

EPFL, CIVIL-127

Programming and software development for engineers

Final Project

I Like Trains - Alok

The grid contains the following shapes:

- A red 2x3 rectangle in the upper-left quadrant.
- A green 2x2 L-shape in the lower-center.
- A blue 5x2 L-shape in the lower-right.
- Two red dots labeled 1 and 3 in the top row.

LEADERBOARD

Rank	Player	Score	Best
1	Alok	9	9
2	Bob	0	7

Time remaining: 00:30

Final Project

- You have to implement a train controller. Your train can turn left, right, or go straight. You score points by picking up and dropping off passengers
- The code runs as a client + server
 - The server runs the game logic
 - One or more clients connect to the server to play against each other
- You can run your own server locally for debugging purpose
- Submission
 - You will be submitting client/agent.py
 - Don't modify other files
 - You can add new files
 - Update requirements.txt if you need additional packages
- Do report any bugs you find in the server code!

Final Project

- Two-student teams
- Deadline May 23rd, 2025, 6pm to submit your code
- Grading
 - Does your code run and do something?
 - Description of your implementation, in one slide
 - Beat the staff's implementations
 - You'll earn points per staff's AI you beat
 - Is your code well structured, documented, and tested (when applicable)

Final Project

- You can use the server to practice as many times as you wish
- You can also play against each other
- We expect each submission to be unique
 - You can discuss ideas with your peers but you shouldn't be sharing code with each other

Networking Basics

Networking Basics

- Computer networks enable exchanging data by implementing protocols
- A protocol is an agreement on how to behave on a network
- Modern computer networks use layers of protocols
 - With layers, we can have the same higher level protocol (e.g. http) work on top of different physical layers (e.g. 5G, wifi, Ethernet, etc.)
 - Each layer provides a specific functionality
 - E.g. the lowest level defines radio frequencies, voltage levels, and how to avoid everyone talking at the same time
 - Higher level protocols define how to talk to computers located anywhere in the world
- We'll only talk about UDP/IP today

Networking Basics

- Alice's computer wants to say "hi" to Bob's computer
- Every computer on our network has a unique address, allowing each computer to communicate with each other
 - Alice has address AA
 - Bob has address BB

Networking Basics

- Alice can send a packet to Bob. A packet contains the source, destination, and data (and some other things):
 - Source: AA
 - Destination: BB
 - Data: "hi"
- When Bob receives the message from Alice, the packet's source address can be used to send a response:
 - Source: BB
 - Destination: AA
 - Data: "howdy?"

Networking Basics

- If Alice's computer is not directly connected to Bob's, she can send her packet to her router. The router forwards the data to another router, until it eventually reaches Bob
- Routers are just computers, interconnected with two or more links. Each router maintains dynamic routing tables so it knows where each address is located in an ever changing network

UDP/IP

- UDP/IP is a packet-based, switched, communication protocol
- A server, identified by an IP address, listens for connections on a given port
- A client, identified by an IP address, connects to the server on the agreed upon port
- IP addresses are numbers. An IP address can be an IPv4 or an IPv6
 - IPv4 is 32-bits, e.g. 172.217.168.14
 - IPv6 is 128-bits long, e.g. 2001:db8:3333 :4444:5555:6666:7777:8888
- Ports are 16 bits numbers (1-65535)
 - We'll use 5555 for the project
- IP addresses can be public (routable over the internet) or local (private)
 - If your computer has a private IP address, it can connect to Internet servers by having a router shuffle traffic between the public and private networks. This is the typical home network setup

UDP/IP

- Packet-based == data is sent in small chunks. Each chunk contains
 - Source IP address
 - Destination IP address
 - Source port
 - Destination port
 - A couple things you don't need to care about
 - Data

UDP/IP

- Switched network == your packet gets sent to various routers before (maybe) reaching its destination
 - (typically) Your home router has one upstream internet connection, it therefore forwards every incoming packet to the upstream internet connection
 - Internet routers are interconnected in a mesh fashion. Each router has multiple ways to forward packets. Routers implement protocols to establish routing tables and then use those tables to decide where a packet must go
- You can use **tracroute** (Mac/Linux) or **tracert** (Windows) to observe a route your packets are taking. The route changes dynamically

