IOSR Journal of Mathematics (IOSR-JM)
e-ISSN: 2278-5728, p-ISSN: 2319-765X. Volume 16, Issue 1 Ser. I (Jan – Feb 2020), PP 35-47
www.iosrjournals.org

Graph Search Algorithms and Its Application

Belela Samuel Kotola
Departments of Mathematics Debre Berhan University , Ethiopia.

Abstract: *The main motive behind to the preparation of this project is to understand the graph search algorithm in a simple way. This work introduces and discusses concepts to implement graph algorithms in a reusable fashion*
The module begin with the introduction of graph search ,preliminary and followed by chapter three which is graph search algorithm and finally chapter four that studies about application of graph search.
Keywords: Search algorithm, connected graphs, trees and forest

---

Date of Submission: 20-12-2019 Date of Acceptance: 03-01-2020

I. Introduction

In mathematics, graph theory is the study of graphs, which are mathematical structures used to model pairwise relations between objects. A graph in this context is made up of vertices, nodes, or points which are connected by edges, arcs, or lines. A graph may be undirected, meaning that there is no distinction between the two vertices associated with each edge, or its edges may be directed from one vertex to another; see Graph (discrete mathematics) for more detailed definitions and for other variations in the types of graph that are commonly considered. Graphs are one of the prime objects of study in discrete mathematics.

The basic ideas of graph theories were introduced in the eighteenth century by the great Swiss mathematician Leonhard Euler. He used graph to solve the famous Konigsberg bridge problem.

Graph searching is a hot topic in mathematics and computer science now, as it leads to a wealth of beautiful mathematics, and since it provides mathematical models for many real-world problems such as eliminating a computer virus in a network, computer games, or even counterterrorism. In all searching problems, there is a notion of searchers (or cops) trying to capture some robber (or intruder, or fugitive). A basic optimization question here is: What is the fewest number of searchers required to capture the robber? There are many graph searching problems motivated by applied problems or inspired by some theoretical issues in computer science and discrete mathematics

One of the most fundamental tasks on graphs is searching a graph by starting at some vertex or set of vertices and visiting new vertices by crossing (out) edges until there is nothing left to search.

In such a search we need to be systematic to make sure that we visit all vertices that we can reach and that we do not visit vertices multiple times.

This will require recording what vertices we have already visited so we don't visit them again. Graph searching can be use to determine various properties of graphs, such as whether the graph is connected or whether it is bipartite, as well as various properties relating vertices, such as whether a vertex u is reachable from v, or finding the shortest path between vertices u and v.

Basic Concepts

♣ Search methods often divide the vertices into tree sets:

Three states of vertices

- ❖ unvisited
- ❖ visited/discovered
- ❖ fully-explored

Vertices are visited only once, but may be encountered multiple times.

All search methods when starting on a single source vertex generate a rooted search tree, either implicitly or explicitly. This tree is a subset of the edges from the original graph. In particular a search always visits a vertex v by entering from one of its neighbor's u via an edge E (u, v). This visit to v adds the edge (u, v) to the tree. These edges form a tree (i.e., have no cycles) since no vertex is visited twice and hence there will never be an edge that wraps around and visits a vertex that has already been visited.

We refer to the source vertex as the root of the tree. Graph searching has played a very important role in the design of sequential algorithms, but the approach can be problematic when trying to achieve good parallelism.

Depth first search (DFS) has a wealth of applications, but it is inherently sequential. Because of this, one often uses other techniques in designing good parallel algorithms.

Breadth first search (BFS), on the other hand, can be parallelized effectively as long as the graph is shallow (the longest shortest path from the source to any vertex is reasonably small). In fact, the depth of the graph will show up in the bounds for span.

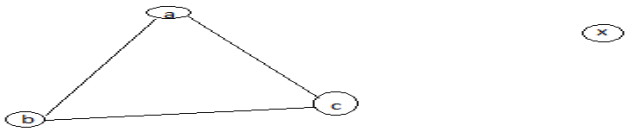
PRELIMINARY – (Definitions and others)

Degree of vertices

The degree of a graph vertex $v\ v$ of a graph $G\ iG\ i$s the number of graph edges which touchvv.
Directed graphs have two types of degrees, known as the in degree and the out degree.

Isolated vertices

For a graph G= (V (G), E (G)), a vertex x∈V (G) is considered Isolated if
deg(x) =0.
For example, the following graph has one isolated vertex:



Note that if a graph has an isolated vertex, then the graph is disconnected.

