* The file containing **costs** should look something like this:

0 1 1 100

1 0 100 1

1 100 0 1

100 1 1 0

* The file containing **nodes/machines** should look something like this:

linux60808 <ip> 5000

linux60808 <ip> 6000

linux60808 <ip> 7000

linux60808 <ip> 8000

* Structure of the program:
  + Initialize global variables
    - struct machines
      * Containing name, ip and, port
    - Initialize the arrays we would require
      * machines, costs, distances …
  + inside the main function
    - obtain initialization parameters via argv
    - Initialization of network elements
      * Socket, address …
    - Read data from files into the matrices using fscanf()
      * Costs and machines
    - Thread-1
      * Updates the matrix containing the costs of the neighbors on receiving a message from other nodes.
    - Thread-2
      * In case of any change in the cost of the neighbors’ path, it accepts user input, updates the matrix containing the *costs* of the neighbors and notifies the other nodes about the change
    - Thread-3
      * Runs Dijkstra’s algorithm (calculating the shortest path) and updates the costs and distances arrays
* Notes:
  + Use mutex (pthread\_mutex\_lock and pthread\_mutex\_unlock ) while accessing the costs matrix
  + try printing out the values of matrices and arrays frequently to verify your logic
    - for example:

void print\_costs (void)

{

int i, j;

for (i = 0; i < N; i++)

{

for (j = 0; j < N; j++)

{

pthread\_mutex\_lock (&lock);

printf ("%d ", costs[i][j]);

pthread\_mutex\_unlock (&lock);

}

printf ("\n");

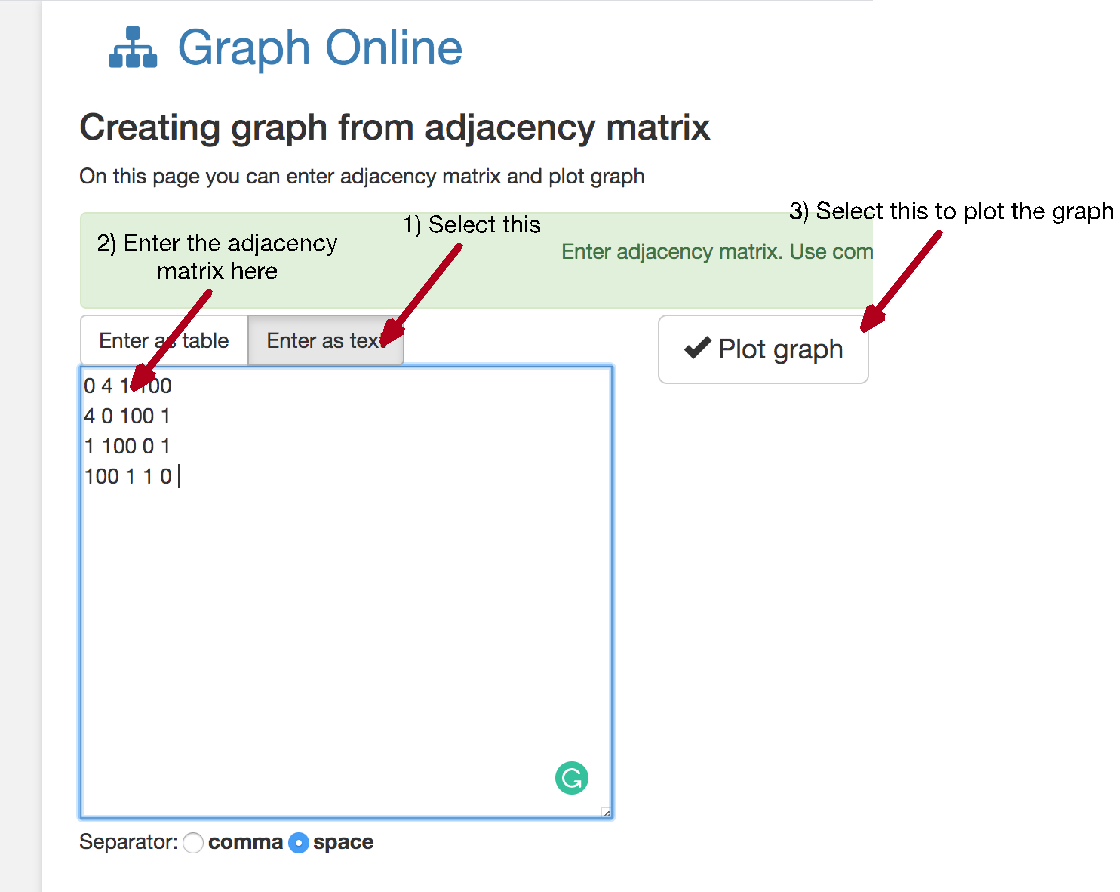
}

}

* + In order to visualize the graph according

to the adjacency matrix, you can use

* + <https://graphonline.ru/en/create_graph_by_matrix>



* + Test case:
  + Costs.txt

100 4 100 3 100 5

4 100 3 4 4 100

100 3 100 2 100 100

3 4 2 100 3 100

100 4 100 3 100 1

5 100 100 100 1 100

* + Shortest paths:

------Source = 0-------

Vertex Distance from Source

0 0

1 4

2 5

3 3

4 6

5 5

------Source = 1-------

Vertex Distance from Source

0 4

1 0

2 3

3 4

4 4

5 5

------Source = 2-------

Vertex Distance from Source

0 5

1 3

2 0

3 2

4 5

5 6

------Source = 3-------

Vertex Distance from Source

0 3

1 4

2 2

3 0

4 3

5 4

------Source = 4-------

Vertex Distance from Source

0 6

1 4

2 5

3 3

4 0

5 1

------Source = 5-------

Vertex Distance from Source

0 5

1 5

2 6

3 4

4 1

5 0