CS3243 : Introduction to Artificial Intelligence

Tutorial 1

NUS School of Computing

June 30, 2022

A bit about me

- I'm Sumanth Yalamarty (you can call me Sumanth)
- Studying Computer Science here at NUS

A bit about me

- I'm Sumanth Yalamarty (you can call me Sumanth)
- Studying Computer Science here at NUS
- Telegram handle : *@s7manth*
- Email address : *sumanthyalamarty@u.nus.edu*

Admin

- Telegram Group
- Weekly DQ (due Fridays) and Tutorial Assignment (due Sundays)
- Clarifications, Ideas, Doubts, Consultations

Review

- Building the right intuition about Artificial Intelligence
- Study of the design of *Rational/Intelligent agents*

Review

- Building the right intuition about Artificial Intelligence
- Study of the design of *Rational/Intelligent agents*

- Agents? Rationality? Intelligence? (last two pretty subjective topics in general)

Review

- Various kinds of Agents
- Properties of the Task Environment
- Search Space
- Graph and Tree-based search implementations
- Uninformed search strategies : BFS, UCS, DFS, DLS, IDS

Tutorial Question 1(a)

- Sudoku is a popular number puzzle that works as follows: we are given a 9×9 square grid; some squares have numbers, while some are blank. Objective is to fill in the blanks with numbers from $1 - 9$ such that each row, column and the highlighted 3×3 squares contain no duplicate entries.

Tutorial Question 1(a)

- Sudoku is a popular number puzzle that works as follows: we are given a 9×9 square grid; some squares have numbers, while some are blank. Objective is to fill in the blanks with numbers from $1 - 9$ such that each row, column and the highlighted 3×3 squares contain no duplicate entries.
- Sudoku puzzles can solved easily after being modelled as a CSP (which we will cover later in the module). We will consider the problem of generating Sudoku puzzles.

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	
Deterministic *vs* Stochastic	
Epsodic *vs* Sequential	
Discrete *vs* Continuous	
Single *vs* Multi-agent	
Static *vs* Dynamic	

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	
Epsodic *vs* Sequential	
Discrete *vs* Continuous	
Single *vs* Multi-agent	
Static *vs* Dynamic	

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	
Discrete *vs* Continuous	
Single *vs* Multi-agent	
Static *vs* Dynamic	

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	Episodic*
Discrete *vs* Continuous	
Single *vs* Multi-agent	
Static *vs* Dynamic	

*Something to think about : How would humans perceive this as compared to the machine?

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	Episodic*
Discrete *vs* Continuous	Discrete
Single *vs* Multi-agent	
Static *vs* Dynamic	

*Something to think about : How would humans perceive this as compared to the machine?

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	Episodic*
Discrete *vs* Continuous	Discrete
Single *vs* Multi-agent	Single
Static *vs* Dynamic	

*Something to think about : How would humans perceive this as compared to the machine?

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	Episodic*
Discrete *vs* Continuous	Discrete
Single *vs* Multi-agent	Single
Static *vs* Dynamic	Static

*Something to think about : How would humans perceive this as compared to the machine?

Tutorial Question 1(a)

Environment Characteristic	**Sudoku Puzzle**
Fully *vs* Partially Observable	Fully
Deterministic *vs* Stochastic	Deterministic
Epsodic *vs* Sequential	Episodic*
Discrete *vs* Continuous	Discrete
Single *vs* Multi-agent	Single
Static *vs* Dynamic	Static

Key Takeaway : Getting to know about the problem in a more proper/formal way

Tutorial Question 1(b)

- State Representation : A state in this problem is a (partial) valid representation of the sudoku puzzle. More formally, it would be a matrix $A \in \{0...9\}^{9\times 9}$ with 0 representing a blank square.

Tutorial Question 1(b)

- State Representation : A state in this problem is a (partial) valid representation of the sudoku puzzle. More formally, it would be a matrix $A \in \{0...9\}^{9\times 9}$ with 0 representing a blank square.
- The problem of generating a sudoku puzzle

Tutorial Question 1(b)

- State Representation : A state in this problem is a (partial) valid representation of the sudoku puzzle. More formally, it would be a matrix $A \in \{0...9\}^{9\times 9}$ with 0 representing a blank square.
- The problem of generating a sudoku puzzle
- Initial and Goal state?

Tutorial Question 1(b)

- State Representation : A state in this problem is a (partial) valid representation of the sudoku puzzle. More formally, it would be a matrix $A \in \{0...9\}^{9\times 9}$ with 0 representing a blank square.
- The problem of generating a sudoku puzzle
- Initial and Goal state?