Vertices are said to be adjacent

In graph theory, vertices (or nodes) are connected by edges. When two vertices are joined by an edge, they are said to be adjacent to one another. More formally, let G (V,E) be an undirected graph on |V| vertices with |E| edges. Let $a, b \in Va, b \in V$ be vertices in V. An edge ab is said to join vertices $a\ and\ b\ \ a\ and\ b$. Then we say that the vertices a and b are adjacent.

Finite and infinite graph

$n\ n$

a. A graph with a finite number of nodes and edges is called finite graph. If it has nodes and has no multiple edges or graph loops (i.e., it is simple), it is a subgraph of the complete graph K_n.
b. A graph which is not finite is called infinite. If every node has finite degree, the graph is called locally finite. The Cayley graph of a group with respect to a finite generating set is always locally finite, even if the group itself is infinite.

Connected and dis connected graph

In mathematics and computer science, connectivity is one of the basic concepts of graph theory: it asks for the minimum number of elements (nodes or edges) that need to be removed to disconnect the remaining nodes from each other.

A graph is connected when there is a path between every pair of vertices. In a connected graph, there are no unreachable vertices. A graph that is not connected is disconnected. A graph G is said to be disconnected if there exist two nodes in G such that no path in G has those nodes as endpoints. A graph with just one vertex is connected. An edgeless graph with two or more vertices is disconnected.

Length of the graph

The distance between two vertices in a graph is the length of a shortest path between them, if one exists, and otherwise the distance is infinity. The diameter of a connected graph is the largest distance (defined above) between pairs of vertices of the graph.

Euler graph

The term Eulerian graph has two common meanings in graph theory. One meaning is a graph with an Eulerian circuit, and the other is a graph with every vertex of even degree. These definitions coincide for connected graphs.

In graph theory, an Eulerian trail (or Eulerian path) is a trail in a graph which visits every edge exactly once. Similarly, an Eulerian circuit or Eulerian cycle is an Eulerian trail which starts and ends on the same vertex. They were first discussed by Leonhard Euler while solving the famous Seven Bridges of Königsberg problem in 1736.

Hamiltonian graph

In the mathematical field of graph theory, a Hamiltonian path (or traceable path) is a path in an undirected or directed graph that visits each vertex exactly once. A Hamiltonian cycle (or Hamiltonian circuit) is a Hamiltonian path that is a cycle.Hamiltonian paths and cycles are named after William Rowan Hamilton who invented the icosian game, now also known as Hamilton's puzzle, which involves finding a Hamiltonian cycle in the edge graph of the dodecahedron. Hamilton solved this problem using the icosian calculus, an algebraic structure based on roots of unity with many similarities to the quaternions (also invented by Hamilton). This solution does not generalize to arbitrary graphs.

Tree graphs and forest

A connected graph that contains no simple circuits is called a tree. Forest is a circuit less graph. Trees was used as longago as 1857, when the English mathematician Arthur Cayley used them to count certaintypes of chemical compounds. Since that time, trees have been employed to solve problems inwide variety of disciplines. Trees are particularly useful in computer science, where they are employed in a wide range of algorithms. For instance, trees are used to construct efficient algorithms for locating items in a list. They can be used in algorithms, such as Huffman coding, that construct efficient codes saving costs in data transmission and storage. Trees can be used to study games such as checkers and chess and can help determine winning strategies for playing these games. Trees can be used to model procedures carried out using a sequence of decisions. Constructing these models can help determine the computational complexity of algorithms based on a sequence of decisions, such as sorting algorithms.

Path

A path is a route through a graph visiting vertices and edges in turn. A cycle is a path that ends at the starting vertex.

If we saw this much now let us consider the problem below.

Problem: given a digraph with positive edge weights $G = (V, E)\ G = (V, E)$ and a distinguished source vertex $s \varepsilon V \quad s \varepsilon V$ then determine the distance and shortest path from the source vertex to every vertex in the digraph. How do you design efficient algorithms for this problem?

These concepts are what we are going to see in the next chapter.

Reachability

Given a graph $G = (V, E) G = (V, E)$ and two vertices $u, v \varepsilon V, u, v \varepsilon V,$ a vertex vv is reachable from another vertex u if there is a path from v to u. we also say sometimes that v can reach u.

Connectedness

A (directed) graph is (strongly) connected if there is a (directed) path between every pair of vertices.

