Pathfinding Algorithms

Dijkstra and A*

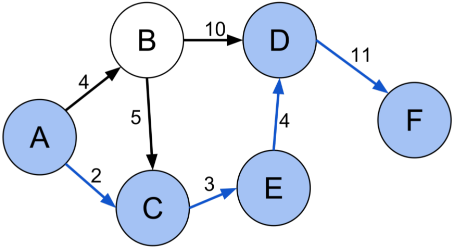
Ahmadreza Hadi

AICup

September 2022

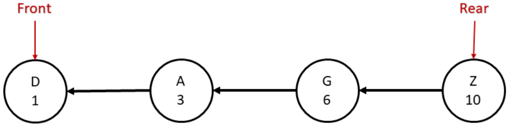
Pathfinding Algorithms

- What is a graph?
  - $G = (V, E)$
  - Directed
  - Undirected
- Shortest path problem
  - Dijkstra's Algorithm
  - A* Search Algorithm
- Applications
  - Google Maps
  - Routing Network Packets
  - ...



Data Structures

- Priority Queue



- Dictionary

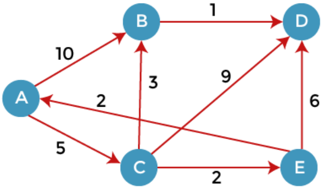
```
car = dict()
car['brand'] = 'Ford'
car['model'] = 'Mustang'
car['year'] = '1994'
```

Dijkstra's Algorithm

Goal: Find the shortest path between two nodes.

1. Initialize distances of all vertices as infinite.
2. Create an empty priority_queue pq. Every item of pq is a pair (weight, vertex). Weight (or distance) is used as first item of pair as first item is by default used to compare two pairs.
3. Insert source vertex into pq and make its distance as 0.
4. While either pq doesn't become empty
    1. Extract minimum distance vertex from pq. Let the extracted vertex be u.
    2. Loop through neighbors of u and do following for every vertex v.
       if dist[v] > dist[u] + weight(u, v)
        - Update distance of v, i.e., do dist[v] = dist[u] + weight(u, v)
        - Insert v into the pq (Even if v is already there)

Example



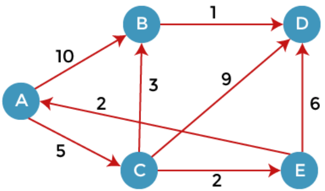
start = A, end = D

Dist	Value
A	0
B	∞
C	∞
D	∞
E	∞

Previous	
A	none
B	none
C	none
D	none
E	none

Priority Queue	
Priority	Node
0	A

Example



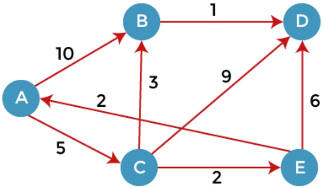
start = A, end = D, Current Node = A

Dist	Value
A	0
B	10
C	5
D	∞
E	∞

Priority Queue	
Priority	Node
5	C
10	B

Previous	
A	none
B	A
C	A
D	none
E	none

Example



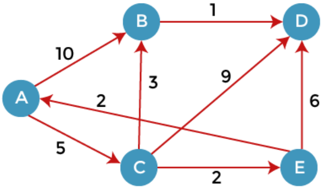
start = A, end = D, Current Node = C

Dist	Value
A	0
B	8
C	5
D	14
E	∞

Priority Queue	
Priority	Node
8	B
10	B
14	D

Previous	
A	none
B	C
C	A
D	none
E	none

Example



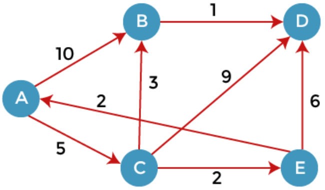
start = A, end = D, Current Node = B

Dist	Value
A	0
B	8
C	5
D	9
E	∞

Priority Queue	
Priority	Node
9	D
10	B
14	D

Previous	
A	none
B	C
C	A
D	B
E	none

Example



start = A, end = D, Current Node = D

Dist	Value
A	0
B	8
C	5
D	9
E	∞

Priority Queue	
Priority	Node
9	D
10	B
14	D

Previous	
A	none
B	C
C	A
D	B
E	none

Shortest path length from A to D = 9

A* Search Algorithm

$$F(n) = G(n) + H(n)$$

- Add start node to pq
- Get first item from pq(u), for all the neighbouring nodes(V), find dist = G(n) + Weight(u,v)
- if dist < G(v).
  - Update G(v) = dist.
  - Update F(v) = G(v) + H(v)
  - Add (F(v), v) to pq.
- Stop working when
  - You find the destination
  - You cannot find the destination going through all possible points

Heuristic Function

- If H(n) = 0 -> A* turns into Dijkstra's algorithm
- If H(n) is always lower or equal to the cost of moving from n to the goal, then A* is guaranteed to find a shortest path.
- If H(n) is is exactly equal to the cost of moving from n to the goal, then A* follows the best path from start to end
- If H(n) is sometimes greater than the cost of moving from n to goal, A* is not guaranteed to find the shortest path