**KINGDOM OF SAUDI**

**ARABIA**

**Ministry of Higher Education**

**Al-Imam Muhammad Ibn**

**Saud Islamic University**

**College of Computer & 1438/1439**

**Information SciencesSpring 2018**

**المملكة العربية السعودية** **وزارة التعليم العالي**

**جامعة الإمام محمد بن سعود**  **الإسلامية** **كلية علوم الحاسب والمعلومات**

## Design and Analysis of Algorithms(CS- 215)

**Submitted To:**

Dr.Yassin Mohammed Aldaadaa

**Submitted By:**

Abdulrahman khalid alghafees 436018031

Talal omar Aljasser 437014512

Saeed abdulwasea almekhlafi 435031769

**Project Title:**

LAN shortcut system

**1-Propose:**

* 1. Pseudo code of Floyd-Warshall algorithm

1 **function** FloydWarshall (w)

2 **let** dist be a |V| × |V| initialized to (infinity)

3 **for each** edge (*u*,*v*)

4 dist[*u*][*v*] ← w(*u*,*v*) *// the weight of the edge (*u*,*v*)*

5 **for each** vertex *v*

6 dist[*v*][*v*] ← 0

7 **for** *k* **from** 1 **to** |V|

8 **for** *i* **from** 1 **to** |V|

9 **for** *j* **from** 1 **to** |V|

10 **if** dist[*i*][*j*] > dist[*i*][*k*] + dist[*k*][*j*]

11 dist[*i*][*j*] ← dist[*i*][*k*] + dist[*k*][*j*]

12 **end if**

Time complexity of this algorithm is |V|^3

* 1. Pseudo code of Johnson algorithm

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20 | **function** Johnson(G)  1.  create G` where G`.V = G.V + {s},  G`.E = G.E + ((s, u) for u in G.V), and  weight(s, u) = 0 for u in G.V  2.  **if** Bellman-Ford(s) == False  **return** "The input graph has a negative weight cycle"  **else**:  **for** vertex v **in** G`.V:  h(v) = distance(s, v) computed by Bellman-Ford  **for** edge (u, v) **in** G`.E:  weight`(u, v) = weight(u, v) + h(u) - h(v)  3.  D = new matrix of distances initialized to infinity  **for** vertex u **in** G.V:  run Dijkstra(G, weight`, u) to compute distance`(u, v) for all v in G.V  **for** each vertex v **in** G.V:  D\_(u, v) = distance`(u, v) + h(v) - h(u)  return D |

The first and second part of this algorithm don’t need to us and h (v) is 0 for all v in G.V, because we have only positive weight.

We will use Dijkstra algorithm. Then the time complexity of Dijkstra is O ((V+E) log V).

So total time complexity of the Johnson algorithm is O (V(V+E) log V).

* 1. Pseudo code of dijkstra algorithm

1 **function** Dijkstra(*Graph*, *source*):

2

3 create heap Q

4

5 **for each** vertex *v* in *Graph*: *// Initialization*

6 dist[*v*] ← INFINITY *// Unknown distance from source to v*

7 add *v* to *Q* *// All nodes initially in Q (unvisited nodes)*

9

10 dist[*source*] ← 0 *// Distance from source to source*

11

12 **while** *Q* is not empty:

13 *u* ← vertex in *Q* with min dist[u] *// Node with the min distance*

14 *// will be selected first*

15 remove *u* from *Q*

16

17 **for each** neighbor *v* of *u*: *// where v is still in Q.*

18 *alt* ← dist[*u*] + length(*u*, *v*)