GRAPH SEARCH ALGORITHMS

A Searching of vertices of connected graph G is systematic procedure for visiting all the vertices of G by travelling along its edge. An edge of G may be traversed more than once and it's a vertex may be visited more than once in the search.

In the other word; a searching of a graph is a method of constructing an edge sequence whose associated vertex sequence includes every vertex of the graph.

An algorithm is a step by step method of solving some problem. Or (an algorithm is a finite set of instruction that, if followed accomplishes a particular task.)

Characteristic of Algorithm

Algorithms generally have the following characteristic

1. Input. The algorithms receive input. Zero or more quantities are externally supplied.
2. Output. The algorithms produce outputs. At least one quantity is produced.
3. Precision. The steps are precisely stated each instruction in clear and unambiguous.
4. Determinism. The intermediate results of each steps of execution are unique and are determined only by the input and the result of the preceding steps.
5. Finiteness. If we trace out the instruction of an algorithm when for all cases, the algorithms terminate after finite number of steps.
6. Correctness. The output produced by the algorithm must be correct. That mean the algorithm correctly solves the problem.
7. Generality. The algorithms applied to asset of inputs.

8. Effectiveness. Every instruction must be very basic so that it can be carried out in principle by apart using only pencil and pen.

Breadth first search algorithms

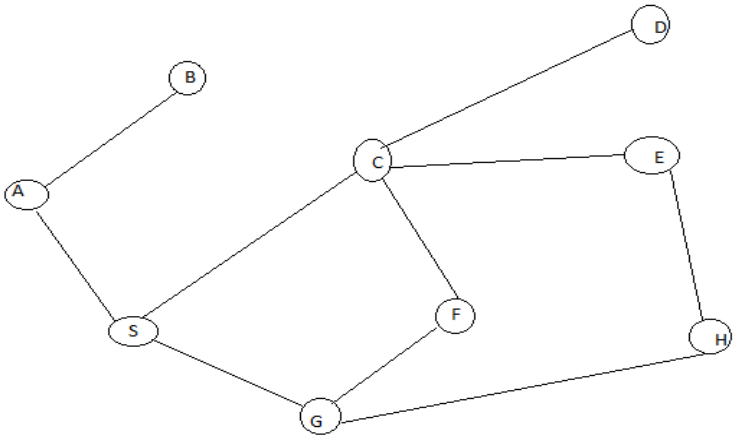
The breath first search algorithm is a particular graph search algorithm that can be applied to solve a variety of problems such as finding all vertices reachable from given vertex, as with the other graph search BFS can be applied to both directed and undirected graphs.

$O(v+e)O(v+e)$ Is its running time where the graph has vertices vv and edge ee.

Le as see the algorithm.

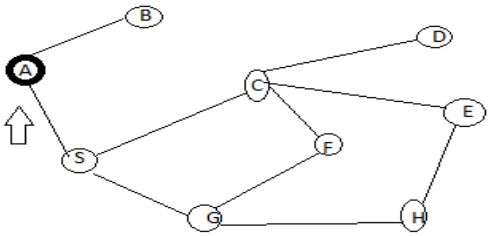
1. In the beginning, no vertex is labeled. Set i ← 0.
2. . Choose a (unlabeled) starting vertex r (root) and label it with i.
3. Search the set J of vertices that are not labeled and are adjacent to some vertex labeled withi.
4. If $J \neq \emptyset, J \neq \emptyset,$ then set i ← i + 1. Label the vertices in J with i and go to step #3.
5. (Only for disconnected graphs so we can jump from one component to another.) If a vertex is unlabeled, then set i ← 0 and go to step #2.
6. Stop.

Example



Solution

Step one

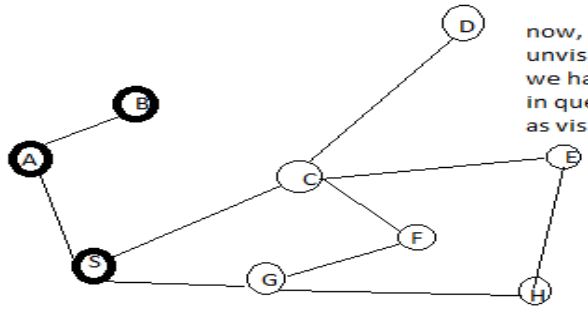


let us start over discusing on node A and make node A bold to notice that a A visited. and use ray as written there to indicate tha A is our current node thatn we are working on.

