**CS 542 Fall 13 Project Report**
***on* Implementation of Link-State Routing Protocol**

*by* Wang, Dingwen (CWID: A20304858)
Email: dwang43@hawk.iit.edu
Section: 01 (live, main campus)

## Introduction

When link-state routing protocol is applied, each router needs to construct a map of the connectivity to the network. Every router floods the network with its connectivity information, so that all routers can obtain a complete map of the network. And then, a router will create/modify its routing table according to the map.

In this project, the concentration is on the part of work to be done on a router after the map is constructed, namely, finding shortest paths and building the routing table. The major algorithm involved is Dijkstra's single-source shortest-path algorithm. The programming language used is the C-related subset of the C++ programming language. And several testing cases are designed to make sure that the program works correctly on valid input. Please read on to find more details.

## Data Structures

Following is a list of major data structures (with description) that were designed:

```
#define INFINITY -1
typedef struct vertex {
	int id;
	int dist;
	struct vertex *pre;
} vertex;
```

The *vertex* data structure corresponds to the vertex that is involved in the computation of Dijkstra's algorithm. Given a vertex, its *id* is a positive integer that identifies itself; *dist* holds the value of its currently computed distance from the source vertex (*dist* equals *INFINITY* when a path has not yet been found during computation); *pre* is a pointer that points to the predecessor vertex on the currently found shortest path during computation (*pre* equals *NULL* when a path has not yet been found during computation).

```
typedef struct graph {
	int size;    // number of vertices
	int *adjmtx;
} graph;
```

The *graph* data structure is basically an adjacency matrix implemented as a 1-dimension array with capacity that equals $size^2$.

```
typedef struct {
	int dest;
	int next;
} entry;
typedef struct router {
	int id;
	int n;
	vertex *varray;
	entry *table;
} router;
```

The *router* data structure provides the environment within which the Dijkstra's algorithm operates, and it denotes the source router of the single-source shortest path problem. Given a router, its *id* is a positive integer that identifies itself; *n* holds the number of routers to be involved in the problem; *varray* points to an array of *n* vertices, which Dijkstra's algorithm runs on and modifies (at the end of a successful run of Dijkstra's algorithm, every *vertex* in *varray* reaches the final state, in which it is either an isolated vertex, say, *dist* equals *INFINITY*, and *pre* equals *NULL*, or one of the vertices whose shortest path from source vertex is found.); *table* points to an array of *n entry*s, which holds routing table of the *router*.

**Algorithms**

*PICKMIN* – remove from set *Q* the vertex whose *dist* value is currently the minimum of all vertices in set *Q*, and add it to set *S*. Due to time constraint, a simple array-based implemented is applied in this project. For more sophisticated implementation, a min-heap or Fibonacci-heap is usually applied.

*RELAXATION* – for a vertex *u* returned by *PICKMIN*, by looking up the adjacency matrix, 0 or more vertices can be found directly connected with *u*. If vertex *v* is found directly connected with *u* over an edge of weight *w*, the value of *v->dist* will be updated to *u->dist + w* when the latter is less than the former, meaning, a shorter path that goes through *u* is found for *v*. The value of *v->dist* will be remained if *u->dist + w* is no less than *v->dist*, meaning, no shorter path is found for *v* in this time of relaxation.

*DIJKSTRA* – the initial state is set like this: a vertex set *Q*, which consists of all vertices; a vertex set *S*, which consists of none; all but one vertices in *Q* have their *dist* set to *INFINITY*; *dist* of the source vertex is set to 0. After the initial state is set, perform *PICKMIN -> RELAXATION* repeatedly until *PICKMIN* can get no more vertices. Dijkstra's algorithm ends here where a shortest path has been found for every vertex in set *S*.

Following is the implementation code for the Dijkstra's algorithm:

```
    void dijkstra(graph *g, int srcid, vertex *varr) {
           int n = g->size;
           vertex **s_set = (vertex **)calloc(n, sizeof(char *));
           vertex **q_set = (vertex **)calloc(n, sizeof(char *));
           int i;

           vertex **p;
           for (i = 1, p = q_set; i < n+1; i++, p++) {
                  *p = varr + i - 1;
                  if (i == srcid) {
                         (*p)->dist = 0;
                  } else {
                         (*p)->dist = INFINITY;
                  }
                  (*p)->pre = NULL;
           }

           vertex *u;
           int s_set_i = 0;
           int delta;
           int dist;
           vertex *vp;
           while ((u = pickMIN(q_set, n)) != NULL) {
                  *(s_set + s_set_i++) = u;
                  // relaxation
```

```
                for (i=1, vp=varr; i < n+1; i++, vp++) {
                        if ((delta = distance(g, u->id, i)) == INFINITY)
                                continue;
                        dist = u->dist + delta;
                        if (less(dist, vp->dist)) {
                                vp->dist = dist;
                                vp->pre = u;
                        }
                }
        }
        free(s_set);
        free(q_set);
}
```

