Developers' Guide

Contents:

1. Introduction and Installation:

The main software components used in making this software are:

- Graphviz (version: 2.26.3-1)
- SWIG
- GnuPlot
- Imagemagik (Which is provided with graphviz or Ubuntu)

The programming languages used

- ANSI-C
- Tcl/tk

The software was developed in a Ubuntu system. The step by step procedure to install all the software and programming languages to make the system ready for further development (in Ubuntu system) are as follows. Before performing the following steps please ensure that you are connected to the Internet, as many software components will have to be installed from the Ubuntu on-line repository. Note that as an user you do not have to install all components.

1. Install gcc by typing `sudo apt-get install gcc` (in most systems this is already provided).
2. Install Tcl8.5 by typing `sudo apt-get install tcl8.5`. Install tcl8.5-dev package by typing `sudo apt-get install tcl8.5-dev`. Then install tk8.5 similarly.
3. Install graphviz. Go to www.graphviz.org. Go to the downloads section (http://graphviz.org/Download..php). Select he following .deb packages from the downloads list of Ubuntu/Debian systems. The required packages are:-
    a) libgraphviz4_2.26.3-1_i386.deb
    b) libgraphviz-dev_2.26.3-1_i386.deb

   c) graphviz_2.26.3-1_i386.deb
   d) graphviz-dev_2.26.3-1_all.deb
   e) libgv-tcl_2.26.3-1_i386.deb

   These packages should be installed in the order as they are listed.(At the time of writing this guide, the version of graphviz available in Ubuntu online repository is not up-to-date. So, using sudo apt-get method will not fetch the right packages.*). You can use Gdebi software which is a nice GUI based application install .deb packages. Keep in mind that you should not download any dependency files from Graphviz site if you are installing offline, it can cause version mismatch.
4. Install swig by typing sudo apt-get install swig
5. Install gnuplot by typing sudo apt-get install gnuplot.

If you have the source codes of the software then you are ready to compile and run the software. At first let us start with brief description of the source files.

2. Brief Description of Source Files:

The source code is divided into three parts: one written in C which deals with the main graph algorithms and the other written in Tcl/tk, that deals with the GUI and the interactive visualization part. The third part is SWIG files which are there to link C and Tcl/tk.

- C (.c) files:

There are 11 .c files (readg.c, addv.c, delete.c, graphviz.c, save.c, free_data.c, Cluster.c, compute.c, gnuplot.c, Short_dijkstra.c, visualize.c) and two headers (structdef.h and mstack.h). The header structdef.h contains the structure declarations used to implement the Data Structure needed to represent the graph in memory and all the function prototypes and preprocessor # define directives. There are structure definitions for vertex (or nodes), edges, a global variable collection named pass and also for a stack implementation. One can also see the def_wrap.c file which is generated by SWIG and its functionality will be described under swig files description.

* The libgv-tcl_2.26.3-1_i386.deb package is thoroughly different than the present version available in repository, and installing it will result in software malfunction. The main reason for migrating to the recent version 2.26.3-1 is the previous version contained major bugs in this package

- SWIG (.i) files:

def.i has a similar format as a C header. As we know that SWIG does the interfacing to make the C functions be available in Tcl, one has to include the structure definitions and the function prototypes in the .i file. It should be noted that only the functions that have to be made visible to the Tcl, has to be included in def.i file. When this def.i file is processed by SWIG, a file named def_wrap.c is produced. It contains the wrapper functions necessary to make the C functions visible to Tcl.

- Tcl (.tcl) files:

There are three .tcl files in the source code namely GUI.tcl, canvas.tcl and scrapbook.tcl. GUI.tcl contains the code for the basic GUI format of the software and also other regular routines such as loading a file, saving a file etc. canvas.tcl contains the code for GUI components for using in canvas (such as buttons and checkboxes) and scrapbook.tcl contains the codes for the scrapbook(functionalities of these elements like scrapbook, canvas, are described later.

The source code directory has two folders namely “bitmap” and “tmp”. “bitmap” contains some bitmap images used as icons in the GUI and “tmp” is the folder where the files generated during the software-operations.

