Graph Theory & Algorithms

Aniket Pandey

23 December 2017

1 Introduction

Graph Theory is the study of Graphs, the mathematical objects modelling the pairwise relation of Vertices(also called nodes) and Edges where two nodes are connected by edges. A graph can be directed or undirected, cyclic or acyclic, linear or weighted etc.

Graph Theoretical concepts are widely used to study and model various applications, in different areas. For example, in Computer Science, problems like travelling salesman problem, the shortest spanning tree in a weighted graph and in Mathematics like hamiltonian graphs and Fermat's Little Theorem & Nielson-Schreier Theorem.

2 Notations

2.1 Big-O Notation

Big-O Notations are used in mathematics to characterize functions according to their growth rate. In Graph Theory, efficiency of an algorithm is measured in terms of the input length n as $n \to \infty$.

Formal definition would be
If $f : N \to N$ and $g : N \to N$ are two functions, then $f = \mathrm{O}(g)$ if and only if $f(n) < c{\cdot}g(n)$ for a constant c as $n \to \infty$.

2.2 Other Notations

There are a few more notations which complement Big-O Notation. I will give a brief information about these.

For functions f & g from N to N

$$f = \Omega(g) \quad \text{if } g = O(f) \tag{1}$$
$$f = \Theta(g) \quad \text{if } f = O(g) \ \&\ g = O(f) \tag{2}$$
$$f = o(g) \quad \text{if there exists } \varepsilon \text{ such that } f(n) < \varepsilon \cdot g(n) \tag{3}$$
$$f = \omega(g) \quad \text{if } g = o(f) \tag{4}$$

3 Terminologies

Here are a few basic terminologies that are used to represent navigation through the Graph.

Walk A walk is any route through a graph from vertex to vertex along edges. A walk can end on the same vertex on which it began or on a different vertex. A walk can travel over any edge and any vertex any number of times.

Path A path is a walk that does not include any vertex twice, except that its first vertex might be the same as its last.

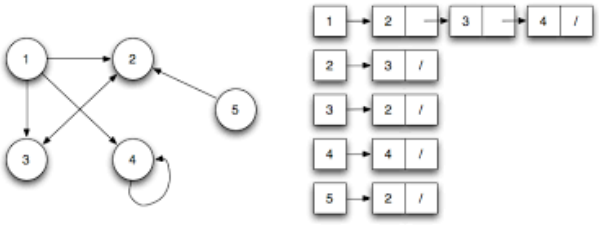
Trail A trail is a walk that does not pass over the same edge twice. A trail might visit the same vertex twice, but only if it comes and goes from a different edge each time.

Cycle A cycle is a path that begins and ends on the same vertex.

4 Data Structures used in Graphs

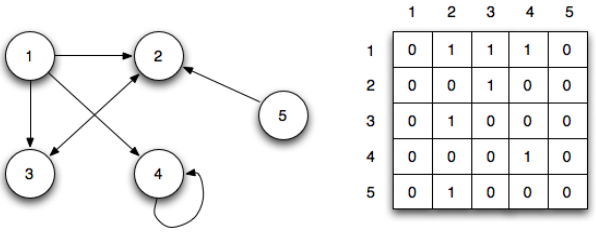
4.1 Adjacency List

An Adjacency List is a collection of unordered lists used to represent a finite graph. Each list describes the set of neighbors of a vertex in the graph.



4.2 Adjacency Matrix

An Adjacency Matrix is a square matrix used to represent a finite graph. The elements of the matrix indicate whether pairs of vertices are adjacent or not in the graph.



4.3 Stacks and Queues

Stacks and Queues are dynamic sets in which the element removed from the set by the *delete* operation is prespecified. In a Stack, the element deleted from the set is the one inserted most recently, while in a Queue, the element that is to be deleted is the element which has been in the Queue for the longest time.

A **Stack** is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack.

A **Queue** is a container of objects that are inserted and removed according to the first-in first-out (FIFO) principle. In the queue only two operations are allowed, enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item.

STACK

```
def StackEmpty(S)
    if S.top == 0
        return TRUE
    else return FALSE

def push(S,x)
    S.top = S.top + 1
    S[S.top] = x

def pop(S)
    if StackEmpty(S)
        error "Stack Underflow"
    else S.top = S.top - 1
        return S[S.top+1]
```

QUEUE

```
def enqueue(Q,x):
    Q[Q.tail] = x
    if Q.tail == Q.length:
        Q.tail = 1
    else Q.tail = Q.tail + 1

def dequeue(Q):
    x = Q[Q.head]
    if Q.head == Q.length:
        Q.head = 1
    elif Q.head == 0:
        error "Queue Underflow"
    else Q.head = Q.head + 1
    return x
```

5 BFS and DFS