
COMP250: Artificial Intelligence

# 6: Navigation

# Learning outcomes

- Outcome 1
- Outcome 2
- Outcome 3

# Pathfinding

# The problem

- We have a **graph**

# The problem

- We have a **graph**
  - **Nodes** (points)

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads
- E.g. a tile-based 2D game

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads
- E.g. a tile-based 2D game
  - Nodes = grid squares

# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads
- E.g. a tile-based 2D game
  - Nodes = grid squares
  - Edges = connections between adjacent squares

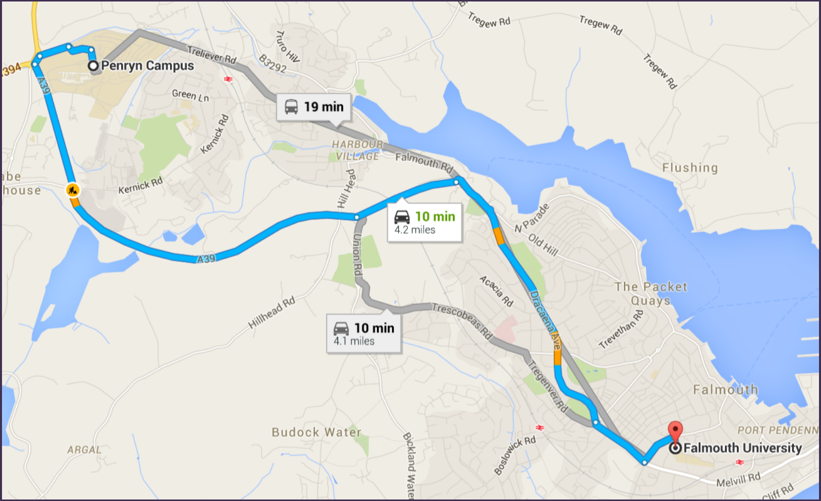
# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads
- E.g. a tile-based 2D game
  - Nodes = grid squares
  - Edges = connections between adjacent squares
- Given two nodes $A$ and $B$, find the **shortest path** from $A$ to $B$

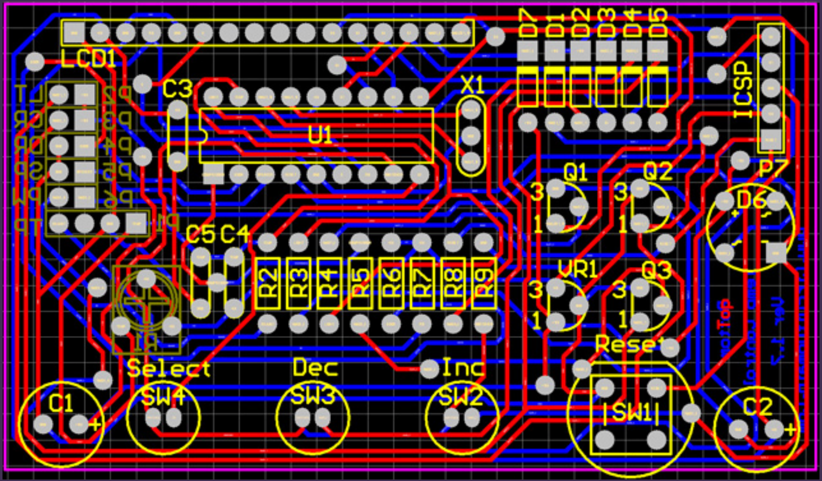
# The problem

- We have a **graph**
  - **Nodes** (points)
  - **Edges** (lines between points, each with a **length**)
- E.g. a road map
  - Nodes = addresses
  - Edges = roads
- E.g. a tile-based 2D game
  - Nodes = grid squares
  - Edges = connections between adjacent squares
- Given two nodes $A$ and $B$, find the **shortest path** from $A$ to $B$
  - "Shortest" in terms of edge lengths — could be distance, time, fuel cost, ...

# Applications of pathfinding



# Applications of pathfinding



# Applications of pathfinding

Many applications in game AI

# Applications of pathfinding

Many applications in game AI

- Non-player character AI

# Applications of pathfinding

Many applications in game AI

- Non-player character AI
- Mouse-based movement (e.g. strategy games)

# Applications of pathfinding

Many applications in game AI

- Non-player character AI
- Mouse-based movement (e.g. strategy games)
- Maze navigation

# Applications of pathfinding

Many applications in game AI

- Non-player character AI
- Mouse-based movement (e.g. strategy games)
- Maze navigation
- Puzzle solving

# $A^*$ search

Idea:

# A* search

Idea:

- Expand out from the start node

# $A^*$ search

Idea:

- Expand out from the start node
- Let $g(n)$ be the length of the path currently found between the start and node $n$

# A* search

Idea:

- Expand out from the start node
- Let $g(n)$ be the length of the path currently found between the start and node $n$
- Let $h(n)$ be an **estimate** of the distance from $n$ to the goal

# $A^*$ search

Idea:

- Expand out from the start node
- Let $g(n)$ be the length of the path currently found between the start and node $n$
- Let $h(n)$ be an **estimate** of the distance from $n$ to the goal
- Prioritise nodes for which $g(n) + h(n)$ is small

# Properties of A* search

- A* is **guaranteed** to find the shortest path if the distance estimate is **admissible**

# Properties of A* search

- A* is **guaranteed** to find the shortest path if the distance estimate is **admissible**
- Essentially, **admissible** means it must be an **underestimate**

# Properties of A* search

- A* is **guaranteed** to find the shortest path if the distance estimate is **admissible**
- Essentially, **admissible** means it must be an **underestimate**
  - E.g. straight line Euclidean distance is clearly an underestimate for actual travel distance

# Properties of A* search

- A* is **guaranteed** to find the shortest path if the distance estimate is **admissible**
- Essentially, **admissible** means it must be an **underestimate**
  - E.g. straight line Euclidean distance is clearly an underestimate for actual travel distance
- A* is an example of **heuristic** search

# Properties of A* search

- A* is **guaranteed** to find the shortest path if the distance estimate is **admissible**
- Essentially, **admissible** means it must be an **underestimate**
  - E.g. straight line Euclidean distance is clearly an underestimate for actual travel distance
- A* is an example of **heuristic** search
  - In AI, a heuristic is an estimate based on human intuition

# Properties of $A^*$ search

- $A^*$ is **guaranteed** to find the shortest path if the distance estimate is **admissible**
- Essentially, **admissible** means it must be an **underestimate**
  - E.g. straight line Euclidean distance is clearly an underestimate for actual travel distance
- $A^*$ is an example of **heuristic** search
  - In AI, a heuristic is an estimate based on human intuition
  - Heuristics are often used to prioritise search, i.e. explore the most promising options first