queue status **out pu**

B

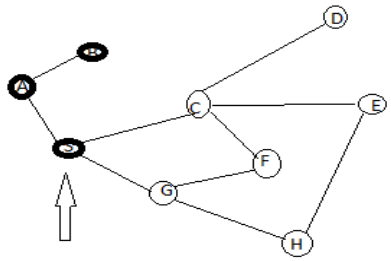
Step two



now, we have to check all the adjecent unvisited node to the current node. here we have B&S, so we writte alphabetically in queue and in out put after we marked it as visited.

queue status	out put
B,S	A B S

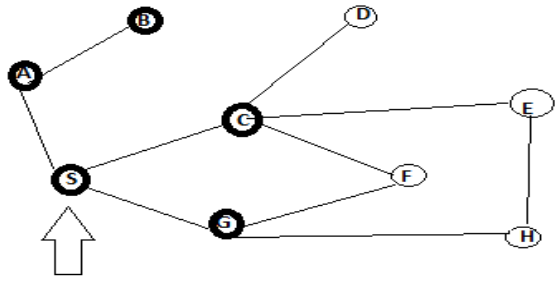
Step three



For node A, B&S are the only adgecent node so, we have to up date the currently working node. so we have to check first of all the first element of queue. we have B here. so we should up date the current working node to B and remove B now from queue. now we see that B has A as adgecent wich is all ready visited. so we up date the current node again to S and remove s from queue

queue	out put
	A B S

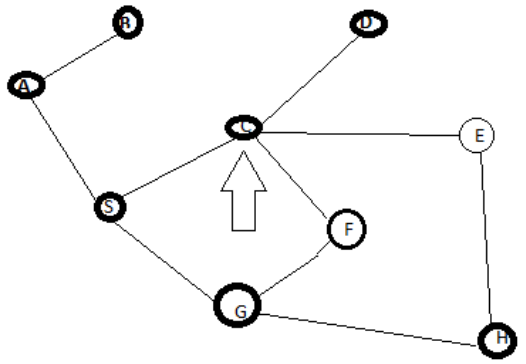
Step four



Now, S has C&G as adjecent nodes. so we writte them alphabetically in queue and writte them in the sequence of out put and marked them visited.

queue	out put
C G	A B S C G

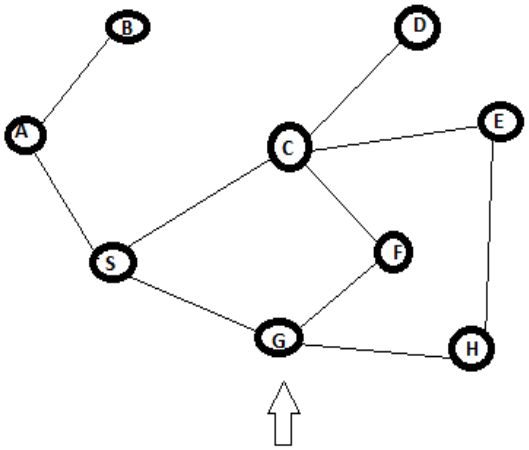
Step five



S has A C&S as adgecent note wich are all ready visited so we need to up date based on the first element of queue. so we have C.so we make the aray on C as currently working node and delet it from queue. now C has D,E&F as wich are note visited so we writte it alphabeticaly to queue and add them to the sequence of out put ,finally make them bold to show that they are visited.

queue	**out pu**
GDEF	**ABSCGDEF**

Step six



It is the time to up date the currently working node. we have G at the first sequence of the queue, so we move the pointer from C to G and remove it from the queue. G has only H which is adgecent and unvisited node. thus adde H to the sequence of queue and to the out put. finally mark it as visited. now we have D at the first element of queue, thus we must make it our currently working node and remove it from queue. but D has no adgecent node which is not visited. so go on in the same process finally we left with empty queue. thus we have done and the out put produced the below.

out pu	**queue**
ABSCGDEFH	**.....**

Depth first search algorithms

While BFS is effective for many problems but another algorithm called depth-first search of DFS for short, can be more natural in other problems such as topological sorting, cycle detection and finding connected components of a graph.

We can build a spanning tree for a connected simple graph using depth-first search. We will form a rooted tree, and the spanning tree will be the underlying undirected graph of this rooted tree.

