AI - Lecture 1b: Route Finding

Chris Thornton

October 30, 2014

Introduction

The goal in AI is to reproduce intelligent behaviour.

The focus is particularly on replicating *thought processes* and *knowledge representation*.

Other approaches, such as Alife, neural networks and robotics, focus more on replication of behaviour.

AI approaches are informed by an

Introduction

The goal in AI is to reproduce intelligent behaviour.

The focus is particularly on replicating *thought processes* and *knowledge representation*.

Other approaches, such as Alife, neural networks and robotics, focus more on replication of behaviour.

AI approaches are informed by an

- introspective understanding of thought processes,

Introduction

The goal in AI is to reproduce intelligent behaviour.

The focus is particularly on replicating *thought processes* and *knowledge representation*.

Other approaches, such as Alife, neural networks and robotics, focus more on replication of behaviour.

AI approaches are informed by an

- introspective understanding of thought processes,
- concepts of symbolic computation and

Introduction

The goal in AI is to reproduce intelligent behaviour.

The focus is particularly on replicating *thought processes* and *knowledge representation*.

Other approaches, such as Alife, neural networks and robotics, focus more on replication of behaviour.

AI approaches are informed by an

- introspective understanding of thought processes,
- concepts of symbolic computation and
- principles of mathematics

Introduction

The goal in AI is to reproduce intelligent behaviour.

The focus is particularly on replicating *thought processes* and *knowledge representation*.

Other approaches, such as Alife, neural networks and robotics, focus more on replication of behaviour.

AI approaches are informed by an

- introspective understanding of thought processes,
- concepts of symbolic computation and
- principles of mathematics

Search as a key concept

A key discovery in AI has been that many forms of knowledge and thought can be represented in terms of a mechanism which

Search as a key concept

A key discovery in AI has been that many forms of knowledge and thought can be represented in terms of a mechanism which

(1) identifies ways in which possible actions can be arranged into sequences

Search as a key concept

A key discovery in AI has been that many forms of knowledge and thought can be represented in terms of a mechanism which

(1) identifies ways in which possible actions can be arranged into sequences

(2) finds a ‘route’ through the sequences which achieves a desired result.

Search as a key concept

A key discovery in AI has been that many forms of knowledge and thought can be represented in terms of a mechanism which

(1) identifies ways in which possible actions can be arranged into sequences

(2) finds a ‘route’ through the sequences which achieves a desired result.

This is the process known as **search**.

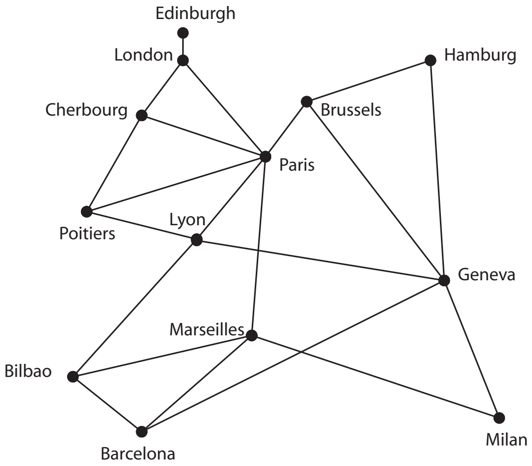
Most AI methods use search in one way or another.

Route finding

In the simplest case, the possible actions are physical transitions from one location to another.

Search can then be used to discover a literal route from a starting point to a goal location.

Toy rail map of Europe



Each (blob-connecting) line represents a direct rail connection somewhere in western europe.

Route finding task

Route-finding task

Route finding task

Route-finding task

- Given knowledge of direct rail connections, what's the shortest rail itinery which gets you from A to B?

Route finding task

Route-finding task

- Given knowledge of direct rail connections, what's the shortest rail itinery which gets you from A to B?

In this problem, action sequences form a tree structure.

At some given point, certain actions are possible.

These actions take you to new points.

At each of those new points, more actions are possible.

And so on.

At each point the possible actions form a branch.

Joining up the branches gives you a tree.

Search by generation

To find a solution, we need to search the tree of possible action sequences looking for one with the right start and finish.

But since the tree doesn't actually exist to begin with, we will have to generate it first.

If we are going to do this, we may as well inspect nodes as we are going along.

