1.Breadth First Search

Source file for the function : igraph-0.7.1/src/visitors.c
Function modified : igraph_bfs

Source file for the driver function :igraph-0.7.1/examples/simple/igraph_bfs2.c

Random version

Name of the file submitted :bfs_ranfrq.c

Summary : Prints the BFS traversal for 10 calls in a tree as specified by the user. It also prints the number of steps to reach a particular node from the root node in every call.Finally, it prints the average path length of the node from the root.

Details :

Function igraph_bfsr( ) :

- The vector 'neis' contains the neighbours of the current operational node. When the flag variable is set, the contents of the vector are shuffled using 'igraph_vector_shuffle' to randomize breadth first search traversal.

Function main( ):

- The total number of nodes and the number of children per node is queried and the tree is constructed accordingly.
- The function 'igraph_bfsr' is called as many times as the number of nodes considering every vertex as the root node once . The minimum distance of every node from the current root is calculated and stored in the matrix 'mat[ ][ ]' .
- The flag is set so that the vectoe 'neis' is shuffled for all following calls to the igraph_bfsr function.(hence,randomizing the results).
- The node for which the average path length is to be computed is queried.
- The order for every call is printed and the path length for the given node for the current order is computed.
- The distances for every call and average path length of the node is printed.

Deterministic version

Name of the file submitted :bfs_deterministic.c

Summary : Prints the BFS traversal for 10 calls in a tree as specified by the user. It also prints the number of steps to reach a particular node from the root node in every call.Finally, it prints the average path length of the node from the root.

Details :

Function igraph_bfsd( ) :

- The vector 'neis' contains the neighbours of the current operational node. The contents of the vector are sorted using 'igraph_vector_sort' to generate the deterministic version of breadth first search traversal.

Function main( ):

- The deterministic version of the function -'igraph_bfsd' is called and the traversal for different cases is printed as specified in the driver function.

2.Depth First Search

Source file for the function	: igraph-0.7.1/src/visitors.c
Function modified	: igraph_dfs
Source file for the driver function	: igraph-0.7.1/examples/simple/igraph_bfs2.c

Random Version

Name of the file submitted : dfs_ranfrq.c

Summary : Prints the DFS traversal for 10 calls in a tree as specified by the user. It also prints the number of steps to reach a particular node from the root node in every call.Finally, it prints the average path length of the node from the root.

Details :

Function igraph_dfsr() :

- The vector 'neis' contains the neighbours of the current operational node.
- The vector 'visited' is used to mark the visited nodes. When the flag variable is set and when the operational node is not already visited, the contents of the vector is shuffled using 'igraph_vector_shuffle' to randomize depth first search traversal.
  This vector 'visited' is needed to ensure that the vector 'neis' is shuffled only once for a particular set of nodes. If we do not use this vector then 'neis' with the same contents is shuffled again and again and hence, some nodes are unable to enter the stack.

Function main():

- The total number of nodes and the number of children per node is queried and the tree is constructed accordingly.
- The function 'igraph_dfsr' is called as many times as the number of nodes considering every vertex as the root node once . The minimum distance of every node from the current root is calculated and stored in the matrix 'mat[ ][ ]' .
- The flag is set so that the vectoe 'neis' is shuffled for all following calls to the igraph_dfsr function.(hence,randomizing the results).
- The node for which the average path length is to be computed is queried.
- The order for every call is printed and the path length for the given node for the current order is computed.
- The distances for every call and average path length of the node is printed.

Deterministic version

Name of the file submitted :dfs_deterministic.c

Summary : Prints the DFS traversal for 10 calls in a tree as specified by the user. It also prints the number of steps to reach a particular node from the root node in every call.Finally, it prints the average path length of the node from the root.

Details :

Function igraph_dfsd() :

- The vector 'neis' contains the neighbours of the current operational node.
- The vector 'visited' is used to mark the visited nodes. When the operational node is not already visited, the contents of the vector is shuffled using 'igraph_vector_shuffle' to generate the deterministic version of depth first search traversal.
  This vector 'visited' is needed to ensure that the vector 'neis' is shuffled only once for a particular set of nodes. If we do not use this vector then 'neis' with the same contents is shuffled again and again and hence, some nodes are unable to enter the stack.

