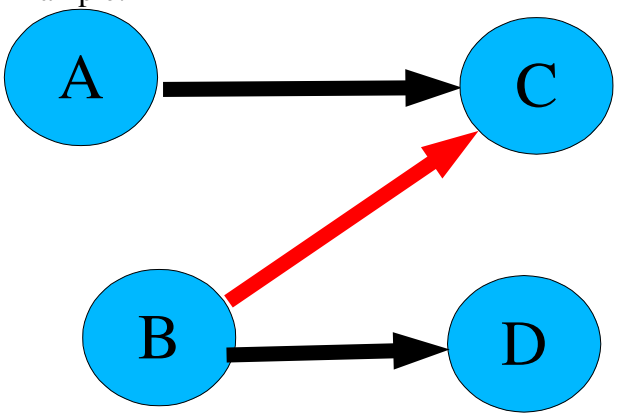
Guide to the subroutines:

($links_ref,adja_ref)=read_sif_file($sif_filename);
Reads the sif file $sif_filename. The network is output in the two fileformats discussed above. Please note that the two return values are references to the link array and the adja hash, respectively.

$out_degree_hash_ref=degrees($link_ref,'nodeF');
or
$in_degree_hash_ref=degrees($link_ref,'nodeT');
Calculates the degree of every node. The first version calculates the out-degree, while the second version calculates the in-degree. The returned value is a reference to a hash, whose keys are the names of the nodes in the network and whose values are the degree of the node.
Example:



$out_degree_hash_ref->{A}=1;
$out_degree_hash_ref->{B}=2;
Notice $out_degree_hash_ref->{C} and $out_degree_hash_ref->{D} do not exist.
$in_degree_hash_ref->{C}=2;
$in_degree_hash_ref->{D}=1;
Notice $out_degree_hash_ref->{A} and $out_degree_hash_ref->{B} do not exist.

$dist_array_ref=dist($degree_hash_ref);
Calculates the distribution of the hash %{$degree_hash_ref}. That is how many times a given value occurs in the hash. The returned value is a reference to an array. The value of element $d, $dist_array_ref->[$d] is the number of times the value $d occurs.
Example:
$dist_array_ref=dist($out_degree_hash_ref);
$dist_array_ref->[0]=undef;
$dist_array_ref->[1]=1;
$dist_array_ref->[2]=1;

$sym_adja_hash_ref=symmetrize($adja_hash_ref);
Returns the symmetric network %{$sym_adja_hash_ref} corresponding to the original network %{$adja_hash_ref}. That the network is symmetric means if $sym_adja_hash_ref->{'F'}{'T'} exists then also $sym_adja_hash_ref->{'T'}{'F'} exists.

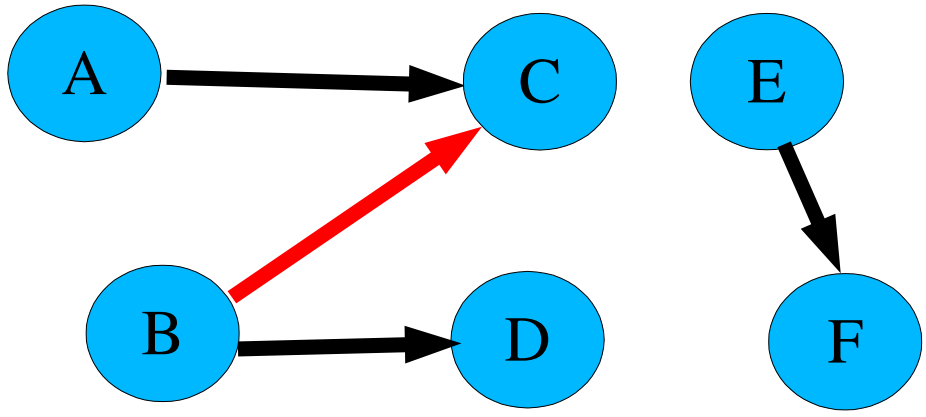
$node_list_array_ref=my_component($sym_adja_hash_ref,@node_list);
$sym_adja_hash_ref is a reference to symmetric network in the adjacency hash format. A network is symmetric means if $sym_adja_hash_ref->{'F'}{'T'} exists then also $sym_adja_hash_ref->{'T'}{'F'} exists.