**Compile and Run Program**

There are 5 source code files: cs542.h, main.c, router.c, graph,c, and dijkstra.c. They are compiled on Linux system using *gcc* compiler. Command

```
gcc –Wall –ggdb main.c router.c graph.c dijkstra.c –o cs542
```

will compile the source code and produce the executable file cs542. Option `–Wall` and `–ggdb` are included for debug use. And then, enter `./cs542` will run the program.

**Test Cases and Screenshots**

Here are six of the test cases that have been applied to the program and yields correct results:

1. The given sample:

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case0.txt
Original routing table is as follows:
0       2       5       1       -1
2       0       8       7       9
5       8       0       -1      4
1       7       -1      0       2
-1      9       4       2       0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 3
destination: 4  next-hop: 4
destination: 5  next-hop: 4

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
1 5
The shortest path from 1 to 5 is 1-4-5, the total cost is 3.
```

2. A network of which the topology resembles a linear line (10 routers included):

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case1_line.txt
Original routing table is as follows:
0       1       -1      -1      -1      -1      -1      -1      -1      -1
1       0       1       -1      -1      -1      -1      -1      -1      -1
-1      1       0       1       -1      -1      -1      -1      -1      -1
-1      -1      1       0       1       -1      -1      -1      -1      -1
-1      -1      -1      1       0       1       -1      -1      -1      -1
-1      -1      -1      -1      1       0       1       -1      -1      -1
-1      -1      -1      -1      -1      1       0       1       -1      -1
-1      -1      -1      -1      -1      -1      1       0       1       -1
-1      -1      -1      -1      -1      -1      -1      1       0       1
-1      -1      -1      -1      -1      -1      -1      -1      1       0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 2
destination: 4  next-hop: 2
destination: 5  next-hop: 2
destination: 6  next-hop: 2
destination: 7  next-hop: 2
destination: 8  next-hop: 2
destination: 9  next-hop: 2
destination: 10 next-hop: 2

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
2
destination: 1  next-hop: 1
destination: 2  next-hop: 0
destination: 3  next-hop: 3
destination: 4  next-hop: 3
destination: 5  next-hop: 3
destination: 6  next-hop: 3
destination: 7  next-hop: 3
destination: 8  next-hop: 3
destination: 9  next-hop: 3
destination: 10 next-hop: 3

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
1 10
The shortest path from 1 to 10 is 1-2-3-4-5-6-7-8-9-10, the total cost is 9.
```

3. A network of which the topology resembles a circle (10 routers included):

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case2_circle.txt
Original routing table is as follows:
0       1       -1      -1      -1      -1      -1      -1      -1      1
1       0       1       -1      -1      -1      -1      -1      -1      -1
-1      1       0       1       -1      -1      -1      -1      -1      -1
-1      -1      1       0       1       -1      -1      -1      -1      -1
-1      -1      -1      1       0       1       -1      -1      -1      -1
-1      -1      -1      -1      1       0       1       -1      -1      -1
-1      -1      -1      -1      -1      1       0       1       -1      -1
-1      -1      -1      -1      -1      -1      1       0       1       -1
-1      -1      -1      -1      -1      -1      -1      1       0       1
1       -1      -1      -1      -1      -1      -1      -1      1       0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 2
destination: 4  next-hop: 2
destination: 5  next-hop: 2
destination: 6  next-hop: 2
destination: 7  next-hop: 10
destination: 8  next-hop: 10
destination: 9  next-hop: 10
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
2
destination: 1  next-hop: 1
destination: 2  next-hop: 0
destination: 3  next-hop: 3
destination: 4  next-hop: 3
destination: 5  next-hop: 3
destination: 6  next-hop: 3
destination: 7  next-hop: 3
destination: 8  next-hop: 1
destination: 9  next-hop: 1
destination: 10 next-hop: 1