Let x and y be two vertices in a rooted tree with root r. If x is on the path connecting r to y, we say that y is a descendant of x. (In particular, all vertices are descendants of r.)

If one of u and v is a descendant of the other, we say that {u, v} is a lineal pair.

A lineal spanning tree or depth-first spanning tree of a connected graph G = (V, E) is a rooted spanning tree of G such that each edge {u, v} of G is a lineal pair.

The running time for this algorithm is $o(v+e)o(v+e)$ for the graph with vertices vv and edge ee.

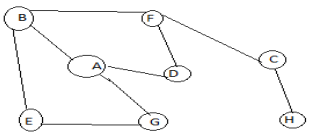
Algorithm of depth-first search

Let G be connected graph and v_0v_0 be a vertex of G. A depth search of G with initial vertex v_0v_0, is the following procedure for constructing a spanning tree for G.

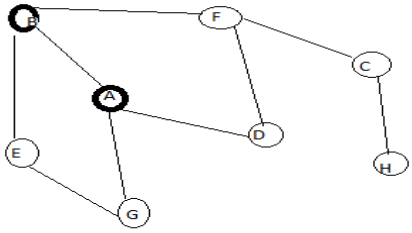
1. Set $w = v_0 w = v_0$ and $n = 0;\ w\, n = 0;\ w$ is called current center of the search. Let $T_0 T_0$ denotes the tree with no edges and vertex $v_0 v_0$; $T_0 T_0$ is the first tree in our sequence.

2. If possible, choose an edge $e_n e_n$ which incident with $w\, w$ and the new vertex $v_{n+1} v_{n+1}$. (By a new vertex, we mean a vertex which does not appear in $T_n T_n$, the current tree; in the other words, a vertex which has not yet been visited.) Adjoin $e_n e_n$ to the current tree $T_n T_n$ to form the next tree in the sequence, T_{n+1} T_{n+1}. Set $w = v_{n+1} w = v_{n+1}$ and increase nn into $n + 1 n + 1$.

3. Repeat step 2 until the current w is not adjacent to any other new vertex. If $T_n T_n$ is spanning tree, the search is completed. It not more precisely set $w = v_{n-1}\, w = v_{n-1}$ and increases nn to $n + 1 n + 1$. Repeat step 2(if possible; several back tracking may be required before a new vertex can be visited).



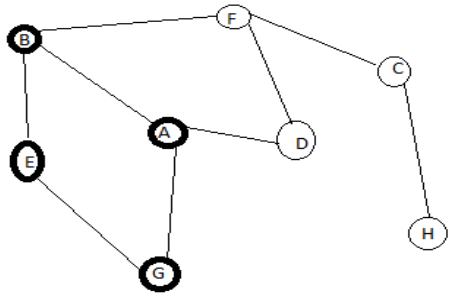
Example
Solution
Step one



First let us start with vertex A , make it bold to show that it is visited ant put it in both stack and result as below. now all vertices wich are adjecent to A are not visited. thus let as take B and put it as A. and make it visited

result	**stack**
AB	B
	A

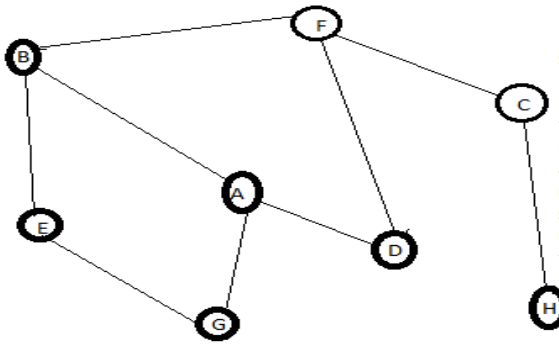
Step two.



Now we reapt the procedure,we look the node on the top of the stack wich is B.and E &F are the adgecent vertices to B. now take E and put it on the top of the stack and and it in the sequence of result, finally make it visited. again we look the top of stack and we gate E , then the only vertex that is not visited and adjecent to E is G. thus make it visited and put on the top of the stack as well as make it visited.

result	**stack**
ABEG	**G**
	E
	B
	A

Step three



Now the element on the top of stack is G. thus we find unvisited node adjecent to G but there rae no such vertex.so we remove ot from the top of the stack and back to E.also E has no unvisited adjecent vertex. again we remove E from the top of the stack. then we back to B, now we reapet the above procedure. finally we left with empty stack mean we have done and the below result is dispaleyed

result **stack**

ABEGFCHD **::**

Dijkstar's algorithms

Dijikstar's algorithm is introduced by the Dutch mathematician EdsgerDijkstar in 1959

It used for waited directed and undirected graph.

