# a2q4q5_YOU

October 7, 2021

## 1 A2: Q4 and Q5

[29]:
```
import numpy as np
from scipy.sparse import dok_matrix
from copy import deepcopy
import matplotlib.pyplot as plt
```

## 2 A2Q4a: SparseMatMult

[30]:
```
def SparseMatMult(G, x):
    '''
      y = SparseMatMult(G, x)

      Multiplies a vector (x) by a sparse matrix G,
      such that y = G @ x .

      Inputs:
        G is an NxM dictionary-of-keys (dok) sparse matrix
        x is an M-vector

      Output:
        y is an N-vector
    '''
    rows,cols = G.nonzero()
    Nrows,Ncols = np.shape(G)
    y = np.zeros(Nrows)

    # === YOUR CODE HERE ===
    size = len(rows)
    for n in range(size):
        i = rows[n]
        j = cols[n]
        y[i] += x[j]*G[i,j]

    return y
```

[31]:
```
# Simple test
#     [1  0  0]       [ 0.1 ]
# A = [0  1 -1]  b = [ 0.2 ]
#     [0  2  0]       [ 0.3 ]
A = dok_matrix((3,3), dtype=np.float32)
A[0,0] = 1
A[1,2] = -1
A[1,1] = 1
A[2,1] = 2
b = np.array([0.1, 0.2, 0.3])
y = SparseMatMult(A, b)
print(f'y = {y}')
print(f'Answer should be [ 0.1 -0.1  0.4]')
```

```
y = [ 0.1 -0.1  0.4]
Answer should be [ 0.1 -0.1  0.4]
```

## 3 A2Q4b: PageRank

[77]:
```
def PageRank(G, alpha):
    '''
     p, iters = PageRank(G, alpha)

     Computes the Google Page-rank for the network in the adjacency matrix G.

     Note: This function never forms a full RxR matrix, where R is the number
           of node in the network.

     Input
       G     is an RxR adjacency matrix, G[i,j] = 1 iff node j projects to node␣
 ↪i
             Note: G must be a dictionary-of-keys (dok) sparse matrix
       alpha is a scalar between 0 and 1

     Output
       p     is a probability vector containing the Page-rank of each node
       iters is the number of iterations used to achieve a change tolerance
             of 1e-8 (changes to elements of p are all smaller than 1e-8)
    '''
    R = np.shape(G)[0]
    rows,cols = G.nonzero()
    iters = 0

    # === YOUR CODE HERE ===
    # calculate degrees
    deg = np.zeros(R)
```

```python
    for j in range(R):
        for x in cols:
            if (x == j):
                deg[j] += 1
    #print(f'degs: {deg}')
    # make P
    P = G
    for n in range(len(rows)):
        i = rows[n]
        j = cols[n]
        P[i,j] = 1/deg[j]
        #print(f'value at {i},{j} = {P[i,j]}')
    # make initial p
    p = np.zeros(R)
    for i in range(R):
        p[i] = 1/R
    #print(f'p = {p}')

    # iterate
    diff = 1
    iters = 0
    while (abs(diff) > 0.00000001):
        oldp = np.zeros(R)
        for i in range(R):
            oldp[i] = p[i]
        #print(f'{oldp}')
        degtimesp = np.zeros(R)
        for i in range(R):
            if (deg[i] == 0):
                degtimesp[i] = (p[i])/R
            else:
                degtimesp[i] = 0
        ptimesp = SparseMatMult(P, p)
        #print(f'd times p = {degtimesp}')
        #print(f'P times p = {ptimesp}')
        # generate new p
        for i in range(R):
            p[i] = (ptimesp[i]+degtimesp[i])*alpha + (1-alpha)/R
        #print(f'    new p = {p}')
        # compare
        diffs = np.zeros(R)
        for i in range(R):
            diffs[i] = abs(p[i]-oldp[i])
            #print(f'diffs[{i}] = abs({p[i]}-{oldp[i]}) = {diffs[i]}')
        diff = max(diffs)
        iters += 1
```

```
    return p, iters
```

[78]:
```
A = dok_matrix((6,6), dtype=np.float32)
A[1,0] = 1
A[0,3] = 1
A[0,5] = 1
A[2,1] = 1
A[3,1] = 1
A[3,2] = 1
A[4,2] = 1
A[5,2] = 1
A[5,4] = 1
# its the example from the lecture
alph = 0.85
p, iters = PageRank(A, alph)
print(f'p = {p}, {iters} iterations')
```

```
p = [0.26752809 0.25239887 0.13226952 0.16974589 0.06247637 0.11558128], 38
iterations
```