So, in practice, 'tree generation' and 'search' are merged into one process.

Search strategies

The two basic methods of search:

Search strategies

The two basic methods of search:

- *Depth-first search* (DFS) always expand nodes at a deeper level of the tree whenever there is a choice.

Search strategies

The two basic methods of search:

- *Depth-first search* (DFS) always expand nodes at a deeper level of the tree whenever there is a choice.
- *Breadth-first search* (BFS): always expand every node at the present level of the tree before moving to any deeper level.

Search strategies

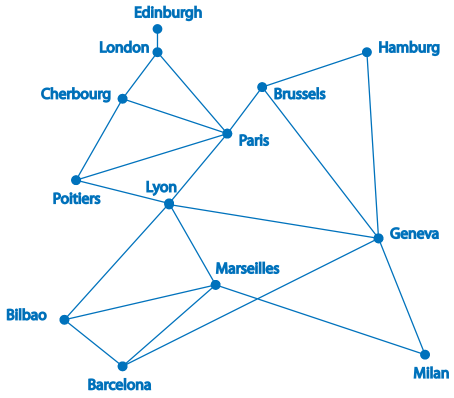
The two basic methods of search:

- *Depth-first search* (DFS) always expand nodes at a deeper level of the tree whenever there is a choice.
- *Breadth-first search* (BFS): always expand every node at the present level of the tree before moving to any deeper level.

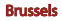
DFS is a ‘maverick’. BFS is ‘conservative’.

The two strategies can be illustrated by showing how they generate the search tree for the route-finding problem.