3. Compiling and Running the Source Code:

If the necessary software are installed, you are ready to compile and run the source code. For compilation follow these steps from the source code directory:-

i. We generate the wrapper functions from def.i using SWIG.

```
%swig –tcl8 def.i
```

This generates the def_wrap.c file containing the wrappers.

ii. Now we make the object files of the .c files.

```
%gcc –fpic –c *.c –I/usr/include/tcl8.5/
```

-fpic option makes position independent code, and –c option prevents gcc from making any executable file. “–I/usr/include/tcl8.5/” this option is to link with the tcl.h file located in “/usr/include/tcl8.5/” directory. You should check if this folder exists and has the tcl.h file. If the folder is nonexistent, that may mean that you have not installed the tcl8.5-dev package properly or else it is located in a different location. If the folder exists but the tcl.h header is not there, it is located elsewhere which you have to sarch for. (for Ubuntu 10.04 lucid version the command above will work fine).
This command will produce all the object files corresponding to each of the .c files.

iii. Now we make a shared library of these object files.

```
%gcc –shared –o graph.so –lm
```

-lm option may be excluded if there are no #include<math.h> statements in the code.
This command will result in a shared library named “graph.so”. This shared library can be loaded in a tclsh8.5 (see any Tcl tutorial) by typing “`load ./graph.so graph`”, and all the C functions will be available for use as a tcl command.

iv. Now we have to execute the GUI.tcl file. But by default it is not an executable file, in order to make it executable, we type the following in the console:

```
%chmod +x GUI.tcl
```

This will mark GUI.tcl as an executable file.

So, we are now ready to execute the program. Just type `./GUI.tcl` in the terminal. You should see the software window now.

4. Using the Software:

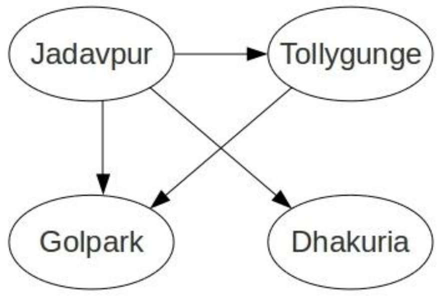
To use the software you need to know :-

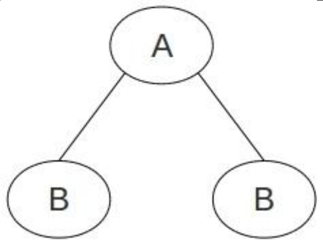
- File Format
- Basic functionalities

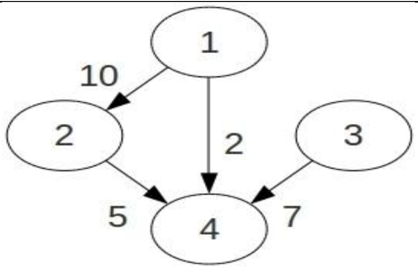
- File Format:

The basic file formats that are supported are .nil and .srj format. For a given graph.we will illustrate these formats.

Graph Type	Example Graph	.nil format	.srj format

Directed Unweighted		4 d
		“Jadavpur”
		“Tollygunge”
		“Jadavpur” “Golpark”
		“Tollygunge” “Golpark”
		“jadavpur” “Dhakuria”
		“Golpark” “Dhakuria”
			4 d
			{
			1 -> “Jadavpur”
			2 -> “Tollygunge”
			3 -> “Golpark”
			4 -> “Dhakuria”
			}
			1 2
			1 3
			2 3
			1 4
			3 4

Undirected Unweighted		3 u
		“A” “B”
		“A” “C”
			3 d
			{
			1 -> “A”
			2 -> “B”
			3 -> “C”
			}
			1 2
			1 3

Directed Weighted		4 d
		“1” “2” [10]
		“2” “4” [5]
		“1” “4” [2]
		“3” “4” [7]
			4 d
			{
			1 -> “1”
			2 -> “3”
			3 -> “2”
			4 -> “4”
			}
			1 3 [10]
			2 4 [7]
			1 4 [2]
			3 4 [5]

- Basic Functionalities:

You can load or create a new graph. For loading go to the open option and load the .nil or .srj file. The layout of the graph will appear in the canvas and then you can add more nodes to it, delete it, find shortest path between two nodes, add edges to existing nodes, view the graph properties, zoom-in and zoom-out, refresh the layout etc. An image of the graph layout is produced in the scrapbook whenever a new graph is loaded. Also the images for shortest path and degree distribution of a graph are loaded in scrapbook. While creating a new graph, a new file have to saved first and then the above operations can be carried out.