Function main():

- The deterministic version ofthe function-'igraph_dfsd' is called and the traversal for different cases is printed as specified in the driver function.

3.Dijkstra's Algorithm

Source file for the function	: igraph-0.7.1/src/structural_properties.c
Function modified	: igraph_get_shortest_paths_dijkstra()

Source file for the driver function :
igraph-.7.1/examples/simple/igraph_get_shortest_paths_dijkstra.c

Random Version

Name of the file submitted : random.c

Summary : Prints randomly the shortest weighted paths for selected nodes(1,3,5,4,6) in a ring and a tree as specified by the user.

Details :

Function igraph_get_shortest_paths_dijkstra() :

- We genrate 10 random numbers between 1 and 100 and store it in an array named ar.
- The altdist stores the distance for a different possibility of traversal for a given node.
  We add the case where altdist is equal to the current distance (curdist) we have for that is the case where randomness arises. In that case, we choose the altenative path only if the random number at any index is greater than 50 or else we choose the previous path only .

Function main():

- The random version of the function-'igraph_get_shortest_paths_dijkstra' is called and the shortest weighted paths for different nodes in different graphs is printed as specified in the driver function.

Deterministic version

Name of the file submitted :deterministic.c

Summary : Prints the specific shortest weighted paths for selected nodes(1,3,5,4,6) in a ring and a tree as specified by the user.

Details :

Function igraph_get_shortest_paths_dijkstra() :

- The altdist stores the distance for a different possibility of traversal for a given node. We add the case where altdist is equal to the current distance (curdist) we have for that is the case where choice arises. In that case, we choose the path whose value of the parent nodes are smaller .

Function main():

- The deterministic version of the function-'igraph_get_shortest_paths_dijkstra' is called and the shortest weighted paths for different nodes in different graphs is printed as specified in the driver function.

4.Topological sorting

Source file for the function :igraph-0.7.1/src/structural_properties.c
Function modified :igraph_topological_sorting

Source file for the driver function :igraph-0.7.1examples/simple/igraph_topological_sorting.c

Random version

Name of the file submitted :sort.c

Summary: Prints a topological sorting of the nodes of the graph specified in the driver function.

Details:

Function igraph_topological_sorting():

- The vector 'temp' contains all the nodes which have no incoming vertices at present. This vector is shuffled using the function 'igraph_vector_shuffle' to randomize the result.

Function main():

- The function 'igraph_topological_sorting' is called which prints the randomized version of the topological sorting of the nodes of the graph generated using the 'igraph_small' function.

Deterministic version

Name of the file submitted :sortdet.c

Summary: Prints a topological sorting of the nodes of the graph specified in the driver function.

Details:

Function igraph_topological_sorting():

- The vector 'temp' contains all the nodes which have no incoming vertices at present. This vector is sorted using the function 'igraph_vector_sort' to obtain the deterministic version of the function. Hence, this always gives a constant result.

Function main():

- The function 'igraph_topological_sorting' is called which prints the deterministic version of the topological sorting of the nodes of the graph generated using the 'igraph_small' function.

<u>5.Minimum Spanning tree(unweighted)</u>

Source file for the function : igraph-0.7.1/src/spanning_trees.c
Function modified : igraph_i_minimum_spanning_tree_unweighted()

Source file for the driver function : igraph-0.7.1/examples/simple/igraph_minimum_spanning_tree.c

<u>Random Version</u>

Name of the file submitted : mstran.c

<u>Summary :</u> Prints randomly the minimum spanning tree of an unweighted graph.

<u>Details :</u>

Function igraph_i_minimum_spanning_tree_unweighted() :

- The vector 'tmp' contains the incident edges of the current operational node which was last dequeued . The contents of the vector are shuffled using 'igraph_vector_shuffle' to randomize Minimum spanning tree formation.

