LEARNING OBJECTIVES

A* algorithm

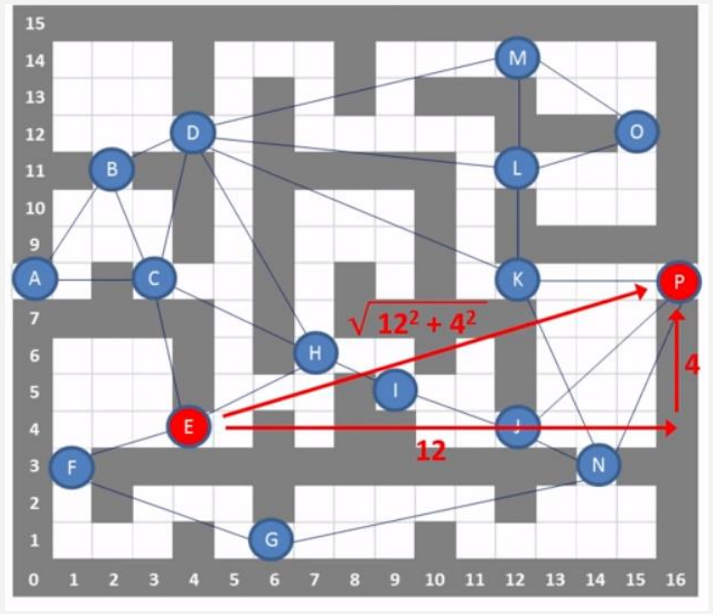
A* SEARCH

- The A* (or A Star) Search is an alternative algorithm that can be used for finding the shortest path.
- It performs better than Dijkstra’s algorithm because of its use of **heuristics.**
- Heuristics is when existing experience is used to form judgement, the so called “rule of thumb”.

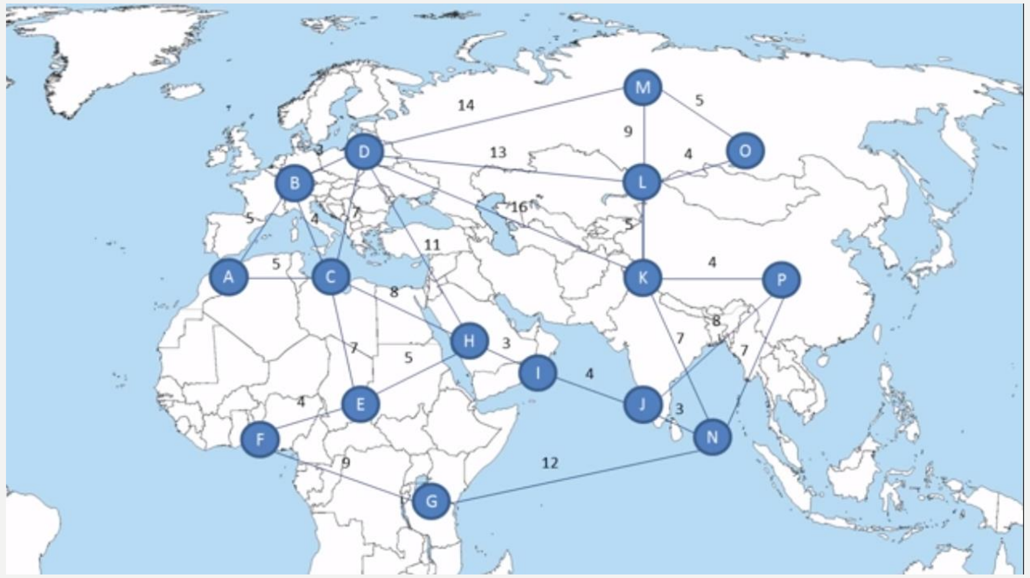
HEURISTICS

- Depend on the problem
- Working out an estimated distance from a node to the end node

A maze could use the Manhattan Distance
Number of steps (12 + 4)
Using x, y coordinates
Or Euclidean Distance as a heuristics
Straight line form E to P

