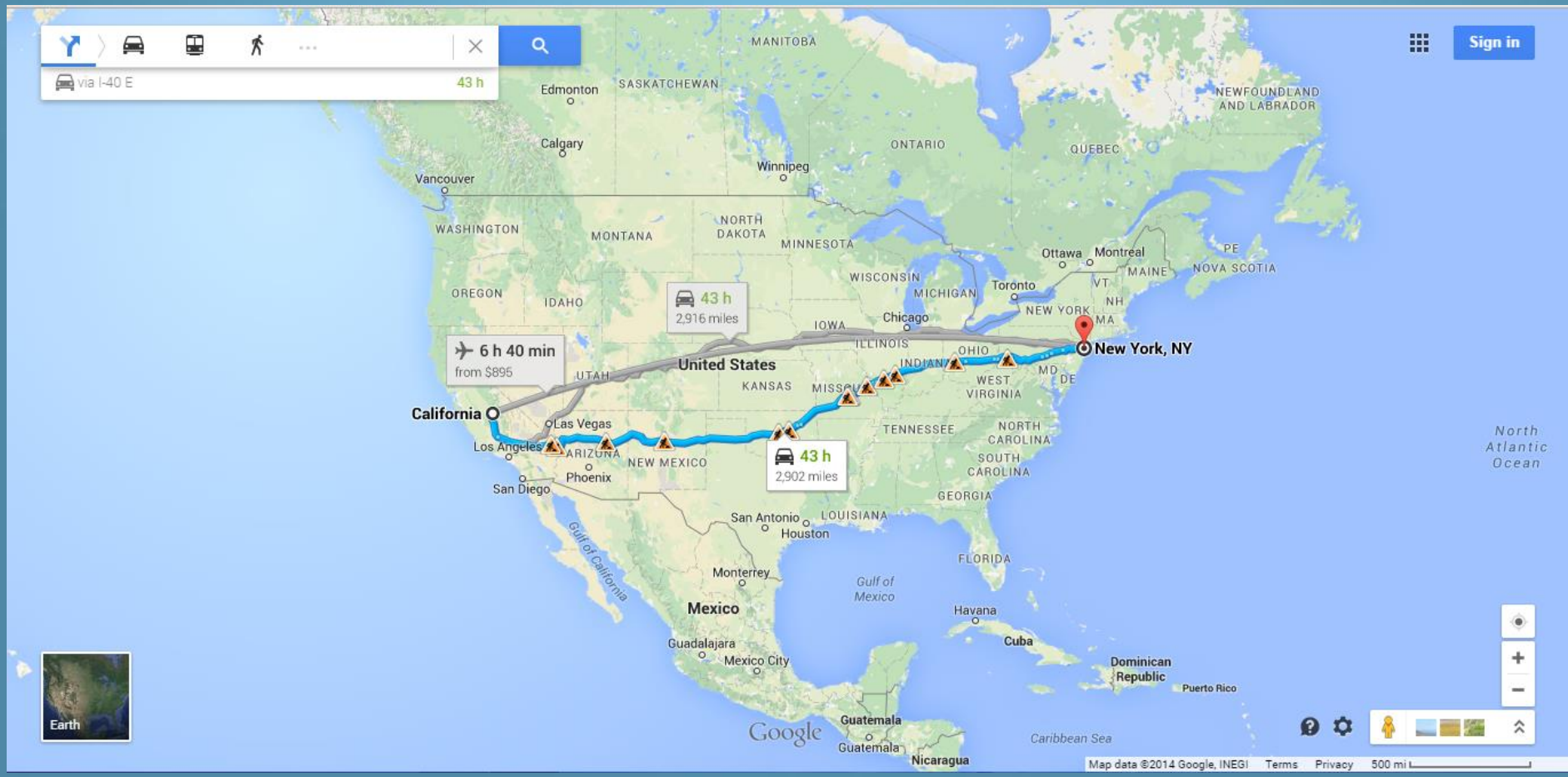
# Comparison of the moore’s method and dijkstra’s algorithm in determing the shortest path between two points on a map

BY GROUP 18
Cai Yuhao
Fan Hsiao-Tien
Liu Bowen
Paglia John Naisby

# MOTIVATION

- Global Positioning System (GPS) navigation onto  smartphone
- Limitations of processing capability
- More efficient algorithm necessary

# MOTIVATION

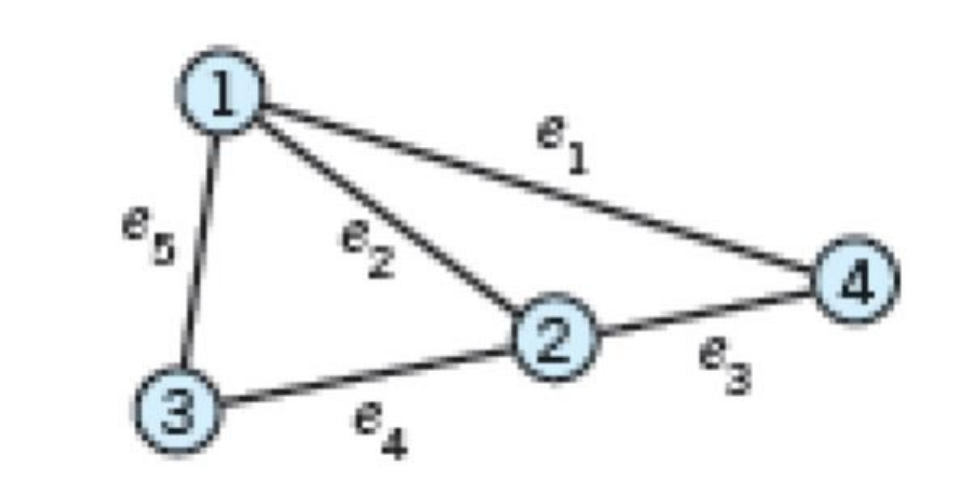

via I-40 E
43 h
Sign in
43 h
2,916 miles
6 h 40 min
from $895
43 h
2,902 miles
New York, NY
California
United States
Las Vegas
Los Angeles
Phoenix
San Diego
Chicago
Toronto
Ottawa
Montreal
Vancouver
Calgary
Edmonton
Winnipeg
Houston
San Antonio
Monterrey
Mexico
Guadalajara
Mexico City
Havana
Cuba
Dominican Republic
Puerto Rico
Guatemala
Nicaragua
Gulf of Mexico
Caribbean Sea
North Atlantic Ocean
Earth
Google
Map data ©2014 Google, INEGI
Terms
Privacy
500 mi

# Problem description

- Maps made with nxn number of tiles
- A starting point and ending point would be determined within this map
- Maps can be represented as graphs
- Compare Moore's and Dijkstras method for these graphs

# Introduction to methods

## Background Knowledge




- A set V of vertices: stand for spots in map models
- A set E of edges: stand for routes connecting spots

# Moore’s Method (Breadth-First-Search Algorithm)

- The widely used fundamental method in searching for the shortest path
- Blind searching method: searching every node of a map to find out the shortest path

# Procedure of the BFS method

1.Label the start point s with 0;

2.Set i=0;

3.Search all unlabeled vertices adjacent to a vertex labeled i;

4.Label all vertices just found with i+1;

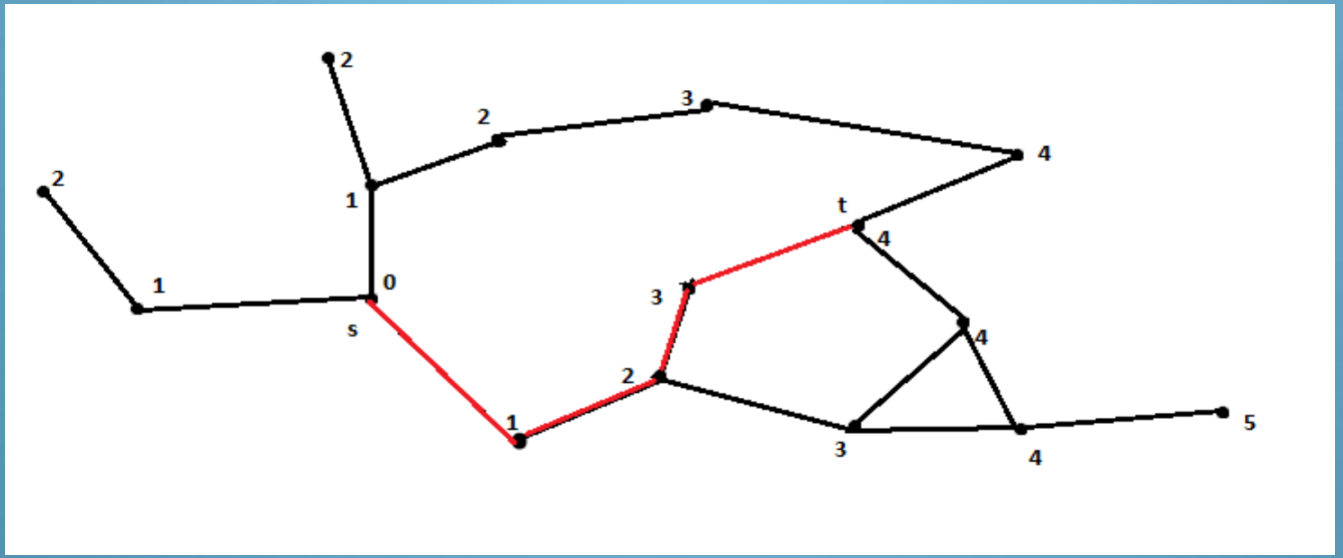
5.If the terminal point t is found, stop the process and output the shortest path. Otherwise, loop back to the step 3 and keep going on.