2	5	8	7	3	6	9	4	1
6	1	9	8	2	4	3	5	7
4	3	7	9	1	5	2	6	8
3	9	5	2	7	1	4	8	6
7	6	2	4	9	8	1	3	5
8	4	1	6	5	3	7	2	9
1	8	4	3	6	9	5	7	2
5	7	6	1	4	2	8	9	3
9	2	3	5	8	7	6	1	4

Tutorial Question 1(b)

- State Representation : A state in this problem is a (partial) valid representation of the sudoku puzzle. More formally, it would be a matrix $A \in \{0...9\}^{9\times 9}$ with 0 representing a blank square.
- Initial and Goal state?
- The initial state is a completely filled out grid of numbers, where the grid is valid : all rows, columns and 3×3 squares contain all numbers between $\{1...9\}$. Note that any state in the problem will be a valid goal state.

Tutorial Question 1(b)

- An action would be removing a number from the grid. More formally, we take as input a matrix A and a matrix $E_{i,j}(a)$ where $E_{i,j}(a) \in \{0...9\}^{9\times 9}$ is a matrix of all zeros except for a non-zero value $a \in \{1...9\}$ at coordinate (i, j). An action would be setting $A - E_{i,j}(a)$. Note that actions must not result in boards with two or more solutions.

Tutorial Question 1(b)

- An action would be removing a number from the grid. More formally, we take as input a matrix A and a matrix $E_{i,j}(a)$ where $E_{i,j}(a) \in \{0...9\}^{9\times 9}$ is a matrix of all zeros except for a non-zero value $a \in \{1...9\}$ at coordinate (i, j). An action would be setting $A - E_{i,j}(a)$. Note that actions must not result in boards with two or more solutions.
- We continue blanking out squares as long as the resulting puzzle can be completed in only one way. Hence, the transition model would be

$$T : A,\ E_{i,j}(a) = A - E_{i,j}(a)$$

Tutorial Question 1(b)

- An action would be removing a number from the grid. More formally, we take as input a matrix A and a matrix $E_{i,j}(a)$ where $E_{i,j}(a) \in \{0...9\}^{9\times 9}$ is a matrix of all zeros except for a non-zero value $a \in \{1...9\}$ at coordinate (i, j). An action would be setting $A - E_{i,j}(a)$. Note that actions must not result in boards with two or more solutions.
- We continue blanking out squares as long as the resulting puzzle can be completed in only one way. Hence, the transition model would be

$$T : A,\ E_{i,j}(a) = A - E_{i,j}(a)$$

- **Key Takeaway** : Right representation of the problem at hand

Tutorial Question 2(a)

- Graph-based *vs* Tree-based implementations of search algorithms

Tutorial Question 2(a)

- Graph-based *vs* Tree-based implementations of search algorithms

- Graph search algorithms will not explore redundtant paths – i.e., they only explore *nodes*
  (i) with associated states that have not been visited, or
  (ii) that have been visited, but previously via less optimal paths.

Tutorial Question 2(a)

- Graph-based *vs* Tree-based implementations of search algorithms

- Graph search algorithms will not explore redundtant paths – i.e., they only explore *nodes*
  (i) with associated states that have not been visited, or
  (ii) that have been visited, but previously via less optimal paths.
- Tree-based algorthms have no such restriction; they consider all paths, including redundant ones

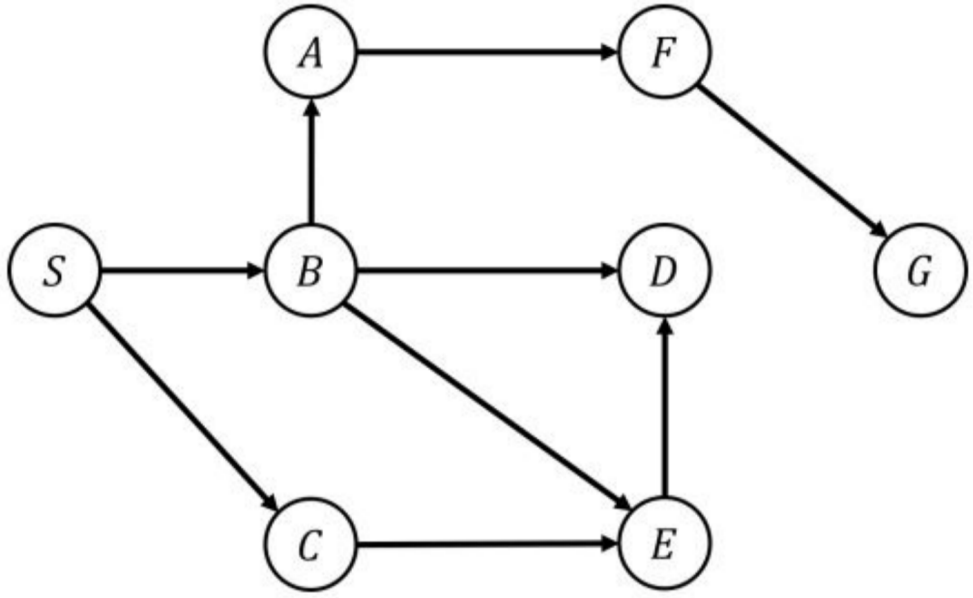
Tutorial Question 2(a)