Breadth-first search from ‘Brussels’: step 1

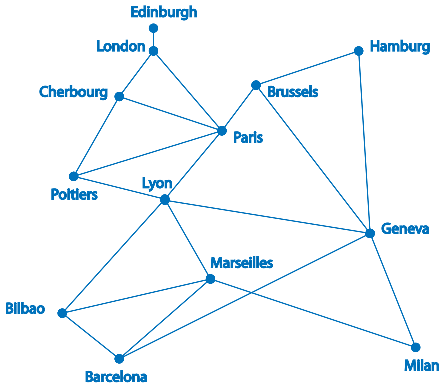


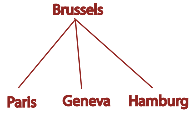
CONNECTIONS



BFS TREE

Step 2

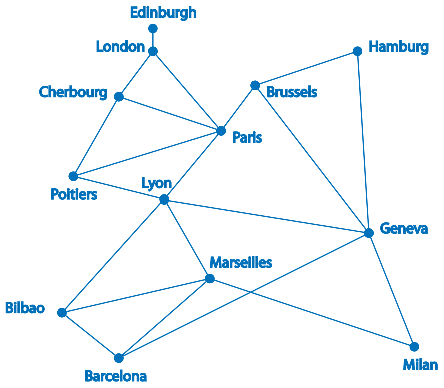




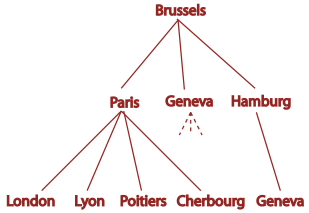
CONNECTIONS

BFS TREE

Step 3

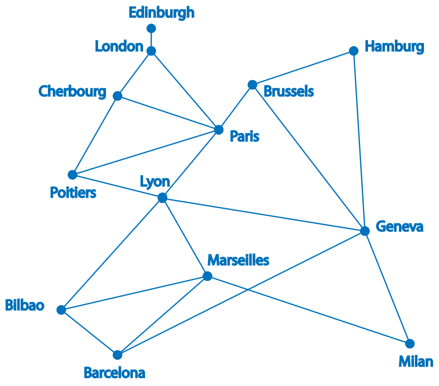


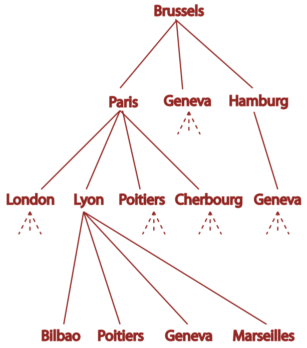
CONNECTIONS



BFS TREE

Step 4

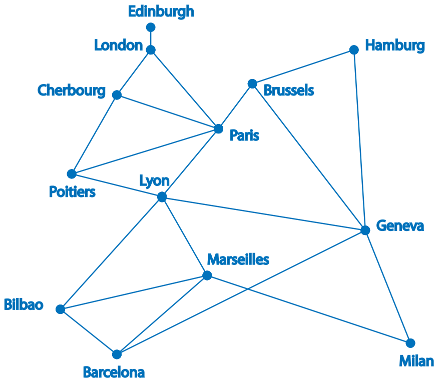




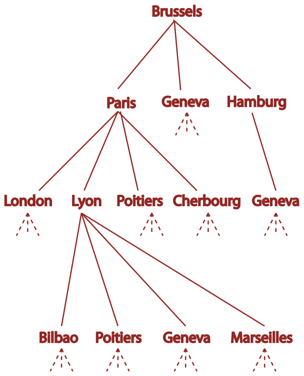
CONNECTIONS

BFS TREE

Step 5

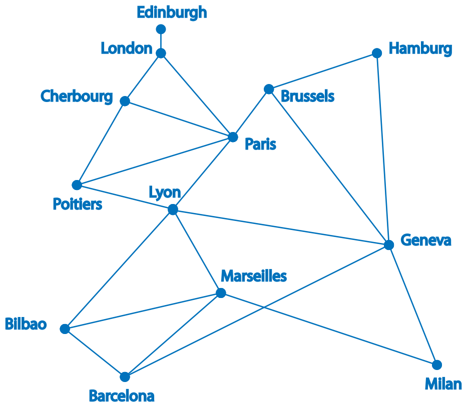


CONNECTIONS



BFS TREE

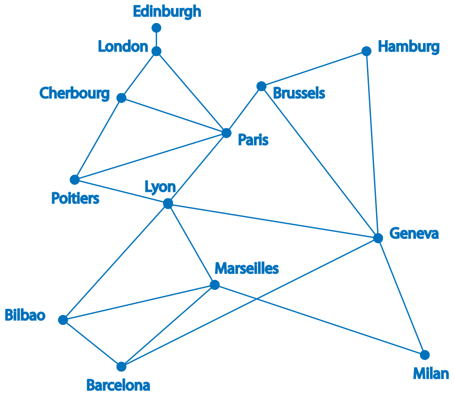
Depth-first search from ‘Brussels’: step 1

