Available online at www.elixirpublishers.com (Elixir International Journal)

Applied Mathematics

Elixir Appl. Math. 76 (2014) 28684-28686

ISSN: 2229-712X

Algorithm for graph with different vertices

El-Zohny H, Radwan S and El- Morsy H
Department of Mathematics, Faculty of science, Alazhar University, Cairo, Egypt.

ARTICLE INFO

Article history:
Received: 29 May 2014;
Received in revised form:
10 November 2014;
Accepted: 20 November 2014;

Keywords
Graph,
Algorithm.

ABSTRACT
In this paper we will discuss the algorithm for new type of graph which vertices are different. The algorithm for each type of this graph also will be illustrated. Then we will illustrate different examples for each type.

Introduction

Definition of Graph: An (undirected) graph G is defined by two finite sets. a non-void set X of elements called vertices, a set E (which can be empty) of elements called edges, with for each edge e two associated vertices, x and y , distinct or not, called the end vertices of e [4].

Definition of Algorithm: In mathematics and computer science, an algorithm is an effective method expressed as a finite list of well-defined instructions for calculating a function .In simple words an algorithm is a step-by-step procedure for calculations.

Definition of Spanning tree :Spanning tree for a graph G is a subgraph of G that contains every vertex of G and is a tree [5].

Definition of link-graph : knot-graph (link-graph) is a graph in which each vertex is a link[1].

Definition of simple graph: A "simple" graph is a graph with no loops or multiple edges [6].

Definition of graph with graph vertices: Consider the geometric graph G(V, E) where
$V(G) = \{\{V^0{}_1,e^0{}_1\},\{v^1{}_1,e^1{}_1\}\ldots\ldots\{v^n{}_1, e^n{}_1\},\{V^0{}_2, e^0{}_2\}, \{V^1{}_2, e^1{}_2\}\ldots\ldots\{V^n{}_2, e^n{}_2\}, \{V^0{}_n, e^0{}_n\},$
$\{V^1{}_n, e^1{}_n\}\ldots\ldots\{V^n{}_m, e^n{}_m\}$ and $E(G) =\{e^1, e^2, e^3,\ldots\ldots e^n\}$. We are called this graph (graph with complex vertices)[2].

Definition of graph with different vertices : The graph with different vertices is a graph which each vertex has different chap such as: link vertex , graph vertex , or simple vertex , whether this graph is simple or have multiedges (multigraph) [3].

1. Algorithm for Graph which its vertices has all three chaps (simple, knot or graph) :

Input:
Connected graph with different vertices V_n , V_{nm} such that $0 \le n \le k$, $0 \le m \le k$, k integer ≥ 1.

Algorithm body:
Crete a subgraph that visit V_n , V_{nm} proceeding from vertex to vertex along internal spanning tree T (V_{nm} not simple vertices) then along outer spanning tree $T^{\backslash}$.

```
1.  Initialized T , $T^{\backslash}$ to have all vertices and no edge.
2.  Visit $V_n$, $0 \le n \le k$ , k integer $\ge 1$.
If $V_n$ simple then,
2a. Attach the edges $\{e_n\}$
Return to step 2.
Output T .
Else
3.  Visit $V_n$ , $V_{nm}$ , $0 \le n,m \le k$.
If $V_n$ knot vertices then ,
3a. Visit outer vertices $V_n$ then internal vertices $V_{nm}$ .
3b. Attach the internal edges $\{e_{nm}\}$.
3c. Attach the outer edges $\{ e^{\backslash}{}_{nm} \}$.
Return to step 3.
Output $T^{\backslash}$ , $T^{\backslash\backslash}$ .
Else
4.  Visit $V^n$ , $V^{nm}$ .
If $V^n$ is graph then ,
4a. Visit outer vertices $V_n$ then internal vertices$V^n$ ,$V^{nm}$ .
4b. Attach the internal edges $\{ V^{n1m1}, V^{nimi} \}$ to $T^{\backslash\backslash\backslash}$ and visit $V^{n1mk}$.
4c. Assign $V^{n1m1}$ to $V^{nimk}$.
Return to step 4.
Output $T^{\backslash\backslash\backslash}$ .
End
Output T , $T^{\backslash}$ , $T^{\backslash\backslash}$ , $T^{\backslash\backslash\backslash}$.
End algorithm.
```

Example 1: For a graph shown in Fig.(1) , which has different vertices (simple, knot ,graph) we can compute the algorithm to find the spanning tree as follows:

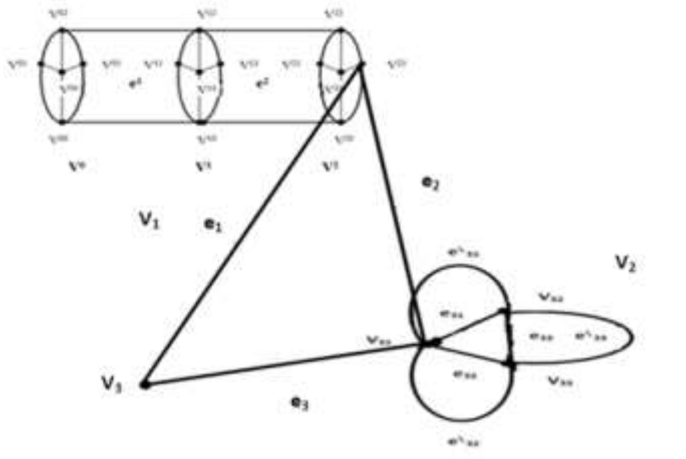

Fig 1.

Input:
Connected graph with different vertices V_n , V_{nm} such that $0 \le n \le k$, $0 \le m \le k$, k integer ≥ 1.

Tele:
E-mail addresses: hendelmorsy@yahoo.com

Algorithm body:

Crete a subgraph that visit V_n , V_{nm} proceeding from vertex to vertex along internal spanning tree T (V_{nm} not simple vertices) then along outer spanning tree $T^{\backslash}$.

1. Initialized T , $T^{\backslash}$ to have all vertices and no edge.
2. Visit V_4 then

2a. Attach e_1 to T.

Return to step 2.

Output T .

Else

3. Visit V_3 then internal vertices V_{31} , v_{33} .

3a. Attach the internal edges {e_{31} , $e_{32}e_{33}$ }.

3b. Attach the outer edges { $e^{\backslash}_{31}$, $e^{\backslash}_{32}$, $e^{\backslash}_{33}$ }.

Return to step 3.

Output $T^{\backslash}$, $T^{\backslash\backslash}$.

Else

4. Visit V_1 then V^0 , then

4a. Visit internal vertices V^{01} to V^{03}.

4b. Attach the internal edges { V^{o1}, V^{04} }, {V^{o0}, V^{04} } ,{V^{o2}, V^{04}},{V^{o3}, V^{04} }to $T^{\backslash\backslash\backslash}$.

4c. Attach the outer edge e^1 to $T^{\backslash\backslash\backslash\backslash}$ and visit V^1.

Return to step 4.

Output $T^{\backslash\backslash\backslash}$, $T^{\backslash\backslash\backslash\backslash}$.

End

Output T , $T^{\backslash}$, $T^{\backslash\backslash}$, $T^{\backslash\backslash\backslash}$,$T^{\backslash\backslash\backslash\backslash}$.

End algorithm.

Then the tree will be:

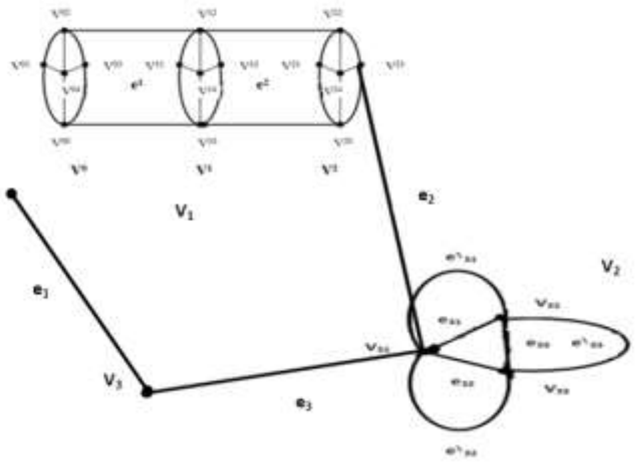

Fig 2

Algorithm for Graph which its vertices has only two chaps:

The algorithm when its vertices are simple and knot:

For a graph shown in Fig.(3) which its vertices are simple and knot the algorithm will be:

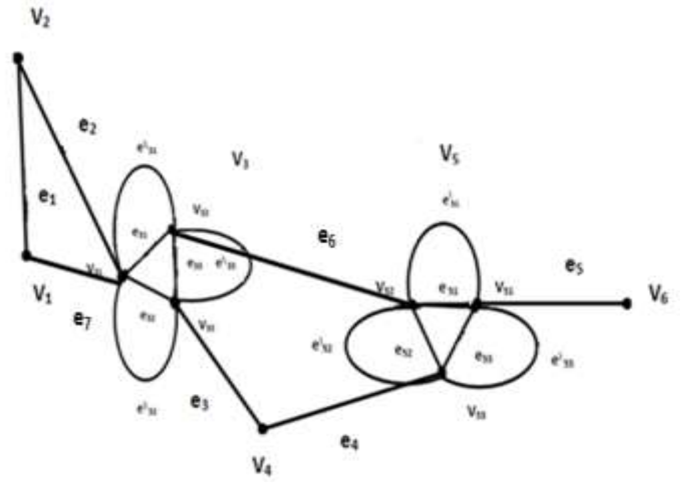

Fig 3

Input:

Connected graph with different vertices V_n , V_{nm}such that $0 \le n \le k$, $0 \le m \le k$, k integer ≥ 1.