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
1 10
The shortest path from 1 to 10 is 1-10, the total cost is 1.
```

4. A network of which the topology resembles a star (10 routers included):

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case3_star.txt
Original routing table is as follows:
0       1       1       1       1       1       1       1       1       1
1       0       -1      -1      -1      -1      -1      -1      -1      -1
1       -1      0       -1      -1      -1      -1      -1      -1      -1
1       -1      -1      0       -1      -1      -1      -1      -1      -1
1       -1      -1      -1      0       -1      -1      -1      -1      -1
1       -1      -1      -1      -1      0       -1      -1      -1      -1
1       -1      -1      -1      -1      -1      0       -1      -1      -1
1       -1      -1      -1      -1      -1      -1      0       -1      -1
1       -1      -1      -1      -1      -1      -1      -1      0       -1
1       -1      -1      -1      -1      -1      -1      -1      -1      0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 3
destination: 4  next-hop: 4
destination: 5  next-hop: 5
destination: 6  next-hop: 6
destination: 7  next-hop: 7
destination: 8  next-hop: 8
destination: 9  next-hop: 9
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
2
destination: 1  next-hop: 1
destination: 2  next-hop: 0
destination: 3  next-hop: 1
destination: 4  next-hop: 1
destination: 5  next-hop: 1
destination: 6  next-hop: 1
destination: 7  next-hop: 1
destination: 8  next-hop: 1
destination: 9  next-hop: 1
destination: 10 next-hop: 1

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
2 3
The shortest path from 2 to 3 is 2-1-3, the total cost is 2.
```

5. A network of which the topology resembles the mix of a circle and a star (10 routers included):

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case4_starcircle.txt
Original routing table is as follows:
0       1       1       1       1       1       1       1       1       1
1       0       1       -1      -1      -1      -1      -1      -1      1
1       1       0       1       -1      -1      -1      -1      -1      -1
1       -1      1       0       1       -1      -1      -1      -1      -1
1       -1      -1      1       0       1       -1      -1      -1      -1
1       -1      -1      -1      1       0       1       -1      -1      -1
1       -1      -1      -1      -1      1       0       1       -1      -1
1       -1      -1      -1      -1      -1      1       0       1       -1
1       -1      -1      -1      -1      -1      -1      1       0       1
1       1       -1      -1      -1      -1      -1      -1      1       0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 3
destination: 4  next-hop: 4
destination: 5  next-hop: 5
destination: 6  next-hop: 6
destination: 7  next-hop: 7
destination: 8  next-hop: 8
destination: 9  next-hop: 9
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
2
destination: 1  next-hop: 1
destination: 2  next-hop: 0
destination: 3  next-hop: 3
destination: 4  next-hop: 1
destination: 5  next-hop: 1
destination: 6  next-hop: 1
destination: 7  next-hop: 1
destination: 8  next-hop: 1
destination: 9  next-hop: 1
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
2 3
The shortest path from 2 to 3 is 2-3, the total cost is 1.
```

6. A network of which every router connects with all the other routers (10 routers included):

```
1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
1
Please load original routing table data file:
case5_allconnected.txt
Original routing table is as follows:
0       1       1       1       1       1       1       1       1       1
1       0       1       1       1       1       1       1       1       1
1       1       0       1       1       1       1       1       1       1
1       1       1       0       1       1       1       1       1       1
1       1       1       1       0       1       1       1       1       1
1       1       1       1       1       0       1       1       1       1
1       1       1       1       1       1       0       1       1       1
1       1       1       1       1       1       1       0       1       1
1       1       1       1       1       1       1       1       0       1
1       1       1       1       1       1       1       1       1       0

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
1
destination: 1  next-hop: 0
destination: 2  next-hop: 2
destination: 3  next-hop: 3
destination: 4  next-hop: 4
destination: 5  next-hop: 5
destination: 6  next-hop: 6
destination: 7  next-hop: 7
destination: 8  next-hop: 8
destination: 9  next-hop: 9
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
2
Please select a router:
2
destination: 1  next-hop: 1
destination: 2  next-hop: 0
destination: 3  next-hop: 3
destination: 4  next-hop: 4
destination: 5  next-hop: 5
destination: 6  next-hop: 6
destination: 7  next-hop: 7
destination: 8  next-hop: 8
destination: 9  next-hop: 9
destination: 10 next-hop: 10

1-Load File
2-Build Routing Table for Each Router
3-Out Optimal Path and Minimum Cost
3
Please input the source and destination router number:
1 5
The shortest path from 1 to 5 is 1-5, the total cost is 1.
```