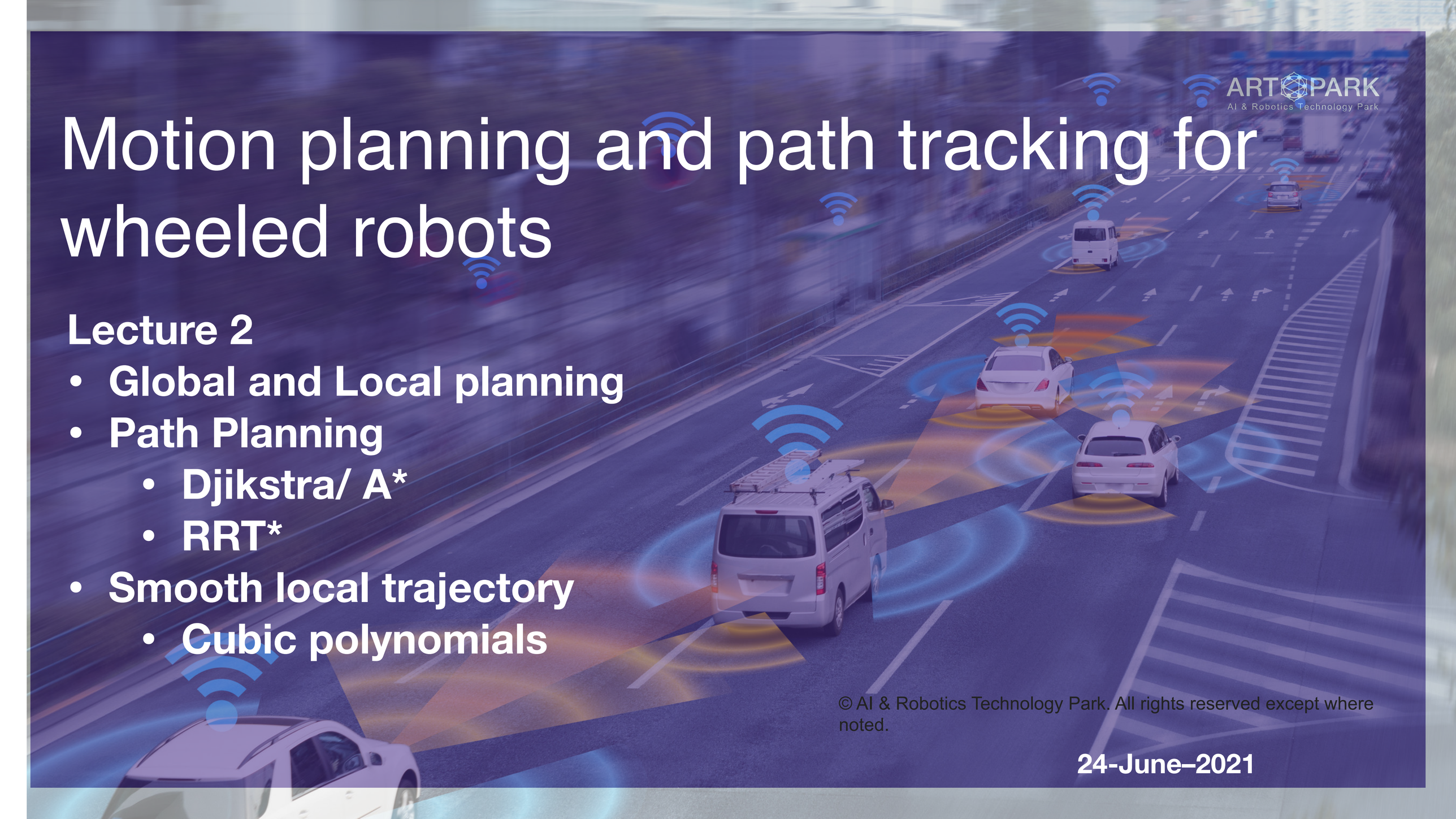


Motion planning and path tracking for wheeled robots

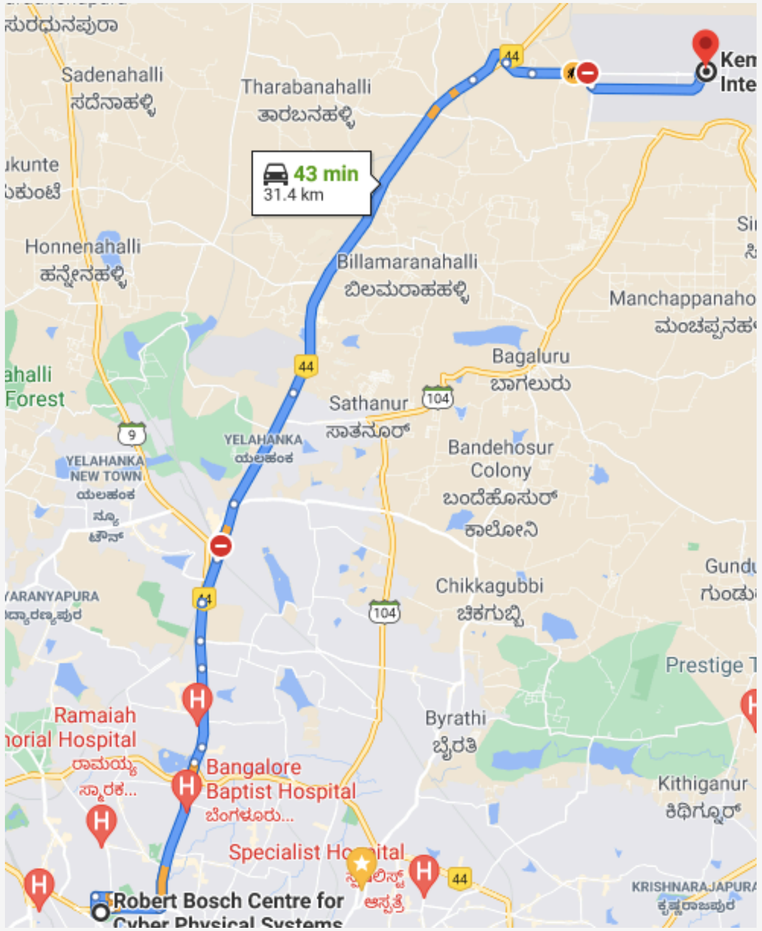
Lecture 2

- **Global and Local planning**
- **Path Planning**
  - **Djikstra/ A***
  - **RRT***
- **Smooth local trajectory**
  - **Cubic polynomials**

24-June–2021

Planning

Global planning: Go from campus to Airport



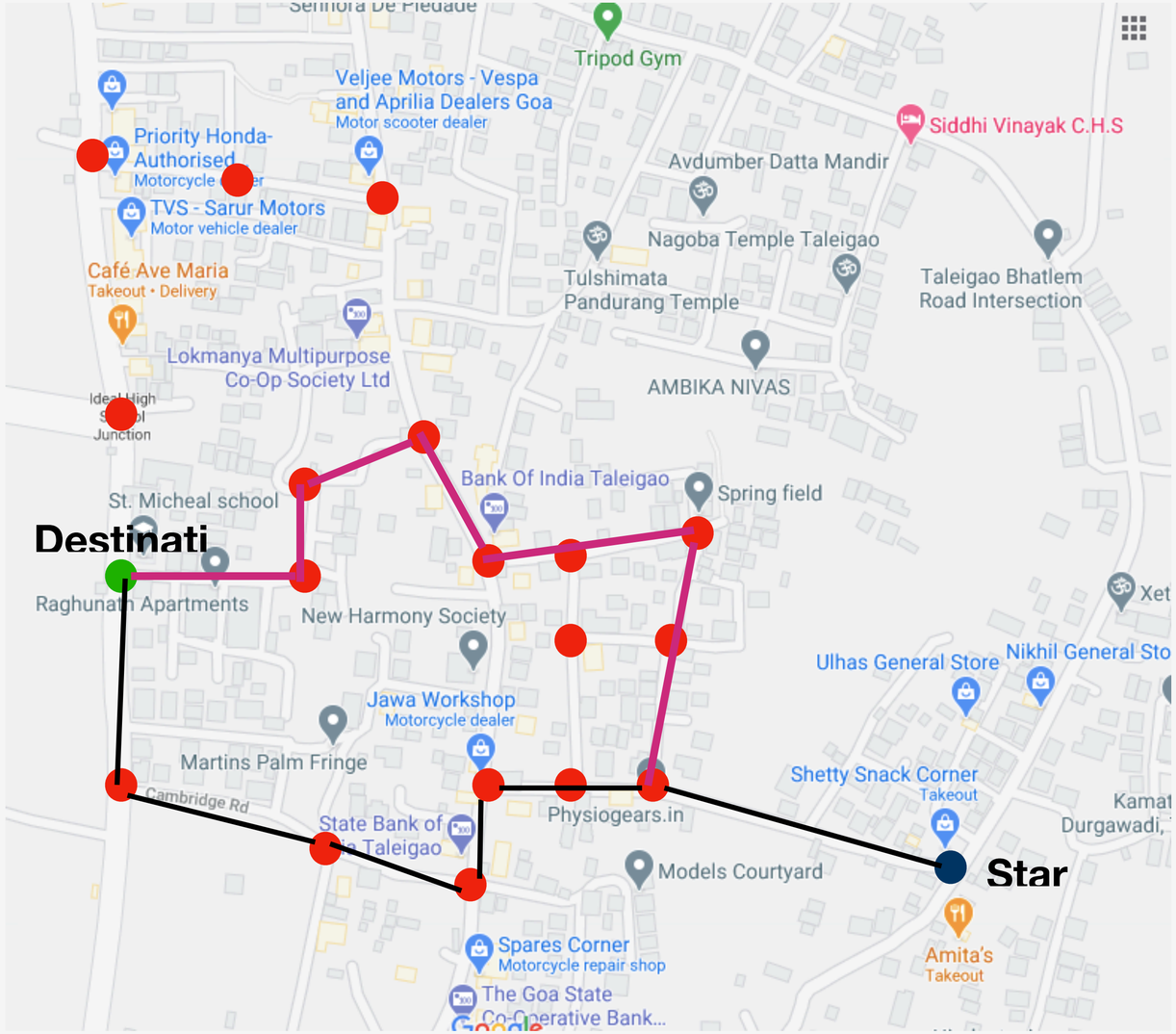
Useful for High-level estimates

Locally safe planning

Vehicle action in immediate time horizon

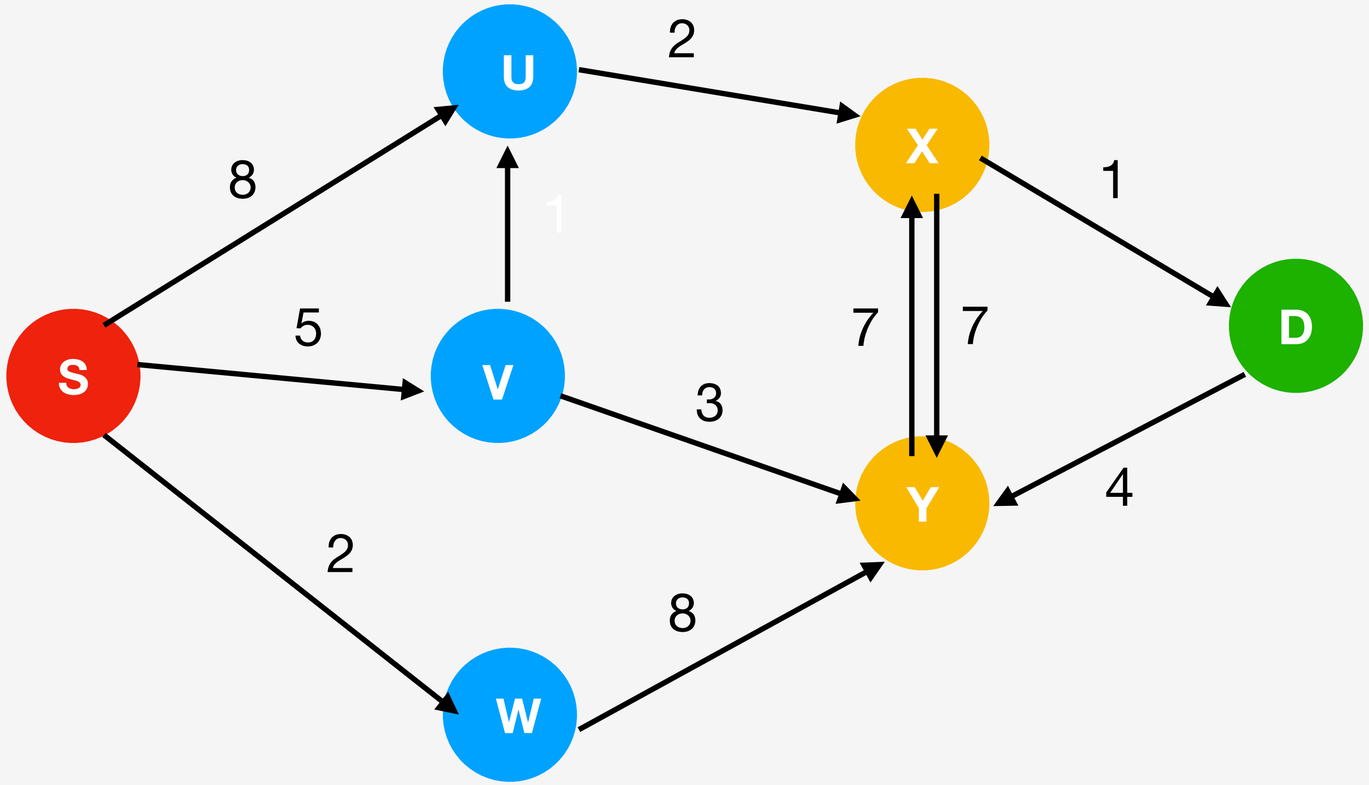
- More annotations/ Semantically richer
- Needed for Replanning