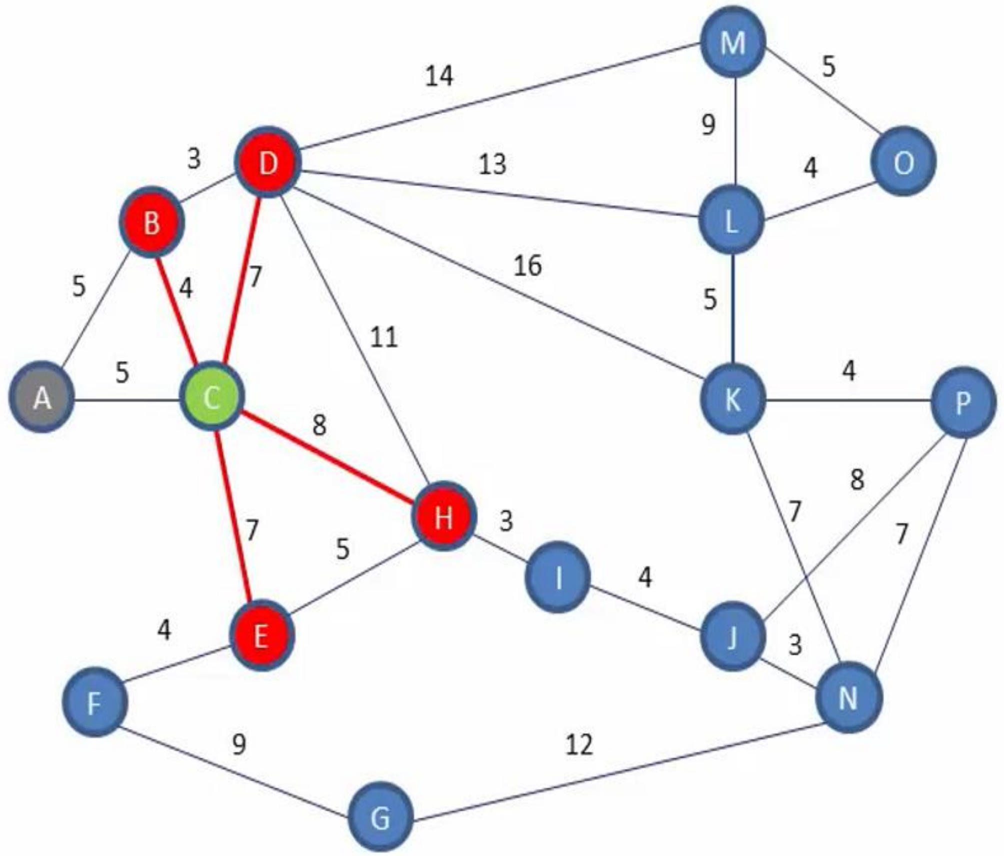


On a map the heuristic distance is in kilometres from a node to the end node
Using latitude and longitude