# IMPORTANT ASSUMPTION

- Every edge between any two vertices has the same length.

- Problem changed from finding out the shortest path to finding out the path with the smallest number of edges.

# EXAMPLE




Solution:

1.Label the start point s with 0;

2.Search all unlabeled vertices adjacent to the point 0 and label them with 1;

3.Repeat the previous works to vertices with new labels, until getting the terminal point t labeled;

4.The number that terminal point labeled with is the shortest length, and the red path is the shortest

# Dijkstra's Algorithm

- Kind of similar to the BFS method

- Each vertex is labeled in either two ways

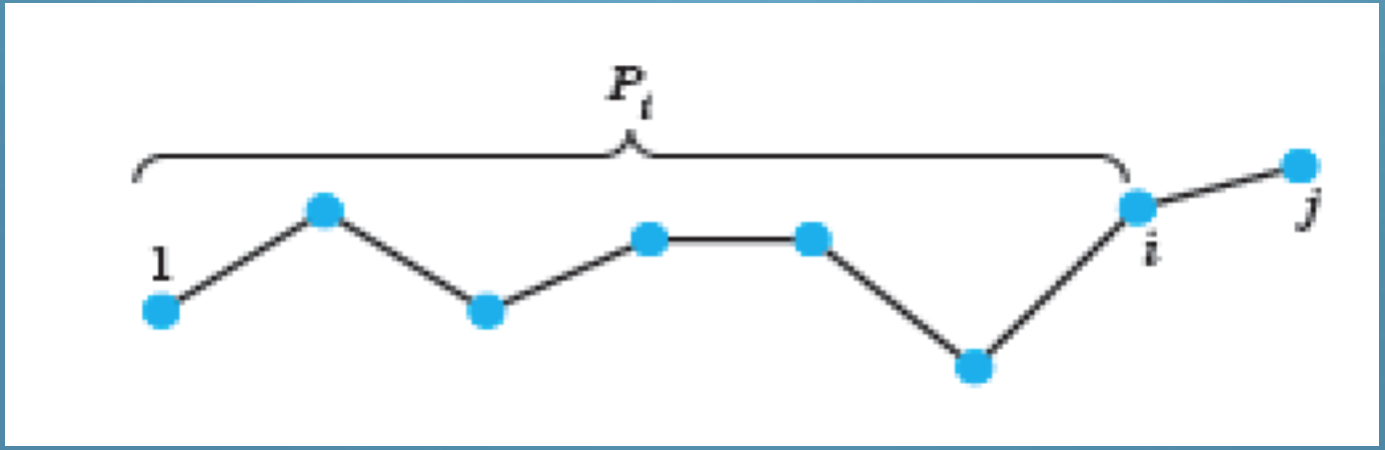
1.A permanent label $L_j$ for the length of a shortest path $1 \rightarrow j$;

2.A temporary label with stared $L_j^*$ for the length of a shortest path $1 \rightarrow j$;

3.Set up two sets(stacks) of ***ps*** and ***ts*** respectively for the permanent labels and temporary ones.

# Bellman's Optimality Principle

If **P**: **1→ j** is a shortest path from 1 to **j** in graph ***G*** and **( i, j)** is the last edge of $\mathbf{P}_j$ ( as picture following), then $\mathbf{P}_i$ **:  1→ j** [ obtained by dropping ( i, j) from $P_j$] is a shortest path **1→ j.**



# Pseudcode

**Input**: Number of vertices **n**, edges **( i, j ),** and length $l_{ij}$

**Output:** Lengths $L_j$ of the shortest path **1→ j, j = 2, 3, …, n**

1.Initial Step

Vertex 1gets PL: $L_1=0$

Vertex **j (= 2,…,n)** gets TL: $L_j^*=l_{1j}$ (=∞ if there is no edge **( i, j )** in ***G***).

Set ***ps***={1}, ***ts***={2,3,…,n}.

2.Fixing a permanent label

Find a **k** in ***ts*** for which $L_k^*$ is minimum, set $L_k= L_k^*$. Take the smallest ***k*** if there are several. Delete **k** from ***ts*** and include it in ***ps***.

If ***ts*** $=\emptyset$ (that is, ***ts*** is empty) then

OUTPUT $L_2,\ldots,L_n$, stop

Else continue (that is, go to Step 3).

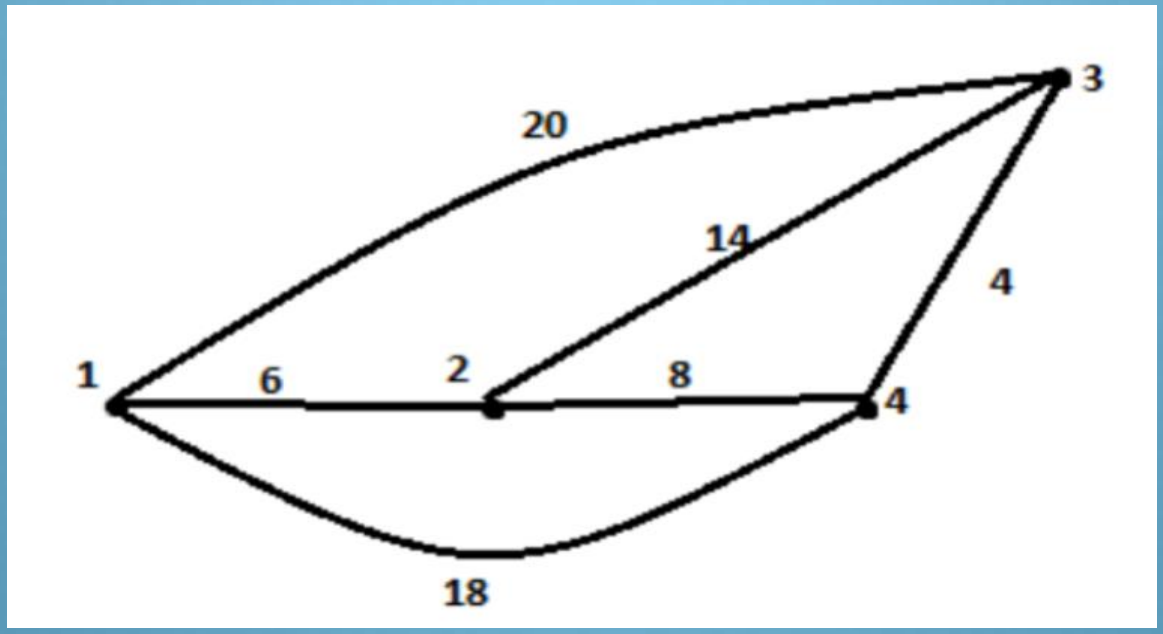
3.Updating temporary labels

For all j in ***ts***, set $L_j^* = \min \{ L_j^* , L_k + l_{kj}\}$

Go to Step 2

END

# Example




Question: find out the shortest path 1→2，3，4

**Solution**. The steps and computations.

Step 1. $L_1$=0, $L_2$*= 6, $L_3$*=20, $L_4$*=18 — ***ps*** ={1}, ***ts***={2，3，4}

Step 2. $L_2$ = min {$L_2$*, $L_3$*, $L_4$*}= 6, k=2 — ***ps*** ={1,2}, ***ts***={3，4}

Step 3. $L_3$*=min {20, $L_2$ +$L_{23}$}= min {20,6+14}=20,

$L_4$*=min {18, $L_2$ +$L_{24}$}= min {18,6+8}=14,

Step 2. $L_4$ =min {$L_3$*, $L_4$*} =14,**k**=4 — ***ps*** ={1,2,4}, ***ts***={4}

Step 3. $L_3$*=min {20, $L_4$ + $l_{43}$}= min {20, 14+4}=18

Step 2. ***L3***=18, k=3 — ***ps*** ={1,2,3,4}, ***ts*** =∅

Figure 1.4 shows the resulting shortest paths, of lengths $L_2$=6, $L_3$=18, $L_4$=14

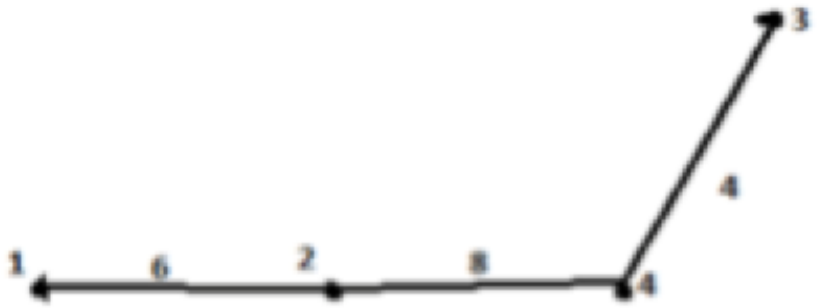



Figure 1.4 - Shortest Paths From Vertex 1 To Vertices 2, 3, 4

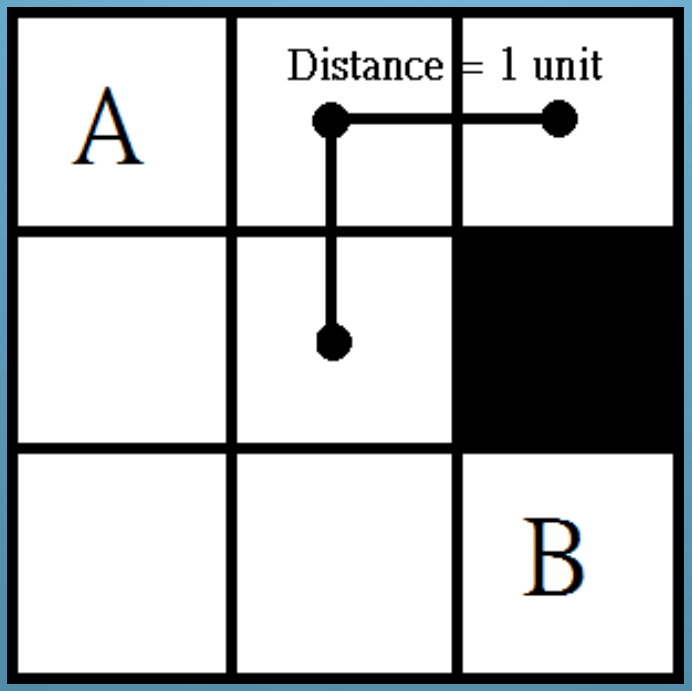
# Mathematical presentation of problem

Assumptions for this problem

- The distance between the nodes is 1 unit.
- Turing a corner has no impact on the distance travelled.
- The edge between nodes are bidirectional