- Graph-based *vs* Tree-based implementations of search algorithms

- Graph search algorithms will not explore redundtant paths – i.e., they only explore *nodes*
  (i) with associated states that have not been visited, or
  (ii) that have been visited, but previously via less optimal paths.
- Tree-based algorthms have no such restriction; they consider all paths, including redundant ones
- **Key Takeaway** : The difference between states and nodes (often undermined)

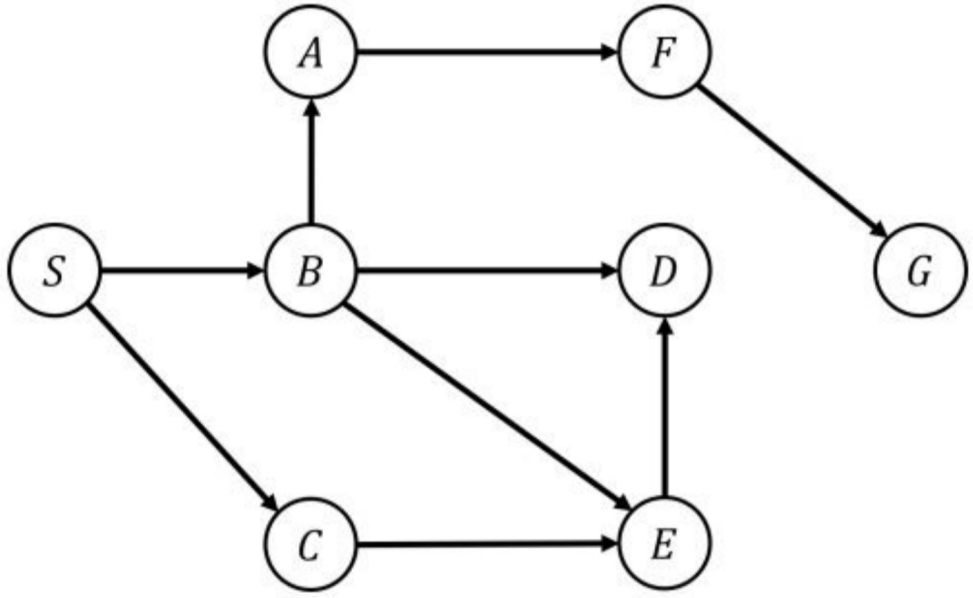
Tutorial Question 2(b)

- Finding paths



Tutorial Question 2(b)

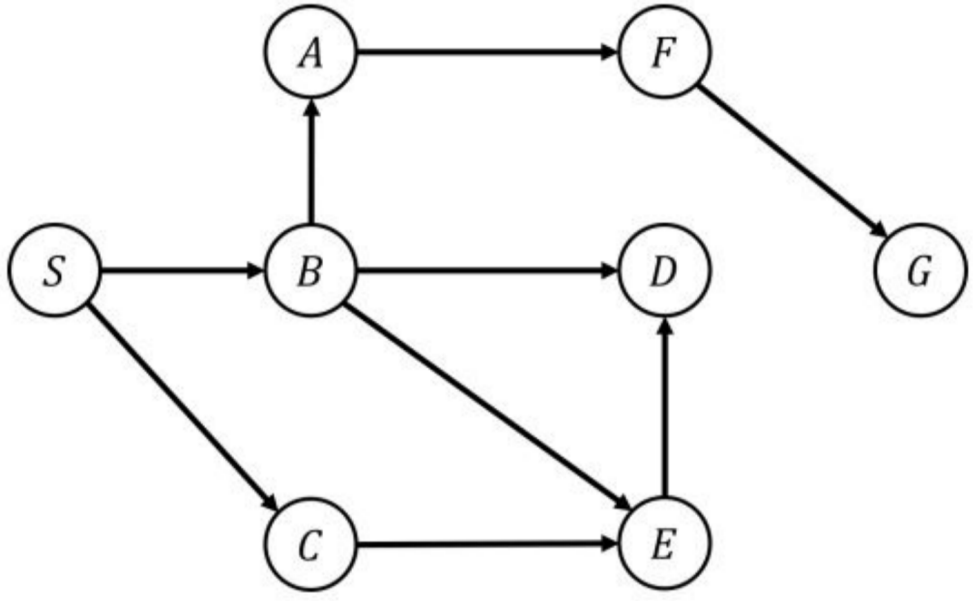
- Finding paths



- DFS with tree-based implementation

Tutorial Question 2(b)