Mission planning



- How to go from start to destination
- Focus: Speed limits, traffic flow rates, road closures
- No visibility to dynamics and obstacles
- Directed graph can represent road-network
    - One-ways may be present

Shortest path in weighted graphs



Road network = Topological graph

Graph $G = (V, E)$

What is the shortest path from S to D?

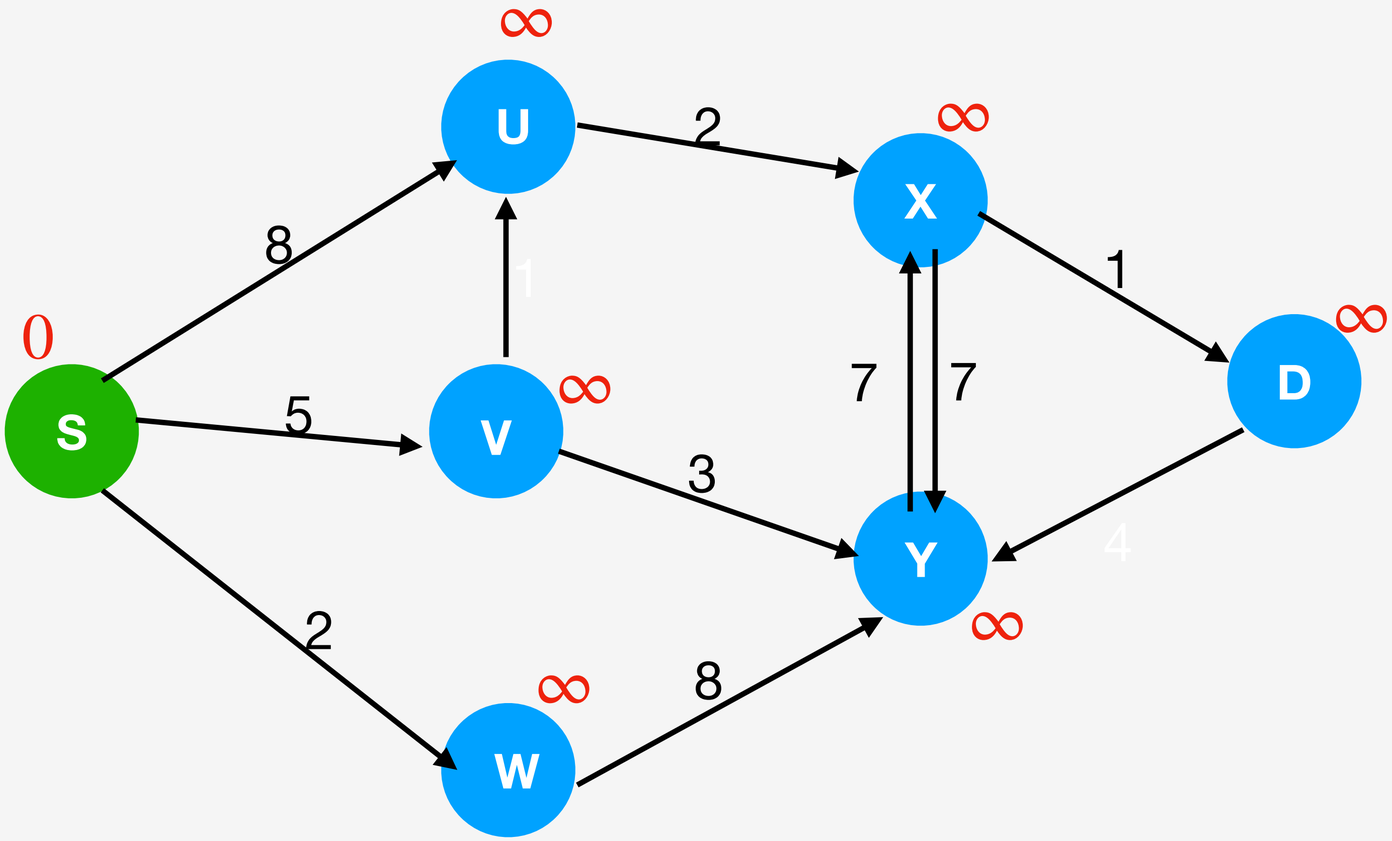
All edges may not have identical costs to traverse