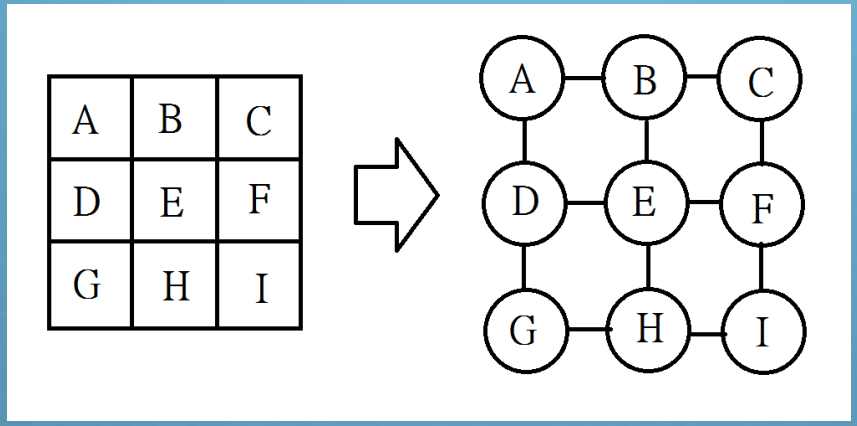
# Mathematical presentation of problem

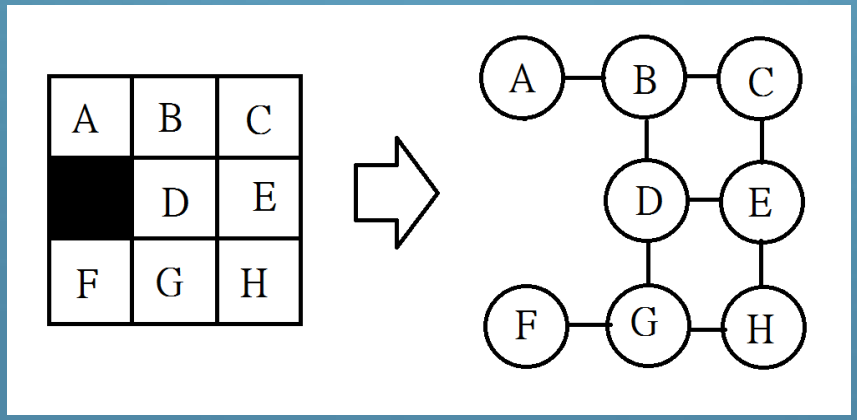
1. create maps



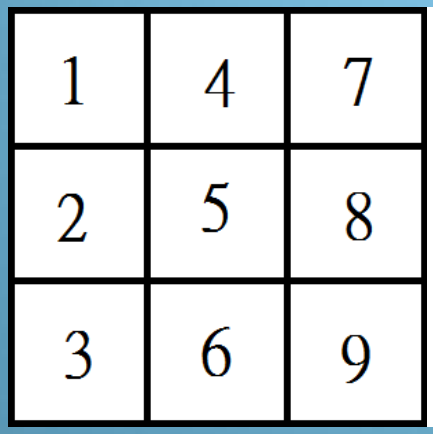
# Mathematical presentation of problem

## 2. transfer maps into graphs



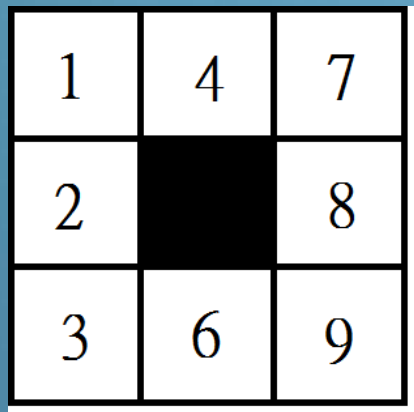



# Transforming the designed maps into graphs




node_map1 =

$$\begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 3 \\ 2 & 5 \\ 3 & 6 \\ 4 & 5 \\ 4 & 7 \\ 5 & 6 \\ 5 & 8 \\ 6 & 9 \\ 7 & 8 \\ 8 & 9 \end{bmatrix}$$

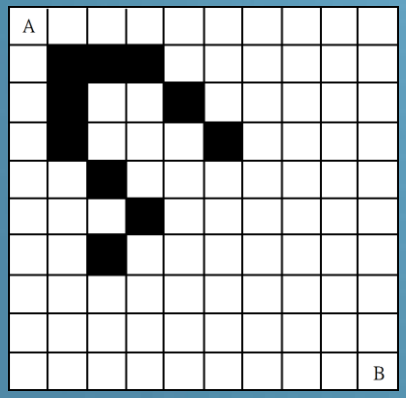



node_map1 =

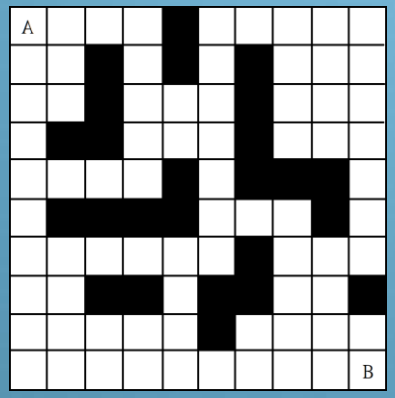
$$\begin{bmatrix} 1 & 2 \\ 1 & 4 \\ 2 & 3 \\ 3 & 6 \\ 4 & 7 \\ 6 & 9 \\ 7 & 8 \\ 8 & 9 \end{bmatrix}$$

# Mathematical presentation of problem

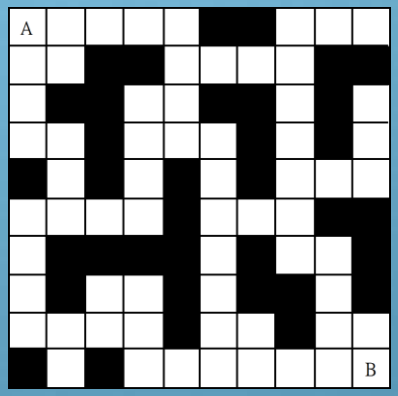
maps



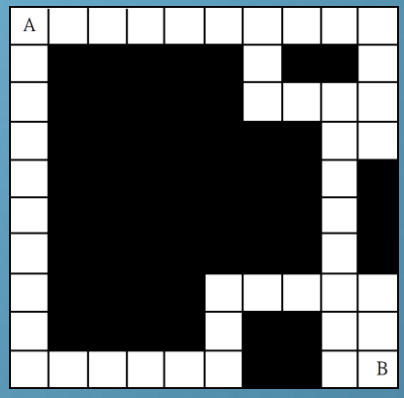

(a)




(b)