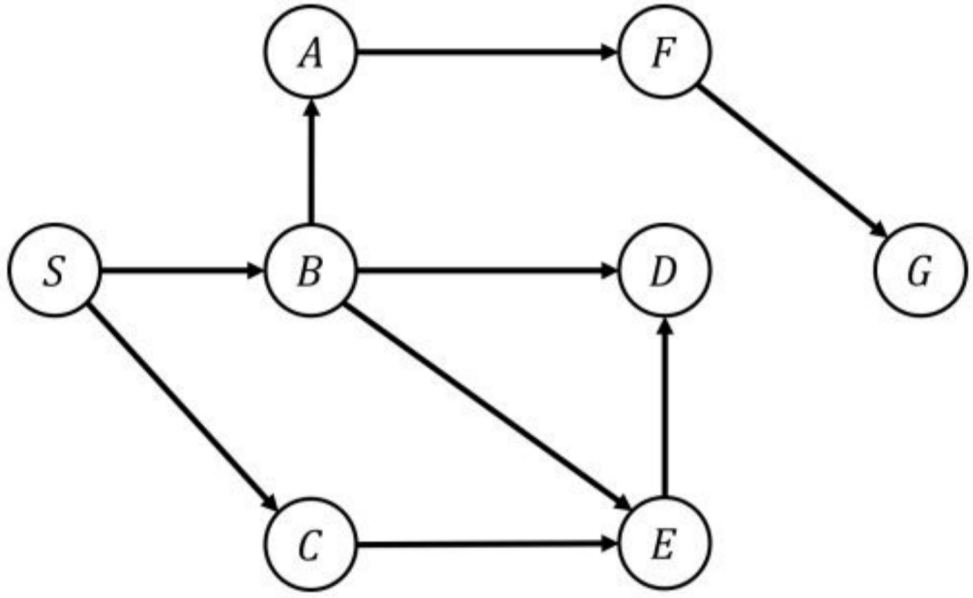
- Finding paths



- DFS with tree-based implementation
- S–C–E–D–B–E–D–D–A–F–G

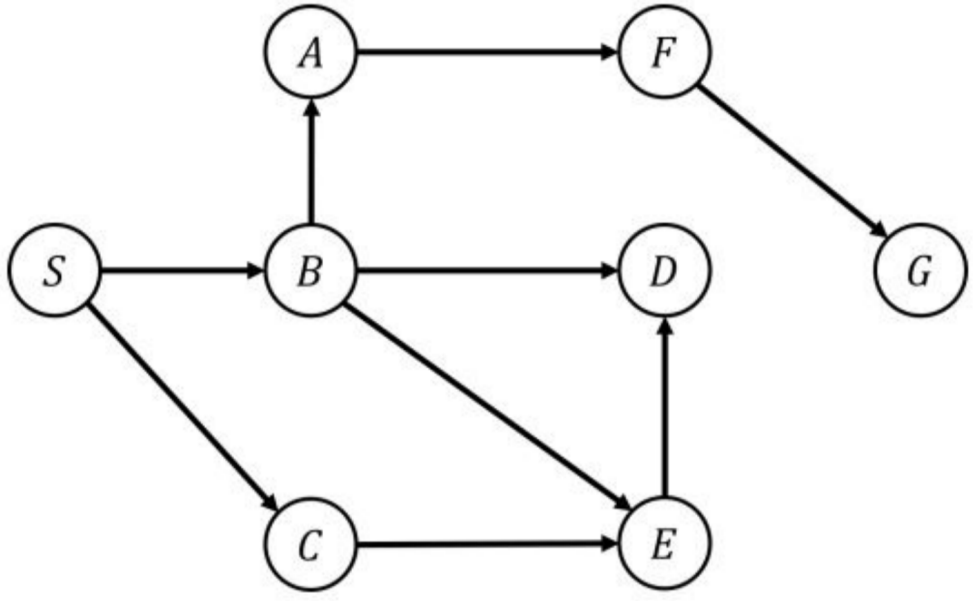
Tutorial Question 2(b)

- Finding paths



Tutorial Question 2(b)