UDP/IP

```
$ traceroute google.com
traceroute to google.com (142.250.203.110), 64 hops max, 52 byte packets
 1  192.168.1.1 (192.168.1.1)  3.932 ms  3.466 ms  4.202 ms
 2  10.245.32.1 (10.245.32.1)  6.767 ms  5.003 ms  8.052 ms
 3  172.22.100.9 (172.22.100.9)  31.527 ms  4.490 ms  4.112 ms
 4  37.0.32.181 (37.0.32.181)  4.433 ms  30.128 ms  4.506 ms
 5  213.230.56.68 (213.230.56.68)  5.249 ms  4.695 ms
    213.230.56.70 (213.230.56.70)  4.544 ms
 6  10.248.107.10 (10.248.107.10)  35.346 ms
    10.248.7.10 (10.248.7.10)  7.033 ms
    10.248.107.10 (10.248.107.10)  7.518 ms
 7  10.253.245.1 (10.253.245.1)  7.555 ms  44.885 ms
    100.65.99.5 (100.65.99.5)  6.785 ms
 8  213.230.54.254 (213.230.54.254)  7.599 ms
    100.65.99.5 (100.65.99.5)  9.188 ms  8.237 ms
 9  213.230.52.147 (213.230.52.147)  103.117 ms
    213.230.54.254 (213.230.54.254)  8.281 ms
    213.230.52.147 (213.230.52.147)  11.335 ms
10  74.125.51.48 (74.125.51.48)  11.791 ms
    213.230.52.147 (213.230.52.147)  13.224 ms  12.316 ms
11  142.251.245.165 (142.251.245.165)  12.244 ms
    142.251.76.107 (142.251.76.107)  13.639 ms
    74.125.51.48 (74.125.51.48)  12.003 ms
12  142.251.70.185 (142.251.70.185)  11.874 ms
    142.251.245.165 (142.251.245.165)  13.537 ms  13.060 ms
13  zrh04s16-in-f14.1e100.net (142.250.203.110)  95.542 ms  12.145 ms
    142.251.70.187 (142.251.70.187)  12.725 ms
```

- Packets between my home and Google's servers

UDP/IP

- Communication Protocol == there's an agreed upon way computers to talk to each other
- Servers: listens for incoming packets
- Clients: picks a random source port and sends an initial packet
- The client and server can then send data to each other
- Connection terminates after some amount of inactivity (timeout)

UDP/IP

- UDP is a best effort protocol
 - Packets can get dropped
 - Packets can arrive out of order
 - Packets can be duplicated
 - When using UDP, it is the application's responsibility to deal with these scenarios
- UDP is useful for real-time communication (voice, video) or when the application layer is able to deal with unreliable packet delivery

UDP/IP

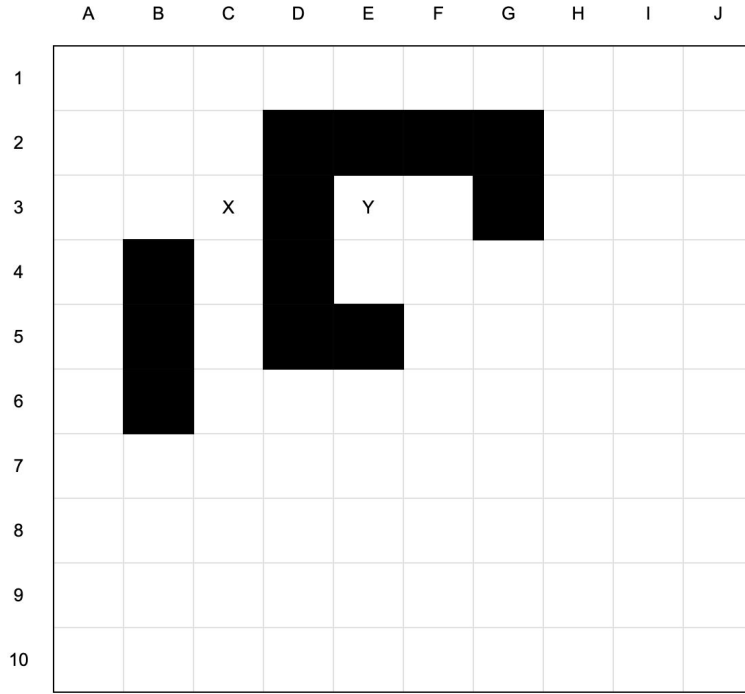
- You can start a server on your local computer and have it listen on a port, e.g. 127.0.0.1:5555 or localhost:5555
 - 127.0.0.1 or localhost refer to your computer
 - Only clients on your local computer will be able to connect to this server
 - Your firewall might tell you about this activity
- You can then start a client and have it connect to 127.0.0.1:5555 or localhost:5555