Find all the nodes that are connected to a least one of the nodes in @node_list. All nodes in @node_list are per definition connected to at least one of the nodes in @node_list even if they are not nodes in the %{$sym_adja_hash_ref} network.

All the found nodes are returned in the array @{$node_list_array_ref}.

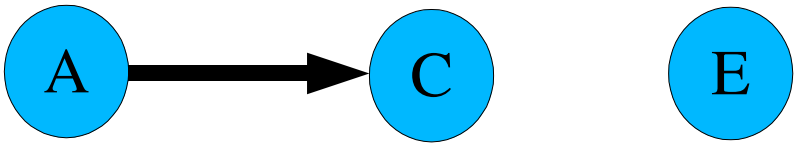
If the adja matrix is not symmetric only the nodes downstream any node in @nodelist are returned in @{$node_list_array_ref}.

($node2component_hash_ref,$component_array_ref)=components($sym_adja_hash_ref);
Finds all single connected components in the symmetric network %{$sym_adja_hash_ref}. That the network is symmetric means if $sym_adja_hash_ref->{'F'}{'T'} exists then also $sym_adja_hash_ref->{'T'}{'F'} exists.
The elements of the array @{$component_array_ref} are the different single connected components of the network. They are sorted after the number of nodes withthe largest first. Technically it is references to an array with all nodes in the given component as elements. %{$node2component_hash_ref} is a hash, whose keys are all the nodes in the network. Its value $node2component_hash_ref->{'node'} is a reference to the single connected component 'node' belongs. Technically it is a reference to one of the elements of @{$component_array_ref}. (Note not a copy, but the very *same* array).



```
$component_array_ref->[0]=['A','B','C','D'];  #The order is arbitrary
$component_array_ref->[1]=['E','F'];          #The order is arbitrary
$node2component_hash_ref->{'A'}=$component_array_ref->[0];
$node2component_hash_ref->{'B'}=$component_array_ref->[0];
$node2component_hash_ref->{'C'}=$component_array_ref->[0];
$node2component_hash_ref->{'D'}=$component_array_ref->[0];
$node2component_hash_ref->{'E'}=$component_array_ref->[1];
$node2component_hash_ref->{'F'}=$component_array_ref->[1];
```

($sub_network_links_ref,sub_network_adja_ref)=get_sub_network($adja_hash_ref ,$node_list_ref);
Returns a subnetwork of the network %{$adja_hash_ref}. Both the link and adja form are returned. Same output format as read_sif_file. The returned network is independent of the original network so changing one does not change the other.
The sub network consists *only* of the nodes @{$node_list_ref} and then all the links in the original network between them.
Example:
For the above network @{$node_list_ref} = ('A','C','E') yields the sub network



$links_ref=adja2links($adja_hash_ref)
Transforms a network on the adja form to the links formats

$num_FF =find_FF_loops($adja_hash_ref);
Calculates the number of feed forward loops in the network. Notice can also be used to calculate the number of undirected loops in a symmetric network as one undirected network corresponds to six feedforward loops (exercise?).

print_sif_file_links($links_ref,$sif_file)
Prints the links to the file $sif_file. The type of the link is set to 'link'. Perhaps in the future it becomes a third argument.

$prune_links_ref=prune_links_to_symmetric($links_ref)
Remove one copy of links that connect the same two nodes irrespective of direction.

Example:
A B
B A
becomes just
A B.

$ran_adja_hash_ref=randomize_directed($adja_hash_ref,$num_iterations)

Returns a randomized network with no double links. Links going in opposite directions are treated as different. Both the in- and out-degree of all the nodes are conserved.
$num_iterations are the number of swops performed.

$ran_sym_adja_hash_ref=randomize_symmetric($sym_adja_hash_ref,$num_iterations)

Returns a randomized symmetric network with no double links. Links going in opposite directions are treated as one link. In fact it is assumed the network %{$sym_adja_hash_ref} is symmetric. A network is symmetric, if $sym_adja_hash_ref->{'F'}{'T'} exists then also $sym_adja_hash_ref->{'T'}{'F'} exists. The degree of all the nodes are conserved.
$num_iterations are the number of swops performed.