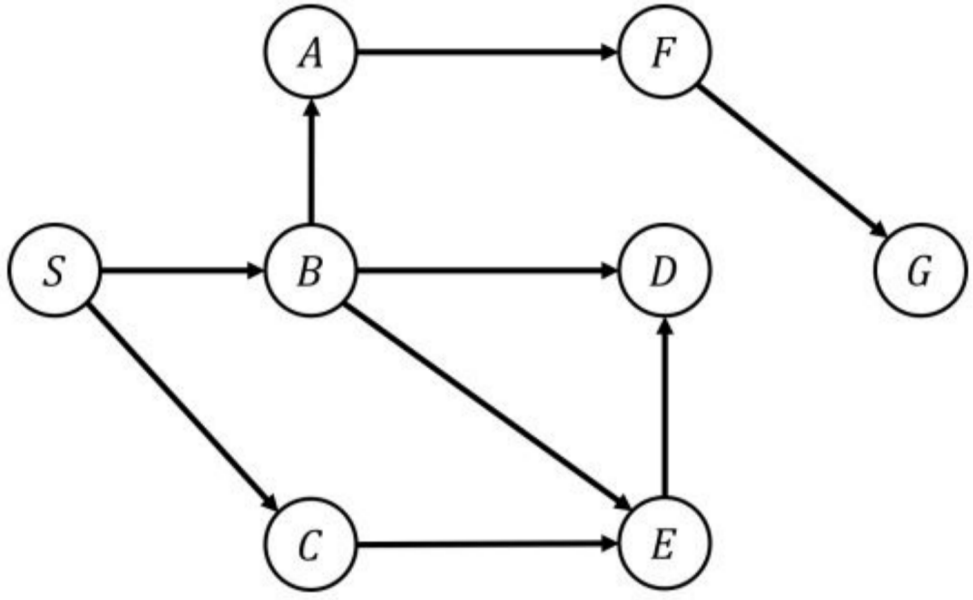
- Finding paths



- DFS with graph-based implementation

Tutorial Question 2(b)

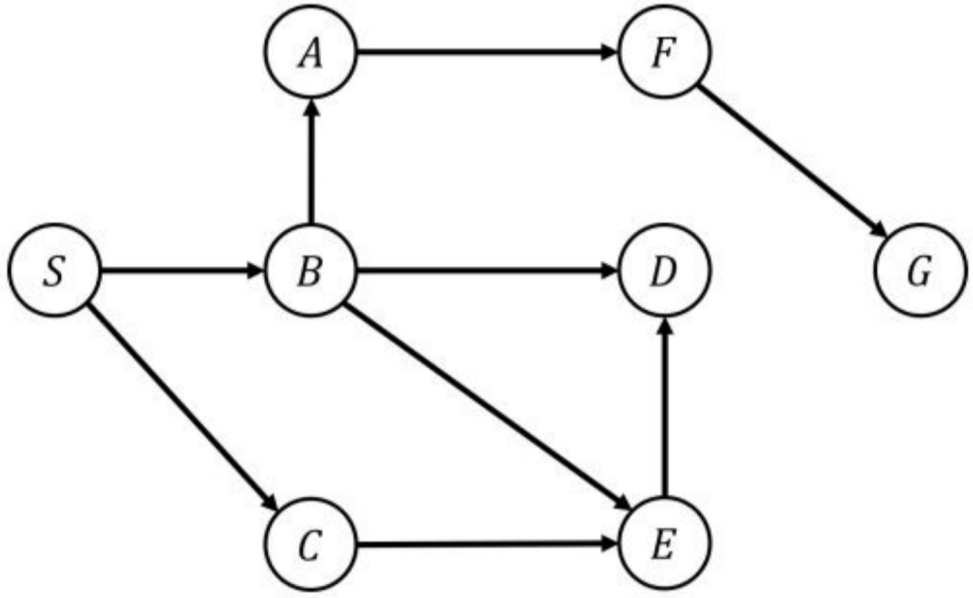
- Finding paths



- DFS with graph-based implementation
- S–C–E–D–B–A–F–G

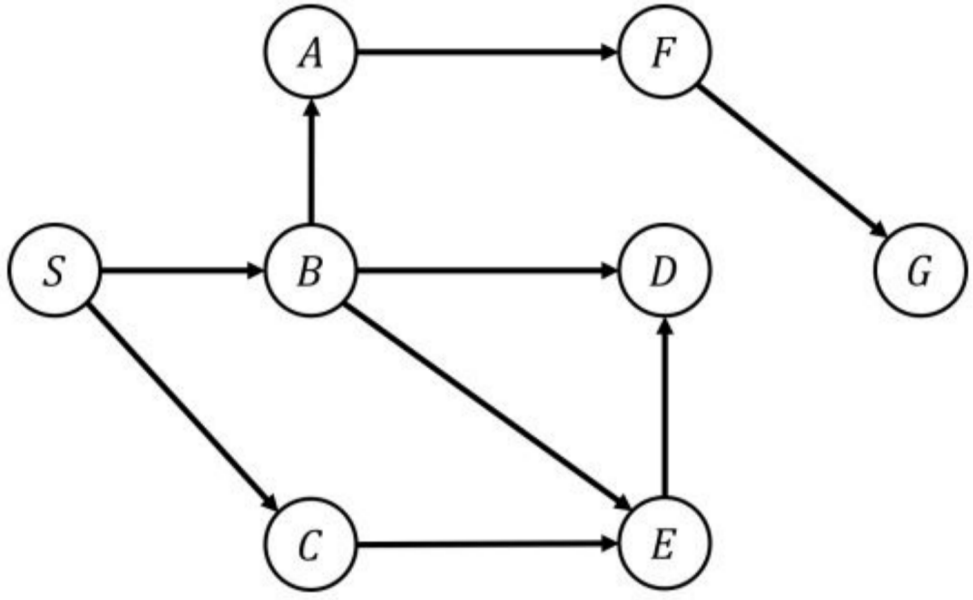
Tutorial Question 2(b)

- Finding paths



Tutorial Question 2(b)

