SPL 6: Graphical Dijkstra's Algorithm

Out: Thursday, March 16th; Due: Thursday, March 23rd

I. Objectives

Primary Objective:

Improve your understanding of pointers through the implementation of a Dijkstra's algorithm.

Secondary Objectives:

- File parsing – File I/O. A look at more advanced File Parsing which includes files with comments and multiple values on a single line, separated by commas, a.k.a. CSV/CDV (Comma Separated/Delimited Values).
- Practice code reuse
- Practice with Graphics Library.
- **Debugging with PUT**

II. Files

Downloads - Please go to the Unified C&P Website and download the file under "Systems Problem Support" (If you haven't already installed them, you may also need "`UnifiedC&PInstallation`" and "`UnifiedC&PInstallation-Graphics`"):

"`C&P_SPL_6.zip`"

Included files:

"`ada_graphics.ads & ada_graphics.adb`"
"`double_buffer.ads & double_buffer.adb`"
"`priority_queue.ads & priority_queue.adb`"
"`priority_queue_graphics.ads & priority_queue_graphics.adb`"
"`Graph_package.ads & Graph_package.ads`"
"`Generic_set.ads & Graph_package.adb`"
"`test_dijkstra.adb`"
"`dijkstra.txt`"
"`dijkstra_input_generator.exe`"
"`test_ dijkstra.exe`"
"`test_ada_graphics.exe`"
"`gnat.ago`"

Files you will *have* to modify (you *are allowed* to modify any of them):

"`test_dijkstra.adb`"

III. Overview

This System Problem is similar to the previous system problem in that you will be dealing with file i/o, graphics, and code reuse. You will be required to answer a series of questions on how some of the code works and then write ONE procedure: Draw_Dijkstra, your implementation of the algorithm.

The end result of your coding efforts will be two graphic windows, one displaying Nodes connected by Links/Edges and the Dijkstra algorithm trying to find the shortest path, the other will display the Priority Queue that the algorithm uses. The final result should be similar to "`test_dijkstra.exe`." You can run (double click on) "`dijkstra_input_generator.exe`" to generate a new input file ("`Dijkstra.txt`") for "`test_dijkstra.exe`."

Dijkstra.txt

```
# Lines that start with a Pound Sign are Comments
Nodes:
# Node 1
257,694
# Node 2
91,545
# Node 3
55,350
```

```
Links:
# Links from Node 1 to other Nodes
 42.238,  0.020,  5.083, 59.964, 83.532, 12.792,
# Links from Node 2 to other Nodes
 87.260, 34.939, 59.386, 28.459, 94.082,
# Links from Node 3 to other Nodes
 40.689, 18.619, 74.831, 84.997,
```

Figure 1: "`Dijkstra.txt`" can be randomly generated by "`dijkstra_input_generator.exe`".

Note that "Dijkstra.txt" is much more complicated that "Input.txt" that you saw in the previous SP. There are # signs to denote comment lines that should be ignored, and lines of data with numbers separated by commas. This Comma Separated Value (CSV) format is very common if you ever have to look at data reports of sensor values.

In Dijkstra's algorithm sets are used to keep track of Nodes in a graph. Code to create and manipulate Sets have been written for you and are described in "`graph_package.ads`". The figure below illustrates the structure of a Graph. You should keep this structure in mind as it may be important when you are implementing your procedure.