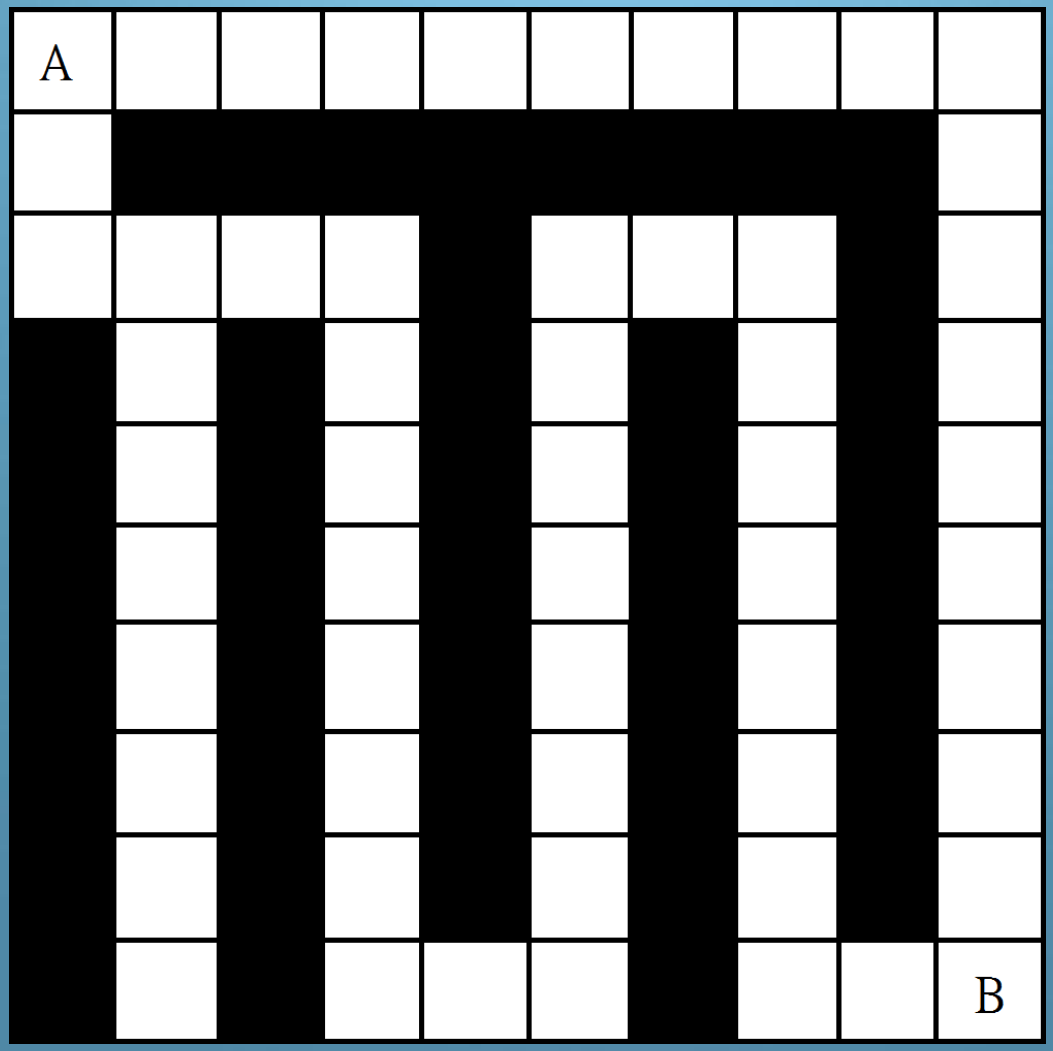

(c)




(d)

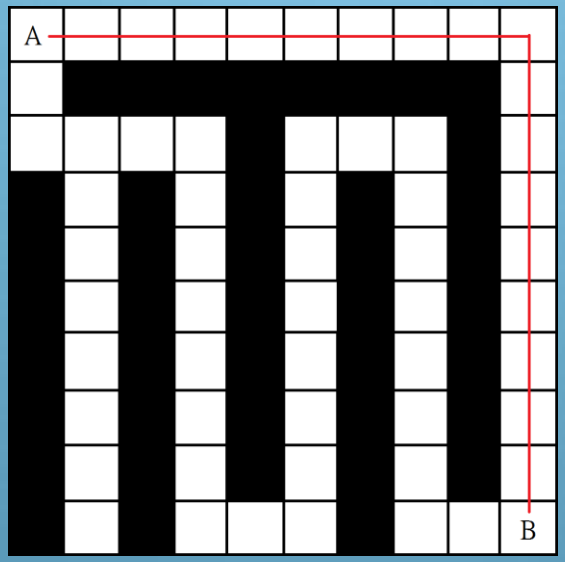
Maps (a), (b), (c), (d) have been developed with 10, 25, 36 and 55 obstacles

# Reference map



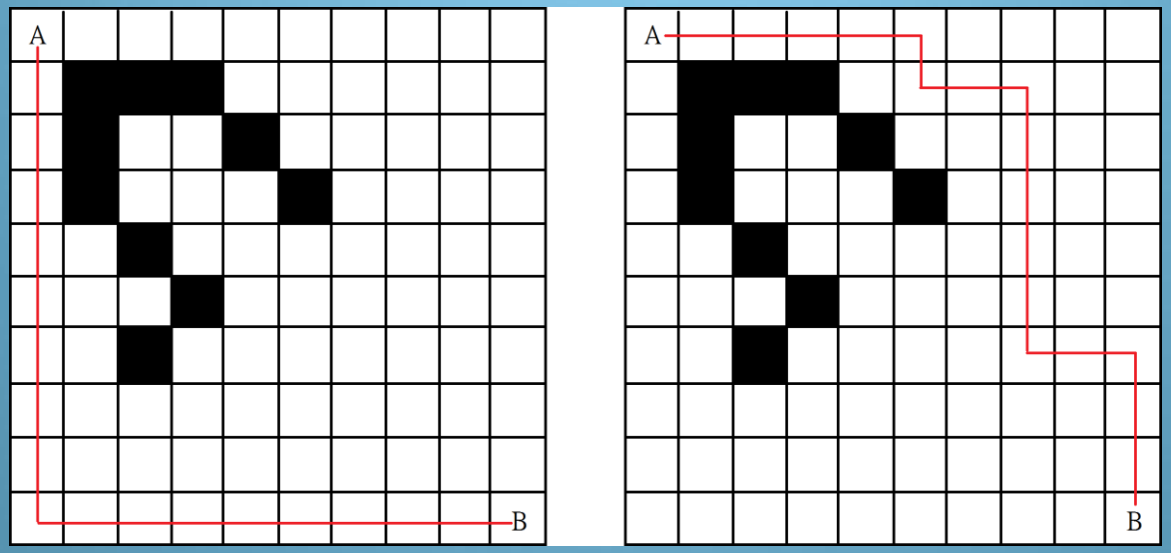
# Reference map

Upon the application of the MATLAB functions to the reference map, the same shortest path was found successfully by both methods.




| | Moore's | Dijkstra's |
|---|---|---|
| Reference Map (average time) | 0.000733s | 0.000743s |
| Shortest path (node – node) | 1-11-21-31-41-51-61-71-81-91-92-93-94-95-96-97-98-99-100(18steps) | 1-11-21-31-41-51-61-71-81-91-92-93-94-95-96-97-98-99-100(18steps) |

# Maps with different number of obstacles

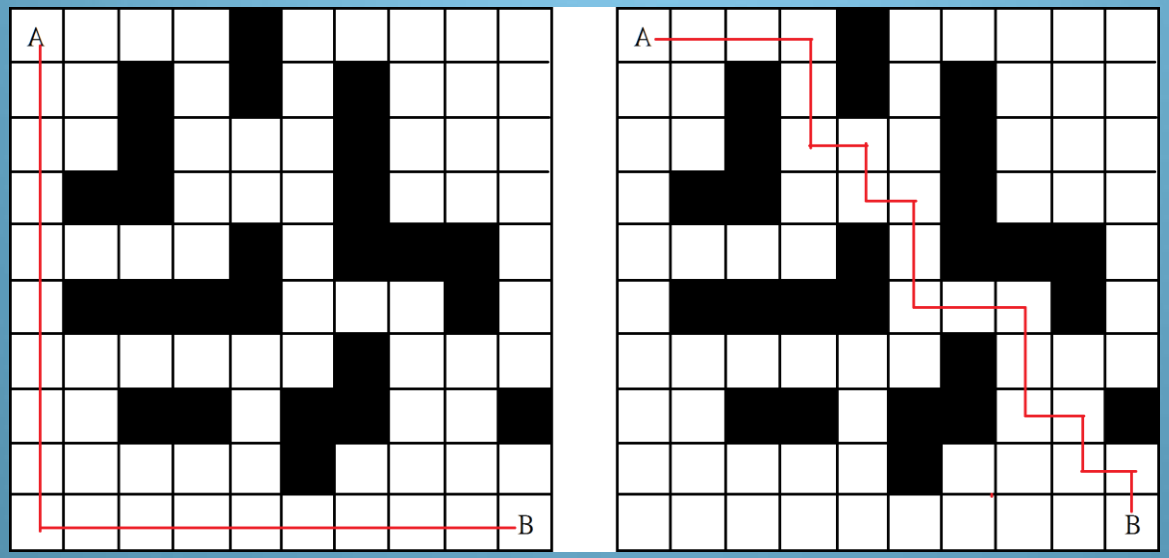



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map10 (average time) | 0.000813s | 0.000848s |
| Shortest path (node – node) | 1- 2- 3- 4- 5- 6- 7- 8- 9-10-20-30-40-50-80-90-100(18 steps) | 1-11-21-31-41-51-52-62-72-73-74-75-76-77-87-97- 98-99-100 (18 steps) |

# Maps with different number of obstacles

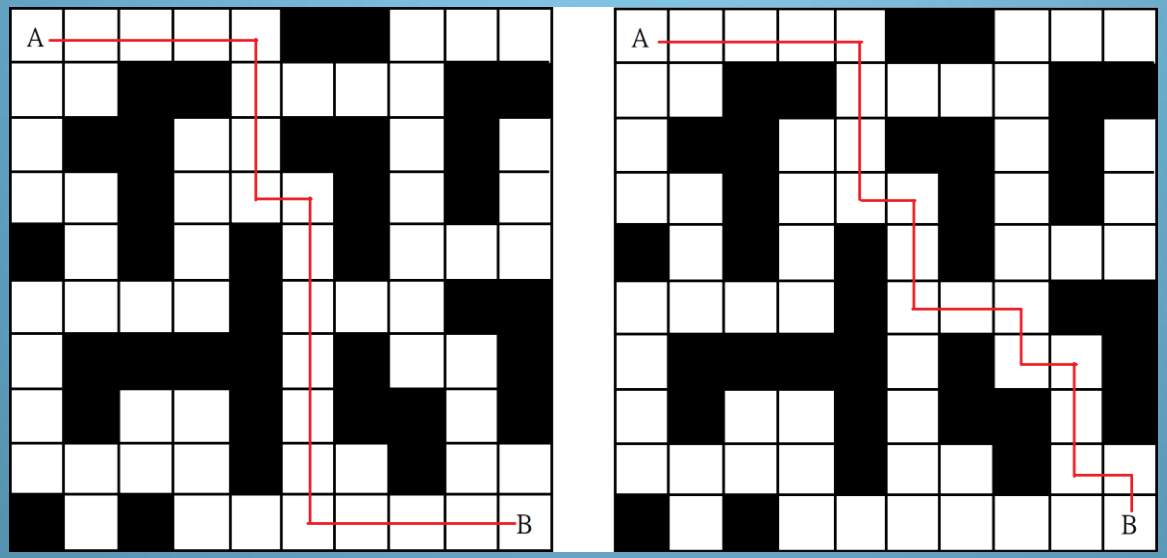



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map25 (average time) | 0.000750s | 0.000781s |
| Shortest path (node – node) | 1- 2- 3- 4- 5- 6- 7- 8- 9-10-20-30-40-50-80-90-100(18 steps) | 1-11-21-31-32-33-43-44-54-55-56-66-76-77-78-88-89-99-100 (18 steps) |