5. Development Notes:

C routines:

The C functions are fairly simple and self explanatory. The names of the files suggest what the functions defined inside them does. C source code is commented. Also there are some extra functions which were useful for running whole program in terminal. They might come in handy in further development and debugging. So they have not been done away with. There are a number of system calls inside these functions. For example, visualize.c has system calls for graphviz routines to generate the image files. “gnuplot.c” has system calls for generating the graph image and resizing it. Mainly some interface functions (like “save_interface()”) are only made visible to Tcl (along with some utility functions) by citing them in def.i file.

Tcl/tk Routines:

Among the files with Tcl/tk codes, GUI.tcl is “main” file that sources other .tcl files. The Graphical User Interface has a tk widget called notebook which hold the canvas, scrapbook and a textbox. These components has been implemented in separate files.

- Canvas.tcl and relevant information:

“canvas.tcl has the most important part of the software which is the interactive visualization part. The rendering of the graph layout on the canvas, when a file loaded is done at the Load command in GUI.tcl. This is done with the help of graphviz (libgvtcl package, find more by typing `%man render`). The objects in the canvas are uniquely tagged by graphviz to facilitate the interaction with rendered graph.

  - Layout by Graphviz in Tk canvas: The layout by graphviz has the following items:

Object Name	Represents	Tag
Oval	The outer boundary of a node	0node*number**
Text	The name of node or weight of edge for a weighted graph	1node*number (for nodes)* 0edge*number (for edges)*
Polygon	The triangular arrow shape for a directed edge	0edge*number*
Line	The straight line or small segments of straight lines to make a curve	0edge*number#*

*this number is automatically assigned by graphviz and there is no way to predict it

#This tag is same as the polygon item that represent the same edge. i.e. for a single directed edge the polygon and the line share the same tag

We have maintained this tagging system for adding a new node, edge or weight-text with a slight modification that instead of *number*, we have used **number*, to avoid any tagging collision with existing layout. For example, for a new node we would tag an oval as,say, “0node*3” and for an edge, “0edge*12”. However we have done away with the polygon for denoting an arrowhead and used an arrow shaped line from the available Tk line items.

This tagging system is the sole means of the robust method for selecting an item in the canvas. This method is as follows:

Whenever the mouse pointer enters an item an tk-event <Enter> occurs. We bound this event with a routine that fetches the tag associated with the item that has been entered into. Tk adds a “current” tag to an item if a mouse pointer hovers over it. So we fetched all the tags associated with current and singled out the one that resembles the “graphviz tagging” (like 0node*number* or 1edge**number*) and saved it in a global variable “alochyo”. Also when we leave an item, an event called <Leave> is invoked. With occurrence of this event we make “alochyo” an empty string after recording its content in variable “lastVisited”. So whenever we click an object we can simply have the tag of it in the variable “alochyo”. Again, when we open a pop-up window, we effectively LEAVE that item, so we can find the item tag in variable “lastVisited”.

We have interchangeably treated the list items of Tcl to be strings.

- scrapbook.tcl :-

  It is a relatively small file as compared to the canvas.tcl and GUI.tcl and contains basic GUI components.

6. Debugging:-

- Debugging the C functions:- One can use the older version for terminal and customize it to fit the need of debugging which will not be very tedious. Another advantage of it is you can use the GNU Debugger (gdb). Another way of debugging could be write the C functions and interface them for Tcl. And then test them in tclsh8.5 prompt before updating the main source code file.
- Debugging Tcl code: Any error in the Tcl code either shows up in the command prompt or as a dialogbox. The command prompt error message contains very few information, so it is advisable to test the codes in small parts or else finding an error can be a tough job.(We would like to mention that a message box would appear earlier, but probably due to some code it does not appear any more. So the error output is redirected to terminal. If you can find out the cause please change that piece of code to an alternative one).

7. References:-

- Tcl online tutorial
- Tcl wiki examples and tutorial
- Tk Docs tutorial for tk
- Book: Tcl/tk A Developers’ Guide - Morgan Kaufman
- SWIG official site
- http://www.swig.org/tutorial.html
- Tcl, C and SWIG: How to combine