Open B C D H E

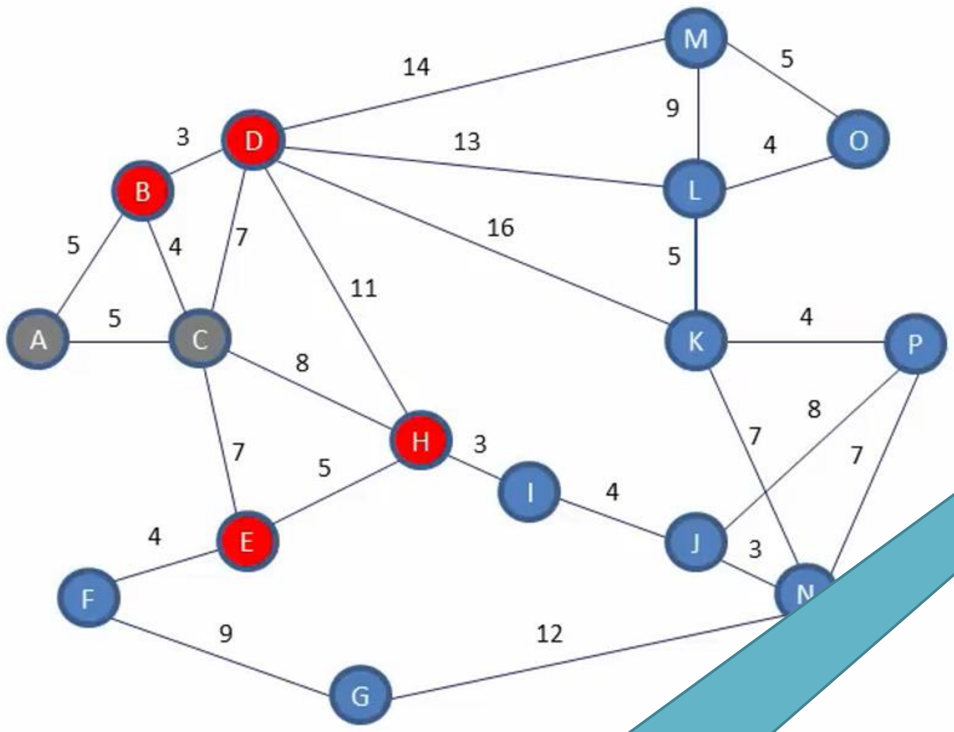
Closed A



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D		16		
E		16		
F		20		
G		17		
H		11		
I		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

Open B D H E

Closed A C

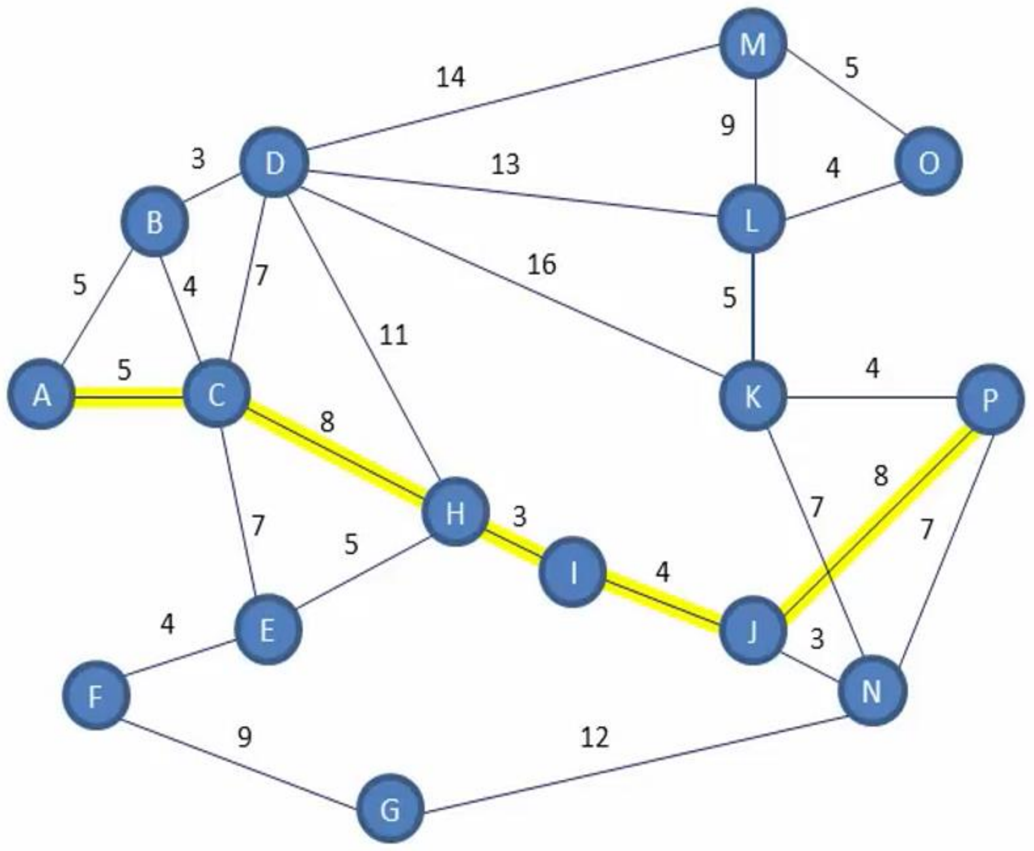


Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	$\boxed{22}$	A
C	5	13	18	A
D	12	16	28	C
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
[illegible]		10		
J		8		
K		4		
L		7		
M		10		
N		7		
O		5		
P		0		