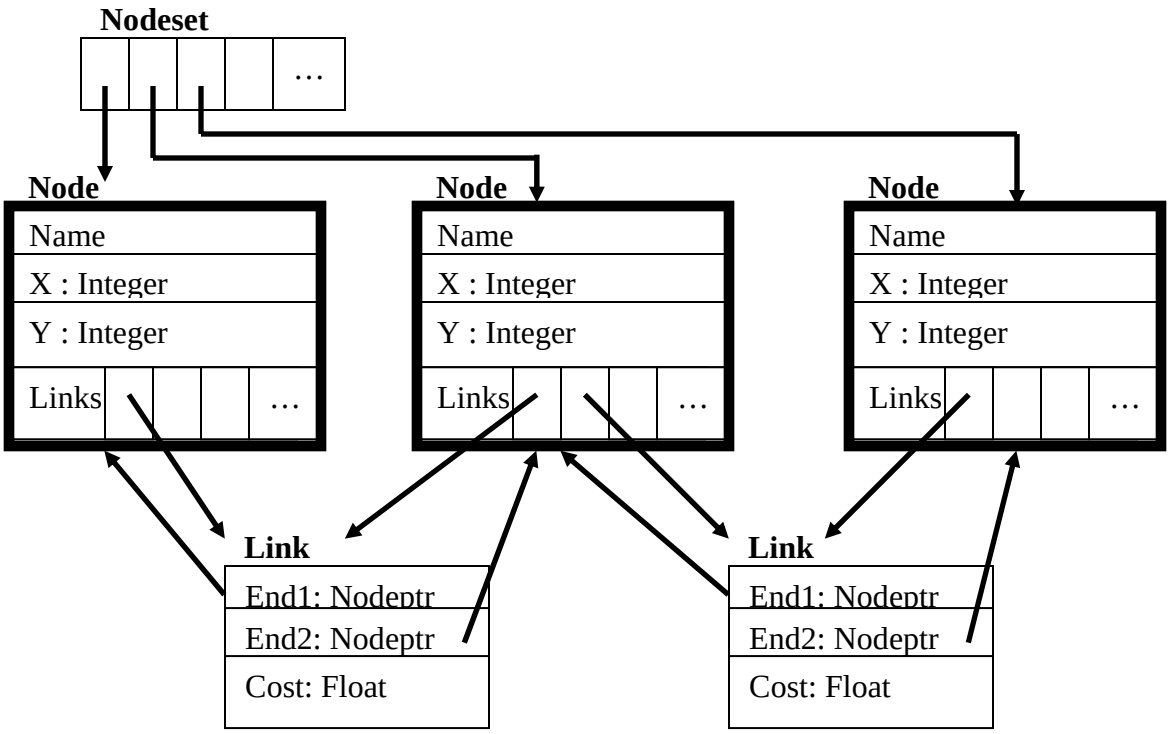

Figure 2: This is the structure of the Nodeset Type

In "`graph_pacakge.ads`" you will be able to find many procedures. Here is a list of some of the ones you might use:

```
        Copy_Set
--ELEMENT OPERATIONS:
        Add
        Remove
        Peek
--SET OPERATIONS:
        Intersection
        Union
        Difference
--SET STATUS OPERATIONS:
        Is_In
        Is_Empty
        Get_Size
--DRAWING OPERATIONS:
        Draw_Graph
        Draw_Node
        Draw_Link
```

IV. Questions:

Here are a series of questions that should help you get acquainted with the code. You will be expected to write up the answers and turn them in.

1. In "`test_dijkstra.adb`" there is a procedure called "ReadFile" that is responsible for reading the data from "Dijkstra.txt."
   A. In the previous SPL, you got numbers from the file line by line by calling Get. ReadFile still reads in the file line by line, but it doesn't get a number, what does it get instead?
   B. What is the significance of the State variable and checking for text in the file "Nodes:" and "Links:"?
   C. There is an Integer variable called "`Number`." If there are X nodes in the file, what is the maximum value that Number gets to when reading from the file? (Hint: Insert calls to "Put" to periodically print the values of Number).
   D. Up until now you've known that you can use "Put" to print to the text output. What does the line of code "`Put(I_Str,I);`" do? Why is it important?
   E. When parsing a String of Links from the File, there is a line of code: "`Get(Line(Temp+2..Eol),Cost,Temp);`" What is the significance of the number 2?

2. In "`test_dijkstra.adb`" there is a procedure called "`Initialize_Dijkstra.`"
   A. Computers do things by executing sequential instructions. So why might it be difficult to make a computer generate random numbers?
   B. Ada, like most languages, has a random number generator, but it doesn't actually generate random numbers. It is possible to make it seem like it is generating random numbers by giving a different starting instruction. What is the line of code that does this? What value does it use to "seed" the random number generator?
   C. What happens when you ***delete*** this line of code? Use Put, to print out the Random numbers you get. Run the program three times. Are the numbers the same or different?