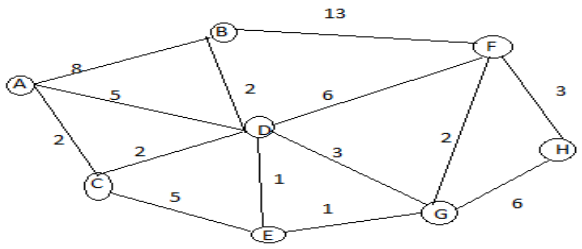
This algorithm follows the following steps.

1. Start with an initial node and goal node.
2. Find the least distance path to the goal node.
3. Assign to every node a tentative distance value, set it to zero for an initial node and infinity for all other nodes.
4. Keep a set of visited note. This startswith just the initial nodes.
5. For the current node consider all of its unvisited neighbor and calculate (distance to the current node) + (distance from current node to the neighbor).

If it is less than their current tentative distance, replace it with this new value.

6. When we are done considering all of theneighbor of the current node, mark the current node has remove it from un visited set.
7. If the destination node has been marked visited, the algorithm has finished.
8. Set the unvisited node marked with smallest tentative distance as the next current node and go back to step.

Example

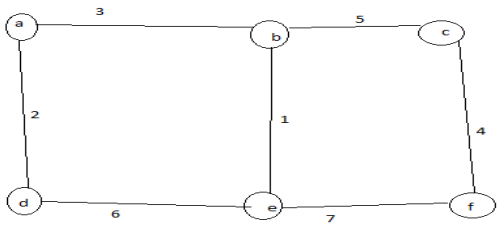


V	A	B	C	D	E	F	G	H
A	$.0_A.0_A.$	8_A	2_A	5_A	∞	∞	∞	∞
C		8_A	$.2_A2_A.$	4_C	7_C	∞	∞	∞
D		6_D		$.4_C.$	5_D	10_D	7_D	∞
E		6_D			$.5_D.5_D.$	10_D	6_E	∞
B		$.6_D.$				10_D	6_E	∞
G						8_G	$.6_E.$	12_G
F						$.8_G.$		11_F
H								$.11_F.$

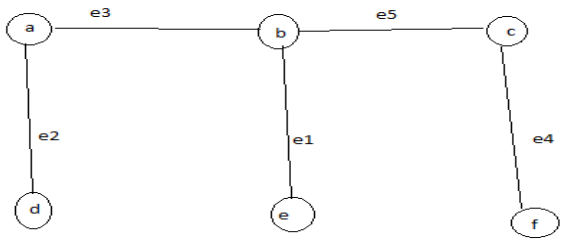
Kruskal's algorithms

- Step 1. Labels the edge of the graph G as $e_1,e_2,e_3,e_4,e_5,e_6,\ldots e_1,e_2,e_3,e_4,e_5,e_6,\cdots$ in order of increasing weighed.
- Step 2.choose edge e_1e_1 to begin the tree, and then choose the next edge in the sequence whose addition does not createcycle.
- Continue until no more edges must be added.

Example
Find the minimal spanning tree for the graph G below.



Solution
Step one
$e_1 = 1, e_2 = 2, e_3 = 3, e_4 = 4, e_5 = 5 e_1 = 1, e_2 = 2, e_3 = 3, e_4 = 4, e_5 = 5 e_6 = 6, e_7 = 7.$
$e_6 = 6, e_7 = 7.$
Step two



Now $e_7 and\ e_6 e_7 and\ e_6$ makes the cycle, so we ignore it. Thus its weight will be 15"

Primes algorithms

Prime algorithm is originally discovered by the Czech mathematician VojtˇechJarník in 1930, who described it in a paper in an obscure Czech journal. The algorithm became well known when it was rediscovered in 1957 by Robert Prim. Because of this, it is known as **Prim's algorithm** (and sometimes as the **Prim-Jarník algorithm**).