# Maps with different number of obstacles

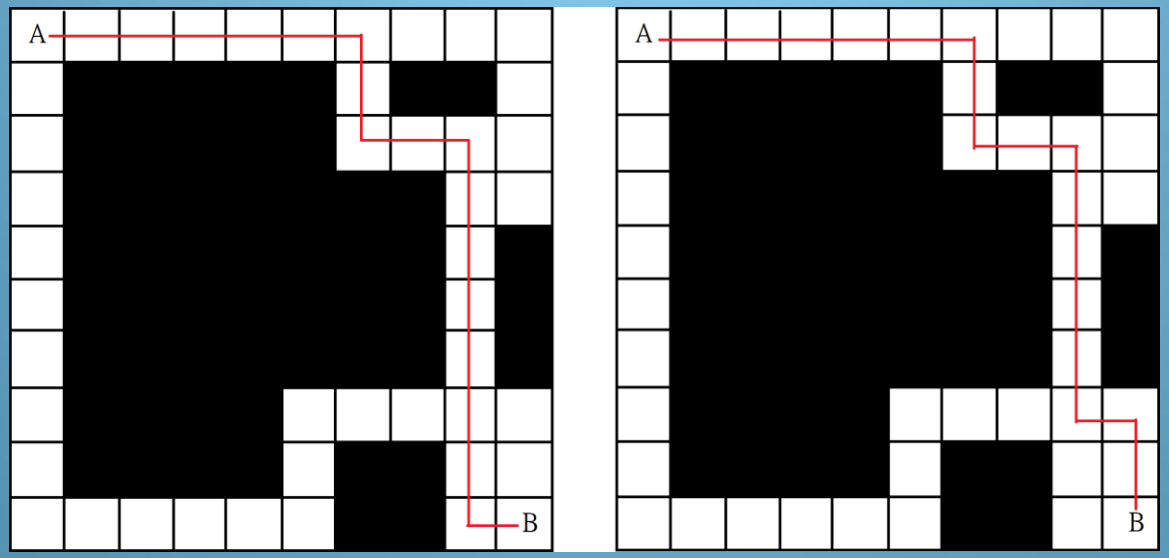



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map36 (average time) | 0.000724s | 0.000769s |
| Shortest path (node – node) | 1-11-21-31-41-42-43-44-54-55-56-57-58-59-60-70-80-90-100(18 steps) | 1-11-21-31-41-42-43-44-54-55-56-66-76-77-87-88-89-99-100(18 steps) |

# Maps with different number of obstacles

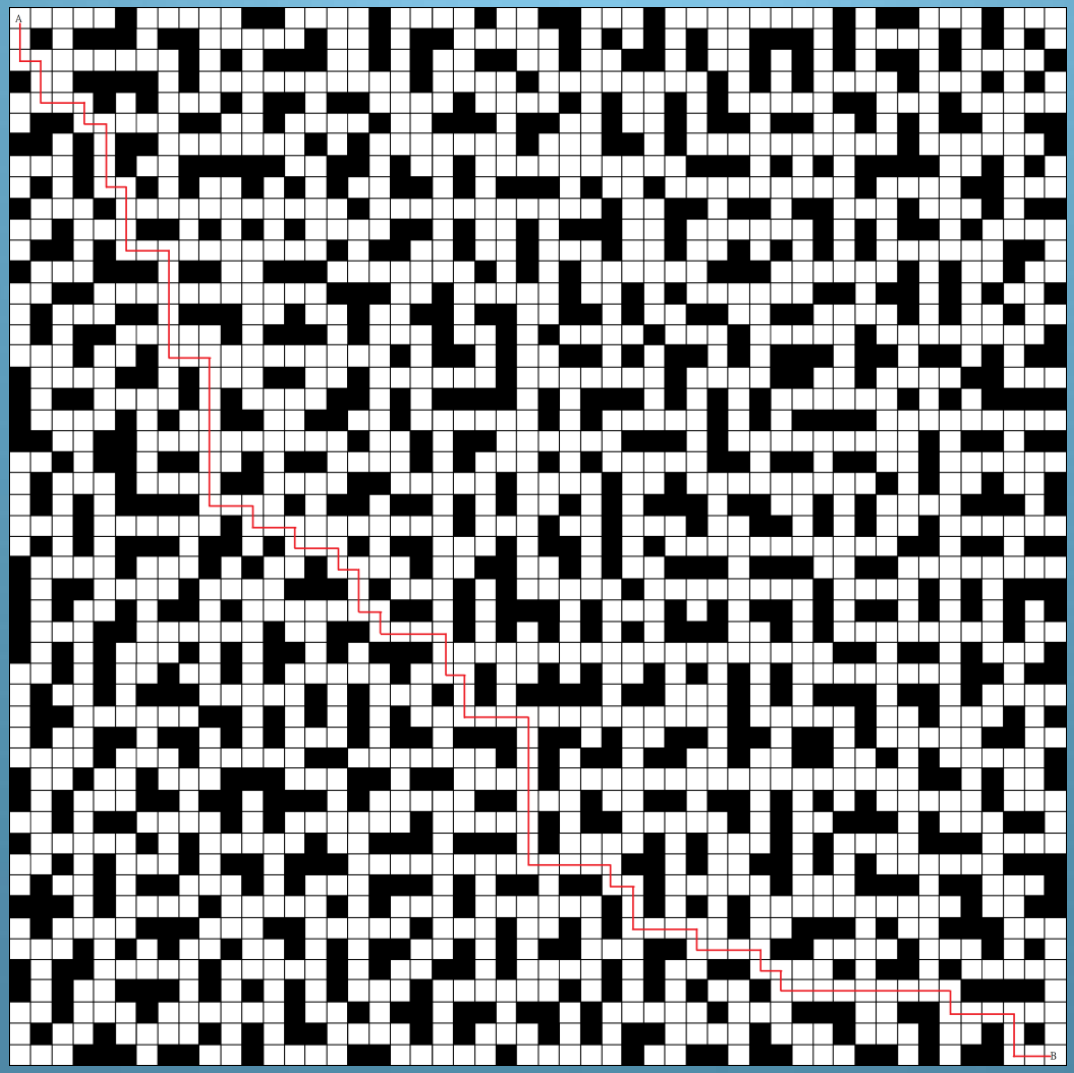



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

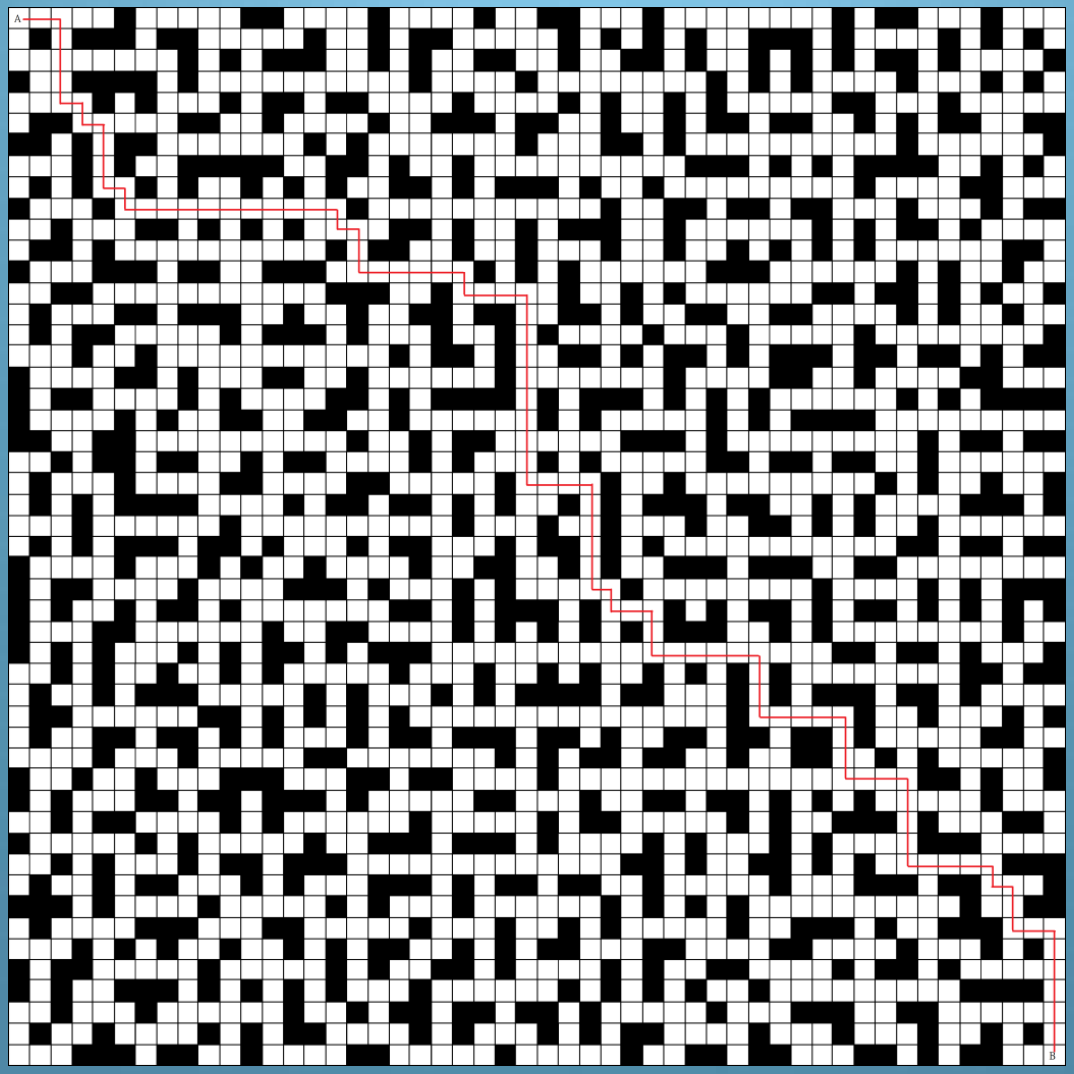
| | Moore's | Dijkstra's |
|---|---|---|
| Map50 (average time) | 0.000713s | 0.000742s |
| Shortest path (node – node) | 1-11-21-31-41-51-61-62-63-73-83-84-85-86-87-88-89-90-100(18 steps) | 1-11-21-31-41-51-61-62-63-73-83-84-85-86-87-88-98-99-100(18steps) |