Next Current Node will be the node with the lower F value

Open E M L K J P N

Closed A C B H D I



Vertex	Distance from A (g)	Heuristic distance (h)	f = g + h	Previous vertex
A	0	16	16	
B	5	17	22	A
C	5	13	18	A
D	8	16	24	B
E	12	16	28	C
F		20		
G		17		
H	13	11	24	C
I	16	10	26	H
J	20	8	28	I
K	24	4	28	D
L	21	7	28	D
M	22	10	32	D
N	23	7	30	J
O		5		
P	28	0	28	J

A* SEARCH

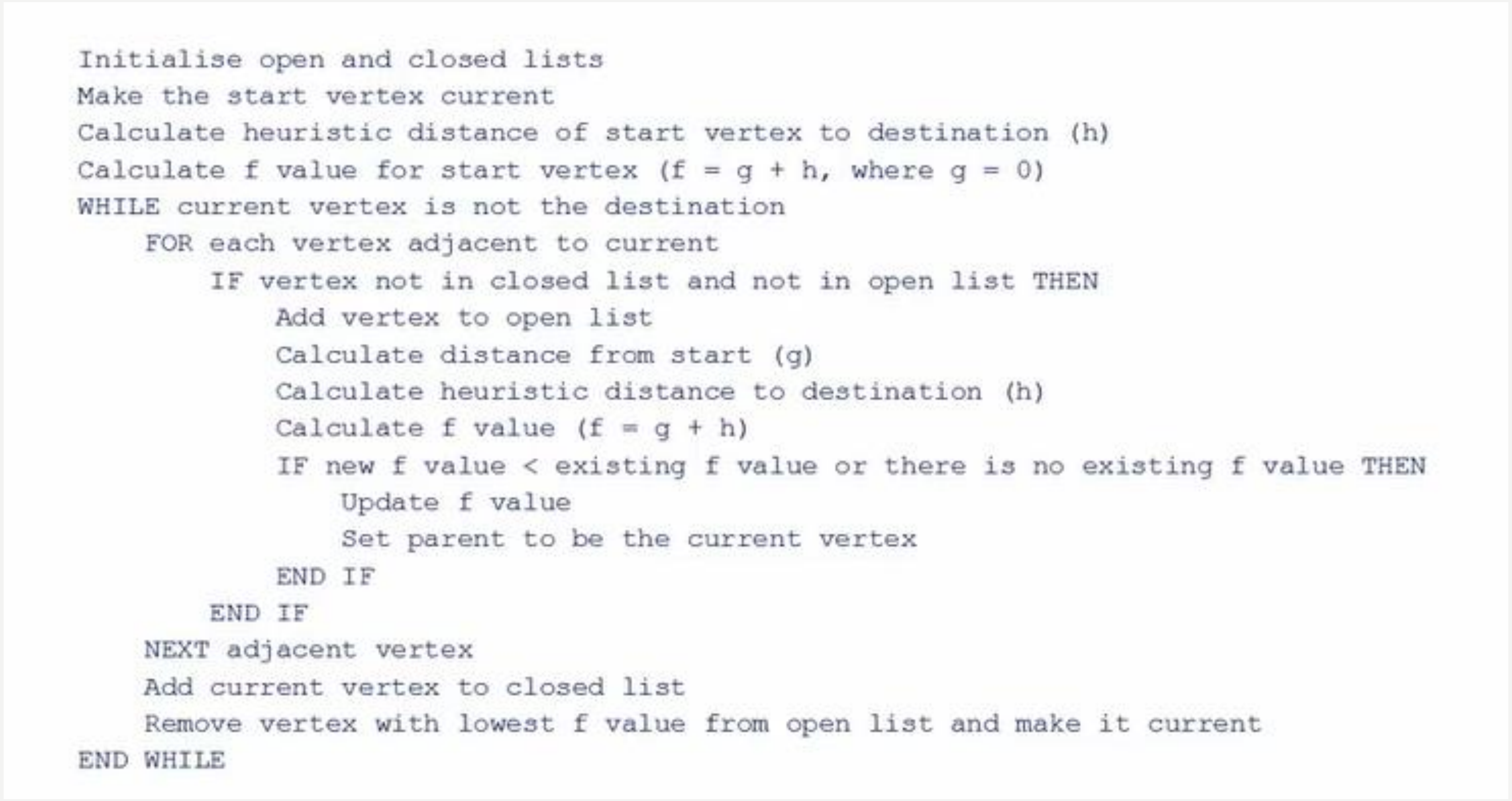
Will not visit all nodes
Will follow the route with the lowest F value
Stops when destination node is current node