Function main():

- The random version of the function-'igraph__i_minimum_spanning_tree_unweighted()' is called and the varying minimum spanning trees of a graph specified in the driver function is printed.

<u>Deterministic version</u>

Name of the file submitted : mstdet.c

<u>Summary :</u> Prints specifically the minimum spanning tree of an unweighted graph..

<u>Details :</u>
Function igraph_i_minimum_spanning_tree_unweighted() :

- The vector 'tmp' contains the incident edges of the current operational node which was last dequeued . The contents of the vector are sorted using 'igraph_vector_shuffle' to genrate deterministic version of Minimum spanning tree formation.

Function main():

- The deterministic version of the function-'igraph__i_minimum_spanning_tree_unweighted()' is called and the varying minimum spanning trees of a graph specified in the driver function is printed.

<u>6.Minimum spanning tree(weighted)</u>

Source file for the function :igraph-0.7.1/src/spanning_tree.c
Function modified :igraph_i_minimum_spanning_tree_prim

Source file for the driver function :

igraph-0.7.1/examples/simple/igraph_minimum_spanning_tree.c

Random version:

Name of the file submitted :primrandom.c

Summary: Prints randomly the minimum spanning tree of a weighted graph.

Details:

Function igraph_i_minimum_spanning_tree_prim():

- The array 'ar' is used to store the frequency of the weights of the edges. The index specifies the weight of a particular edge and the content at that index is the frequency of that edge weight.
- The overflow condition is checked for. If any edge weight exceeds the limit i.e. 100(can be changed later as per the user's requirement) the overflow condition is satisfied, a suitable message is displayed and the program is aborted.
- The array 'ra' is used to store 10 random numbers.
- The maximum frequency is determined. All the edge weights are shifted by the number of bits in the maximum frequency. A random value which is less than the maximum frequency is added to all the edge weights. This ensures that all the egdes having similar weights are randomized.
- Another overflow condition is checked for. If any of the weights after shifting exceeds 32 bits, an error message is printed and the program is aborted.
- This function also gives different results depending on the node with which the generation of minimum spanning tree is started, hence, this node is randomly chosen.

Function main():

- This function calls the 'igraph_i_minimum_spanning_tree_prim' and randomly prints the minimum spanning tree of the weighted graph specified in the driver function.

Deterministic version:

Name of the file submitted :primdet.c

Summary: Prints the deterministic version of the minimum spanning tree of the specified graph.

Details:

Function igraph_i_minimum_spanning_tree_prim():

- The nodes of the specified graph are accessed in a particular order. This makes the approach deterministic.

Function main():

- This function calls the 'igraph_i_minimum_spanning_tree_prim' and prints the deterministic version of minimum spanning tree of the weighted graph specified in the driver function.

7.BGLL Algorithm

Source file for the function : igraph-0.7.1/src/community.c

Function modified : igraph_i__community_multilevel_step()

Source file for the driver function : igraph_0.7.1/examples/simple/igraph_community_multilevel.c

Random version

Name of the file submitted :bgll.c

Summary : This function implements the multi-level modularity optimization algorithm for finding community structure of a graph randomly.

Details :

Function igraph_i_community_multilevel_step( ) :

- The vector 'links_community' contains the commnities incident on the vertex.. The contents of the vector are shuffled using 'igraph_vector_shuffle' to randomize multilevel community detection in a graph.

Function main( ):

- Queries and prints the multilevel communities of the graphs as specified in the driver function by calling the function 'igraph_community_multilevel()' .

Deterministic version

Name of the file submitted :bgldet.c

Summary : This function implements the multi-level modularity optimization algorithm for finding community structure in a deterministic approach.

Details :

Function igraph_i_community_multilevel_step( ) :

- The vector 'links_community' contains the commnities incident on the vertex.. The contents of the vector are sorted using 'igraph_vector_sort' to give the deterministic approach to multilevel community detection in a graph.

Function main( ):

- .Queries and prints the multilevel communities of the graphs as specified in the driver function by calling the function 'igraph_community_multilevel()' .