- road terrain conditions
- road length/ time to travel
- pay tolls

Djikstra's algorithm

Factors in cost of edge traversal

Djikstra’s algorithm: Iteration 0



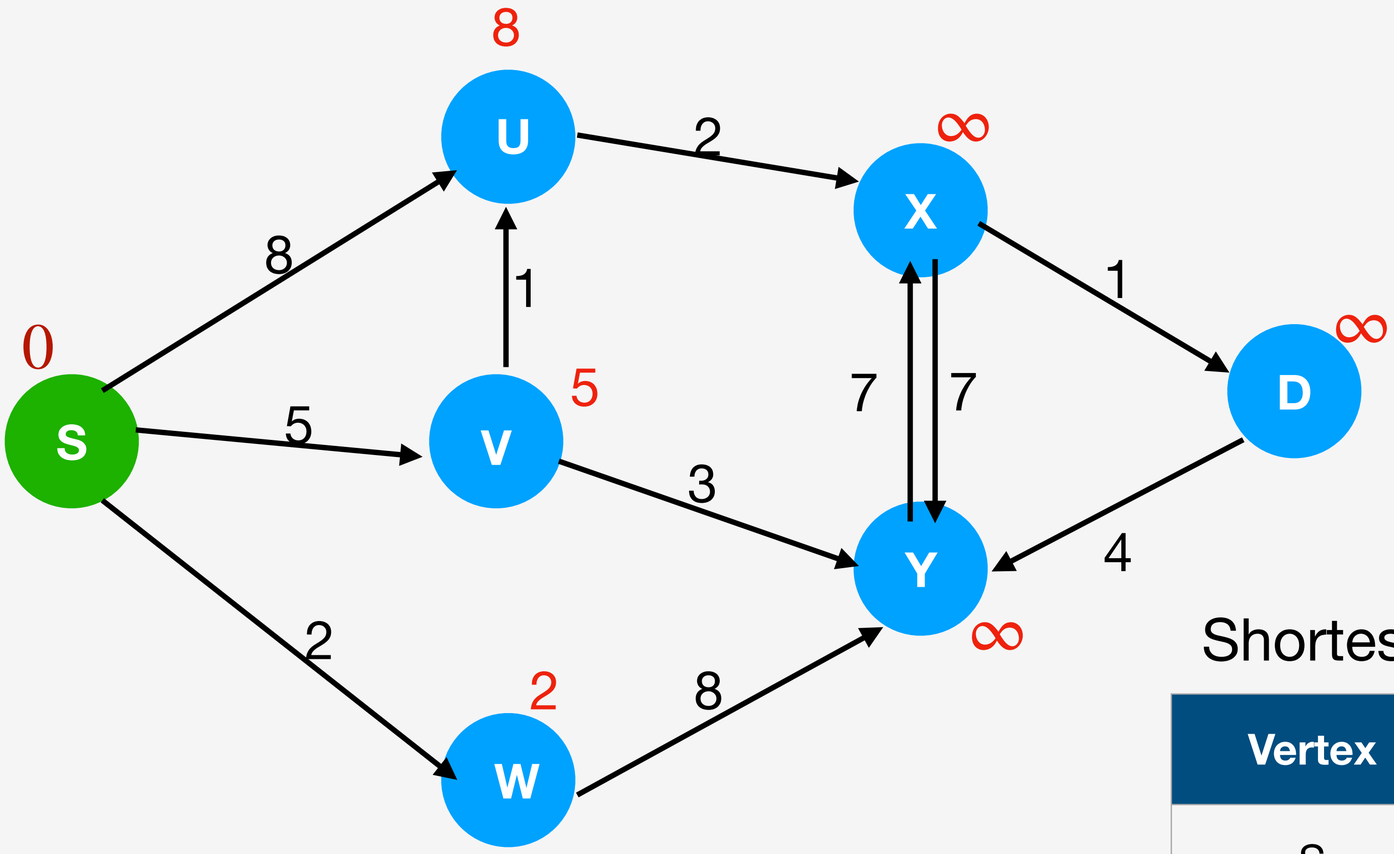
Shortest path set: empty

Djikstra’s algorithm: Iteration 1

ART PARK
AI & Robotics Technology Park

Min cost table

Vertex	Min-cost
U	8
V	5
W	2



Shortest path table

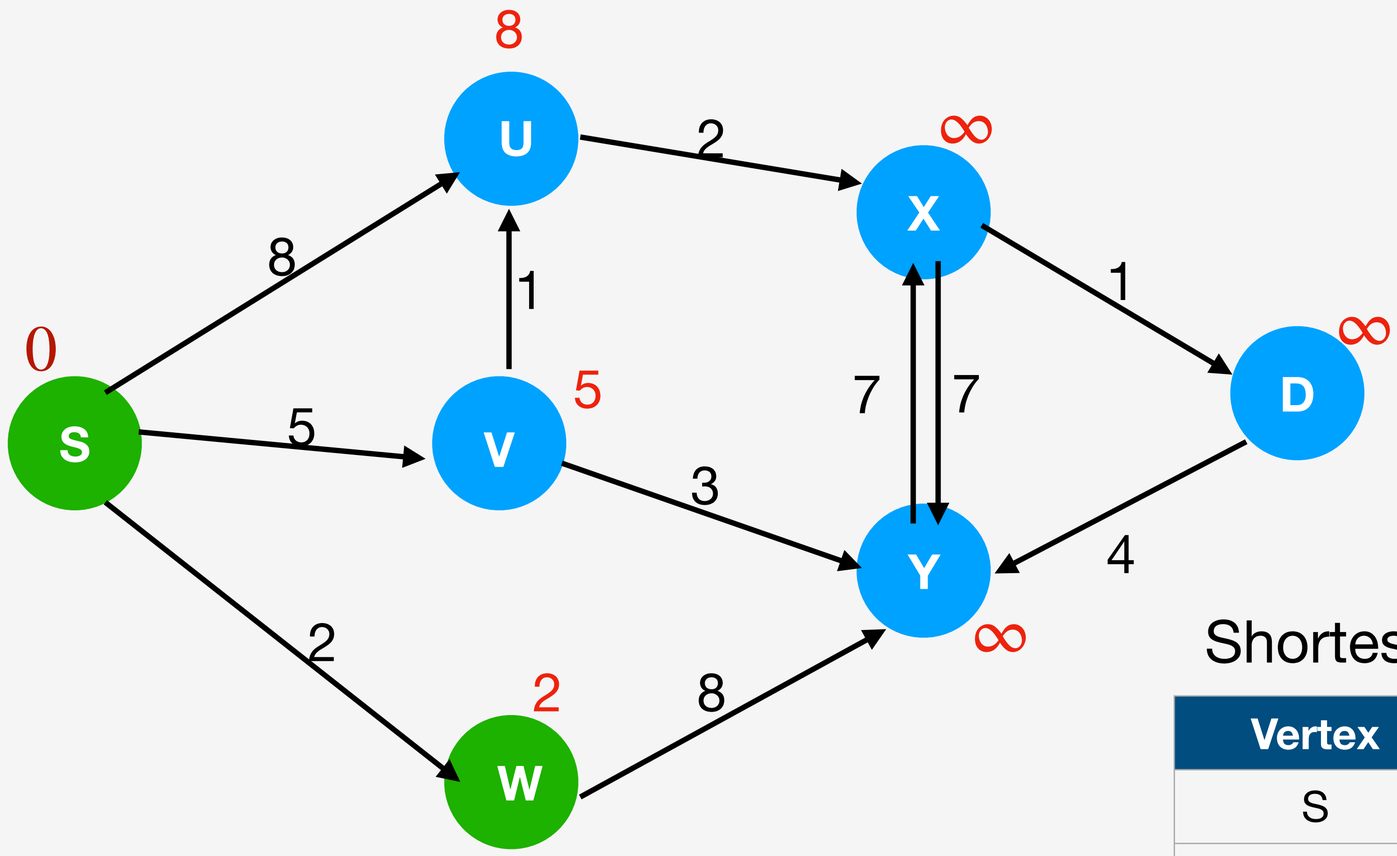
Vertex	Predecessor
S	-

Min cost of U, V and W got updated after S was promoted to Shortest path set

Djikstra’s algorithm: Iteration 2

Min cost table

Vertex	Min-cost
U	8
V	5
Y	10



Shortest path table

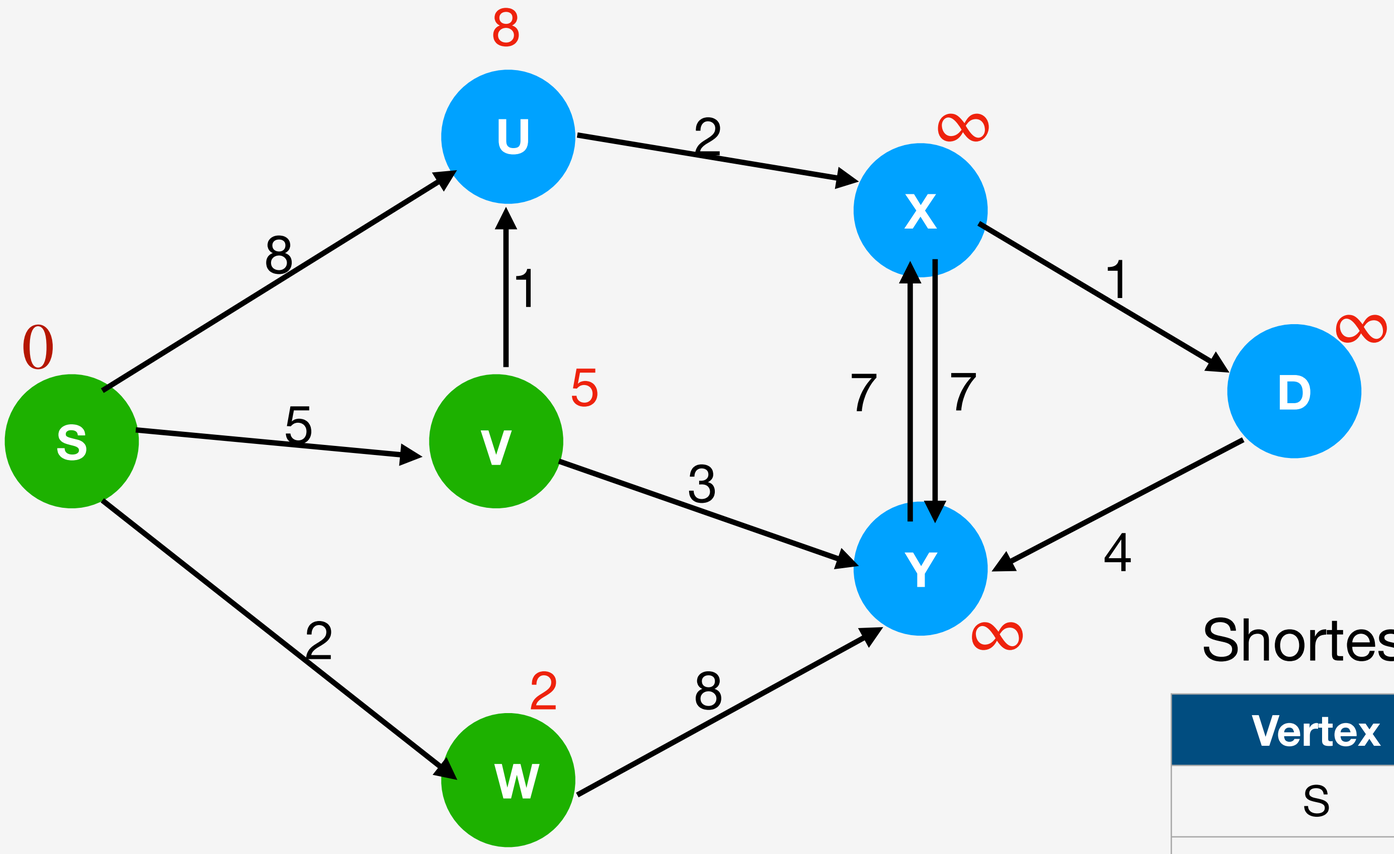
Vertex	Predecessor
S	-
W	S

Min cost of Y got updated after W was promoted to Shortest path set

Djikstra’s algorithm: Iteration 3

Min cost table

Vertex	Min-cost
U	6
Y	8



Shortest path table

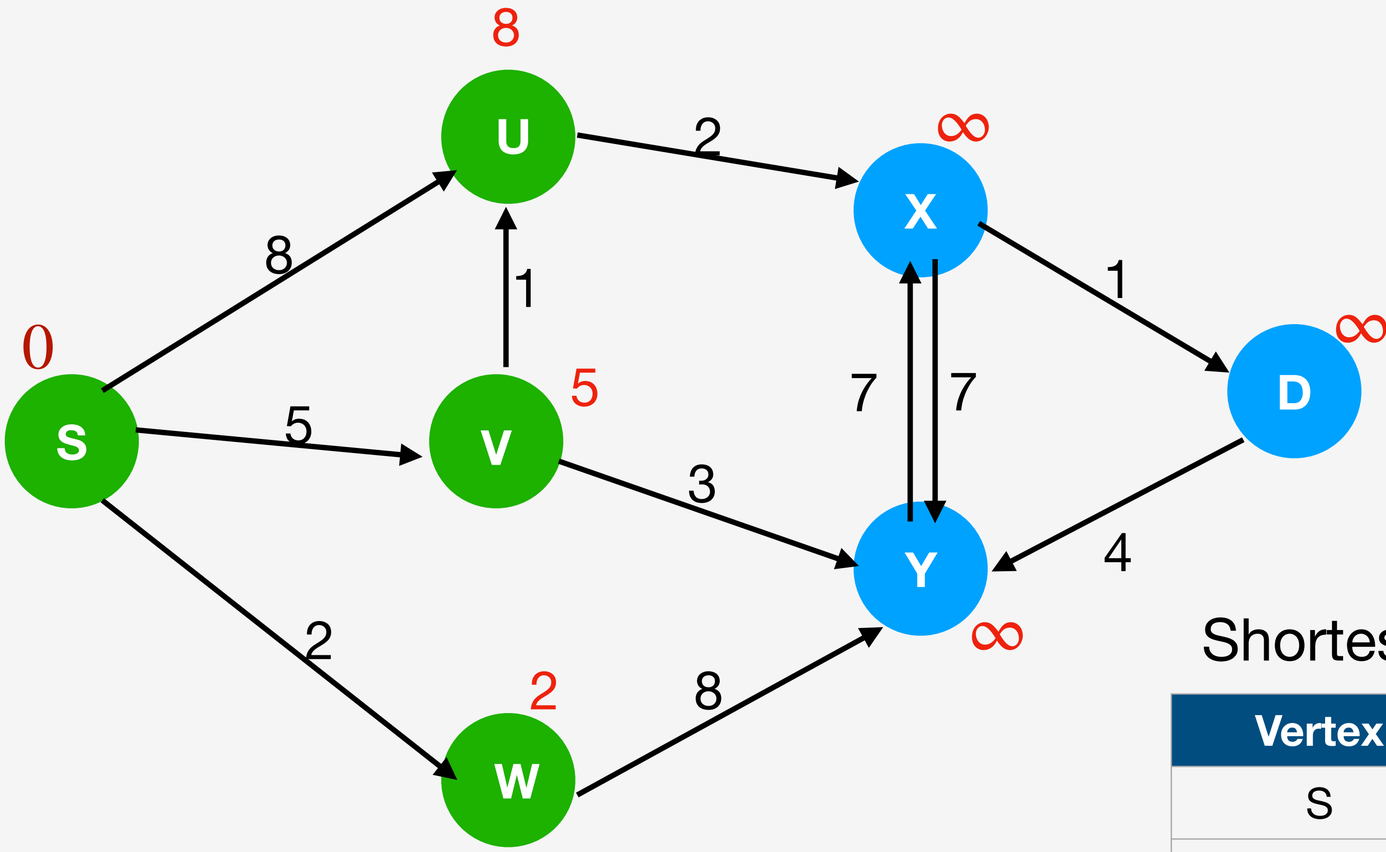
Vertex	Predecessor
S	-
W	S
V	S

Min cost of U, Y got updated after V was promoted to Shortest path set

Djikstra’s algorithm: Iteration 4

Min cost table

Vertex	Min-cost
X	8
Y	8



Shortest path table

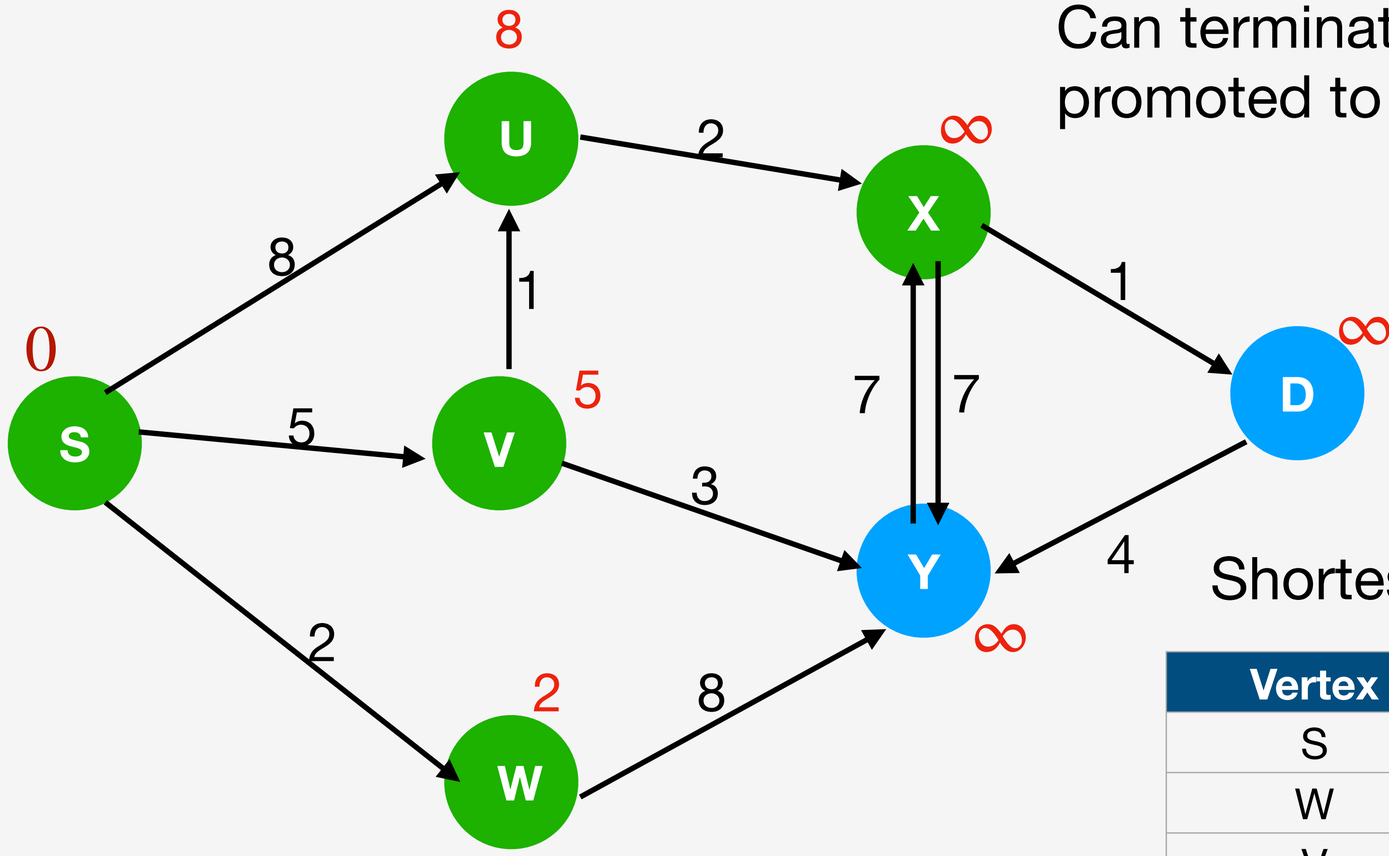
