Graph Data Mining HW1

Sai Kumar Goud Vengali (svengal@ncsu.edu)

4 September 2022

1. (a) To Make right decision we need to know the details and specifications of the graph.
   i. Whether the graph is simple or multigraph.
   ii. If graph is directed or undirected.
   iii. Whether the graph is static or any other edges are added later i.e Dynamic.
   iv. Whether there are any labeled or weighted edges.
   v. If graph is sparse or dense.
   vi. If graph is unipartite, bipartite or multipartite

   (b) **Scenario 1:** If the graph is sparse, undirected, unweighted and labeled.
   Scenario 2: If the graph is dense, directed, weighted and labeled.
   Scenario 3: If the graph is dense, undirected, unweighted and labeled.

   (c) The most efficient data structure, Time Complexity and space complexity for the given operations in all scenarios.

	Data Structure	Add Edge	Delete Edge	Add vertex	Delete Vertex	Space
Scenario 1	Adjacency list	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(deg(v))$	$\mathcal{O}(n)$
Scenario 2	Adjacency Matrix	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Scenario 3	Adjacency Matrix	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$

2. (a) In graphs, calculating path matrix is the easiest way to know whether two nodes are connected.
   For the graph G, let A be the adjacency matrix in which A[i][j] represents the path distance from i to j.
   If we multiply A by itself we get A^2 where A[i][j] represents number of paths of path length 2 between i,j.
   we multiply A by itself to find A^n and add these matrices to get the intermediate matrix I.

$$I = A + A^2 + ...A^n$$

As the path matrix is a boolean matrix(0,1) we replace I[i][j]values with 1 if the value is non zero to get a resulting matrix P.

Therefore, i and j are connected and if I[i][j]$>$ 0 we replace it with 1 resulting in path matrix P.
Therefore for P:
$P[i][j] = 1$ if two vertices are connected.

(b) **PsuedoCode for the Algorithm Algorithm**

Function Matrixmultiplication(A,M)

```
n = order of matrix A
let X be the matix of order n
    for i= 1 to n do
      for j=1 to n do
          let sum=0
          for k =1 to n do
              sum = A_ik * M_kj
              X_ij = sum
    return X
```

Function PathMatrix(A)

```
n = order of matrix A
let P be the matix of order n
    for i= 1 to n do
        P = P + Matrixmultiplication(A,P)

    for i =1 to n do
        for j=1 to n do
            If P_ij > 0then
            P_ij = 1
    return P
```

we find the path ma

(c) **Time Complexity:**$\mathcal{O}(n^4)$
As we are calculating the matrices $A, A^2, A^3, ...A^n$ here multiplication of the matrices takes $\mathcal{O}(n^3)$.
As we are doing these multiplications for n times, the complexity of the algorithm is $\mathcal{O}(n * n^3)$

(d) As we know that the above algorithm runs in order of 4.
Given laptop running at $2.4 * 10^9$ floating points per second.
per day it is

$$2.4 * 10^9 * 86400$$

Therefore, It is impratical if the value of n is:

$$n >= \sqrt[4]{207360 * 10^9}$$

$$n >= 3795$$

(e) If we consider the values of the matrix i.e A[i][j] is integer and an integer requires 4 bytes of storage.
The space you require for a matrix is $\mathcal{O}(n^2)$.
for the integer matrix it requires 4 * (n^2)

i. For 512 MB

$$4 * n^2 = 512MB = 512 * 10^6$$

$$n = 16K$$

ii. For 1GB

$$4 * n^2 = 1GB = 1 * 10^9$$

$$n = 15811$$

iii. For 16GB

$$4 * n^2 = 16GB = 16 * 10^9$$

$$n = 63245$$

As per our selected data structure, the upper limit for number of nodes that can be possible is n= 3795.
So the required bytes is

$$= 4 * n^2$$

$$= 4 * 3795 * 3795$$

$$= 57608100 = 57.6MB$$