure, move to Cell  23
Cell  2 3  contains no treasure, move to Cell  43
Cell  4 3  contains no treasure, move to Cell  51
Cell  5 1  contains no treasure, move to Cell  21
Cell  2 1  contains no treasure, move to Cell  14
Cell  1 4  contains no treasure, move to Cell  41
Cell  4 1  contains no treasure, move to Cell  33
Cell  3 3  contains no treasure, move to Cell  52
Cell  52  contains TREASURE
Treasure found after  18  cells visited
```

Exercise 0

Consider a new treasure map contained in file http://54.243.252.9/engr-1330-webroot/8-Labs/Lab07/treasure2.txt.

Example

Develop a script to multiply a vector by a matrix.

- Apply the program to find $\mathbf{A}\mathbf{x}$ where.

\begin{gather} \mathbf{A} = \begin{pmatrix} 4.0 & 1.5 & 0.7 & 1.2 & 0.5 \\ 1.0 & 6.0 & 0.9 & 1.4 & 0.7 \\ 0.5 & 1.0 & 3.9 & 3.2 & 0.9 \\ 0.2 & 2.0 & 0.2 & 7.5 & 1.9 \\ 1.7 & 0.9 & 1.2 & 2.3 & 4.9 \\ \end{pmatrix} ~ \mathbf{x} = \begin{pmatrix} 0.595194878133 \\ 0.507932173989 \\ 0.831708392507 \\ 0.630365599089 \\ 1.03737526565 \\ \end{pmatrix} \end{gather}
Use the code blocks below to craft your answer.

In [5]:
```
#%reset -f # only if necessary
```

In [6]:
```
# create matrix A
amatrix = [[4.0,1.5,0.7,1.2,0.5],
           [1.0,6.0,0.9,1.4,0.7],
           [0.5,1.0,3.9,3.2,0.9],
           [0.2,2.0,0.2,7.5,1.9],
```

```
           [1.7,0.9,1.2,2.3,4.9]]
# create vector x
xvector = [0.595194878133,
           0.507932173989,
           0.831708392507,
           0.630365599089,
           1.03737526565  ]
# create null vector to store Ax
AXvector = [0 for i in range(0,len(xvector))] # populate with zeros
# print A
for i in range(0,len(amatrix),1):
    print(amatrix[i][:])
# print x
for i in range(0,len(xvector),1):
    print(xvector[i])
# perform the multiplication Ax put the result into Ax
for i in range(0,len(amatrix),1):
    for j in range(0,len(xvector),1):
        AXvector[i]= AXvector[i] + amatrix[i][j]*xvector[j]
# print Ax
for i in range(0,len(AXvector),1):
    print(round(AXvector[i],3))
```

```
[4.0, 1.5, 0.7, 1.2, 0.5]
[1.0, 6.0, 0.9, 1.4, 0.7]
[0.5, 1.0, 3.9, 3.2, 0.9]
[0.2, 2.0, 0.2, 7.5, 1.9]
[1.7, 0.9, 1.2, 2.3, 4.9]
0.595194878133
0.507932173989
0.831708392507
0.630365599089
1.03737526565
5.0
6.0
7.0
8.0
9.0
```

Exercise 1

Develop a script to multiply two matrices, just like in the Lesson. Apply the script to find $\mathbf{A}\mathbf{B}$ where.

\begin{gather} \mathbf{A} = \begin{pmatrix} 4.0 & 1.5 & 0.7 & 1.2 & 0.5 \\ 1.0 & 6.0 & 0.9 & 1.4 & 0.7 \\ 0.5 & 1.0 & 3.9 & 3.2 & 0.9 \\ 0.2 & 2.0 & 0.2 & 7.5 & 1.9 \\ 1.7 & 0.9 & 1.2 & 2.3 & 4.9 \\ \end{pmatrix} ~ \mathbf{B} = \begin{pmatrix} 0.27196 & -0.05581 & -0.03285 & -0.01687 & -0.007203 \\ -0.036787 & 0.186918 & -0.03206 & -0.011457 & -0.012618 \\ -0.02595 & -0.001333 & 0.268266 & -0.10875 & -0.004267 \\ 0.027048 & -0.050632 & 0.016499 & 0.14865 & -0.056198 \\ -0.093939 & 0.009124 & -0.056155 & -0.03519 & 0.236322 \\ \end{pmatrix} \end{gather}
The two matrices are located in files:

http://54.243.252.9/engr-1330-webroot/8-Labs/Lab08/A-Matrix.txt

and:

http://54.243.252.9/engr-1330-webroot/8-Labs/Lab08/A-Inverse.txt

You should download these files before proceeding

In [11]:
```
matrixA = []
matrixfile = open("A-Matrix.txt","r")
for line in matrixfile:
    matrixA.append([float(n) for n in line.strip().split()])
matrixfile.close()