Begin by choosing any edge with smallest weight, putting it into the spanning tree. Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree, never forming a simple circuit with those edges already in the tree. Stop when $n - 1$ edge has been added.
Prim's Algorithm can be performed by the following steps

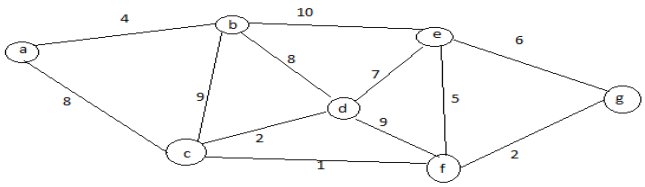
Step 0: Choose any element rr and set $s = \{r\}s = \{r\}$ and $A = \emptyset A = \emptyset$
(Take r as the root of our spanning tree.)

Step 1: Find a lightest edge such that one endpoint is in s and ss and the other is in $V\backslash SV\backslash S$. Add this edge to AA and its (other) endpoint to SS.

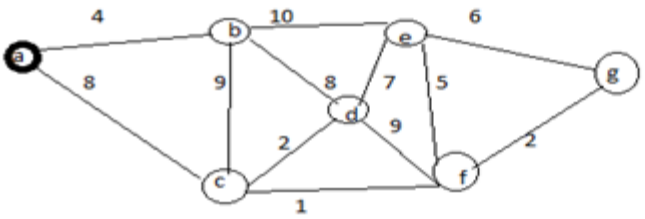
Step 2: If $V\backslash S = \emptyset\ V\backslash S = \emptyset$, then stop and output the minimum spanning tree $(S,A)(S,A)$.Otherwise go to Step1.

Note that the choice of an edge to add at a stage of the algorithm is not determined when there is more than one edge with the same weight that satisfies the appropriate criteria. We need to order the edges to make the choices deterministic. We will not worry about this in the remainder of the section. Also note that there may be more than one minimum spanning tree for a given connected weighted simple graph.

Example
Use prime algorithm to solve the below graph.

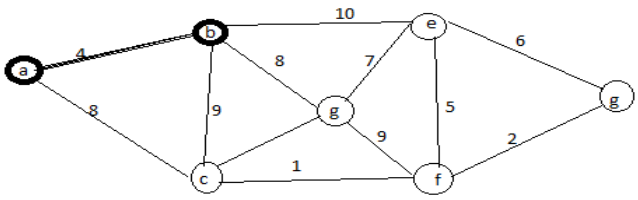


Solution
Step one.



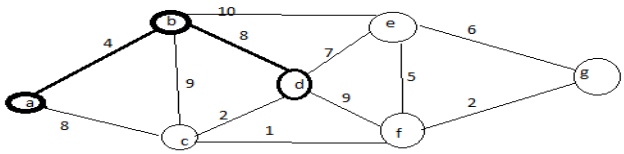
s={a}
A={}
v/s={b,c,d,e,f,g}
lightest edge={a,b}

Step two



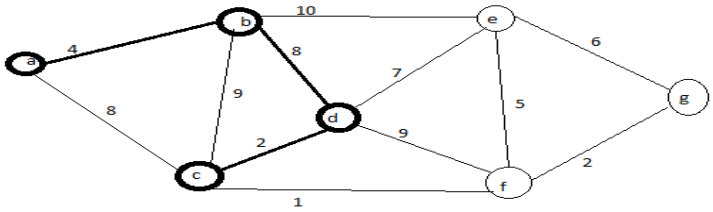
s={a,b}
v/s={c,e,f,g}
A={{a,b}}
lightest
edge={b,d},{a,c}

Step three



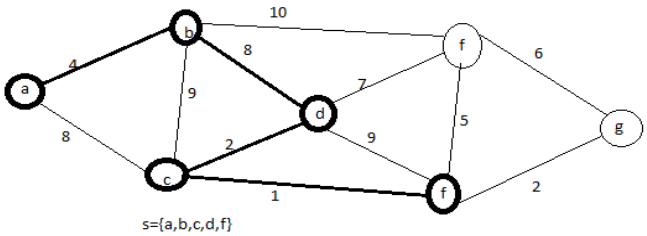
s={a,b,c}
v/s={c,e,f,9}
A={{a,b},{b,d}}
lightest edge{d,c}

Step four



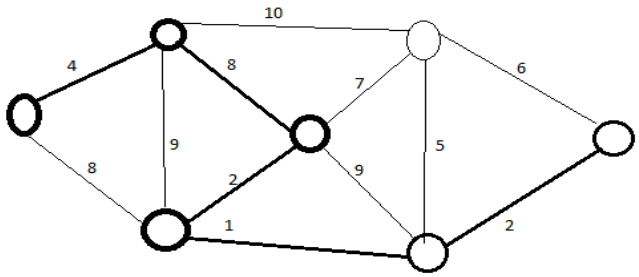
s={a,b,c,d}
v/s={e,f,g}
A={{a,b},{b,d},{c,d}}
lightest edge={c,f}

