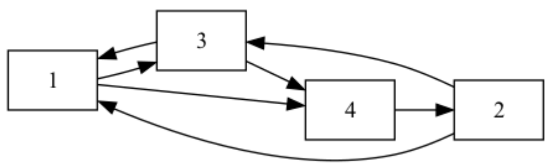
Assignment 2

Barak-Nadav Diker

June 15, 2023

Question 1

Given the following graph



Calculate all walks of size 6 in the given graph

Answer

In order to do so we'll use the following theorem

Theorem 1 (Walks Theorem)**.** *If A is the adjacency matrix of a graph or digraph G with vertices $\{v1, ...vn\}$, then the i, j entry of A^k is the number of walks of length k from v_i to v_j*

By 1 we can just multiple the adjacency matrix by itself 6 times and we'll get all the walks available from node i into node j

Since the problem specify undirected graph we'll have to sum all elements of the matrix and divide it by 2

Algorithm 1 All walks of length 6

$n \leftarrow 6$ ▷ 6 => length of walk
Adj ▷ adjacency matrix
$M \leftarrow I$
while $n \neq 0$ **do**
  $M \leftarrow M \times Adj$ ▷ Matrix Multiples
end while
$sum \leftarrow 0$
while $a \in M$ **do**
  $sum \leftarrow sum + a$
end while

We will apply **algorithm** 1 for getting the number of walk of length 6

```
import numpy
from numpy.linalg import matrix_power
input_array = numpy.array([[0,1,1,1],
                           [1,0,1,0],
                           [1,1,0,1],
                           [1,0,1,0]])
A_to_power6 = matrix_power(input_array, 6)
sum_var = 0
for row in A_to_power6:
    for elem in row:
        sum_var += elem
int(sum_var/2)
# output is 557
```

Question 2

Consider the following quote

> "Undirected Graph can be considered as directed graph"

Prove it

Formally , Given an Undirected graph find a directed graph such that $A_G = A_{G'}$

Answer

Given An undirected graph mark it $G = (V, E)$ and the adjacency matrix of his as A_G

Define the directed graph $G' = (V', E')$ where $V' = V$ and

$$\forall e = (v_1, v_2) \in E : e_1 = (v_1, v_2), e_2 = (v_2, v_1) \in E'$$

The adjancey matrix of the directed graph is equal to the adjancey matrix of the undirected graph

$$A_G = A_{G'}$$

Question 3

Prove that given a directed graph $G = (V, E)$ where $V = (1, 2, .., n)$, let A be the adjacency matrix

$$l \in \mathbb{N} \cap \{0\} : \forall i, j \in V : F(j, i, l) = (A^l)_{i,j}$$

where $F : V \times V \times \mathbb{N} \cap \{0\} \to \mathbb{N} \cap \{0\}$ are all the walks from node j to i of length l

Answer

We'll prove the theorem by induction

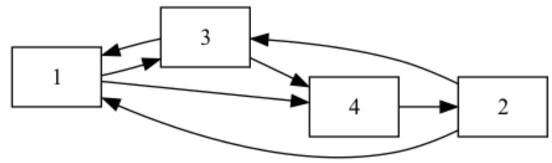
Proof. By induction

Base Case: For k = 1, $A^k = A$, and there is a walk of length 1 between i and j if and only if $a_{ij} = 1$, thus the result holds.

Step Case: Assume the proposition holds for $k = n$ and consider the matrix $A_{n+l} = A_n A$, By the inductive hypothesis, the $(i, j)^{th}$ entry of A_n counts the number of walks of length n between vertices i and j. Now, the number of walks of length n + 1 between i and j equals the number of walks of length n from vertex i to each vertex v that is adjacent to j. But this is the $(i, j)^{th}$ entry of $A^n A = A^{n+1}$ the non-zero entries of the column of A corresponding to v are precisely the first neighbours of v. Thus the result follows by induction on n $\square$

Question 4

Find the number of possible walks of length 8 from 1 to 4 by the following undirected graph



Answer

We'll Write the adjancey matrix and calculate the A^8

To do so we'll use the same code

```
import numpy
from numpy.linalg import matrix_power
input_array = numpy.array([[0,1,1,0],
                           [0,0,0,1],
                           [1,1,0,0],
                           [1,0,1,0]])
A_to_power6 = matrix_power(input_array, 8)
A_to_power6[3][0]
# output is 23
```

Introduction

Kruskal's algorithm[1] finds a minimum spanning forest of an undirected edge-weighted graph. If the graph is connected, it finds a minimum spanning tree. (A minimum spanning tree of a connected graph is a subset of the edges that forms a tree that includes every vertex, where the sum of the weights of all the edges in the tree is minimized. For a disconnected graph, a minimum spanning forest is composed of a minimum spanning tree for each connected component.) It is a greedy algorithm in graph theory as in each step it adds the next lowest-weight edge that will not form a cycle to the minimum spanning forest.[2]

This algorithm first appeared in Proceedings of the American Mathematical Society, pp. 48–50 in 1956, and was written by Joseph Kruskal.[3] It was rediscovered by Loberman & Weinberger (1957).[4]

Other algorithms for this problem include Prim's algorithm, the reverse-delete algorithm, and Borůvka's algorithm.

Simple pseudo code

Here is some code

This is some random text

```
i ← 10
if i ≥ 5 then
    i ← i − 1
else
    if i ≤ 3 then
        i ← i + 2
    end if
end if
```

bla bla bla

Another Example , please note the following

Algorithm 2 An algorithm with caption

```
Require: n ≥ 0
Ensure: y = x^n
  y ← 1
  X ← x
  N ← n
  while N ≠ 0 do
      if N is even then
          X ← X × X
          N ← N/2                ▷ This is a comment
      else if N is odd then
          y ← y × X
          N ← N − 1
      end if
  end while
```