The Pseudocode for a the A* Search looks like this:

```
Begin at the start node and make this the current node.

WHILE the destination node is unvisited
  FOR each open node directly connected to the current node.
      Add to the list of open nodes
      Add the distance from the start (g) to the heuristic estimate
      of the distance (h).
      Assign this value (f) to the node.
  NEXT connected node
Make the unvisited node with the lowest F value the current node
ENDWHILE
```

A* ALGORITHM



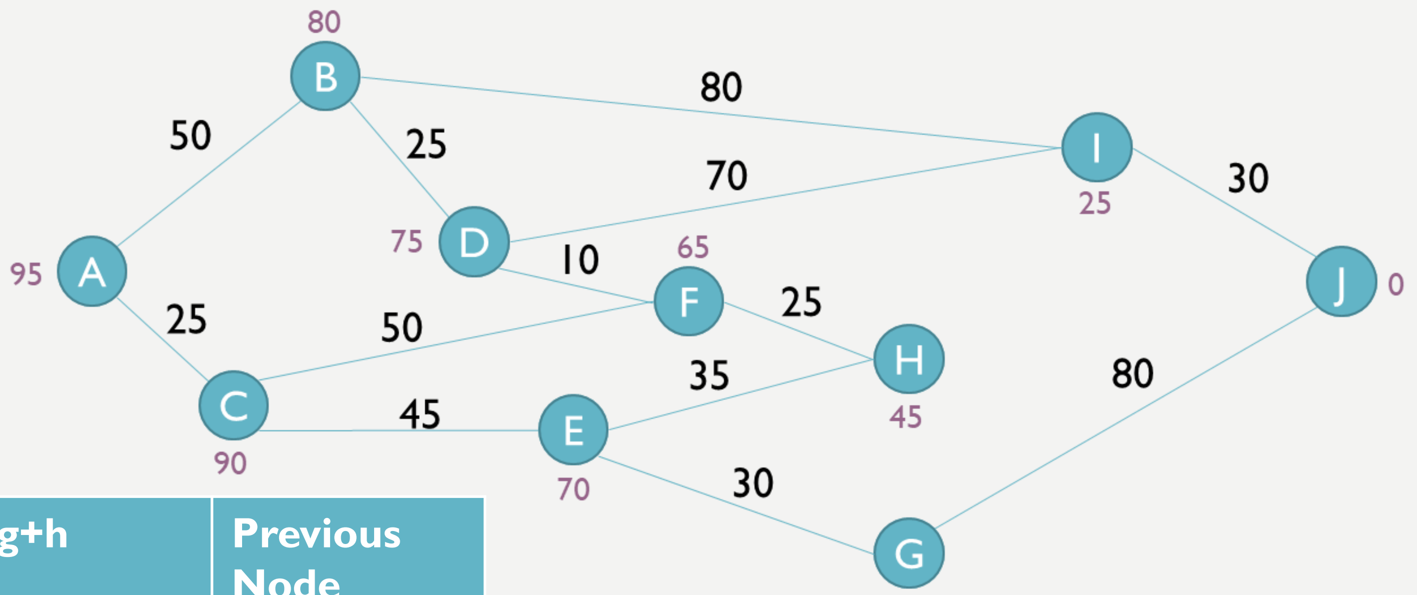
```
Initialise open and closed lists
Make the start vertex current
Calculate heuristic distance of start vertex to destination (h)
Calculate f value for start vertex (f = g + h, where g = 0)
WHILE current vertex is not the destination
    FOR each vertex adjacent to current
        IF vertex not in closed list and not in open list THEN
            Add vertex to open list
            Calculate distance from start (g)
            Calculate heuristic distance to destination (h)
            Calculate f value (f = g + h)
            IF new f value < existing f value or there is no existing f value THEN
                Update f value
                Set parent to be the current vertex
            END IF
        END IF
    NEXT adjacent vertex
    Add current vertex to closed list
    Remove vertex with lowest f value from open list and make it current
END WHILE
```