Algorithm body:

Crete a subgraph that visit V_n , V_{nm} proceeding from vertex to vertex along internal spanning tree T (V_{nm} not simple vertices) then along outer spanning tree $T^{\backslash}$.

1. Initialized T , $T^{\backslash}$ to have all vertices and no edge.
2. Visit V_1 then

2a. Attach e_1 to T.

Return to step 2.

Output T .

Else

3. Visit V_3 then

3a. Attach the internal edges {e_{31} , $e_{32}e_{33}$ }.

3b. Attach the outer edges { $e^{\backslash}_{31}$, $e^{\backslash}_{32}$, $e^{\backslash}_{33}$ }.

Return to step 3.

Output $T^{\backslash}$.

End

Output T , $T^{\backslash}$.

End algorithm.

Then the final spanning tree will be:

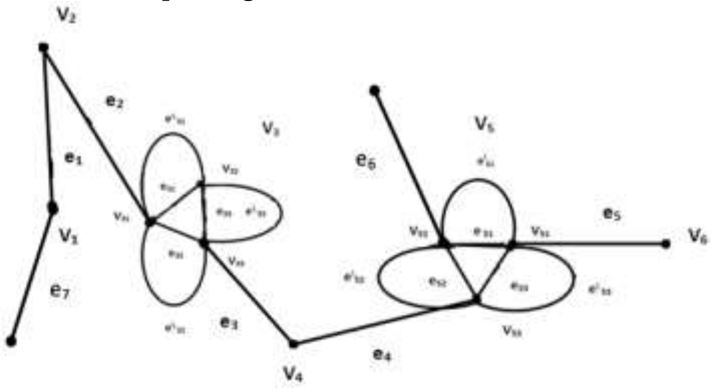

Fig 4

The algorithm when its vertices are simple and graph:

For a graph shown in Fig.(5) which its vertices are simple and graph the algorithm will be:

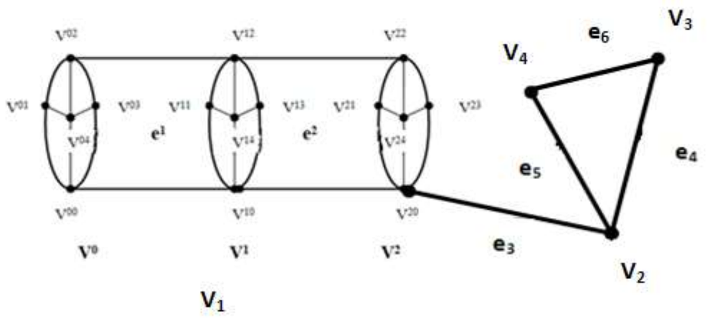

Fig 5

Input:

Connected graph with different vertices V_n , V_{nm}such that $0 \le n \le k$, $0 \le m \le k$, k integer ≥ 1.

Algorithm body:

Crete a subgraph that visit V_n , V_{nm} proceeding from vertex to vertex along internal spanning tree T (V_{nm} not simple vertices) then along outer spanning tree $T^{\backslash}$.

1. Initialized T , $T^{\backslash}$ to have all vertices and no edge.
2. Visit V_4 then

2a. Attach e_4 to T.

Return to step 2.

Output T.

Else

3. Visit V_1 then V^0, then
4a. Visit internal vertices V^{01} to V^{03}.
4b. Attach the internal edges { V^{o1}, V^{04} }, {V^{o0}, V^{04} } ,{V^{o2}, V^{04}},{V^{o3}, V^{04} }to $T^{\backslash}$.
4c. Attach the outer edge e^1 to $T^{\backslash\backslash}$ and visit V^1.
Return to step 4.
Output $T^{\backslash}$,$T^{\backslash\backslash}$.
End
Output T , $T^{\backslash}$, $T^{\backslash\backslash}$.
End algorithm
The spanning tree will be:

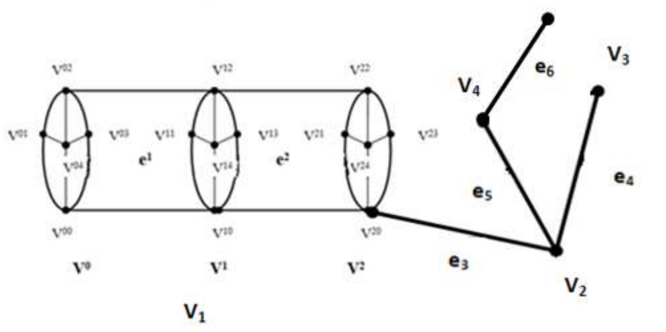

Fig 6

The algorithm when its vertices are knot and graph:
For a graph shown in Fig.(7) which its vertices are knot and graph the algorithm will be:

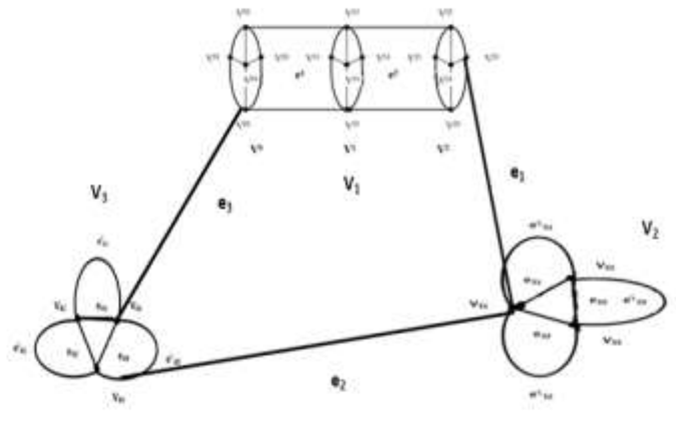

Fig 7

Input:
Connected graph with different vertices V_n , V_{nm}such that $0 \le n \le k$, $0 \le m \le k$, k integer ≥ 1.
Algorithm body:
Crete a subgraph that visit V_n , V_{nm} proceeding from vertex to vertex along internal spanning tree T (V_{nm} not simple vertices) then along outer spanning tree $T^{\backslash}$.

1. Visit V_1 then V^0, then
4a. Visit internal vertices V^{01} to V^{03}.
4b. Attach the internal edges { V^{o1}, V^{04} }, {V^{o0}, V^{04} } ,{V^{o2}, V^{04}},{V^{o3}, V^{04} }to T.
4c. Attach the outer edge e^1 to $T^{\backslash}$ and visit V^1.
Return to step 4.
OutOutput T,$T^{\backslash}$.
Else
1. Visit V_3 then
3a. Attach the internal edges {e_{31} , $e_{32}e_{33}$ }.
3b. Attach the outer edges { $e^{\backslash}_{31}$, $e^{\backslash}_{32}$, $e^{\backslash}_{33}$ }.
Return to step 3.
Output $T^{\backslash\backslash}$, $T^{\backslash\backslash\backslash}$.
End
OutputT,$T^{\backslash}$, $T^{\backslash\backslash}$,$T^{\backslash\backslash\backslash}$.
End algorithm.
Then the spanning tree will be:

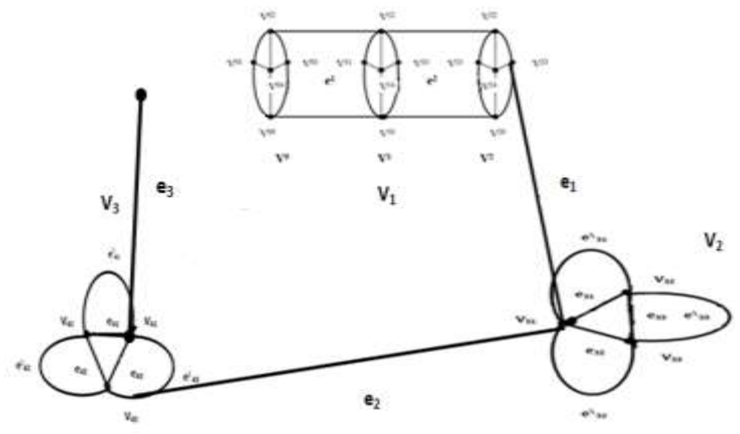

Fig 8

References:

[1]El-Ghoul M; F. Salama ; A. S. Pelila . :On new type of graph**.** Tanta University ,Tanta,Egypt,2012**.**
[2] El-Ghoul,M ; El-Zohny,H; Khalil,M,M.: New types of graphs with graphs vertices. Al-Azhar University ,Cairo, Egypt,2011.
[3]MabrukElghoul , HabibaElzohny , HendElmorsy : New type of graph with different vertices . International journal of mathematical archive , vol.4 , No.1 ,(2013) 1-9 .
[4] Fournier,Jean-Claude, Graph Theory and applications with Exercises and problems, ISTE Ltd,2009.
[5] Susanna S.Epp, Discrete Mathematics With Application, Third Edition, Thomson Learning, Inc , 2004**.**
[6] Wilson R.J.: Introduction to graph theory. Oliver and Boyed, Edinburgh 1972.