UDP/IP

- If you start your server on 0.0.0.0:5555
 - Any client on the same network as you will be able to connect to this server
 - You might have to configure your firewall

Path Finding

Problem Statement



- Given a grid with walls and two points (X and Y):
 - Find the shortest path between X and Y
- Variations:
 - Find a “reasonable” path (e.g. no longer than 2x shortest)
 - Walls are moving (dynamic environment)
 - etc.

Existing Algorithms

- Floodfill
- Dijkstra's algorithm
- A* algorithm
- D* algorithm
- And more...
 - ...but you can also invent your own algorithm!

Floodfill

- Set every cell's value to ∞
- Set the value at X to 0
- Until there's no more changes
 - Loop over every cell which isn't a wall
 - Set each cell's value to $\min(\text{neighbors}) + 1$

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞					∞	∞	∞
3	∞	∞	X		Y	∞		∞	∞	∞
4	∞		∞		∞	∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞					∞	∞	∞
3	∞	∞	0		Y	∞		∞	∞	∞
4	∞		∞		∞	∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	1	[Obstacle]				∞	∞	∞
3	∞	1	0					∞	∞	∞
4	∞	[Obstacle]		[Obstacle]	∞	∞	∞	∞	∞	∞
5	∞				∞	∞	∞	∞	∞	∞
6	∞	[Obstacle]		∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J	
1	∞	∞	2	3	4	5	6	7	8	9	
2	∞	2	1					8	9	10	
3	2	1	0	Y		∞			9	10	11
4	3			∞		∞	∞	10	11	12	
5	4					∞	∞	11	12	13	
6	5			3	4	5	6	7	8	9	10
7	6	7	4	5	6	7	8	9	10	11	
8	7	8	5	6	7	8	9	10	11	12	
9	8	9	6	7	8	9	10	11	12	13	
10	9	10	7	8	9	10	11	12	13	14	

	A	B	C	D	E	F	G	H	I	J
1	∞	3	2	3	4	5	6	7	8	9
2	3	2	1					8	9	10
3	2	1	0					9	10	11
4	3		1		Y	∞		10	11	12
5	4		2		∞	∞		11	12	13
6	5		3				11	12	13	14
7	6	5	4	5	6	7	8	9	10	11
8	7	6	5	6	7	8	9	10	11	12
9	8	7	6	7	8	9	10	11	12	13
10	9	8	7	8	9	10	11	12	13	14

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	9
2	3	2	1					8	9	10
3	2	1	0					9	10	11
4	3		1		Y	∞		9	10	11
5	4		2		∞	8		10	11	12
6	5		3			7	8	9	10	11
7	6	5	4	5	6	7	8	9	10	11
8	7	6	5	6	7	8	9	10	11	12
9	8	7	6	7	8	9	10	11	12	13
10	9	8	7	8	9	10	11	12	13	14

	A	B	C	D	E	F	G	H	I	J	
1	4	3	2	3	4	5	6	7	8	9	
2	3	2	1					8	9	10	
3	2	1	0	Y		9			9	10	11
4	3			1	9	8	9	10	11	12	
5	4			2			7	8	9	10	11
6	5			3	4	5	6	7	8	9	10
7	6	5	4	5	6	7	8	9	10	11	
8	7	6	5	6	7	8	9	10	11	12	
9	8	7	6	7	8	9	10	11	12	13	
10	9	8	7	8	9	10	11	12	13	14	