Vertex	Predecessor
S	-
W	S
V	S
U	V

Min cost of X got updated after U was promoted to Shortest path set

Djikstra’s algorithm: Iteration 5

Min cost table

Vertex	Min-cost
D	9
Y	8



Can terminate once D is promoted to Shortest path

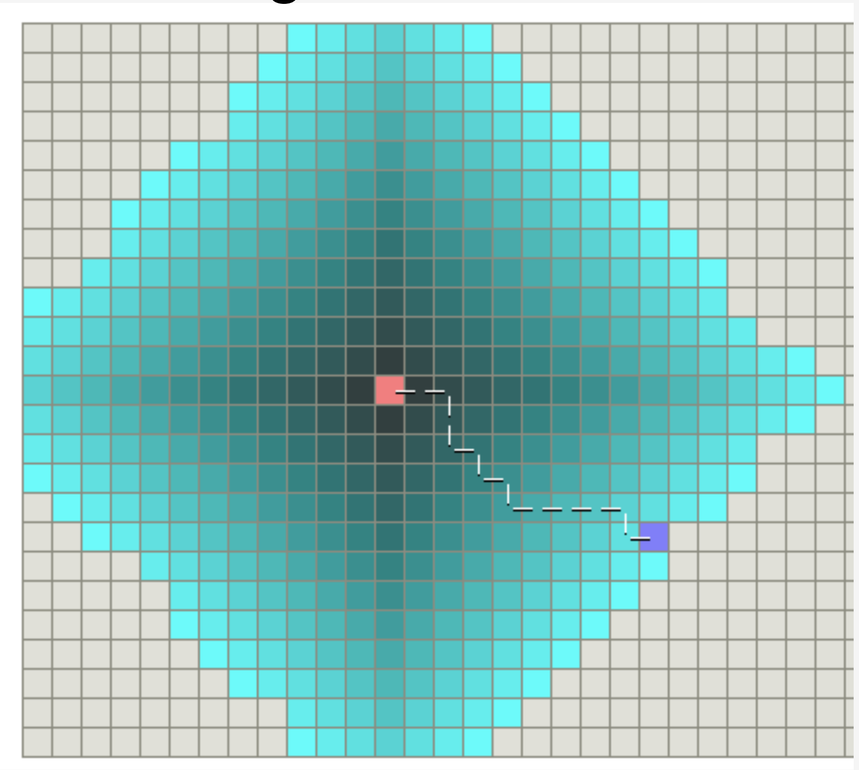
Shortest path table

Vertex	Predecessor
S	-
W	S
V	S
U	V
X	U

Note that min cost of D got updated after X was promoted to Shortest path set

Djikstra

- Outward expansion from source node till destination is reached
- Slow but guaranteed shortest-path

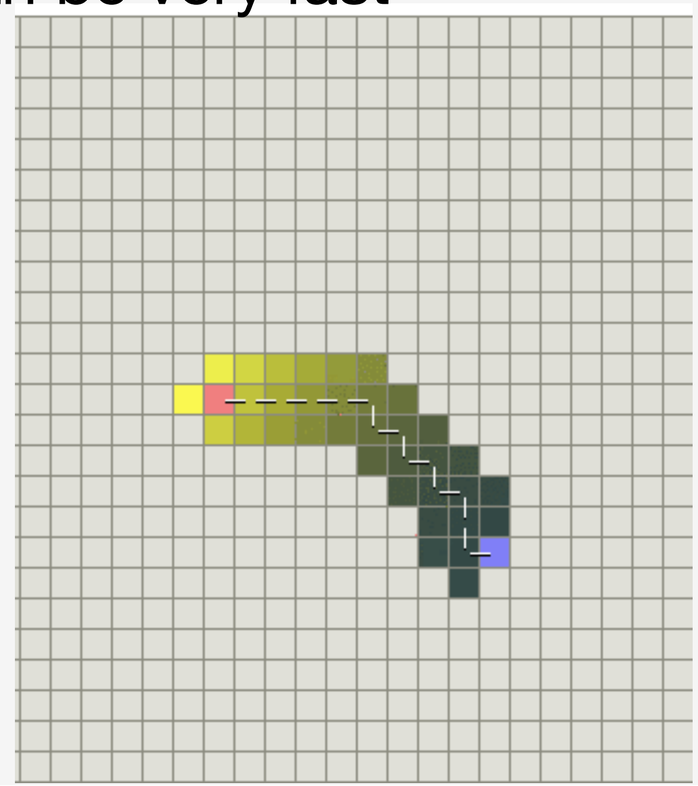


Remember - Convergence only after D is promoted to Shortest path

Greedy best-first

ART PARK
AI & Robotics Technology Park

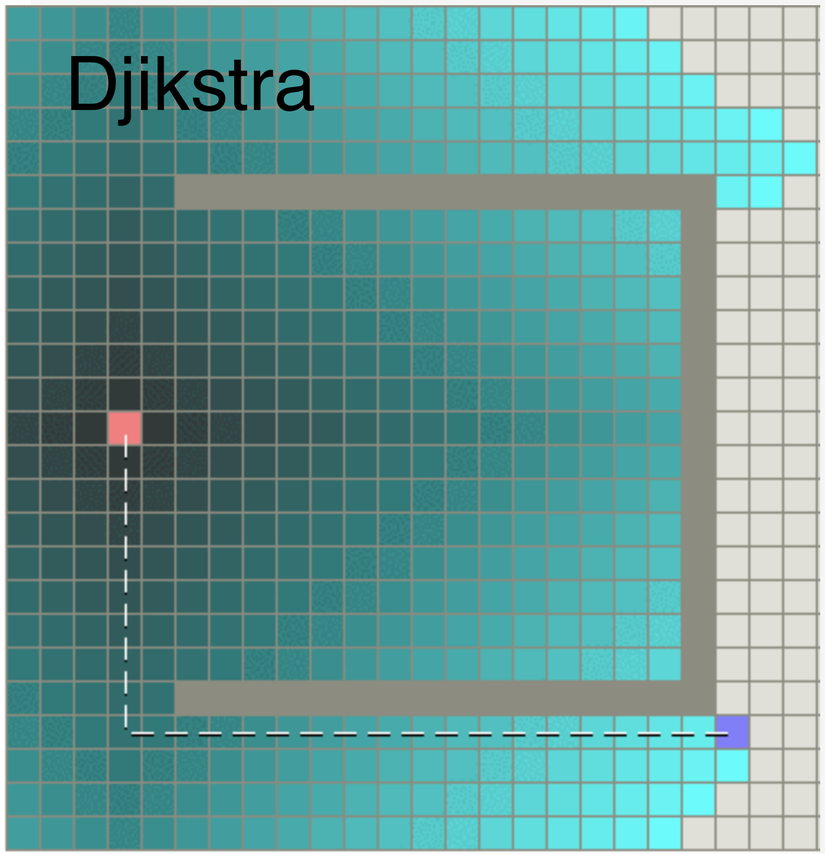
- Select the neighbor which has lowest distance to goal
- Distance to goal = some heuristic function
- Can be very fast



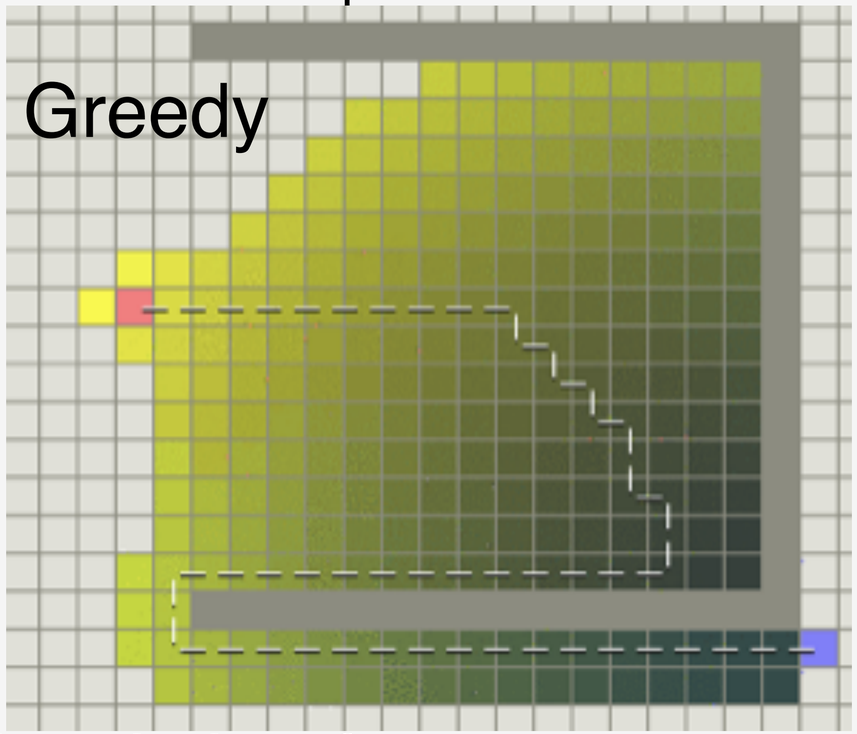
A*: Djikstra + Greedy best-first

ART PARK
AI & Robotics Technology Park

- Optimal paths
- Slow-ish for straight-forward courses



- Fast for straight-forward cases
- Not so desirable paths in face of some convex obstacles



