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

Applied Mathematics

Elixir Appl. Math. 39 (2011) 4948-4950

ISSN: 2229-712X

New algorithm for graph with graphs vertices

EL-Zohny.H, Salam. R and EL-Morsy.H

Department of Mathematics, Faculty of Science Al-Azahar University, Cairo, Egypt.

ARTICLE INFO

Article history:

Received: 29 July 2011;

Received in revised form:

22 September 2011;

Accepted: 29 September 2011;

Keywords

Algorithm ,

Graphs vertices.

ABSTRACT

In this paper we will compute a new algorithm for new graph which its vertex is a graph.

© 2011 Elixir All rights reserved.

Introduction

Definition and Background

Definition 1:

Abstract graphs: An abstract graphs G is a diagram consisting of a finite non empty set of the elements, called "vertices" denoted by V(G) together with a set of unordered pairs of these elements, called "edges" denoted by E(G). The set of vertices of the graph G is called "the vertex .set of G " and the list of edges is called "the edge .list of G " [2,3].

Definition 2:

Consider a geometric graph G(V,E) where $G(V)=\{\{v_0,e_0\},\{v_1,e_1\},\{v_2,e_2\},\ldots\ldots.\{v_n,e_n\}\}$ and $E(G) = \{e^1\}$, we are called this graph(graph with complex vertices).

To represent these graphs we must show that there are three types of these graphs:

1- Null graphs which their vertices are graphs.

2- Graphs which their vertices are similar. [1].

Definition 3:

Null graphs which their vertices are graphs:

We know that a null graph is a graph containing no edges and every vertex is isolated. By definition(1) we can define a new null graphs (which their vertex are graphs), consider the graph $G_n(V^{\square})$ such that $V^{\square}(G) = \{\{v_0, e_0\}, \{v_1, e_1\}, \{v_2, e_2\}, \ldots\ldots\ldots\ldots\{v_n, e_n\}\}$ and $E(G) = \Phi$. See Fig.(1) [1].

Definition 4:

Graphs which their vertices are similar: Consider the geometric graph G(V , E) where $V(G) = \{V^0, V^1\}$ where $V^{\square 0} = \{\{v_0, e_0\}, \{v_1, e_1\}, \{v_2, e_2\}, \ldots\ldots\ldots\ldots\{v_n, e_n\}\}$ and $V^1 == \{\{v_0, e_0\}, \{v_1, e_1\}, \{v_2, e_2\}, \ldots\ldots\ldots\ldots\{v_n, e_n\}\}$

where $V^{0\,\square\square}$ is the same as V^1 to represent this graph we will take the smallest cir-cle which contains vertices of V^0 and the smallest circle which contains vertices of $V^{\square\square}$ and connected between them this connecte is $E(G) = \{e^1\}$.[1].

Definition 5:

Spanning tree for a graph G is a subgraph of G that contains every vertex of G and is a tree[4].

Main results:

We will discuss two new algorithms for graph with graphs vertices.

I. algorithm for null graph which vertices is a graph:

uter

Create a subgraph that visit each outer vertices V" then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree T of that graph.

1. initialized T to have all vertices of G "which have outer vertices".

2. select the smallest superscript k for, $1 \le k \le i$, $0 \le n \le j$. $V^{n\ mk}$ has not already been visited.

If no superscript is found , then,

Go to step 3 , otherwise,

Perform the following:

2a. attach the internal edge $\{V^{1m1}, V^{n2m2}\}$ to T, and visit V^{1mk}.

2b. assign V^{1mk} to V^{nmk} and ,

2c. return to step 2.

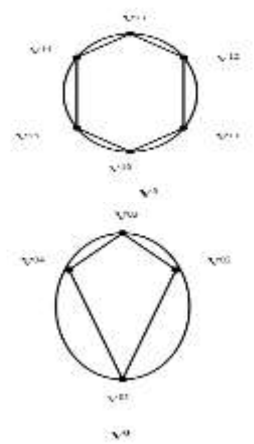
End while.