# 50x50 Map with 900 obstacles




the shortest path – Moore’s method

# 50x50 Map with 900 obstacles



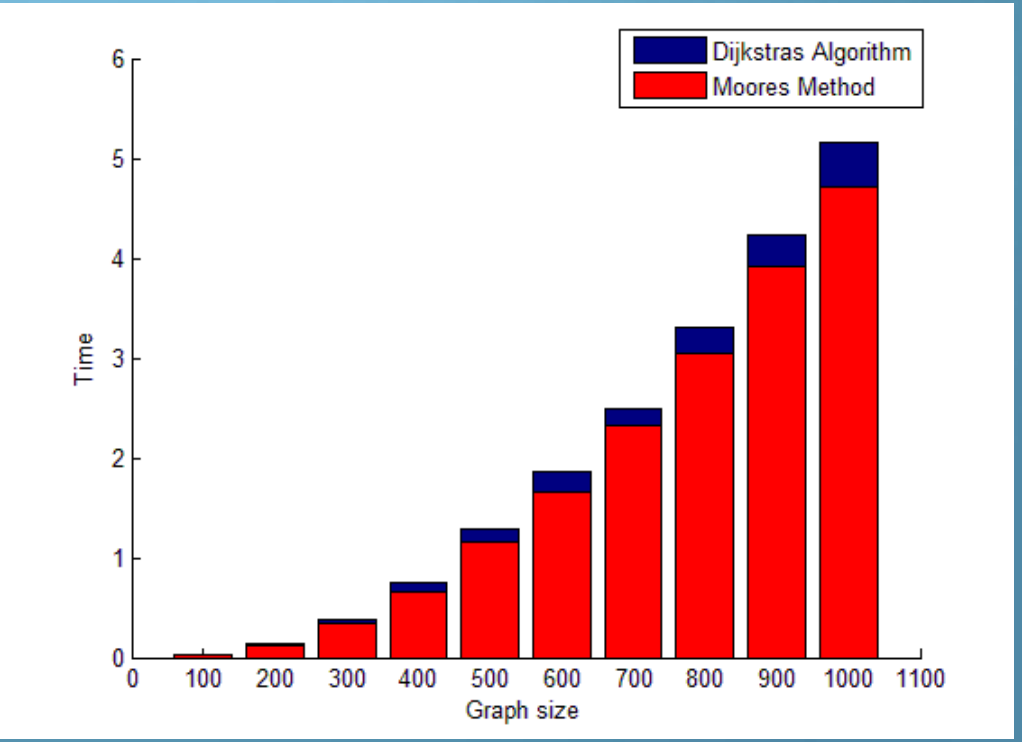

the shortest path – Moore’s method

# 50x50 Map with 900 obstacles

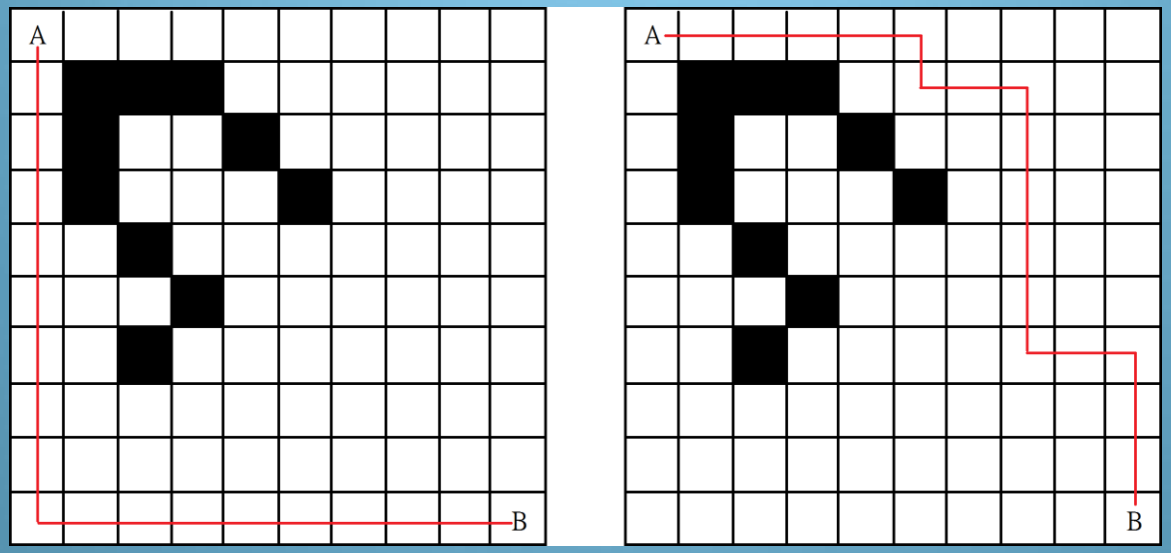
| | Moore's | Dijkstra's |
|---|---|---|
| Map 50x50 (no obstacles) | 0.006550s | 0.006604s |
| Map 50x50 (average time) | 0.00444s | 0.004850s |
| Shortest path (node – node) | 1-2-3-53-54-55-105-155-156-206-207-208-209-259-260-261-262-312-362-363-364-365-366-367-417-467-468-469-470-471-472-473-474-524-574-575-625-675-676-726-776-777-827-828-829-879-880-930-980-1030 1031-1032-1082-1083-1084-1134-1184-1234-1235-1236-1237-1238-1239-1240-1241-1291-1341-1391-1441-1442-1492-1493-1494-1544-1594-1644-1645-1695-1745-1795-1796-1846-1847-1897-1947-1997-2047-2097-2147-2197-2247-2248-2298-2348-2398-2399-2400-2450-2500 (98 steps) | 1-51-101-102-103-104-105-155-156-206-207-208-209-259-260-310-360-410-460-510-560-610-660-710-760-761-811-812-813-863-913-963-1013-1063-1064-1114-1164-1214-1215-1216-1217-1218-1219-1220-1221-1222-1223-1273-1323-1373-1374-1375-1376-1377-1378-1428-1429-1479-1529-1530-1531-1581-1631-1681-1731-1781-1782-1783-1784-1834-1884-1934-1984-1985-1986-1987-2037-2087-2137-2138-2139-2140-2141-2191-2241-2291-2341-2342-2392-2393-2394-2444-2494-2495-2496-2497-2498-2499-2500 (98 steps) |

# Timing Results

| | Moore's | Dijkstra's |
|---|---|---|
| 100x100 | 0.023097 | 0.02613 |
| 200x200 | 0.127722 | 0.141696 |
| 300x300 | 0.346553 | 0.383323 |
| 400x400 | 0.661585 | 0.756057 |
| 500x500 | 1.160261 | 1.287945 |
| 600x600 | 1.660201 | 1.852375 |
| 700x700 | 2.321567 | 2.486482 |
| 800x800 | 3.039056 | 3.296351 |
| 900x900 | 3.923256 | 4.225479 |
| 1000x1000 | 4.714555 | 5.157708 |



# Maps with different number of obstacles

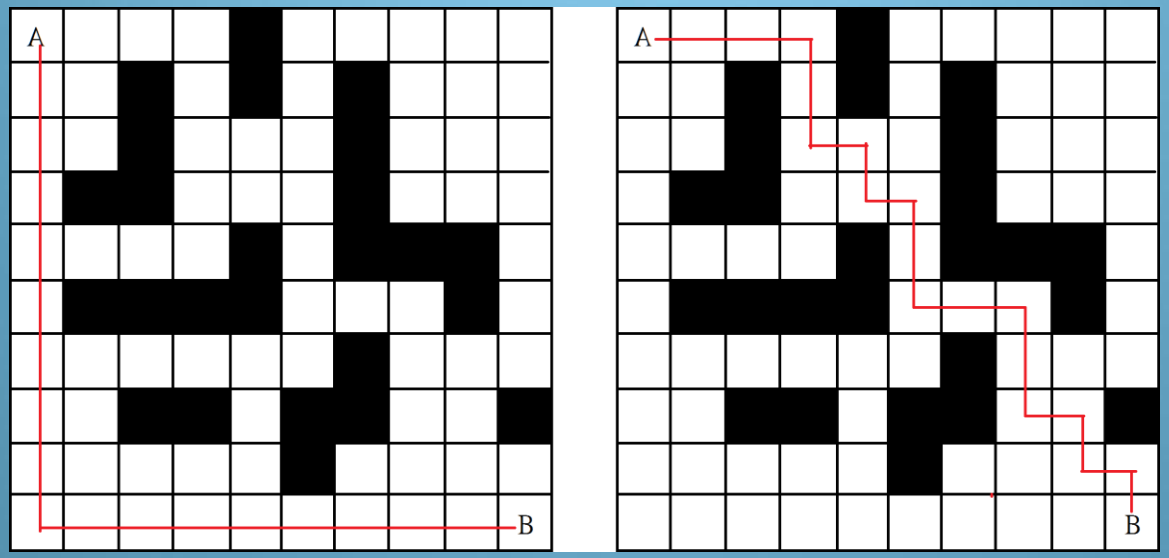



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map10 (average time) | 0.000813s | 0.000848s |
| Shortest path (node – node) | 1- 2- 3- 4- 5- 6- 7- 8- 9-10-20-30-40-50-80-90-100(18 steps) | 1-11-21-31-41-51-52-62-72-73-74-75-76-77-87-97- 98-99-100 (18 steps) |