matrixinv = []
matrixfile = open("A-Inverse.txt","r")
for line in matrixfile:
    matrixinv.append([float(n) for n in line.strip().split()])
matrixfile.close()

matrixC = []
for i in range(0,len(matrixA)):
    matrixC.append([0 for i in range(0,len(matrixinv[0]))])

print("-----------MatrixA-------")
for i in range(0,len(matrixA),1):
    print(matrixA[i][:])

print("-----------MatrixA^-1-------")
for i in range(0,len(matrixinv),1):
    print(matrixinv[i][:])
for i in range(0,len(matrixA),1):
    for j in range(0,len(matrixinv[0])):
        matrixC[i][j] = matrixC[i][j] + matrixA[i][j]*matrixinv[i][j]

print("-----------MatrixAA^-1-------")
for i in range(0,len(matrixC),1):
    print(matrixC[i][:])
```

```
-----------MatrixA-------
[4.0, 1.5, 0.7, 1.2, 0.5]
[1.0, 6.0, 0.9, 1.4, 0.7]
[0.5, 1.0, 3.9, 3.2, 0.9]
[0.2, 2.0, 0.2, 7.5, 1.9]
[1.7, 0.9, 1.2, 2.3, 4.9]
-----------MatrixA^-1-------
[0.27196423630168165, -0.05581183146290884, -0.032853102922602934, -0.016869919448735553, -0.0072026931722172435]
[-0.036786468827077756, 0.18691841183385363, -0.032062455842026744, -0.011456196435011407, -0.012617687833839365]
[-0.025949127789423248, -0.0013334022990376664, 0.26826513178341493, -0.10875073215127727, -0.004266180002777282]
[0.027047195749338872, -0.05063248905238324, 0.01649816113355711, 0.1486518640705042, -0.05619749842697155]
[-0.0939389748254409, 0.009124153146082323, -0.05615458031041434, -0.03518550386250331, 0.23632125710787594]
-----------MatrixAA^-1-------
[1.0878569452067266, -0.08371774719436326, -0.02299717204582205, -0.020243903338482663, -0.0036013465861086218]
[-0.036786468827077756, 1.1215104710031218, -0.02885621025782407, -0.01603867500901597, -0.008832381483687554]
[-0.012974563894711624, -0.0013334022990376664, 1.0462340139553181, -0.34800234288408727, -0.0038395620024995543]
[0.005409439149867775, -0.10126497810476648, 0.0032996322267114225, 1.1148889805287816, -0.10677524701124594]
[-0.15969625720324954, 0.008211737831474091, -0.0673854963724972, -0.08092665888375761, 1.1579741598285922]
```

References

1. List processing tips https://www.programiz.com/python-programming/del

2. Character replacement tips https://www.geeksforgeeks.org/python-string-replace/

3. Python file manipulations https://www.tutorialspoint.com/python/python_files_io.htm

4. A linear algebra primer https://numericalmethodssullivan.github.io/ch-linearalgebra.html

5. Python file manipulations https://www.tutorialspoint.com/python/python_files_io.htm

6. A Complete Beginners Guide to Matrix Multiplication for Data Science with Python Numpy https://towardsdatascience.com/a-complete-beginners-guide-to-matrix-multiplication-for-data-science-with-python-numpy-9274ecfc1dc6