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	9
2	3	2	1	[REDACTED]	[REDACTED]	[REDACTED]	[REDACTED]	8	9	10
3	2	1	0					10	9	9
4	3	[REDACTED]	1		9	8	9	10	11	12
5	4		2		7	8	9	10	11	
6	5		3		4	5	6	7	8	9
7	6	5	4	5	6	7	8	9	10	11
8	7	6	5	6	7	8	9	10	11	12
9	8	7	6	7	8	9	10	11	12	13
10	9	8	7	8	9	10	11	12	13	14

10

	A	B	C	D	E	F	G	H	I	J			
1	4	3	2	3	4	5	6	7	8	9			
2	3	2	1					8	9	10			
3	2	1	0					10	9	9	10	11	
4	3		1					9	8	9	10	11	12
5	4		2						7	8	9	10	11
6	5		3						4	5	6	7	8
7	6	5	4	5	6	7	8	9	10	11			
8	7	6	5	6	7	8	9	10	11	12			
9	8	7	6	7	8	9	10	11	12	13			
10	9	8	7	8	9	10	11	12	13	14			

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	9
2	3	2	1					8	9	10
3	2	1	0		10	9		9	10	11
4	3		1		9	8	9	10	11	12
5	4		2			7	8	9	10	11
6	5		3	4	5	6	7	8	9	10
7	6	5	4	5	6	7	8	9	10	11
8	7	6	5	6	7	8	9	10	11	12
9	8	7	6	7	8	9	10	11	12	13
10	9	8	7	8	9	10	11	12	13	14

Floodfill

- Once completed, the shortest path starts from the destination and goes to a neighboring cell with value exactly one less
- Floodfill is a “slow” algorithm but
 - Relatively small to implement
 - Gives you the shortest path from X to every other point

Dijkstra's Algorithm

- Uses a queue
- Set every cell's value which isn't a wall to ∞
- Set the value at X to 0
- Put X in the queue
- Until you haven't reached Y:
 - Dequeue next element from queue
 - Update each neighbor's distance and put them in the queue

Note: Dijkstra's algorithm is usually described for graphs with weights on each edge (which is a more complicated case)