Step five



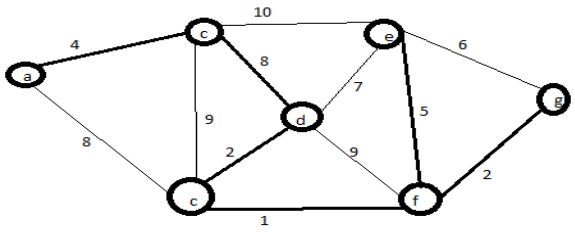
s={a,b,c,d,f}
v/s={e,g}
A={{a,b},{b,d},{c,d},{c,f}}
lightest edge={f,g}

Step six



s={a,b,c,d,f,g}
v/s={e}
A={{a,b},{b,d},{c,d},{c,f},{f,g}
lightest edge {f,e}

Step seven



s={a,b,c,d,e,f,g}
v/s={}
A={{a,b},{b,d},{c,d},{c,f
},{f,g},{f,e}}

completed

APPLICATION OF GRAPH SEARCH

Graphs are used in wide variety of model. Many problems can be modeled paths formed by traveling along the edge of graph. For instance, the problem of determining whether message can be sent between two computers using intermediate links can be studied with graph model. Problem of efficiently planning routes for mail delivery, garbage pickup, diagnostics in computers networks and so on can be solved using models that involves path in the graph.

They include, study of molecules, construction of bonds in chemistry and the study of atoms.

Similarly, graph theory is used in sociology for example to measure actor's prestige or to explore diffusion mechanisms.

Graph theory is used in biology and conservation efforts where a vertex represents regions where certain species exist and the edges represent migration path or movement between the regions. Let us see some application of graph in simple notion.

NICHE OVERLAP IN ECOLOGY

Graphs are used in many models involving the interaction of different species of animals.

For example the computation between species in ecosystem can be modeled using niche overlap graph. Each species is represented by vertex and undirected edge connects two vertices if the two species represented by these vertices compete (that is some of the food resources they uses are the same).

ACQUAINTANCESHIP GRAPHS.

Also we can use graph models to represent various relationship between people. For example we can use a graph to represent whether two people know each other, which is whether they are acquainted.

Each particular person in the group of people is represented by vertex. An undirected is used to connect two vertexes when these people know each other.

❖ The acquaintanceship graph of all people in the world has more than seven billion vertices and probably more than one trillion.

INFLUENCE GRAPH

In studies of group behavior it is observed that certain people can influence the thinking of the other. A directed graph called an influence graph can be used to model this behavior.

Each person of the group is represented by a vertex, there is directed edge from vertex a to vertex b, when the person represented by a influence the person represented by vertex b.

II. Conclusion

Graph theory enables us to study and model networks and solves some difficult problems inherently capable of being modelled using networks.

Various terms e.g. vertex and edge are associated with graph theory which gives these terms special meanings. These meanings need to be understood and remembered in order to apply graph theoretic approaches to solving problems.

When solving a problem by developing a graph-based program, careful attention must be given at the design stage to the structuring of data to help make solving the problem tractable, to enable linkages to be traced efficiently and to avoid duplication of data.

In computer science, graph traversal (also known as graph search) refers to the process of visiting (checking and/or updating) each vertex in a graph. Such traversals are classified by the order in which the vertices are visited. Tree traversal is a special case of graph traversal.

Reference

[1]. Rowan Garnier, John Taylor: Discreet mathematics
[2]. Narsingh Deo: Graph theory with application to engineering and computer science.
[3]. Keneth H.Rose: Discreet mathematics and its application.
[4]. A.Tamilarasi: Ddiscrete mathematics and its applications
[5]. Wikipedia: http://en.wikipedia.org/wiki/Shortest_path_problem
[6]. Robert Sedgewick. Algorithms in Java. Third Edition. ISBN 0-201-36121-3. Section 21.7:
[7]. Negative Edge Weights. http://safari.oreilly.com/0201361213/ch21lev1sec7

Belela Samuel Kotola. "Graph Search Algorithms and Its Application." *IOSR Journal of Mathematics (IOSR-JM)*, 16(1), (2020): pp. 35-47.