# Maps with different number of obstacles

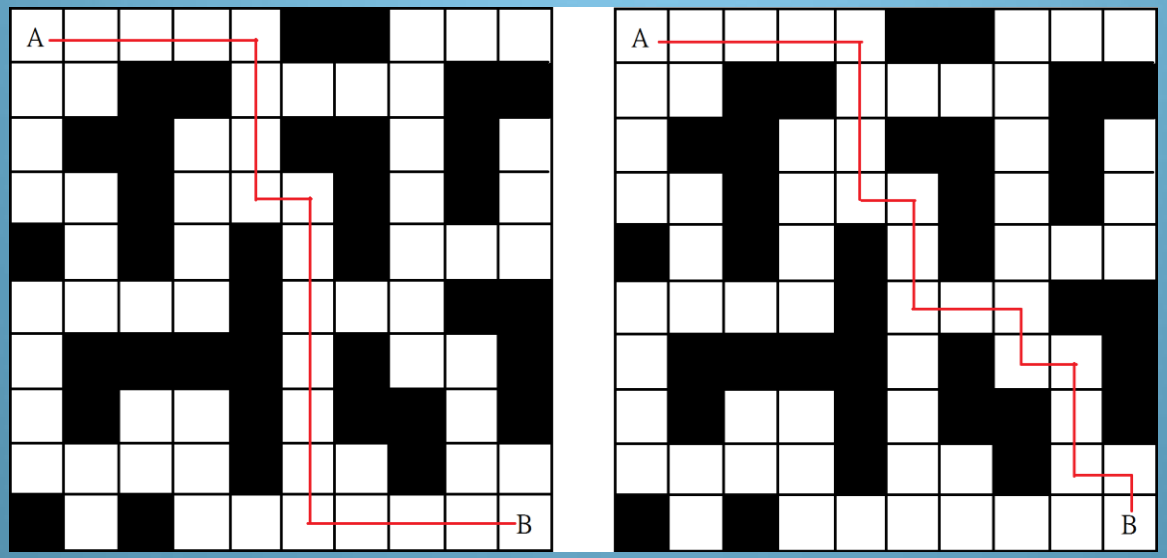



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map25 (average time) | 0.000750s | 0.000781s |
| Shortest path (node – node) | 1- 2- 3- 4- 5- 6- 7- 8- 9-10-20-30-40-50-80-90-100(18 steps) | 1-11-21-31-32-33-43-44-54-55-56-66-76-77-78-88-89-99-100 (18 steps) |

# Maps with different number of obstacles

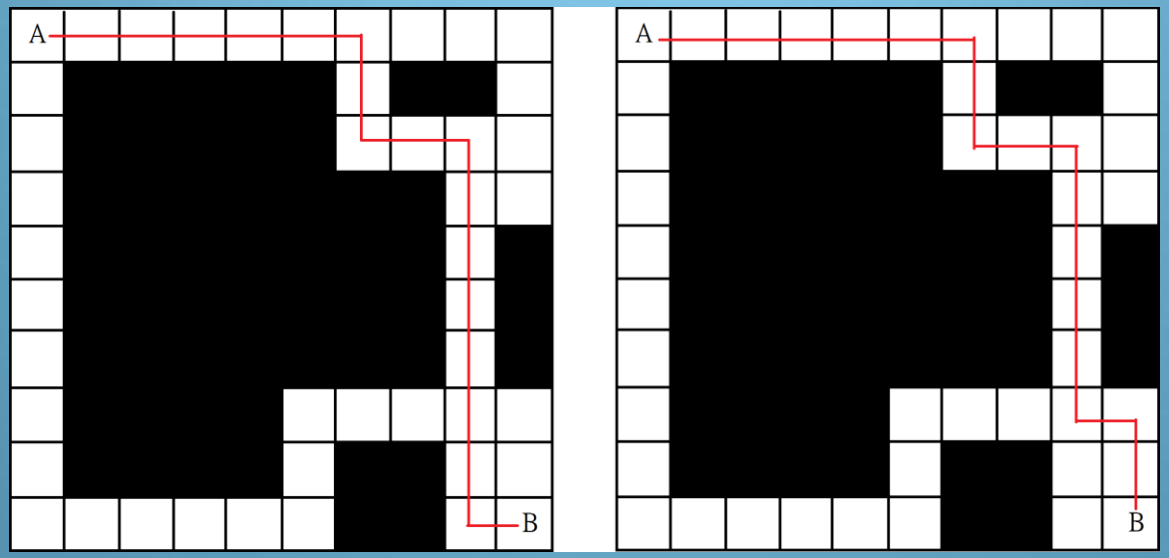



(a) (b)

(a)Moore's method (b)Dijkstra's Algorithm

| | Moore's | Dijkstra's |
|---|---|---|
| Map36 (average time) | 0.000724s | 0.000769s |
| Shortest path (node – node) | 1-11-21-31-41-42-43-44-54-55-56-57-58-59-60-70-80-90-100(18 steps) | 1-11-21-31-41-42-43-44-54-55-56-66-76-77-87-88-89-99-100(18 steps) |

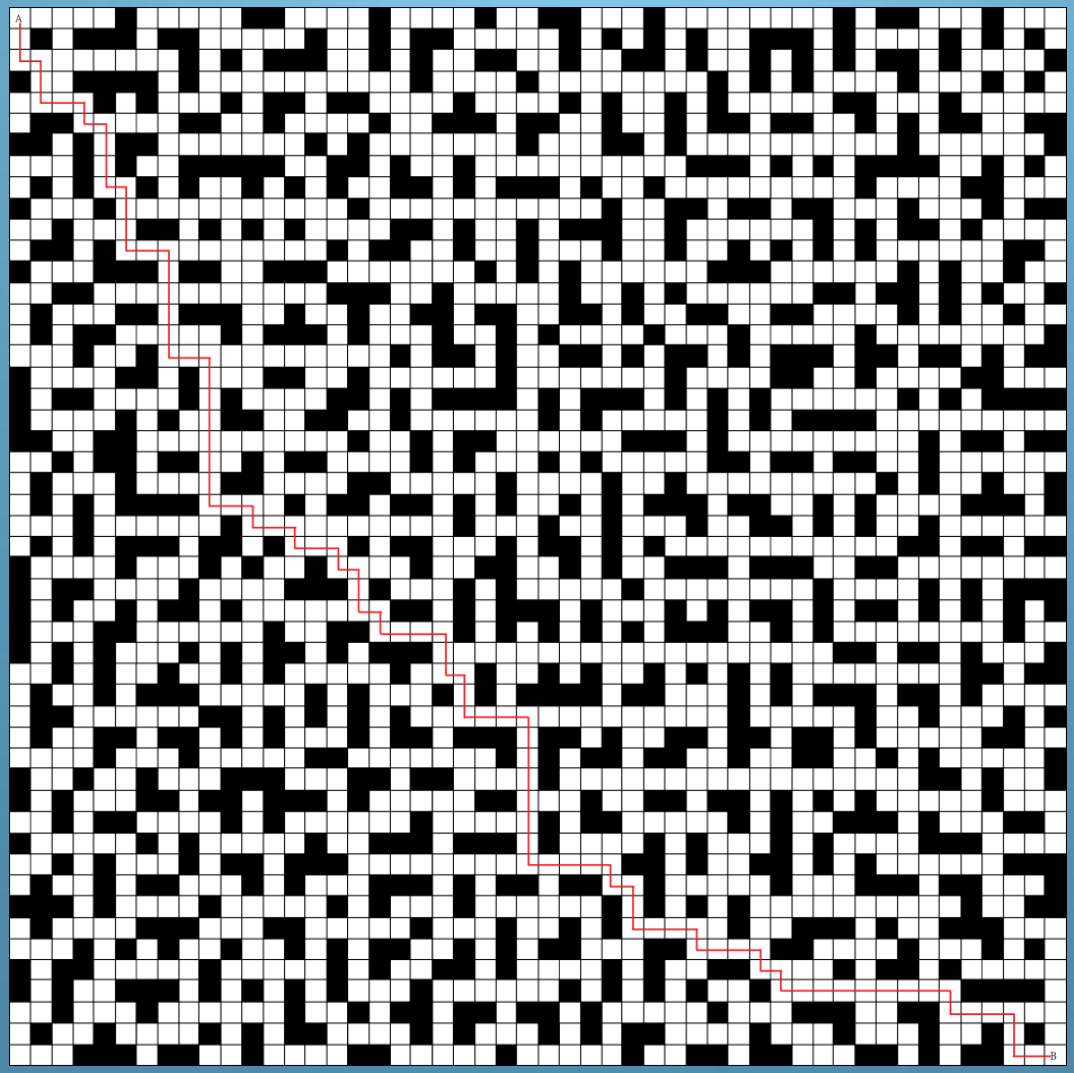
# Maps with different number of obstacles