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞					∞	∞	∞
3	∞	∞	X		Y	∞		∞	∞	∞
4	∞		∞		∞	∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞					∞	∞	∞
3	∞	∞	0					∞	∞	∞
4	∞		∞			∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞			∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	1	[Obstacle]				∞	∞	∞
3	∞	1	0					∞	∞	∞
4	∞	[Obstacle]		[Obstacle]	∞	∞	∞	∞	∞	∞
5	∞				∞	∞	∞	∞	∞	∞
6	∞	[Obstacle]		∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	2	∞	∞	∞	∞	∞	∞	∞
2	∞	2	1					∞	∞	∞
3	2	1	0					∞	∞	∞
4	∞		1		∞	∞	∞	∞	∞	∞
5	∞		2			∞	∞	∞	∞	∞
6	∞		∞		∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3	2	3	∞	∞	∞	∞	∞	∞
2	3	2	1					∞	∞	∞
3	2	1	0					Y	∞	∞
4	3				∞	∞	∞	∞	∞	∞
5	∞				2	∞	∞	∞	∞	∞
6	∞				∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	∞	∞	∞	∞	∞
2	3	2	1					∞	∞	∞
3	2	1	0	Y		∞			∞	∞
4	3			∞		∞	∞	∞	∞	∞
5	4			∞		∞	∞	∞	∞	∞
6	∞			4	∞	∞	∞	∞	∞	∞
7	∞	∞	4	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	∞	∞	∞	∞
2	3	2	1					∞	∞	∞
3	2	1	0					∞	∞	∞
4	3		1		Y	∞		∞	∞	∞
5	4		2		∞	∞		∞	∞	∞
6	5		3		∞	∞	∞	∞	∞	∞
7	∞	5	4	5	∞	∞	∞	∞	∞	∞
8	∞	∞	5	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	∞	∞	∞
2	3	2	1					∞	∞	∞
3	2	1	0	Y		∞			∞	∞
4	3			∞		∞	∞	∞	∞	∞
5	4			∞		∞	∞	∞	∞	∞
6	5			4	5	6	∞	∞	∞	∞
7	6	5	4	5	6	∞	∞	∞	∞	∞
8	∞	6	5	6	∞	∞	∞	∞	∞	∞
9	∞	∞	6	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	∞	∞
2	3	2	1					∞	∞	∞
3	2	1	0	Y		∞			∞	∞
4	3			∞		∞	∞	∞	∞	∞
5	4					7	∞	∞	∞	∞
6	5			4	5	6	7	∞	∞	∞
7	6	5	4	5	6	7	∞	∞	∞	∞
8	7	6	5	6	7	∞	∞	∞	∞	∞
9	∞	7	6	7	∞	∞	∞	∞	∞	∞
10	∞	∞	7	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	∞
2	3	2	1					8	∞	∞
3	2	1	0	Y		∞			∞	∞
4	3			∞		8	∞	∞	∞	∞
5	4			7		8	∞	∞	∞	∞
6	5			4	5	6	7	8	∞	∞
7	6	5	4	5	6	7	8	∞	∞	∞
8	7	6	5	6	7	8	∞	∞	∞	∞
9	8	7	6	7	8	∞	∞	∞	∞	∞
10	∞	8	7	8	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	9
2	3	2	1					8	9	∞
3	2	1	0	Y		9			9	∞
4	3			9		8	9	∞	∞	∞
5	4					7	8	9	∞	∞
6	5			4	5	6	7	8	9	∞
7	6	5	4	5	6	7	8	9	∞	∞
8	7	6	5	6	7	8	9	∞	∞	∞
9	8	7	6	7	8	9	∞	∞	∞	∞
10	9	8	7	8	9	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J		
1	4	3	2	3	4	5	6	7	8	9		
2	3	2	1					8	9	10		
3	2	1	0			10	9		9	10	∞	
4	3			1			9	8	9	10	∞	∞
5	4			2			7	8	9	10	∞	
6	5			3	4	5	6	7	8	9	10	
7	6	5	4	5	6	7	8	9	10	∞		
8	7	6	5	6	7	8	9	10	∞	∞		
9	8	7	6	7	8	9	10	∞	∞	∞		
10	9	8	7	8	9	10	∞	∞	∞	∞		

	A	B	C	D	E	F	G	H	I	J
1	4	3	2	3	4	5	6	7	8	9
2	3	2	1					8	9	10
3	2	1	0		10	9		9	10	∞
4	3		1		9	8	9	10	∞	∞
5	4		2			7	8	9	10	∞
6	5		3	4	5	6	7	8	9	10
7	6	5	4	5	6	7	8	9	10	∞
8	7	6	5	6	7	8	9	10	∞	∞
9	8	7	6	7	8	9	10	∞	∞	∞
10	9	8	7	8	9	10	∞	∞	∞	∞

Dijkstra's Algorithm

- Notice how we didn't visit every cell
 - Think of a wave spreading from the starting point
- Cells are visited at most once, making Dijkstra's Algorithm more efficient than floodfill in almost every scenario

A* Algorithm

- Similar to Dijkstra's Algorithm, but with a heuristic to guide the search
- For each cell, track its current distance and a total distance estimate
 - Total distance estimate = current distance + remaining distance if there were no walls (known as Manhattan distance)
- Prioritize cells which have the lowest total distance estimate