A* = combines the best features

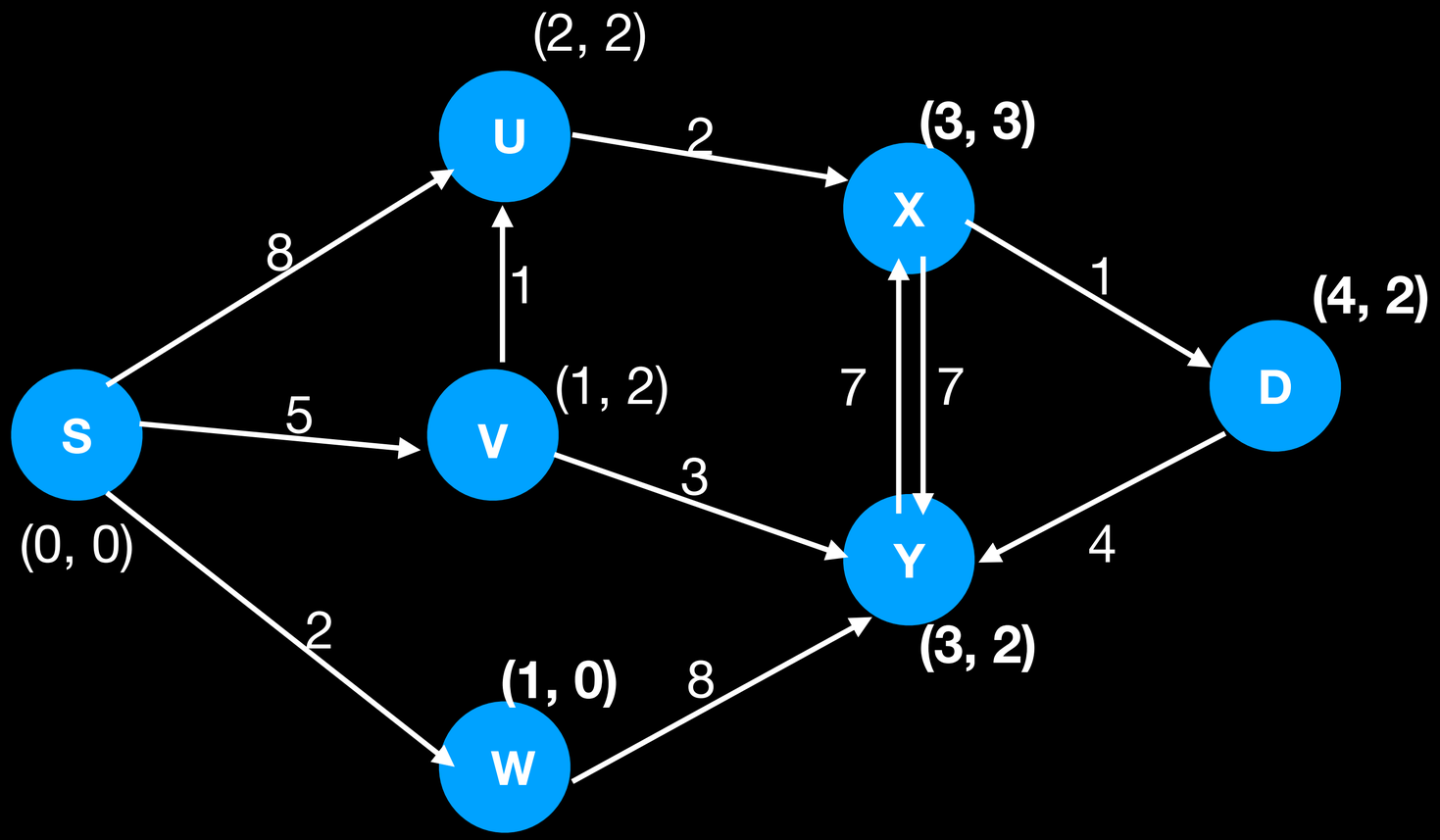
Node Cost = Cost to Source + Heuristic cost to Destination

g(n) = c(n) + w*h(n)

h(n) = 0 => Djikstra

h(n) >> c(n) => Greedy

A* search: Iteration 0

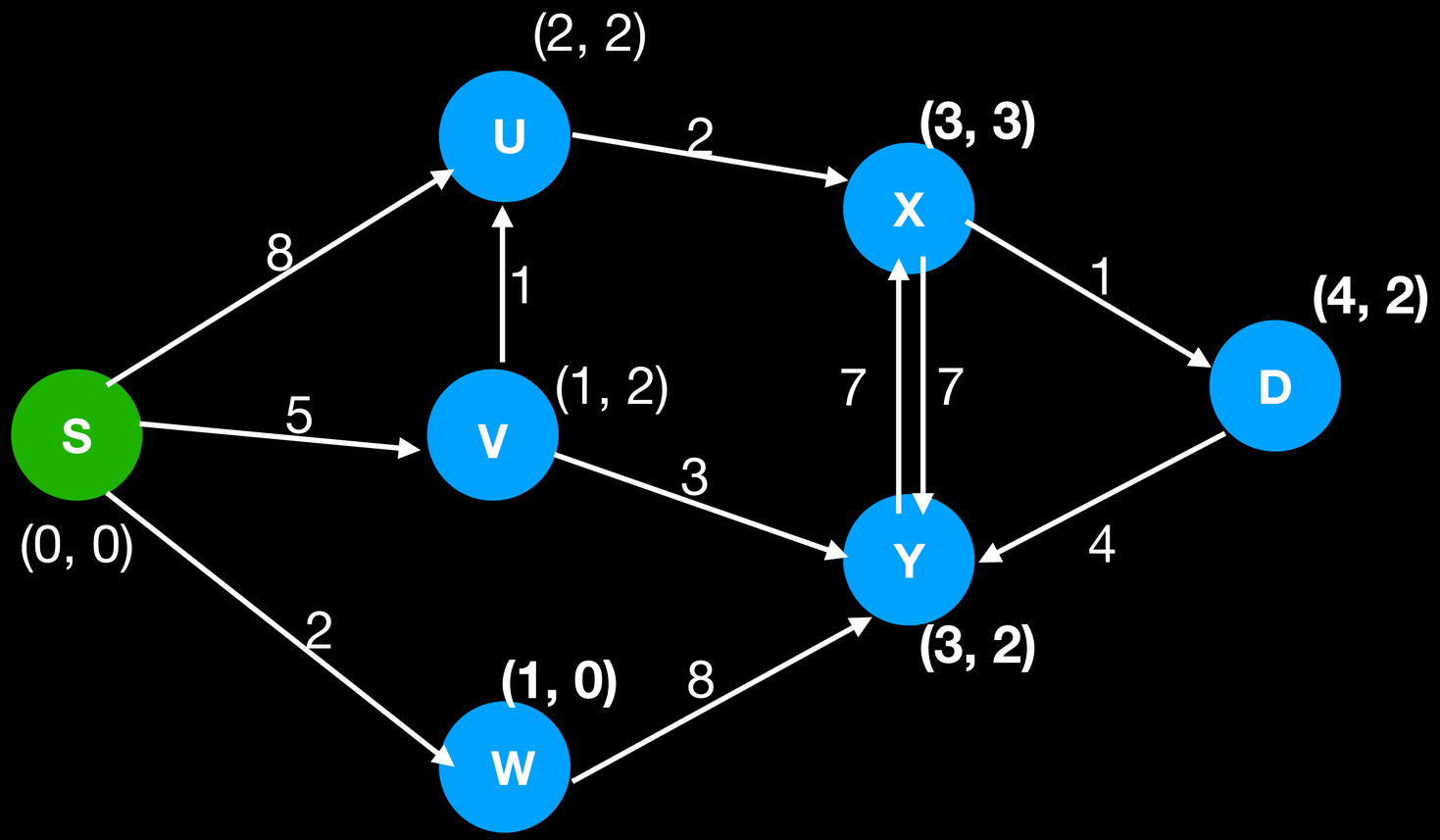


Vertex	Cost	Heuristic	Total
S	0	4.89	4.89

Shortest path set: empty

Simplest heuristic to destination = Euclidean distance
Can also use time estimate as heuristic