3. output T .

end algorithm.

Example 1:

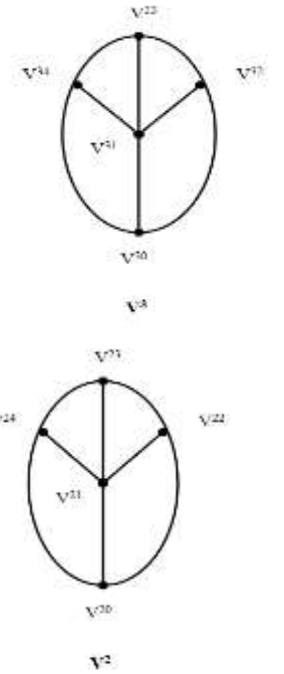
For a null graph shown in fig(1) we can compute its algorithm as follows:



Tele:

E-mail addresses: elzohny_7@yahoo.com, randa_salam@yahoo.com, hendelmorsy@yahoo.com

© 2011 Elixir All rights reserved



Fig(1)

Input:

Null graph with graphs vertices V^{nm} , $0\le n \le 3$, $0\le m \le 5$.

Algorithm body:

Create a subgraph that visit each outer vertices V^n then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree of that graph.

1. initialized T to have all vertices of G "which have outer vertices".
2. select V^0 and visit all internal vertices V^{01} to V^{03}.

2a. attach the internal edges { V^{01},V^{02}}……..{V^{04},V^{01}} to T.

2b . go to step 2 for the other vertices V^2 to V^4.

End while.

3. output T.

end algorithm.

Algorithm for graph which vertices is graph:

Input:

Connected graph G(V , E), V(G) = {V^0,V^1}, V^0 = {{V^0,e^0},……{v^n,e^n}}, V^1={{v^0,e^0},{v^1,e^1},…..{v^n,e^n}}.

Algorithm body:

Create a subgraph that visit each outer vertices V^n then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree T of that graph , then along it's outer spanning tree T'.

1. initialized *T* to have all the vertices of *G* and no edge.
2. let *E* the set of all edges of *G* , $m = 0$.
3. while ($m \le n-1$)

3a. visit outer vertices V^n , then visit internal vertices V^{nmi}.

3b. attach the internal edge { V^{1m1}, V^{n2m2}} to *T*, and visit $V^{1m\mathrm{k}}$.

3c. attach the outer edge{ V^1,V^2}to *T'* and visit V^{nmk},

3d. return to step 3.

End while.

4. output *T,T'*.

end algorithm.

Example 2:

Consider a graph shown in fig(2).

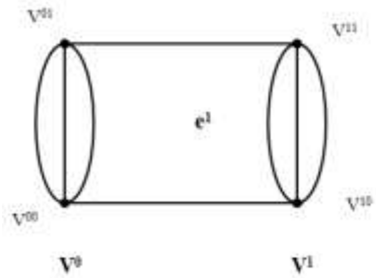


Fig (2)

Input:

Connected graph G(V , E), V(G) = {V^0,V^1}, V^0 = {{V^{00},V^{01}}}, V^1={{v^{10},V^{11}}}.

Algorithm body:

Create a subgraph that visit each outer vertices V^n then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree *T* of that graph, then along it's outer spanning tree T'.

1. initialized *T* to have vertex v^0.
2. let *E* the set of all edges of *G* , $m = 0$.
3. while ($m \le 1$).

3a. visit outer vertex V^0
then visit V^{00},V^{01}.

3b. attach the internal edge {V^{00},V^{01}} to *T*.

3c. attach the outer edge e^1 to T' and visit V^1.

3d. return to step 3.