(a) (b)

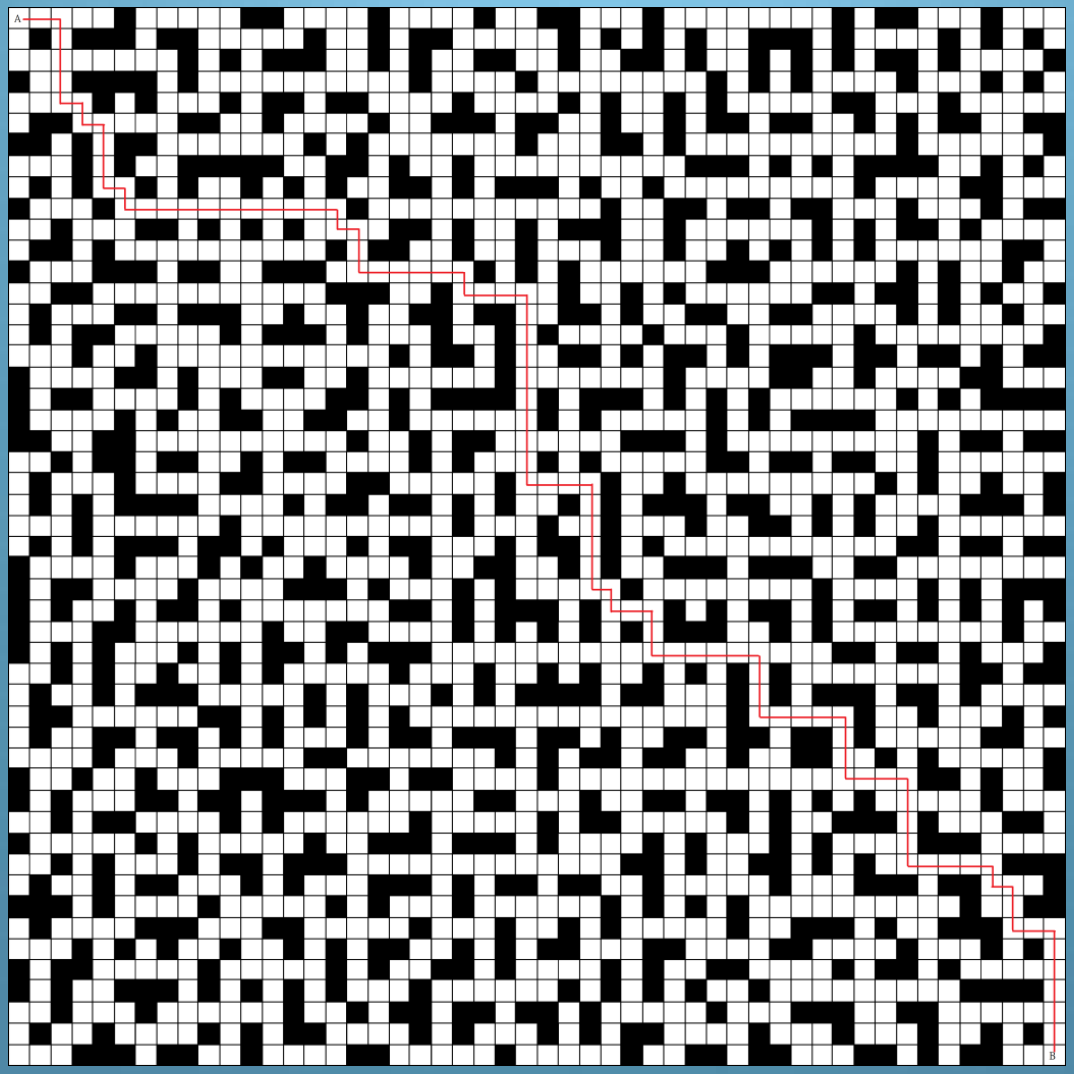
| | Moore's | Dijkstra's |
|---|---|---|
| Map50 (average time) | 0.000713s | 0.000742s |
| Shortest path (node – node) | 1-11-21-31-41-51-61-62-63-73-83-84-85-86-87-88-89-90-100(18 steps) | 1-11-21-31-41-51-61-62-63-73-83-84-85-86-87-88-98-99-100(18steps) |

# 50x50 Map with 900 obstacles




the shortest path – Moore’s method

# 50x50 Map with 900 obstacles



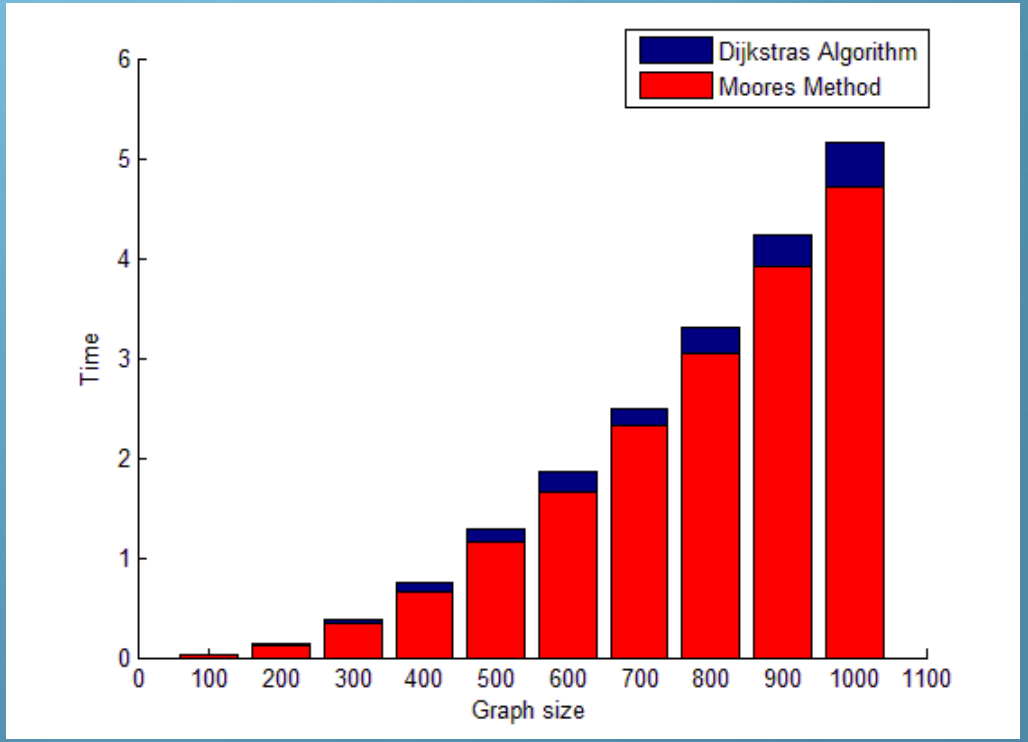

the shortest path – Moore’s method

# 50x50 Map with 900 obstacles

| | Moore's | Dijkstra's |
|---|---|---|
| Map 50x50 (no obstacles) | 0.006550s | 0.006604s |
| Map55 (average time) | 0.00444s | 0.004850s |
| Shortest path (node – node) | 1-2-3-53-54-55-105-155-156-206-207-208-209-259-260-261-262-312-362-363-364-365-366-367-417-467-468-469-470-471-472-473-474-524-574-575-625-675-676-726-776-777-827-828-829-879-880-930-980-1030 1031-1032-1082-1083-1084-1134-1184-1234-1235-1236-1237-1238-1239-1240-1241-1291-1341-1391-1441-1442-1492-1493-1494-1544-1594-1644-1645-1695-1745-1795-1796-1846-1847-1897-1947-1997-2047-2097-2147-2197-2247-2248-2298-2348-2398-2399-2400-2450-2500 (98 steps) | 1-51-101-102-103-104-105-155-156-206-207-208-209-259-260-310-360-410-460-510-560-610-660-710-760-761-811-812-813-863-913-963-1013-1063-1064-1114-1164-1214-1215-1216-1217-1218-1219-1220-1221-1222-1223-1273-1323-1373-1374-1375-1376-1377-1378-1428-1429-1479-1529-1530-1531-1581-1631-1681-1731-1781-1782-1783-1784-1834-1884-1934-1984-1985-1986-1987-2037-2087-2137-2138-2139-2140-2141-2191-2241-2291-2341-2342-2392-2393-2394-2444-2494-2495-2496-2497-2498-2499-2500 (98 steps) |

# Timing Results

| | Moore's | Dijkstra's |
|---|---|---|
| 100x100 | 0.023097 | 0.02613 |
| 200x200 | 0.127722 | 0.141696 |
| 300x300 | 0.346553 | 0.383323 |
| 400x400 | 0.661585 | 0.756057 |
| 500x500 | 1.160261 | 1.287945 |
| 600x600 | 1.660201 | 1.852375 |
| 700x700 | 2.321567 | 2.486482 |
| 800x800 | 3.039056 | 3.296351 |
| 900x900 | 3.923256 | 4.225479 |
| 1000x1000 | 4.714555 | 5.157708 |



# Conclusion

- The Moore's method is more time efficient than the Dijkstra's algorithm for  the applications in this project

- As the map gets larger, the time required for both methods take longer

- The Dijkstra's algorithm would be more suitable for analysing weighted graphs

# Future work

- Improve MATLAB algorithm to produce weighted maps for Dijkstra's algorithm

- Investigate the effect of adding distance weighting to corners