A* search: Iteration 1

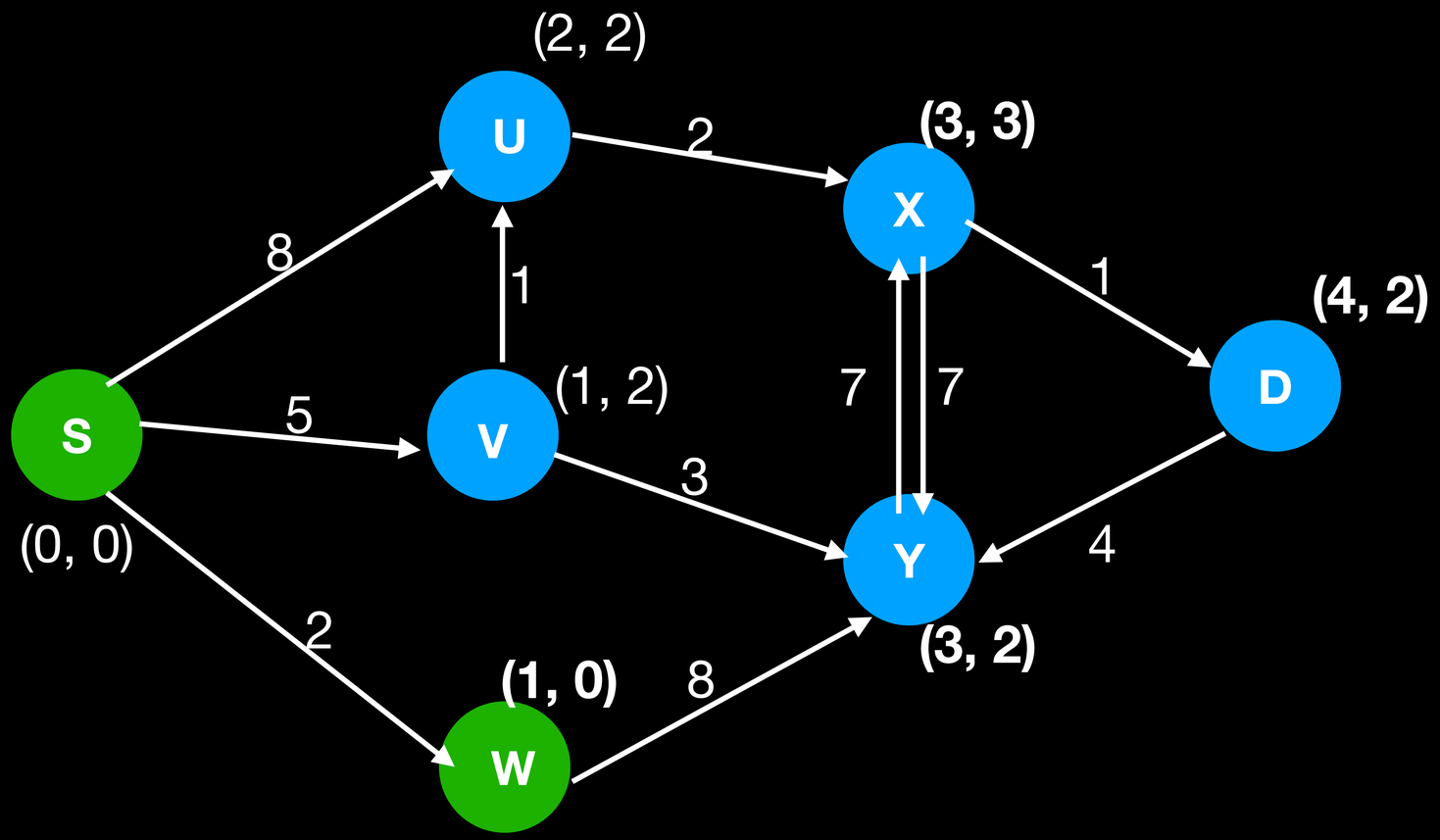


Vertex	Cost	Heuristic	Total
U	8	2	10
V	5	3	8
W	2	3.6	5.6

Shortest path table

Vertex	Predecessor
S	-

A* search: Iteration 2

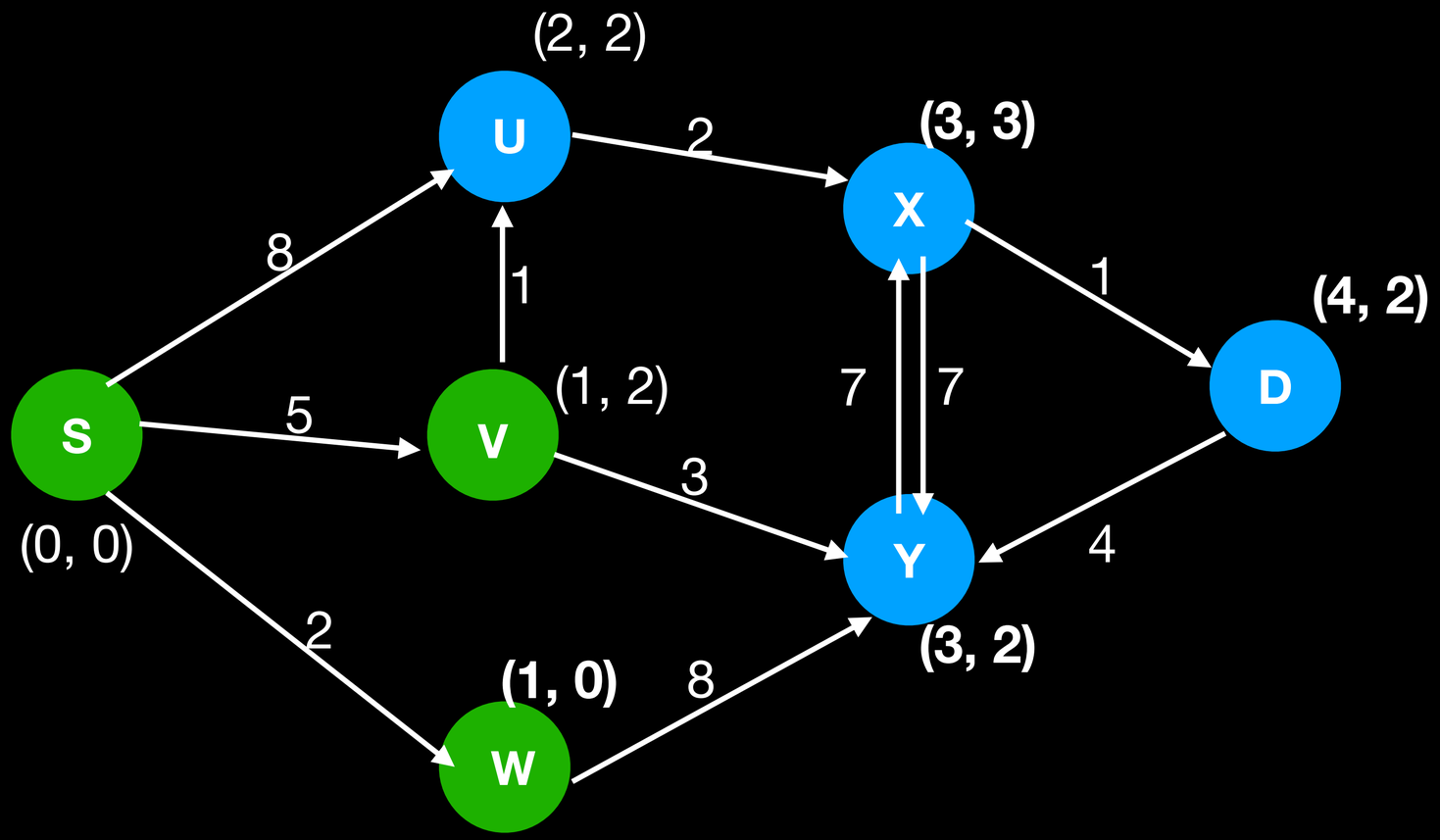


Vertex	Cost	Heuristic	Total
U	8	2	10
V	5	3	8
Y	10	1.414	11.414

Shortest path table

Vertex	Predecessor
S	-
W	S

A* search: Iteration 3

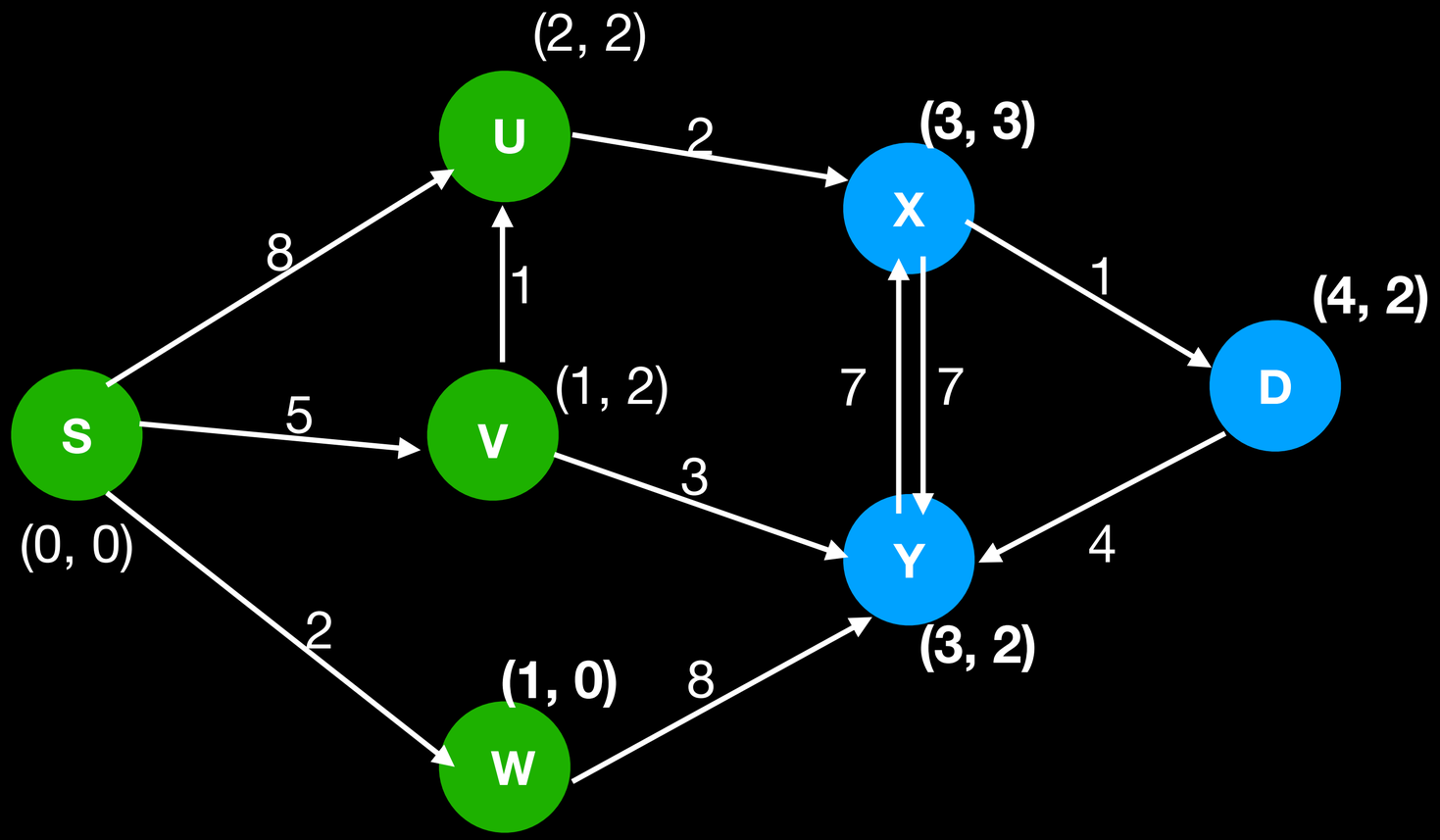


Vertex	Cost	Heuristic	Total
U	6	2	8
Y	8	1	9

Shortest path table

Vertex	Predecessor
S	-
W	S
V	S

A* search: Iteration 4

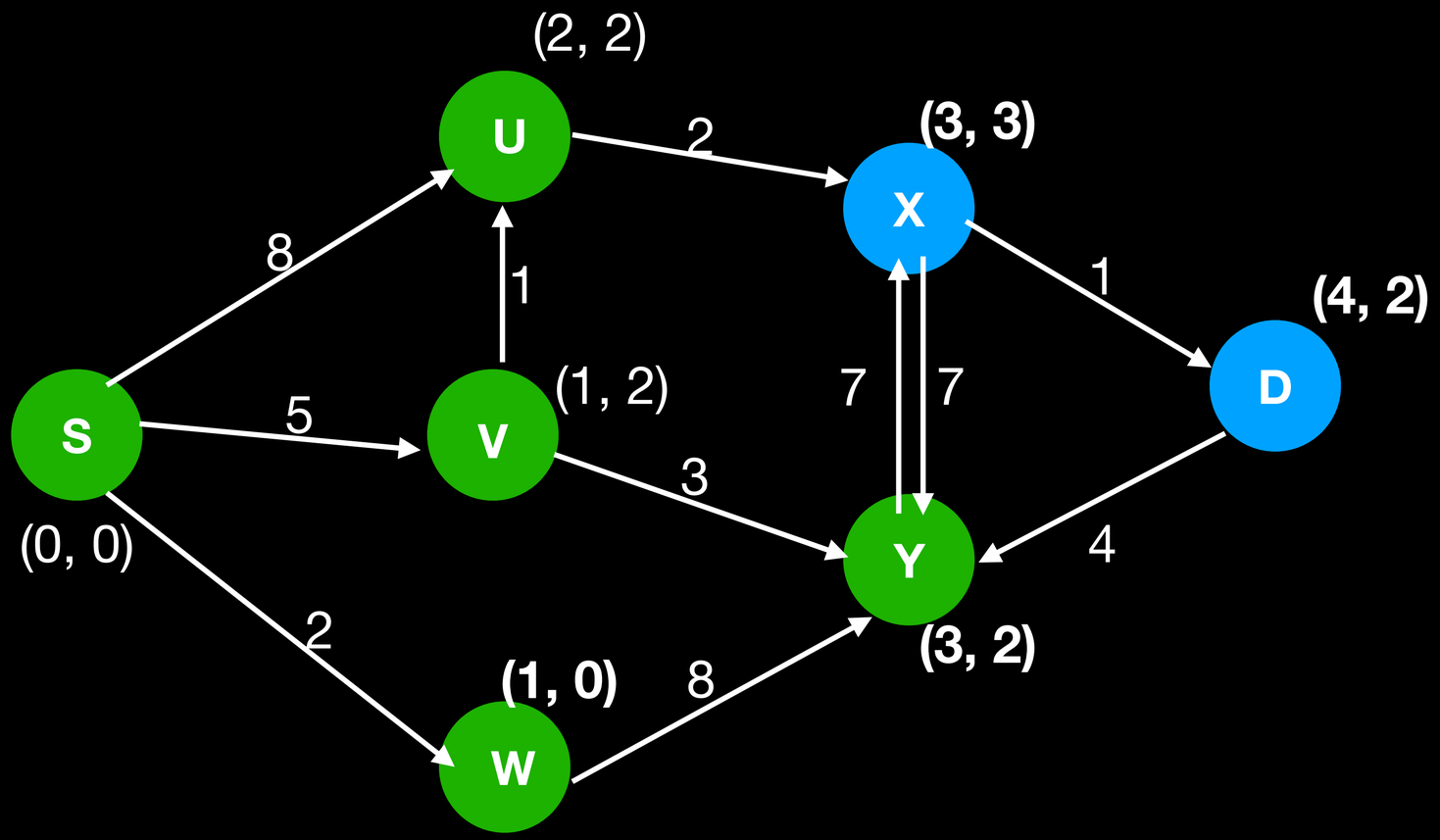


Vertex	Cost	Heuristic	Total
Y	8	1	9
X	8	1.414	9.414

Shortest path table

Vertex	Predecessor
S	-
W	S
V	S
U	V

A* search: Iteration 4

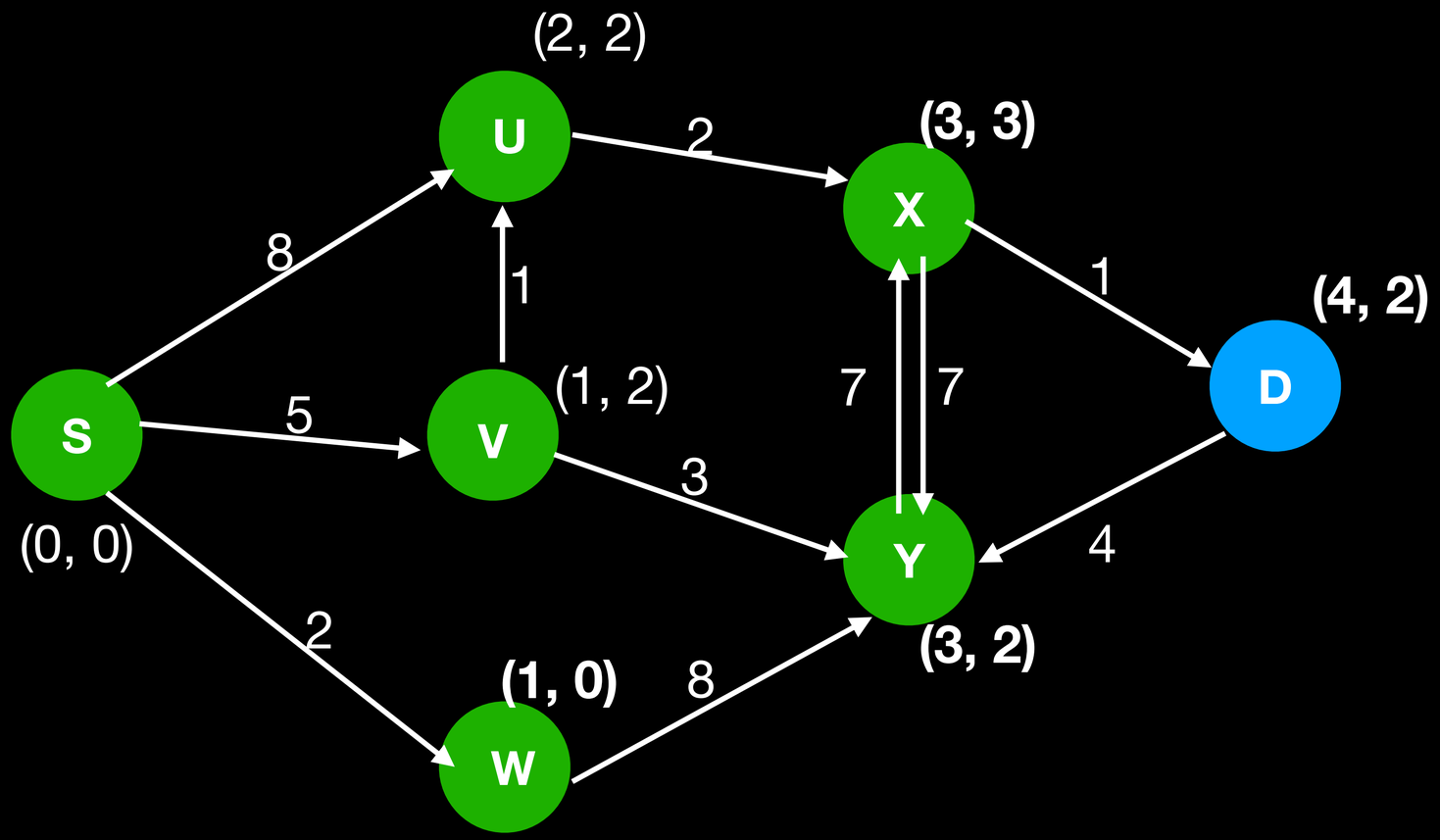


Vertex	Cost	Heuristic	Total
X	8	1.414	9.414

Shortest path table

Vertex	Predecessor
S	-
W	S
V	S
U	V
Y	V

A* search: Iteration 4



Vertex	Cost	Heuristic	Total
D	8	1	9

Shortest path table

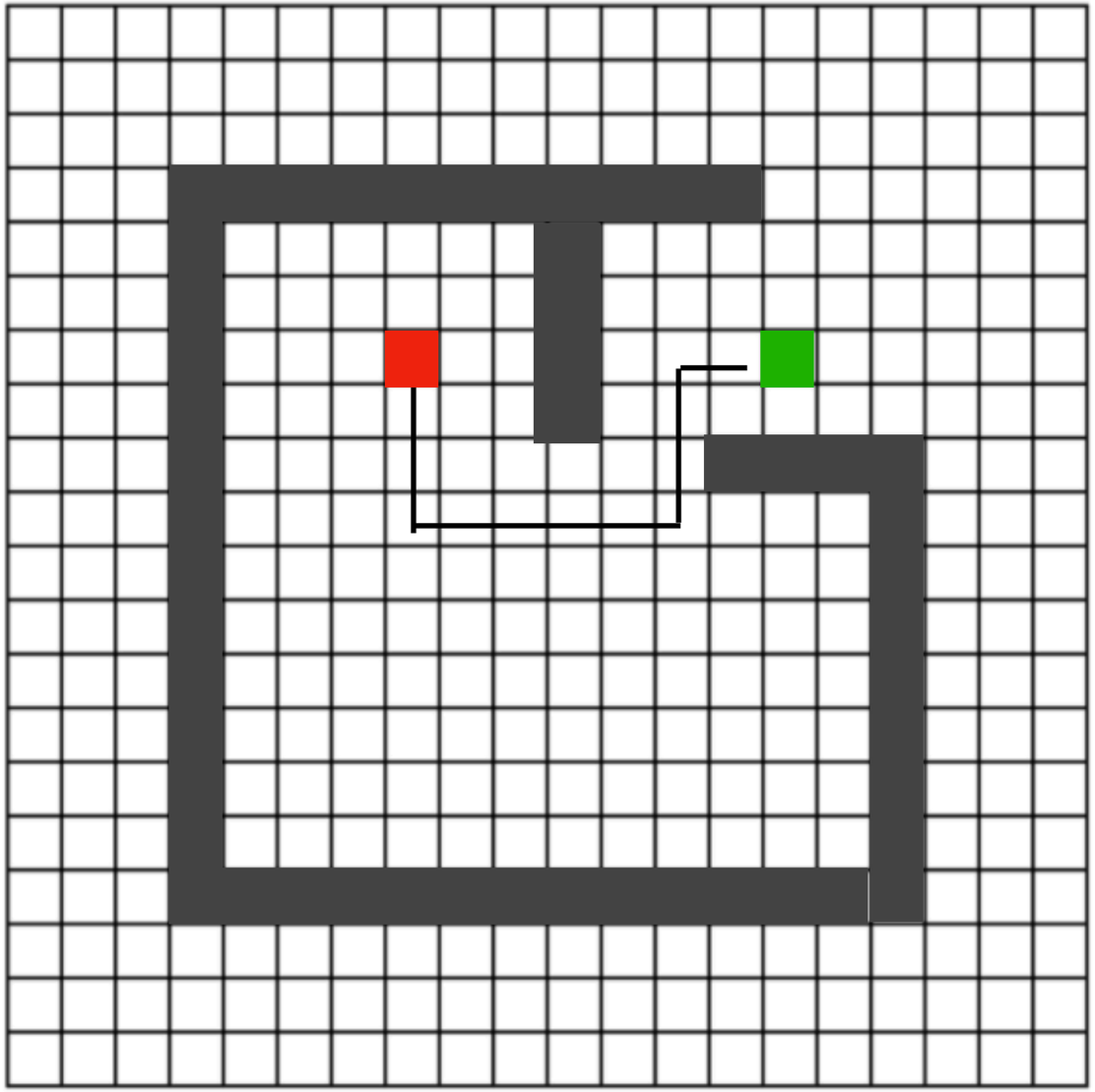
Vertex	Predecessor
S	-
W	S
V	S
U	V