19 **if** *alt* < dist[*v*]: *// A shorter path to v has been found*

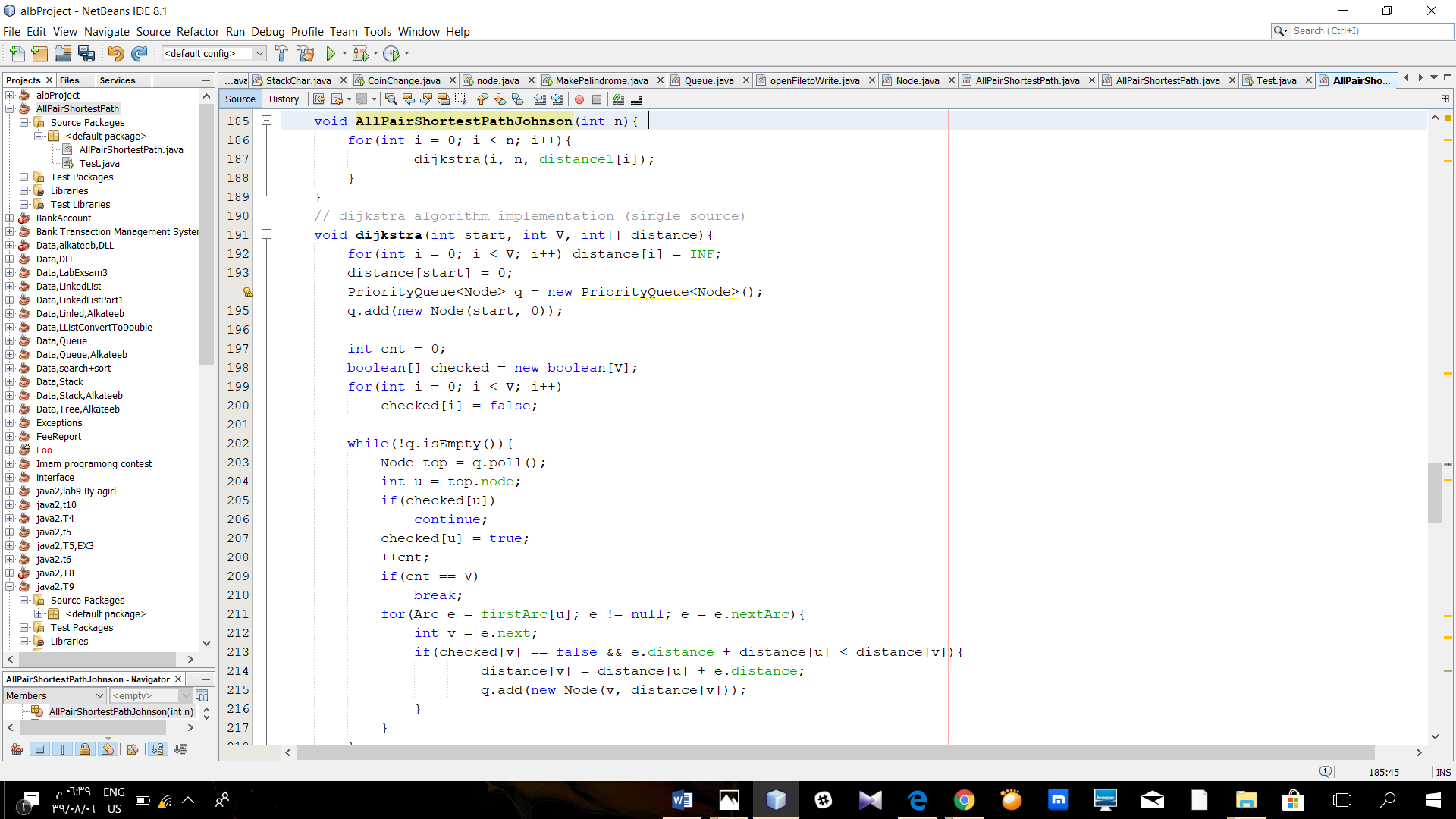
20 dist[*v*] ← *alt*

22

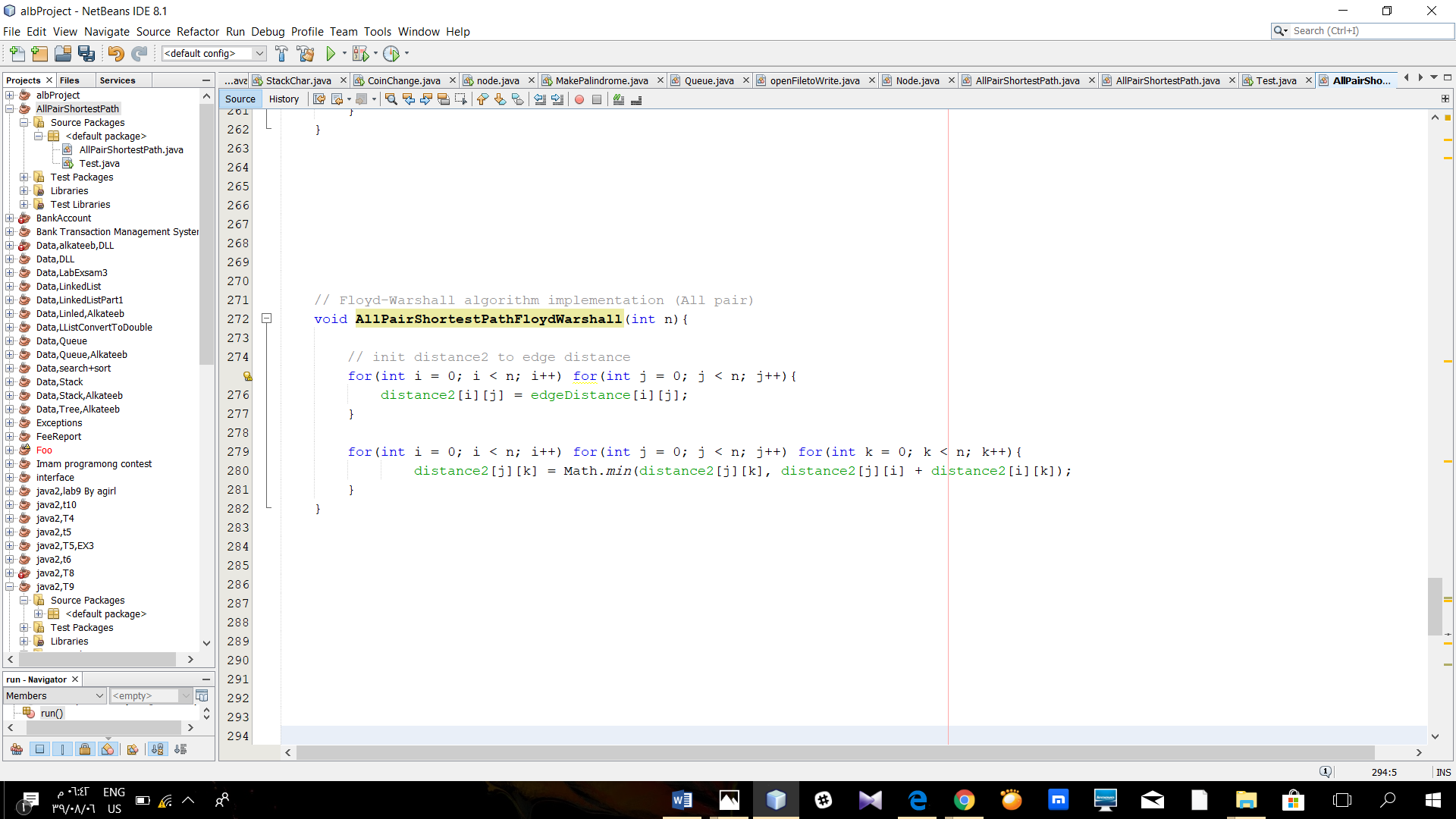
23 **return** dist

**2-Source code**

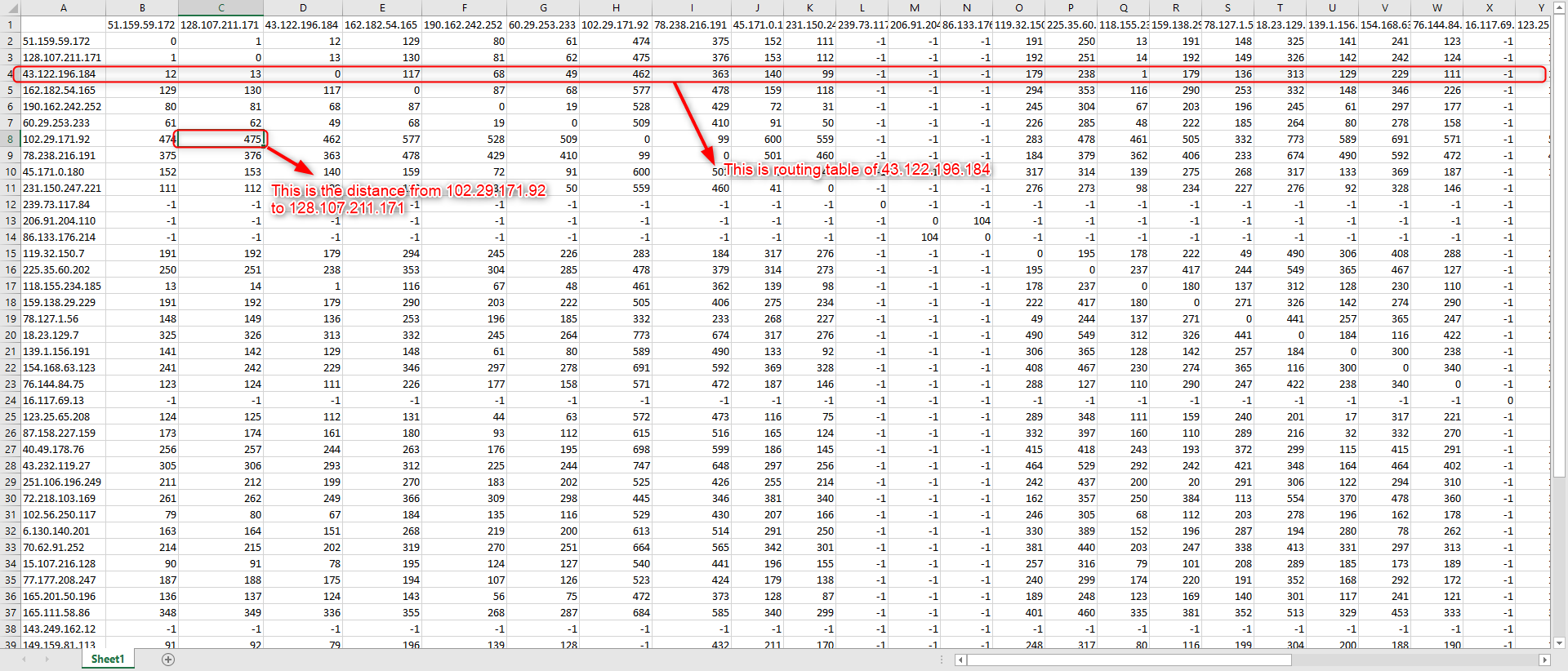
2.2- Johnson algorithm



2.2- Floyd- Warshall algorithm



**3- Output**



**4-Conclusion:**

We use Floyd-Warshall and Johnson algorithm and we conclude that Johnson is faster than Floyd-Warshall algorithm on a sparse graph.

For more details, open the attached excel file and lock the compare between them.

**5-Reference:**

5.1- Text book Introduction to Algorithms Third Edition

5.2- <https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm>

5.3- <https://www.geeksforgeeks.org/johnsons-algorithm/>

5.4- <https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm>

5.5- <http://agdn-online.com/posts.aspx?threadid=475>

5.6- <https://brilliant.org/wiki/johnsons-algorithm/>

5.7- <https://github.com/search?utf8=%E2%9C%93&q=johnson&type>=

5.8- <https://github.com/search?utf8=%E2%9C%93&q=floyd+warshall&type>=