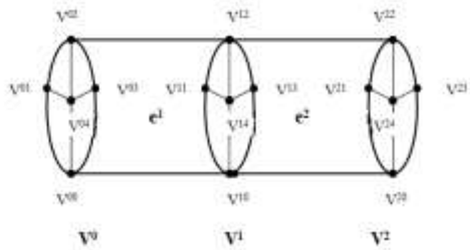
End while.

4. output *T,T'*.

end algorithm.

Example 3:

For a graph shown in fig(3).we have:



Fig(3)

Input:

Connected graph $G(V , E)$, $V(G) = \{V^0,V^1,V^2\}$, V^0 = {{V^{00},V^{04}},{V^{04},V^{03}},{V^{04},V^{02}},{V^{04},V^{01}}},
V^1={{v^{10},V^{14}},{V^{14},V^{13}},{V^{14},V^{12}},{V^{14},V^{11}}}.
V^2 = {{V^{20},V^{24}},{V^{24},V^{23}},{V^{24},V^{22}},{V^{24},V^{21}}}.

Algorithm body:

Create a subgraph that visit each outer vertices V^n then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree *T* of that graph, then along it's outer spanning tree T'.

1. initialized *T* to have vertex v^0.
2. let *E* the set of all edges of *G* , $m = 0$.
3. while ($m \le 2$).

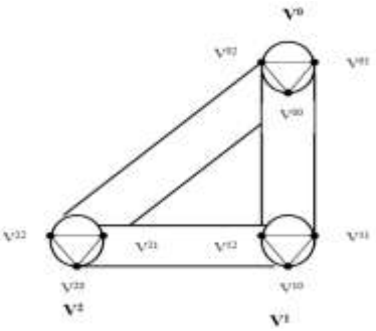
3a. visit outer vertex V^0
then visit V^{00},V^{04},V^{03},V^{01}.

3b. attach internal vertices {V^{00},V^{04}},{V^{04},V^{03}},{V^{04},V^{02}},{V^{04},V^{01}} to *T.*

3c. attach the outer edge e^1 to T and visit V^1.
3d. return to step 3 for the other vertices.
End while.
4. output T,T'.
end algorithm.

Example 4:

For a graph shown in fig(4).



Fig(4)

Input:

Connected graph G(V, E), $V(G) = \{V^0,V^1,V^2\}$, $V^0 = \{\{V^{00},V^{01}\},\{V^{00},V^{02}\},\{V^{02},V^{01}\}\}$,
$V^1=\{\{v^{10},V^{11}\},\{V^{11},V^{12}\},\{V^{12},V^{10}\}\}$.
$V^2 = \{\{V^{20},V^{21}\},\{V^{21},V^{22}\},\{V^{20},V^{22}\}\}$.

Algorithm body:

Create a subgraph that visit each outer vertices V^n then its internal vertices $V^{nm}{}_i$

Proceeding from vertex to vertex but moving along internal spanning tree T of that graph, then along it's outer spanning tree T'.

1. initialized T to have vertex v^0.
2. let E the set of all edges of G , $m = 0$.
3. while $(m \leq 2)$.
3a. visit outer vertex V^0.
then visit V^{00},V^{01},V^{02}.
3b. attach internal vertices $\{V^{00},V^{01}\},\{V^{01},V^{02}\},\{V^{00},V^{02}\}$ to T.
3c. attach the outer edge e^1 to T' and visit V^1.
3d. return to step 3 for the other vertices.
End while.
4. output T,T'.
end algorithm.

References:

[1] El-Ghoul,M; El-Zohny,H; Khalil,M,M.: New types of graphs with graphs vertices. Al-Azhar University ,Cairo,Egypt,2011.
[2] Gibbons, A.: Algorithmic graph theory. Cambridge University Press, Cam-bridge, UK,1995.
[3] Giblin, P.J.: Graphs, surfaces and homology, an introduction to algebraic topology. Chapman and Hall . Ltd, London 1977.
[4] Susanna S.Epp, Discrete Mathematics With Application, Third Edition, Thomson Learning, Inc, 2004.