Y	V
X	U

Lab exercise: Mission-planning in city maps

ART PARK
AI & Robotics Technology Park

- Open-street maps
- OSNX package to load map into a graph
- Small cities ~2000 edges, 5000 vertices
- Lat-Long based node query

Occupancy grids to graphs



- Occupancy grid is a common representation of the environment for autonomous robots
- Every free cell is a vertex
- Edges to only adjacent free cells
- Occupied cells are not added
- Easy to convert from occupancy grid to graph

Rapidly-exploring random trees (RRT)

ART PARK
AI & Robotics Technology Park

Djikstra/ A*

- Accepts pre-built graph with vertices/ edges
- Computes the shortest path between 2 nodes

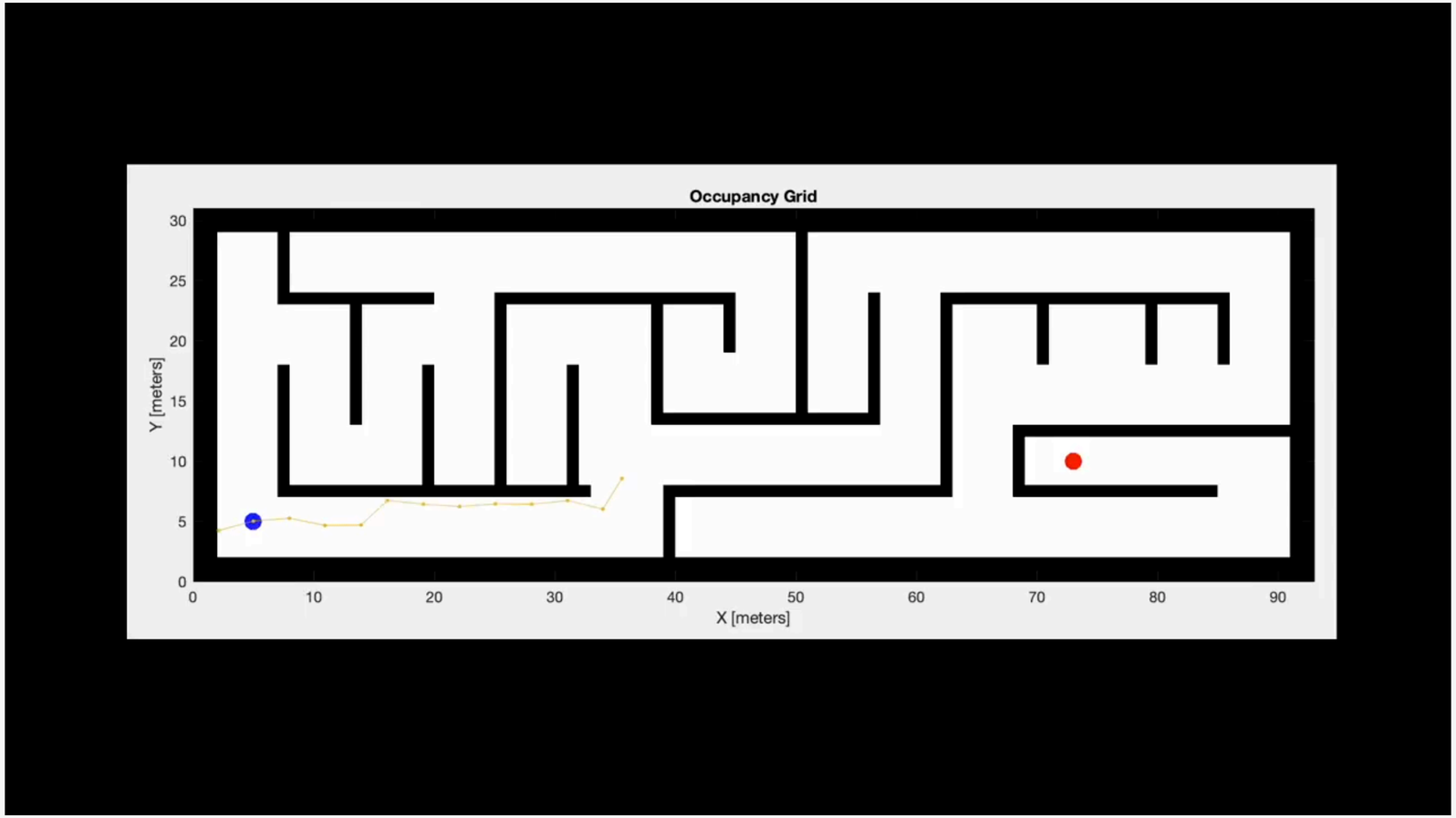
How to construct the graph?

What happens in case of high-dimensional grids?

RRT

- Randomly generate a node
- Find closest node in current graph
  - Method to madness: insert within some known distance of the nearest node
- Add edge to closest node
- Node/ Edge should not collide with obstacle
- Rinse and repeat