SUMMARY

- A* has a wise range of applications
- A* finds the shortest path between two vertices
- A* does not have to visit all vertices, ideally
- A* picks the most promising looking node next
- The better the heuristic, the quicker A* finds the path
- Heuristic is problem specific
- Open nodes known as “the fringe” or “the frontier”
- List of open nodes can be implemented as a priority queue (so next node with the smallest F value is chosen next.)
- Each node on the path keeps track of the one that came before it

A* SEARCH

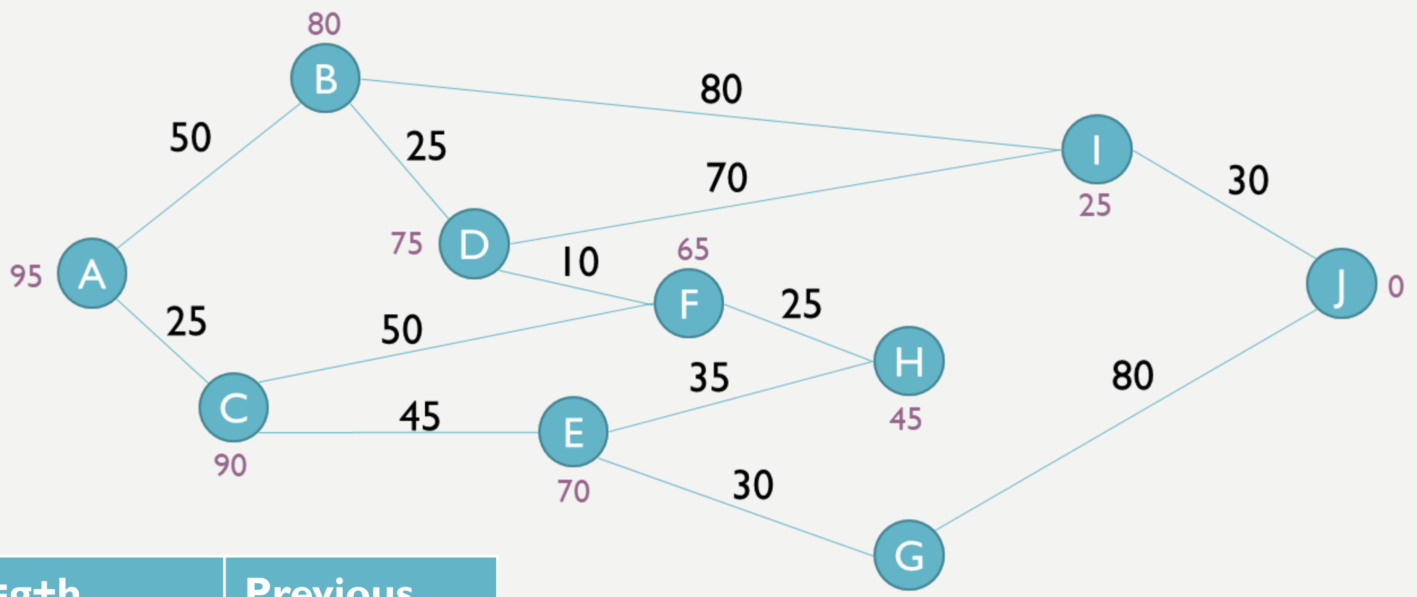
Heuristic value is the distance in a straight line each node is from J



Node	Path Distance (g)	Heuristic Distance (h)	f=g+h	Previous Node
A	0	95		
B		80		
C		90		
D		75		
E		70		
F		65		
G		50		
H		45		
I		25		
J		0		

A* SEARCH

Heuristic value is the distance in a straight line each node is from J



Node	Path Distance (g)	Heuristic Distance (h)	f=g+h	Previous Node
A (v)	0	95	95	
B (v)	50	80	130	A
C (v)	25	90	115	A
D	75	75	150	B
E (v)	70	70	140	C
F	75	65	140	C
G (v)	100	50	150	E
H	105	45	150	E
I	130	25	155	B
J	180	0	~~180~~ 165	~~G~~ I

A B I J

DIJKSTRA VS A*

- Heuristic helps produce a solution in a faster time
- A* uses estimated distance from final node
- Dijkstra uses a weight/distance
- A* chooses which path to take next based on lowest current distance travelled.
- A* does not have to visit all vertices to find a solution
- Difference in programming complexity is minimal