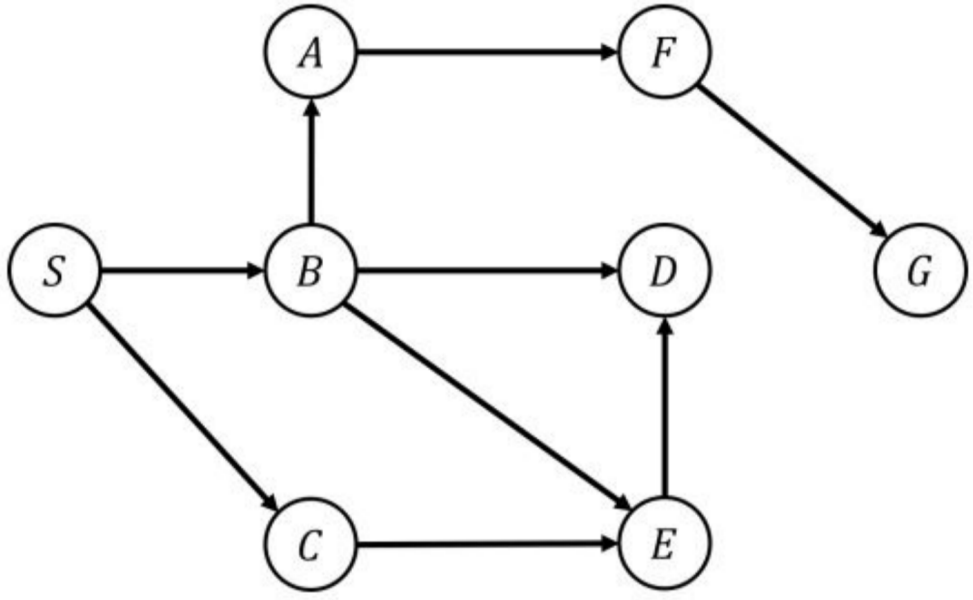
- Finding paths



- BFS with tree-based implementation

Tutorial Question 2(b)