3. In "`test_dijkstra.adb`" there is a procedure called "`Initialize_Dijkstra.`"
   A. In the definition portion of the procedure (between the words "is" and "begin") there is a line of code:
   "`Nptr : Graph_Package.Nodeptr;`"
   What happens when you remove "`Graph_Package.`"? Why does this happen?
   B. There is a call to procedure "`Peek`". To what package does this procedure belong?
   C. What is the difference between calling "`Peek`" versus calling "`Remove`"?

4. In "`graph_package.ads`" there are a bunch of complicated type and package definitions. There are three important things to notice: Type Link is a Record, Type Node is a Record, and Subtype Nodeset is something called a SET.
    A. Find the line of code that defines what subtype "Nodeset" is made from. Write that line down.
    B. You will notice that the line of code says that Nodeset is "Something1.Something2". Find where Something1 is defined in the code. Write down that definition.
    C. Even though you may NOT completely understand the definition: From the definition of "Something1" what do you think Something1 is designed to hold?

5. What is the significance of there being a "`My_Window`" and a "`My_Window2`" (Buffer / Buffer2, Buffer_Pen / Buffer_Pen2)? What do you think would happen if all uses of "`Buffer2`" were replaced with "`Buffer`"?

6. A new procedure "`ChangeValue`" has been added to package "`graph_package.`" "`ChangeValue`" is supposed to change the value of a Node in the Priority Queue.
    A. But, why is this not as simple as just rewriting the value of a Node in the Queue?
    B. Look at the code for "`ChangeValue`." How does `ChangeValue` work?

V. Draw_Dijkstra Procedure:

The procedure in "`test_dijkstra.adb`" that requires modification is:

Draw_Dijkstra

Notes:

a. The sets that you will need to use have ALREADY been created for you.
b. While the pseudo code in Figure 3 uses a bracket notation, the code in "graph_package" that implements the sets do NOT use bracket notation. You will have to figure out how to use the code available in that package to implement the pseudo code.

ToDo:

- Implement the "while loop"/"set manipulation" portion of the Dijkstra Algorithm given the procedures available in the "`graph_package`" and "`priority_queue`" files.
- Add calls to the graphics procedures that can be found in "`graph_package`"

```
While Q is not an empty set loop
        U := Extract_Min(Q)
        S := S union {u}
        For each vertex v which is a neighbor of u loop
                if d[v] > d[u] + w(u,v) then
                        d[v] := d[u] + w(u,v)
                        previous[v] := u
```

Figure 3: Recall the Pseudo Code for Dijkstra's Algorithm

VI. How To Start

1. Run "test_`dijkstra.exe`." This is what your finished product should look like.

2. Double click on "`dijkstra_input_generator.exe`" to create a new "`Dijkstra.txt`" file. Open this file and get an idea of its layout.

3. Answer the Questions! Remember that comments can usually be found in .ads files and the bulk of the code in .adb files. Also, use PUT to print out intermediate values to the screen so you can see what going on in the code.

4. Using the list of procedures in "`graph_package.adb`", above, and the procedures in Priority Queue, rewrite the Pseudo Code given in Figure 3. Remember that definition and initialization of the sets and queue has already been done. See what they are and what they are named in "`test_dijkstra.adb`"

5. Code the Draw_Dijkstra procedure. Do NOT worry about the graphical part. Print text to the output to help you debug what your code is doing!

6. Copy ALL of your code over to a new folder for safe keeping!

7. Look over the Drawing Procedures in "`graph_package.ads`." With these procedures and changing the pen color, draw/erase/highlight the links and nodes your code is investigating. It is up to you what you choose to highlight. You can decide what is important to draw and what is not. But, you MUST draw something to show your algorithm works!

VII. Grading Scheme (Out of 100)

50 – Questions
30 – Correctly Implemeneted "`Draw_Dijkstra`" procedure.
10 – Graphics for the Dijkstra Algorithm
10 – Attending Mandatory Office Hours.

VIII. Turn In Instructions

1. Please zip and turn in **ALL** the **.ads and .adb** files you used in developing this assignment via the on-line submission system (At least make sure `"test_dijkstra.adb"` is included).
2. Name the zip file: "2006SPL6_LastName_FirstName.zip"
3. Go to the class web site to submit the file.
4. **Submit the answers to the questions in hardcopy.**

Reminder:

- READ THE COMMENTS!
- Remember to check for a null pointer before using it