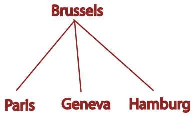


CONNECTIONS

Brussels

Step 2

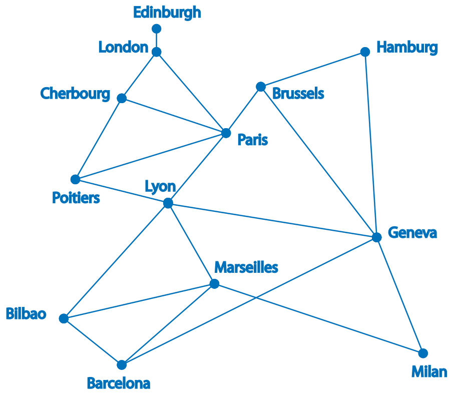




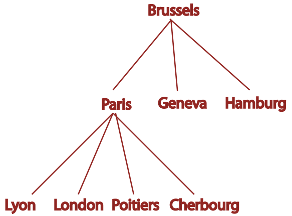
CONNECTIONS

DFS TREE

Step 3

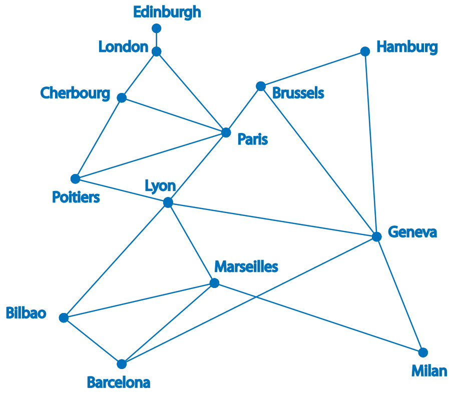


CONNECTIONS

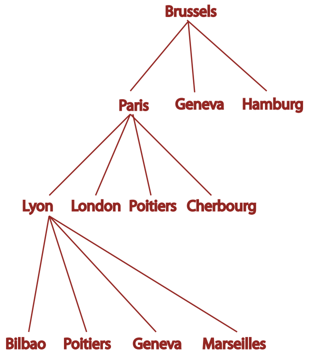


DFS TREE

Step 4

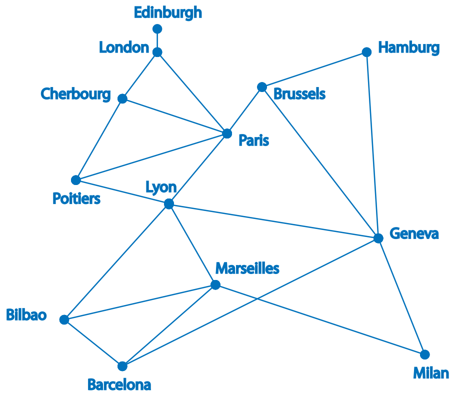


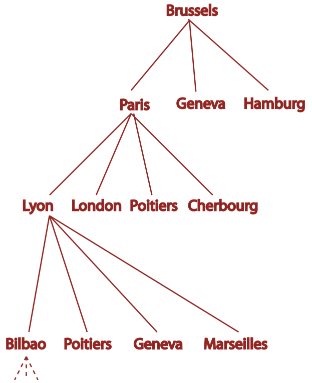
CONNECTIONS



DFS TREE

Step 5





CONNECTIONS

DFS TREE

Node expansion

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.
- The new nodes are the **children** or **successors** of the original or **parent** node.

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.
- The new nodes are the **children** or **successors** of the original or **parent** node.
- To continue the search we expand each of the children in turn, creating more nodes at the next level down.

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.
- The new nodes are the **children** or **successors** of the original or **parent** node.
- To continue the search we expand each of the children in turn, creating more nodes at the next level down.
- Nodes which cannot be expanded are **terminal nodes** or **tip nodes**

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.
- The new nodes are the **children** or **successors** of the original or **parent** node.
- To continue the search we expand each of the children in turn, creating more nodes at the next level down.
- Nodes which cannot be expanded are **terminal nodes** or **tip nodes**
- The search tree may also be called the **search space**.

Node expansion

- To generate the nodes in a particular tree, we first generate a node to represent the starting point. This is the **start node**.
- We then work out the possible transitions from the start node, creating one node for each point that can be reached. This is **expanding** the start node.
- The new nodes are the **children** or **successors** of the original or **parent** node.
- To continue the search we expand each of the children in turn, creating more nodes at the next level down.
- Nodes which cannot be expanded are **terminal nodes** or **tip nodes**
- The search tree may also be called the **search space**.

Search completion

The process can end once it has achieved the desired result.

When a node is about to be expanded, a check should be made to see if it is the node we're searching for, i.e., if it's a node representing the goal location.

This is the **target** or **goal node**.

As soon as we identify the goal node, a solution to the problem can be generated by listing out the sequence of nodes connecting the start node to the goal node.

Any such sequence of nodes is a **path**.

A path connecting the start node to the goal node is a **solution path**.

How much work is involved?

The number of nodes in a search tree *multiplies* with each new level.

Even simple problems can create search trees which are *extremely* large.

If we don't want to waste a lot of time using trial-and-error, we need a way of estimating how much work is going to be involved in a particular search.

How much work is involved?

The number of nodes in a search tree *multiplies* with each new level.

Even simple problems can create search trees which are *extremely* large.

If we don't want to waste a lot of time using trial-and-error, we need a way of estimating how much work is going to be involved in a particular search.

- We want to know how much *time* it's going to take. This is known as the **time complexity** of the process.

How much work is involved?

The number of nodes in a search tree *multiplies* with each new level.

Even simple problems can create search trees which are *extremely* large.

If we don't want to waste a lot of time using trial-and-error, we need a way of estimating how much work is going to be involved in a particular search.

- We want to know how much *time* it's going to take. This is known as the **time complexity** of the process.
- We want to know how much *memory* it's going to need. This is known as the **space complexity** of the process.

How much work is involved?

The number of nodes in a search tree *multiplies* with each new level.

Even simple problems can create search trees which are *extremely* large.

If we don't want to waste a lot of time using trial-and-error, we need a way of estimating how much work is going to be involved in a particular search.