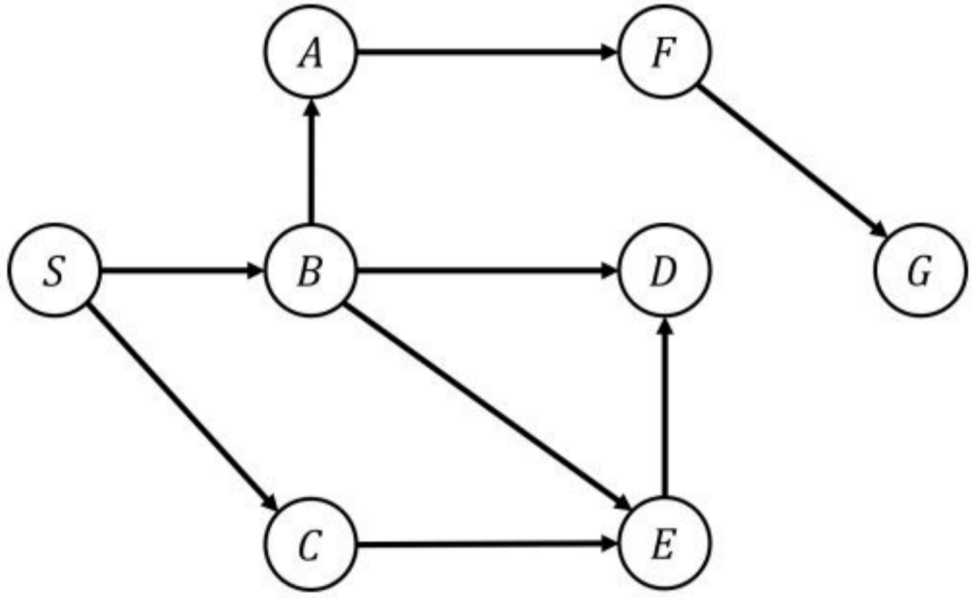
- Finding paths



- BFS with tree-based implementation
- S–B–C–A–D–E–E–F–D–D–G

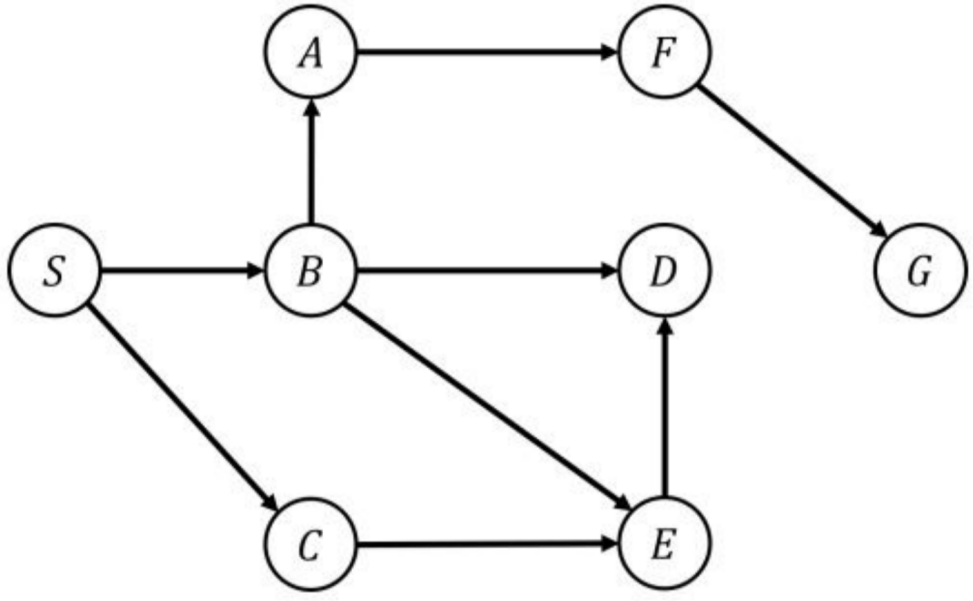
Tutorial Question 2(b)

- Finding paths



Tutorial Question 2(b)

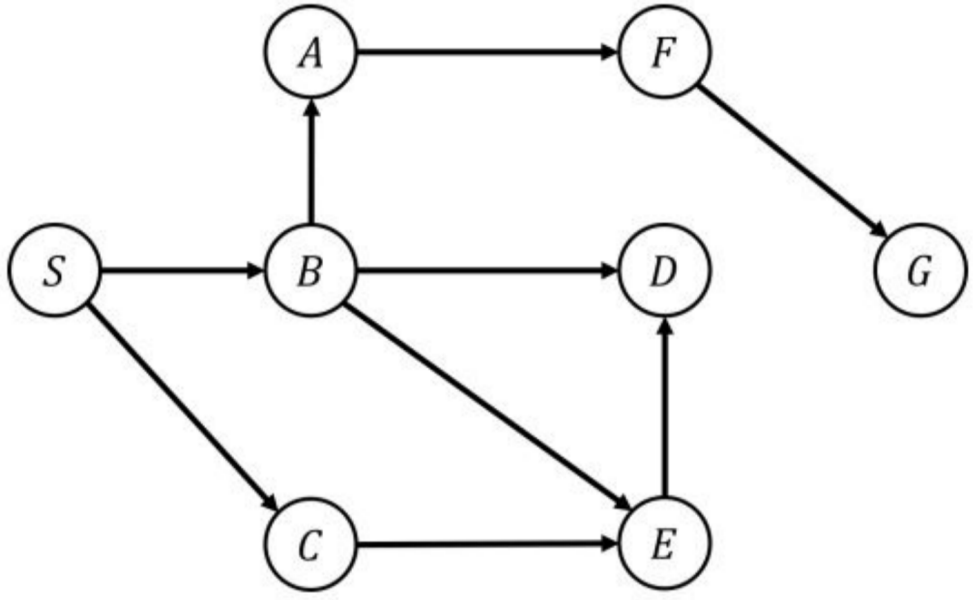
- Finding paths



- BFS with graph-based implementation

Tutorial Question 2(b)

- Finding paths



- BFS with graph-based implementation
- S–B–C–A–D–E–F–G

Tutorial Question 3

- UCS and optimality
- Given that each action cost exceeds some small positive constant ϵ, completeness may be assumed.

Tutorial Question 3

- UCS and optimality
- Given that each action cost exceeds some small positive constant ϵ, completeness may be assumed.
- Claim : Whenever UCS expands a node n, the optimal path to that node has been found.

Tutorial Question 3

- If this was not the case (ie, the path to n was not optimal, let us denote this path T), there would have to be another frontier node n' on the optimal path from the start node to n (denote this optimal path U).

Tutorial Question 3

- If this was not the case (ie, the path to n was not optimal, let us denote this path T), there would have to be another frontier node n' on the optimal path from the start node to n (denote this optimal path U).
- By definition, $g(n)$ via U would be less than $g(n)$ via T, which implies that n' should have been selected first.

Tutorial Question 3

- If this was not the case (ie, the path to n was not optimal, let us denote this path T), there would have to be another frontier node n' on the optimal path from the start node to n (denote this optimal path U).
- By definition, $g(n)$ via U would be less than $g(n)$ via T, which implies that n' should have been selected first.
- If action costs are non-negative, path costs, ie, g values never get smaller as nodes are added.

