Programming Assignment 1 - Dijkstra's Algorithm

Implement Dijkstra's Shortest Path Algorithm using an array for a directed weighted graph with positive arc weights.

As in the lecture notes, your program will represent

- Graph *G*: An array *G*[1...*n*] of pointers to singly-linked adjacency lists.
- Shortest Path Tree *T*(*r*) rooted at r: A parent array *p*[1...*n*] to store the tree structure paths), and a distance array *D*[1...*n*] to hold the path lengths.
- Selection structure for the shortest path algorithm: an array

The inputs to this algorithm should be a graph *G* (an array of adjacency lists), and the index r of the root node (may be chosen at run-time by a user, may be passed in by a calling routine).

The output is a shortest path spanning tree rooted at *r*, implemented as arrays *p* and *D* as above.

Your program should be able to read in the graph from a file, either as

- a stream of weighted arcs; or
- an arc-weight array.

You only need to support one of these possible file formats. The first line of the input file is to contain a flag to indicate which format the file is in:

- **S** for a stream of arcs; or
- **A** for an arc-weight array.

The second line of the file should give *n*, the order of the graph. The program then stores the graph internally in memory as an array of adjacency lists (it is especially easy to build an array of adjacency lists from a stream of arcs).

Apart from the program code, you should provide documentation with three major sections, as follows.

1. Outline the problem and algorithms used to solve them, derive complexities, solve problems by hand, etc. Here also summarise your results from this project, and draw conclusions.
2. Detailed design: address the implementation issues, make and justify your design decisions.
3. Unit test plan, detailing (a) test cases, (b) test data to direct program flow to those cases, (c) expected results, and (d) actual results.

Test your implementation of Dijkstra's algorithm with at least:

- a large sparse graph
- a large dense graph

Here you can take "large" to mean "having a few hundred nodes". Use whatever method you like (e.g., random arcs) to generate the files for these graphs.