- We want to know how much *time* it's going to take. This is known as the **time complexity** of the process.
- We want to know how much *memory* it's going to need. This is known as the **space complexity** of the process.

Branching factor

Time and space complexity are both proportional to the number of nodes in the tree (although as we'll see, space complexity is also strongly affected by the strategy used).

To estimate this, we need to calculate the **branching factor**, which is just the average number of **children** of each node.

Next we calculate the **depth** of the tree, i.e., the expected number of levels.

To estimate the total number of nodes at a particular level, we then raise the branching factor to the relevant degree, i.e., we calculate

$$b^d$$

where b is the branching factor and d is the depth. This gives the number of nodes at depth d.

Example

Say the branching factor is 3.

The number of states to be processed at level 1 is then 3.

The number to be processed at level 2 is 3 x 3, or

$$3^2$$

The number to be processed at level 3 is 3 x 3 x 3, or

$$3^3$$

And so on.

Time and space complexity

The number of nodes at the deepest level of search is a lower bound on the total number of nodes.

For many purposes, deriving this value is sufficient to decide whether or not search is a practical option.

If the expected depth is 8 and the branching factor is 5, a lower bound on the total number of nodes in the space is

$$5^8$$

To estimate time complexity, we would multiply this by the time it takes to check out a single node.

To estimate space complexity, we would multiply this by the amount of memory it takes to represent a single node.

Again these values would in fact be lower bounds.

Summary

Summary

- The concept of search

Summary

- The concept of search
- Route finding

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation
- Node expansion

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation
- Node expansion
- Search strategies (DFS v BFS)

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation
- Node expansion
- Search strategies (DFS v BFS)
- Branching factor

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation
- Node expansion
- Search strategies (DFS v BFS)
- Branching factor
- Time and space complexity

Summary

- The concept of search
- Route finding
- Tree-structure of possible action sequences
- Search by generation
- Node expansion
- Search strategies (DFS v BFS)
- Branching factor
- Time and space complexity

Questions

Questions

- Why do route-finding problems produce tree-structured searches?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?
- What additional information is required in order to be able to identify the space complexity of a search, other than the branching factor and depth of the tree?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?
- What additional information is required in order to be able to identify the space complexity of a search, other than the branching factor and depth of the tree?
- In what circumstances will the expansion of a search-tree node produce no children?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?
- What additional information is required in order to be able to identify the space complexity of a search, other than the branching factor and depth of the tree?
- In what circumstances will the expansion of a search-tree node produce no children?
- AI is just one of several approaches which seek to replicate intelligent behaviour. What is distinctive about it?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?
- What additional information is required in order to be able to identify the space complexity of a search, other than the branching factor and depth of the tree?
- In what circumstances will the expansion of a search-tree node produce no children?
- AI is just one of several approaches which seek to replicate intelligent behaviour. What is distinctive about it?
- Why does AI tend to rely on search-based methods?

Questions

- Why do route-finding problems produce tree-structured searches?
- A particular search tree has a branching factor of 2 and a depth of 4. What is the total number of nodes?
- What additional information is required in order to be able to identify the space complexity of a search, other than the branching factor and depth of the tree?
- In what circumstances will the expansion of a search-tree node produce no children?
- AI is just one of several approaches which seek to replicate intelligent behaviour. What is distinctive about it?
- Why does AI tend to rely on search-based methods?

More questions

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?
- What information is needed (i.e., must be saved by the search process) in order to be able to list a solution path?

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?
- What information is needed (i.e., must be saved by the search process) in order to be able to list a solution path?
- What degree of space complexity implies that a search is intractible?

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?
- What information is needed (i.e., must be saved by the search process) in order to be able to list a solution path?
- What degree of space complexity implies that a search is intractible?
- What degree of time complexity implies that a search is intractible?