Tutorial Question 3

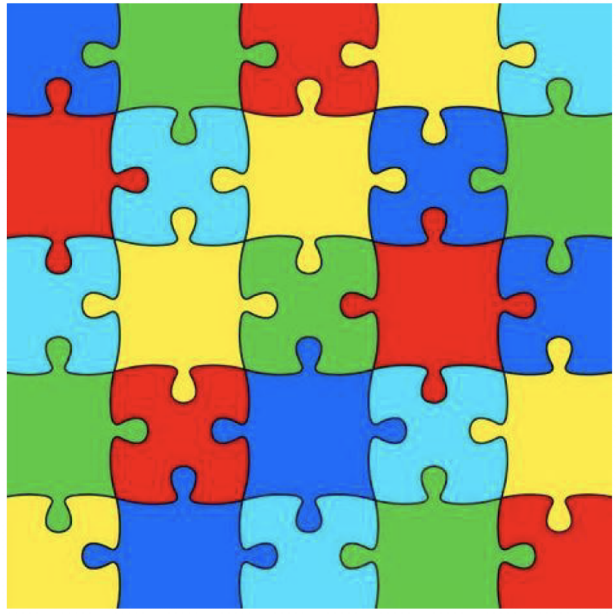
- If this was not the case (ie, the path to n was not optimal, let us denote this path T), there would have to be another frontier node n' on the optimal path from the start node to n (denote this optimal path U).
- By definition, $g(n)$ via U would be less than $g(n)$ via T, which implies that n' should have been selected first.
- If action costs are non-negative, path costs, ie, g values never get smaller as nodes are added.
- The above two points together imply that UCS expands nodes in the other of the optimal path cost.

Tutorial Question 3

- If this was not the case (ie, the path to n was not optimal, let us denote this path T), there would have to be another frontier node n' on the optimal path from the start node to n (denote this optimal path U).
- By definition, $g(n)$ via U would be less than $g(n)$ via T, which implies that n' should have been selected first.
- If action costs are non-negative, path costs, ie, g values never get smaller as nodes are added.
- The above two points together imply that UCS expands nodes in the other of the optimal path cost.
- Something to think about : Would UCS be optimal with tree-based implementation?

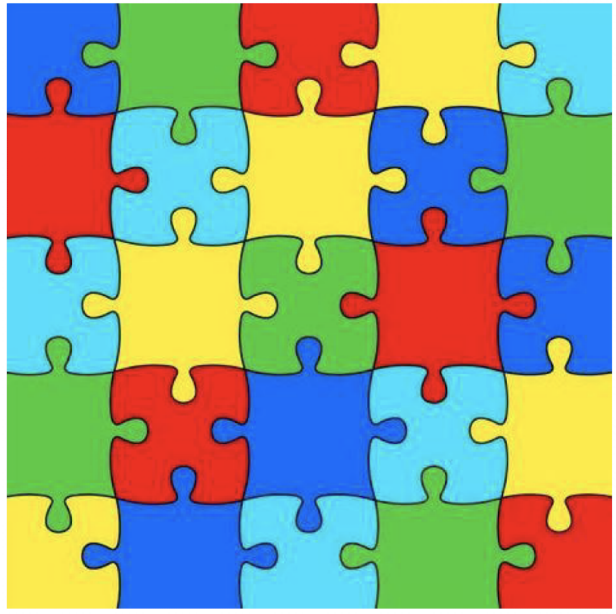
Tutorial Question 4

- Jigsaw puzzle



Tutorial Question 4

- Jigsaw puzzle



- Some thoughts

Tutorial Question 4

- State representation
  - Keeps track of the puzzle pieces that are connected/unconnected
  - Keeps track of the connections between connected puzzle pieces

Tutorial Question 4

- State representation
  - Keeps track of the puzzle pieces that are connected/unconnected
  - Keeps track of the connections between connected puzzle pieces
- Initial state
  - Depending on the representation, either the set of connected puzzle pieces is empty, or the set of unconnected puzzle pieces is full. There should also be no connections between puzzle pieces initally

Tutorial Question 4

- State representation
  - Keeps track of the puzzle pieces that are connected/unconnected
  - Keeps track of the connections between connected puzzle pieces
- Initial state
  - Depending on the representation, either the set of connected puzzle pieces is empty, or the set of unconnected puzzle pieces is full. There should also be no connections between puzzle pieces initally
- Actions
  - Taking two unconnected puzzle pieces and establishing a connection between them. Note that you must mention that the pair of puzzle pieces picked can be legally connected (i.e., you cannot simply take any two puzzle pieces).

Tutorial Question 4

- Transition model
  - Depending on the representation, can either add to the connected set or remove from the unconnected set

Tutorial Question 4

- Transition model
  - Depending on the representation, can either add to the connected set or remove from the unconnected set
- Step costs are 1 (or any positive constant value, cannot be 0)

Tutorial Question 4

- Transition model
  - Depending on the representation, can either add to the connected set or remove from the unconnected set
- Step costs are 1 (or any positive constant value, cannot be 0)
- Goal test
  - Depending on the representation, either the connected set is full, or the unconnected set if empty
  - Check that for every puzzle piece, there should not be more connections than its number of available legal connections

Thank you!

If you have any questions, please don't hesitate. Feel free to ask!
We are here to learn together!