RRT* visualization



Source: https://www.youtube.com/watch?v=QR3U1dgc5RE

RRT* pseudo-code

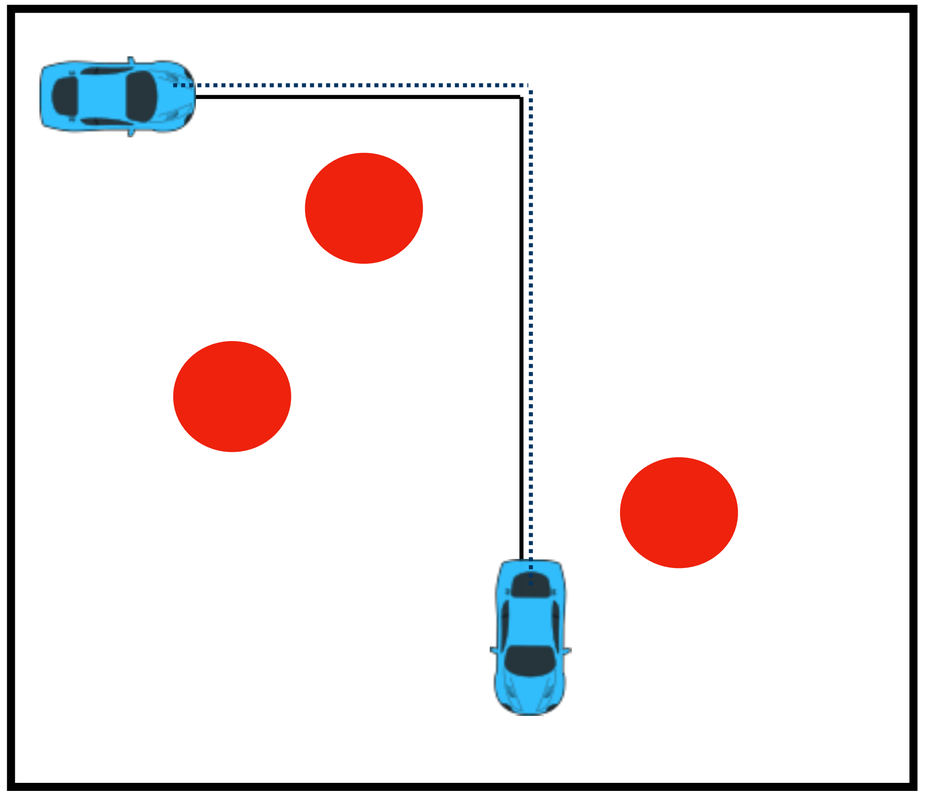
Lab exercise

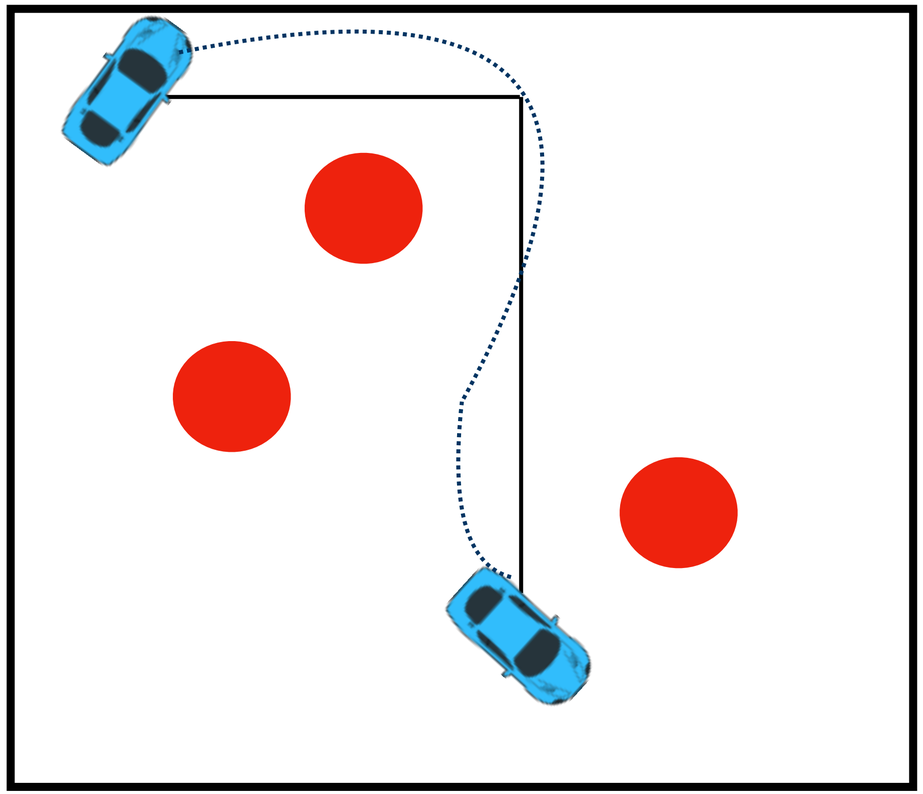
ART PARK
AI & Robotics Technology Park

```
G(V,E) = {V = [S], E = [ ]}
Iterate over n steps
    Xnew = FindRandomPosition()
    If Xnew close to obstacle, try again
    Xnearest = Nearest(G, Xnew)
    Cost(Xnew) = Distance(Xnew, Xnearest)
    Xbest, Xneighbors = findNeighbors(G, Xnew, radius)
    Link = Chain(Xnew, Xbest)
    For x’ in Xneighbors
        If Cost(Xnew) + Distance(Xnew, x’) < Cost(x’)
            Cost(x’) = Cost(Xnew)+Distance(Xnew, x’)
            Parent(x’) = Xnew
            G[Edges] += (Xnew, x’)
    G[Edges] += Link
Return G
```

https://theclassytim.medium.com/robotic-path-planning-rrt-and-rrt-212319121378

Path planning vs Trajectory planning



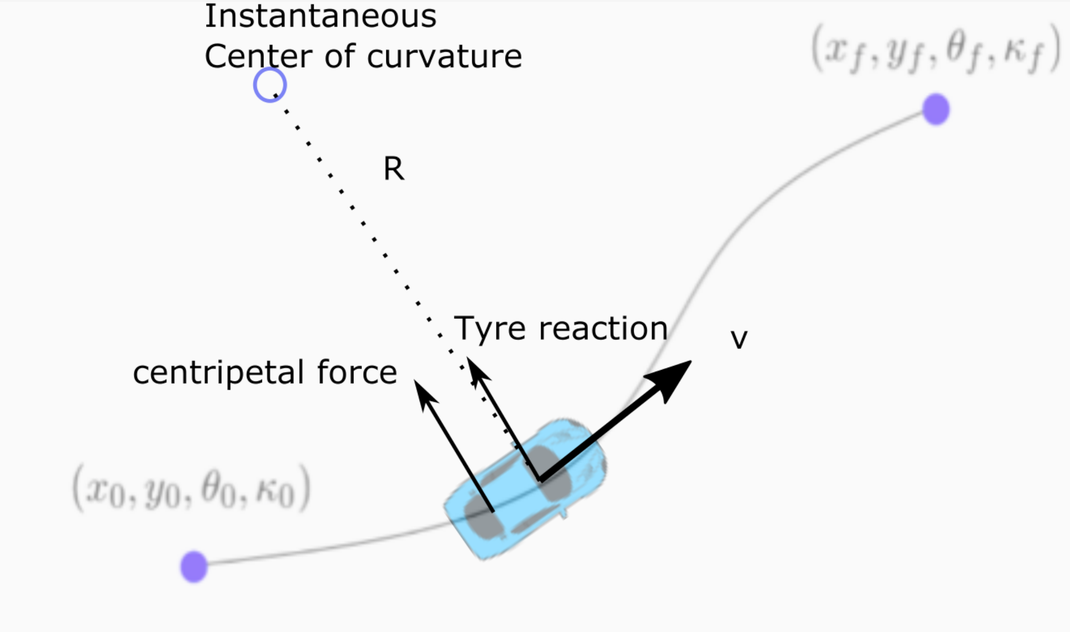


A*/ Djikstra/ RRT* are path planners

No notion of robot orientation/ velocities/ acceleration

Does not solve the parking problem

Path curvature



- Curved Path
  - Forward plus rotational motion
  - Weight shifts to the inner wheels
  - Lateral tyre reaction forces produces centripetal acceleration
  - $\alpha = \frac{v^2}{R} = \kappa v^2$

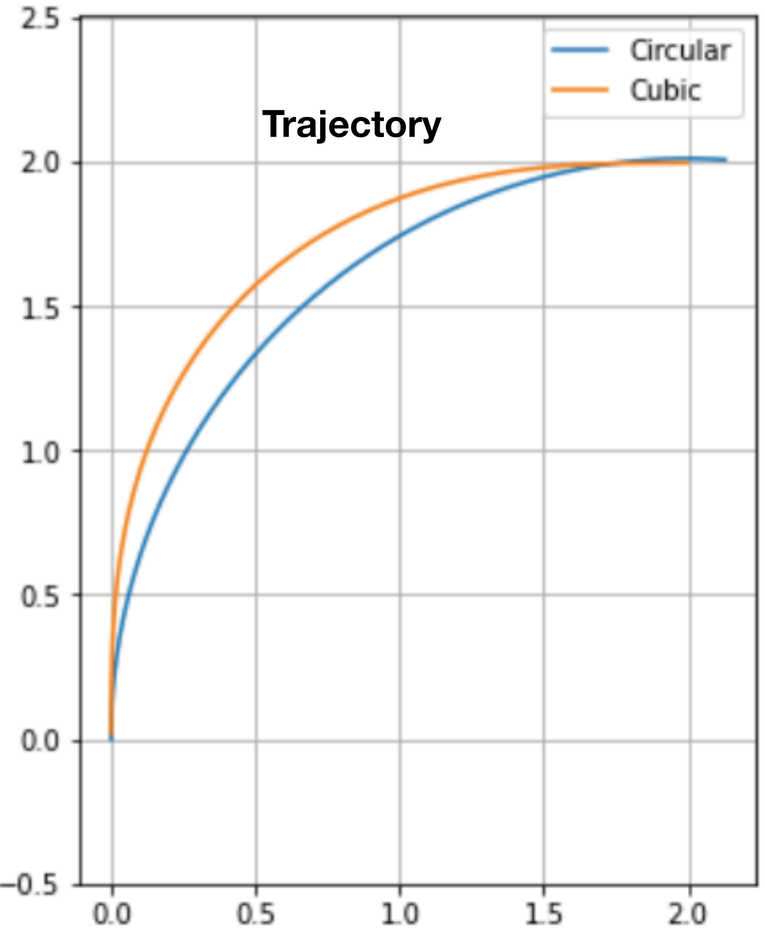
- Curvature: time-derivative of orientation
  - Higher-curvature means sharper turns, higher accelerations
- Higher-order derivatives of vehicle position => velocity, acceleration, jerk, snap
- Passenger comfort may impose boundary constraints on higher-order terms
- Path smoothness / continuity are much desired properties

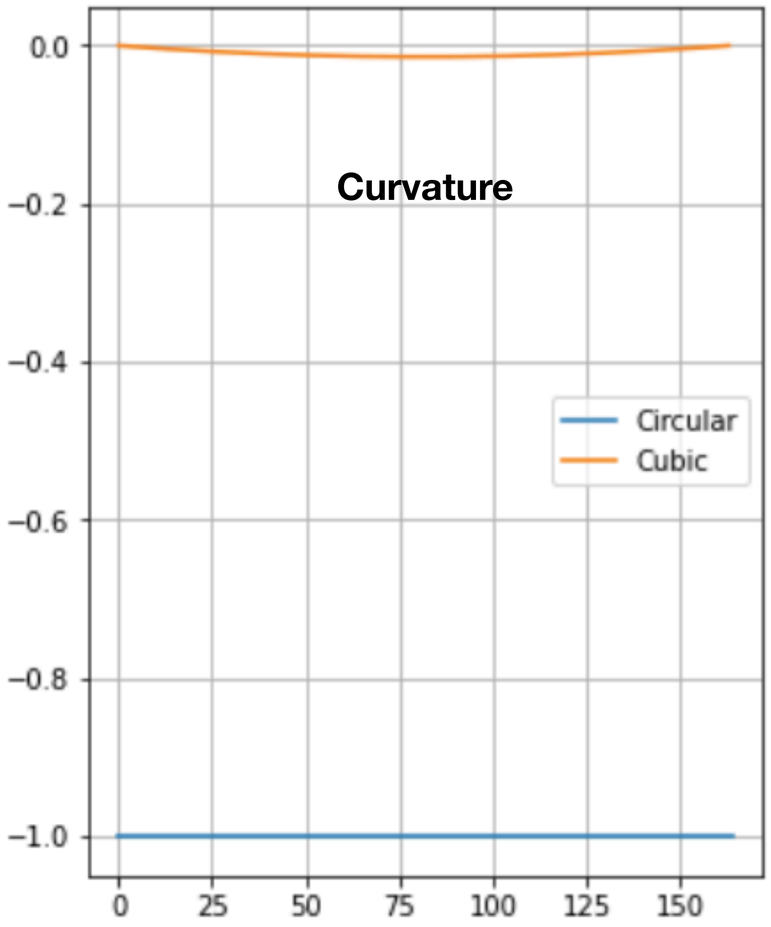
Path Curvature

ART PARK
AI & Robotics Technology Park

- $R = \frac{V}{\omega} \quad \kappa = \frac{1}{R}$

- Straight line: $\kappa = 0$
- Circular path: $\kappa =$ constant





- 2 m radius turn compared
- Circular trajectories have a fixed curvature of 1
- Cubic polynomials
  - Similar turn as circular (not as economic)
  - Curvature can start/ end at 0.0 - seamlessly merge with straight segments

Simple Curve-fitting approach

Fit a polynomial for smooth transition from θ_i to θ_f

For simplicity, $\theta_i = 1$ and $\theta_f = 0$

Our function can be parameterized as

$$f'(\theta) = K(\theta(1-\theta))^{2n}$$

$f'(0) = f'(1) = 0$

- n = 1 -> cubic polynomial
  - Example in week 1: $K = 6$
- n = 2 -> quintic polynomial

$\{\theta_i\}$ is fixed

$\{x_i\}$ and $\{y_i\}$ can be computed for a fixed v