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	∞					∞	∞	∞
3	∞	∞	0/2		Y	∞		∞	∞	∞
4	∞		∞		∞	∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2	∞	∞	1/4					∞	∞	∞
3	∞	1/4	0/2					∞	∞	∞
4	∞		1/4		∞	∞	∞	∞	∞	∞
5	∞		∞		∞	∞	∞	∞	∞	∞
6	∞		∞		∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	2/6	∞	∞	∞	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	∞	1/4	0/2					∞	∞	∞
4	∞		1/4		∞	∞	∞	∞	∞	∞
5	∞		∞			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	2/6	∞	∞	∞	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	∞		1/4		∞	∞	∞	∞	∞	∞
5	∞		∞				∞	∞	∞	∞
6	∞			∞			∞	∞	∞	∞
7	∞			∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	∞	2/6	∞	∞	∞	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	∞		1/4		∞	∞	∞	∞	∞	∞
5	∞		2/6			∞	∞	∞	∞	∞
6	∞		∞	∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	∞	∞	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	∞					Y	∞	∞	∞	∞
5	∞					∞	∞	∞	∞	∞
6	∞					∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	4/6	∞	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2		Y	∞			∞	∞
4	∞				∞	∞	∞	∞	∞	∞
5	∞						∞	∞	∞	∞
6	∞			∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	4/6	5/8	∞	∞	∞	∞
2	∞	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	∞		1/4		∞	∞	∞	∞	∞	∞
5	∞		2/6			∞	∞	∞	∞	∞
6	∞		∞		∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	4/6	5/8	∞	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	3/8					∞	∞	∞	∞	∞
5	∞					∞	∞	∞	∞	∞
6	∞			∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	4/6	5/8	∞	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	∞					∞	∞	∞	∞	∞
6	∞	3/8		∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	∞	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2		Y	∞		∞	∞	∞
4	3/8				∞	∞	∞	∞	∞	∞
5	∞						∞	∞	∞	∞
6	∞			∞	∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	3/8				Y	∞			∞	∞
5	∞				∞	∞			∞	∞
6	∞				∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	3/8		1/4		∞	∞	∞	∞	∞	∞
5	4/10		2/6			∞	∞	∞	∞	∞
6	∞		3/8		∞	∞	∞	∞	∞	∞
7	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	4/10					∞	∞	∞	∞	∞
6	∞			3/8	4/8	∞	∞	∞	∞	∞
7	∞	∞	4/10	∞	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	4/10					∞	∞	∞	∞	∞
6	∞			3/8	4/8	5/8	∞	∞	∞	∞
7	∞	∞	4/10	5/10	∞	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	∞	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	4/10			∞		∞	∞	∞	∞	∞
6	∞			3/8	4/8	5/8	6/10	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	4/10			∞		∞	∞	∞	∞	∞
6	∞			4/8	5/8	6/10	∞	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8			∞		∞	∞	∞	∞	∞
5	4/10			∞		∞	∞	∞	∞	∞
6	5/12			4/8	5/8	6/10	∞	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	∞	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	Y		∞		∞	∞	∞
4	3/8		1/4	∞		∞	∞	∞	∞	∞
5	4/10		2/6			7/10	∞	∞	∞	∞
6	5/12		3/8	4/8	5/8	6/10	7/12	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	7/12	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	3/8					∞	8/10	∞	∞	∞
5	4/10					7/10	8/12	∞	∞	∞
6	5/12			4/8	5/8	6/10	7/12	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	7/12	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2					∞	∞	∞
4	3/8					9/10	8/10	9/12	∞	∞
5	4/10					7/10	8/12	∞	∞	∞
6	5/12			4/8	5/8	6/10	7/12	∞	∞	∞
7	∞	∞	4/10	5/10	6/10	7/12	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2	10/10		9/10		∞	∞	∞
4	3/8			9/10		8/10	9/12	∞	∞	∞
5	4/10					7/10	8/12	∞	∞	∞
6	5/12			3/8	4/8	5/8	6/10	7/12	∞	∞
7	∞	∞	4/10	5/10	6/10	7/12	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

	A	B	C	D	E	F	G	H	I	J
1	4/10	3/8	2/6	3/6	4/6	5/8	6/10	7/12	∞	∞
2	3/8	2/6	1/4					∞	∞	∞
3	2/6	1/4	0/2		10/10	9/10		∞	∞	∞
4	3/8		1/4		9/10	8/10	9/12	∞	∞	∞
5	4/10		2/6			7/10	8/12	∞	∞	∞
6	5/12		3/8		4/8	5/8	6/10	7/12	∞	∞
7	∞	∞	4/10	5/10	6/10	7/12	∞	∞	∞	∞
8	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
9	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
10	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞

A* Algorithm

- We visited even less cells than Dijkstra's Algorithm (less than half in this case)
- Efficient when a good heuristic exists, degrades to Dijkstra's Algorithm otherwise