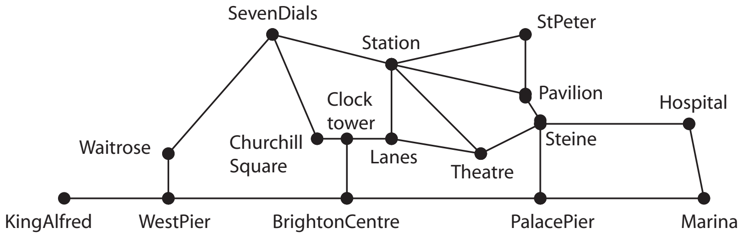
More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?
- What information is needed (i.e., must be saved by the search process) in order to be able to list a solution path?
- What degree of space complexity implies that a search is intractible?
- What degree of time complexity implies that a search is intractible?
- Define what these terms mean in the context of search: node, start node, goal node, children, depth, branching factor, path, solution path.

More questions

- Depth-first search and breadth-first search are the two basic search methods. Which one will solve a route-finding problem quickest?
- What are the main components of a search tree?
- What information is needed (i.e., must be saved by the search process) in order to be able to list a solution path?
- What degree of space complexity implies that a search is intractible?
- What degree of time complexity implies that a search is intractible?
- Define what these terms mean in the context of search: node, start node, goal node, children, depth, branching factor, path, solution path.

Exercises



This schematic map of Brighton shows bus routes between a number of locations. A valid bus route is simply a connected sequence of locations.

Exercises cont.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from ‘Clocktower’.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from ‘Clocktower’.
- Add numeric labels (1, 2, 3...) to the nodes in your search tree to indicate the order in which they would be expanded in a valid depth-first search.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from ‘Clocktower’.
- Add numeric labels (1, 2, 3...) to the nodes in your search tree to indicate the order in which they would be expanded in a valid depth-first search.
- Add alphabetical labels (a, b, c...) to the nodes in your tree to indicate the order in which they would be expanded in a valid breadth-first search.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from ‘Clocktower’.
- Add numeric labels (1, 2, 3...) to the nodes in your search tree to indicate the order in which they would be expanded in a valid depth-first search.
- Add alphabetical labels (a, b, c...) to the nodes in your tree to indicate the order in which they would be expanded in a valid breadth-first search.
- Identify the shortest route (i.e., the route with the smallest number of legs) connecting WestPier with Steine.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from ‘Clocktower’.
- Add numeric labels (1, 2, 3...) to the nodes in your search tree to indicate the order in which they would be expanded in a valid depth-first search.
- Add alphabetical labels (a, b, c...) to the nodes in your tree to indicate the order in which they would be expanded in a valid breadth-first search.
- Identify the shortest route (i.e., the route with the smallest number of legs) connecting WestPier with Steine.
- In the worst case, how many nodes would a depth-first search process expand in order to identify the shortest route between Clocktower and StPeter.

Exercises cont.

- Using the map, draw out the first four levels in the search tree for bus routes starting from 'Clocktower'.
- Add numeric labels (1, 2, 3...) to the nodes in your search tree to indicate the order in which they would be expanded in a valid depth-first search.
- Add alphabetical labels (a, b, c...) to the nodes in your tree to indicate the order in which they would be expanded in a valid breadth-first search.
- Identify the shortest route (i.e., the route with the smallest number of legs) connecting WestPier with Steine.
- In the worst case, how many nodes would a depth-first search process expand in order to identify the shortest route between Clocktower and StPeter.

Exercises cont.

Exercises cont.

- Estimate the branching factor for the search tree for this map, stating any assumptions made.

Exercises cont.

- Estimate the branching factor for the search tree for this map, stating any assumptions made.
- Using your estimate of branching factor, estimate the space complexity of a breadth-first search carried out to a depth of five levels.

Exercises cont.

- Estimate the branching factor for the search tree for this map, stating any assumptions made.
- Using your estimate of branching factor, estimate the space complexity of a breadth-first search carried out to a depth of five levels.
- Using your estimate of branching factor, estimate the space complexity of an exhaustive, loop-avoiding depth-first search of this space, stating any assumptions made.

Exercises cont.

- Estimate the branching factor for the search tree for this map, stating any assumptions made.
- Using your estimate of branching factor, estimate the space complexity of a breadth-first search carried out to a depth of five levels.
- Using your estimate of branching factor, estimate the space complexity of an exhaustive, loop-avoiding depth-first search of this space, stating any assumptions made.

Resources

Resources

- Russell and Norvig have a good section on basic search and time and space complexity; see pp. 74-75

Resources

- Russell and Norvig have a good section on basic search and time and space complexity; see pp. 74-75