# Artificial Intelligence (AI)

CCS-3880 – 3rd Semester 2023

**CO2**: Problem Spaces and Search

Dr. Abdullah Alshanqiti

# Problem Spaces and Search

**CO2. Investigate** the problem state spaces and uninformed search algorithms.

- Agent and Rational Agent Definitions (*Recall*)
- Problem Solving and Representation
- Problem as a State Space Search
  - Problem Types
  - States
  - State Space
  - State Modification
  - Problem Solution
  - Formal Description of the Problem
  - Examples: 8 Puzzle & Traveling Salesman Problems

- State Space Search – *the basic idea*
- Search Tree
- State Space vs. Search Trees

# Agent

**Intelligent Agent:**

An IA refers to an autonomous entity which acts, directing its activity towards achieving goals. It is anything that can be viewed as perceiving its environment through e.g. sensors and acting upon that environment through actuators …

**Rational Agent:**

For each possible percept sequence, a rational agent should select an action that is *expected* to maximize its performance measure, based on the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

# Agent

## Performance Measure:

An objective criterion for success of an agent's behavior

For example, *performance measure* of a vacuum-cleaner agent could be amount of

- dirt cleaned up,
- time taken,
- electricity consumed,
- noise generated, etc.

## Agents Task:

Before we design an intelligent agent, we must specify its tasks w.r.t :

Performance measure, Environment, Actuators, Sensors ...

## Example: Part-picking Robot:

- Performance measure: Percentage of parts in correct bins
- Environment: vacuum parts e.g. bins
- Actuators: Jointed arm and hand
- Sensors: Camera, joint angle sensors

# Problem Solving

## We want:

to automatically solve a problem

## We need:

- A representation of the problem
- Algorithms that use some strategy to solve the problem defined in that representation

# Problem Representation

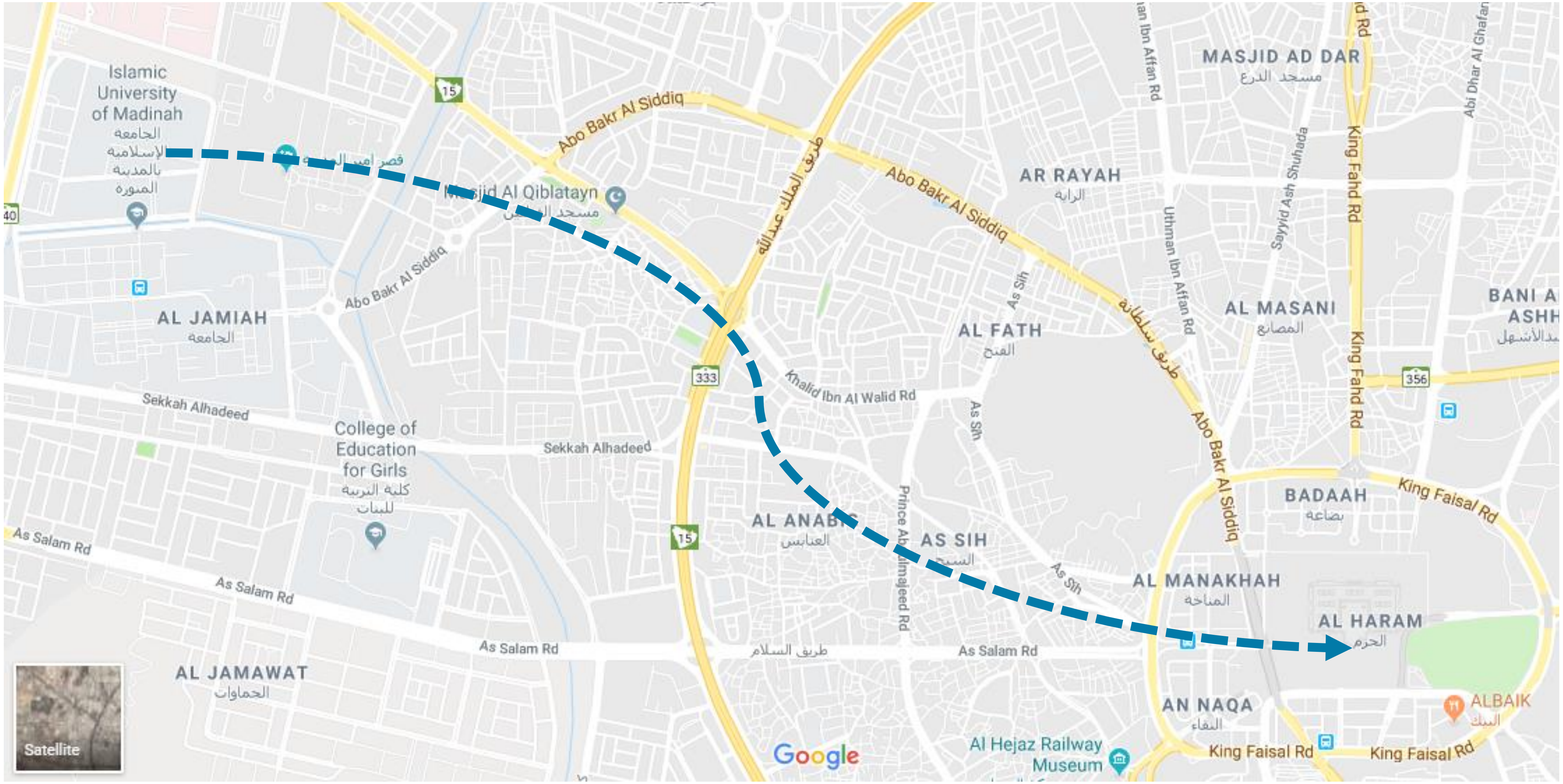
## General:

- **State space:** a problem is divided into a set of resolution steps from the initial state to the goal state
- **Reduction to sub-problems**: a problem is arranged into a hierarchy of sub-problems

## Specific e.g. :

- Game resolution
- Constraint's satisfaction
- and more …

# Example: going to Al-haram from Islamic University …



# Example: going to Al-haram from Islamic University …

**Formulate goal:**

be in Al-haram

**Formulate problem:**

states: various locations

actions: drive/move between locations

**Find solution:**

sequence of locations, e.g., *Islamic uni, Prince Naif Road, Al-salam Road, … , Al-haram*

# Problem as a State Space Search

In AI, a **problem** is normally defined by the following components:

1. **State space**: it can be explicitly or implicitly defined
2. **Initial state** e.g., "at Islamic University“
3. **Actions**: Actions(X) = set of actions available in State X
4. **Transition model**: Result(S,A) = state resulting from doing action A in state S
5. **Goal state**: e.g., x = "at Al-haram”, Checkmate(x)
6. **Path cost**: (additive, i.e., the sum of the step costs)

   c(x,a,y) = step cost of action a in state x to reach state y
   assumed to be ≥ 0, and it can be restricted.

A **solution** is a sequence of actions leading from the initial state to a goal state

# Problem Types

Deterministic, fully observable → single-state problem

Agent knows exactly which state it will be in; solution is a sequence

Non-observable → conformant problem

Agent may have no idea where it is; solution is a sequence

Nondeterministic and/or partially observable → unpredictable problem

- percepts provide new information about current state
- often interleave search, execution

Unknown state space → exploration problem

# States

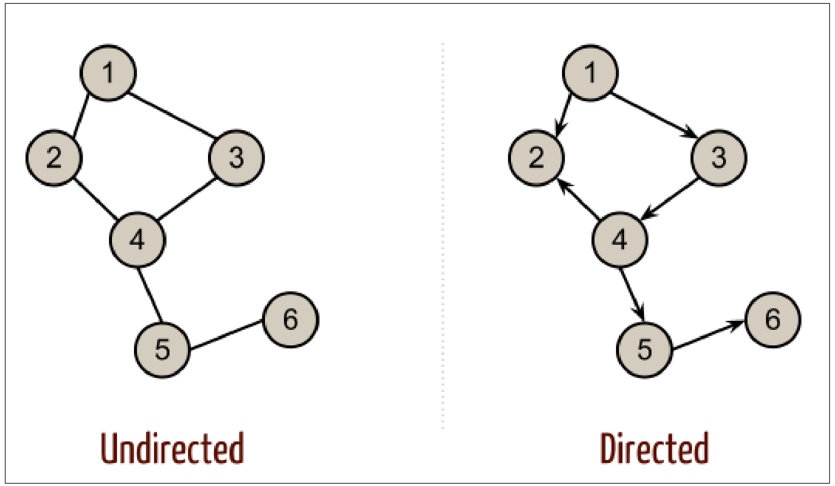
- A *problem* is defined by its elements and their relations.
- In each instant of the resolution of a problem, those elements have specific descriptors (*How to select them*?) and relations.
- A **state** is a representation of those elements in a given moment.
- Two special states are defined:
  - **Initial state** (starting point)
  - **Final state** (goal state)

# State Space

- The **state space** is the set of all states reachable from the initial state. It forms a graph (or map) in which the *nodes* are *states* and the *arcs* between nodes are *actions*.

- A **path** in the state space is a sequence of states connected by a sequence of actions.

  The solution of the problem is part of the map formed by the state space.



# State Modification (successor function):

## A successor function is:

- needed to move between different states.
- a description of possible actions, a set of operators.
- a transformation function on a state representation, which convert it into another state.
- defines a relation of accessibility among states.

## Representation of the successor function:

- Conditions of applicability
- Transformation function

# Problem Solution

**A solution in the state space is:**

a path from the initial state to a goal state or, sometimes, just a goal state.

**Path/solution cost:**

function that assigns a numeric cost to each path, the cost of applying the operators to the states

**Solution Quality:**

is measured by the path cost function, and
an *optimal solution* has the lowest path cost among all solutions.

**Solutions:**

can be *any*, *an optimal one*, *all*.
cost is important depending on the problem and the type of solution sought (attempt to find).

# Formal description of the problem

Let *S* be a set of states (state space)

A search problem consists of *initial state*, *transitions* between states, and a *goal state* (or many goal states)

**Search Problem:**

*problem* = (s0, succ, goal)

*s0* $\in$ *S* is the initial state
succ : *S* → f *(S)* is a successor function defining the state transitions
goal : *S* → {T, ⊥} is a test predicate to check if a state is a goal state

The *successor function* can be defined either explicitly (as a map from input to output states) or implicitly (using a set of operators that act on a state and transform it into a new state)

# Another example: 8-puzzle

What sequence of actions leads to the goal state?

initial state:

| | | |
|---|---|---|
| 8 | ■ | 7 |
| 6 | 5 | 4 |
| 3 | 2 | 1 |

goal state:

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | ■ |

$$problem = (s_0, \mathrm{succ}, \mathrm{goal})$$

$$s_0 = \begin{array}{|c|c|c|}\hline 8 & \blacksquare & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}$$

$$\mathrm{succ}\left(\begin{array}{|c|c|c|}\hline 8 & \blacksquare & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = \left\{\begin{array}{|c|c|c|}\hline \blacksquare & 8 & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}, \begin{array}{|c|c|c|}\hline 8 & 7 & \blacksquare \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}, \begin{array}{|c|c|c|}\hline 8 & 5 & 7 \\ \hline 6 & \blacksquare & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right\}$$

$$\vdots$$

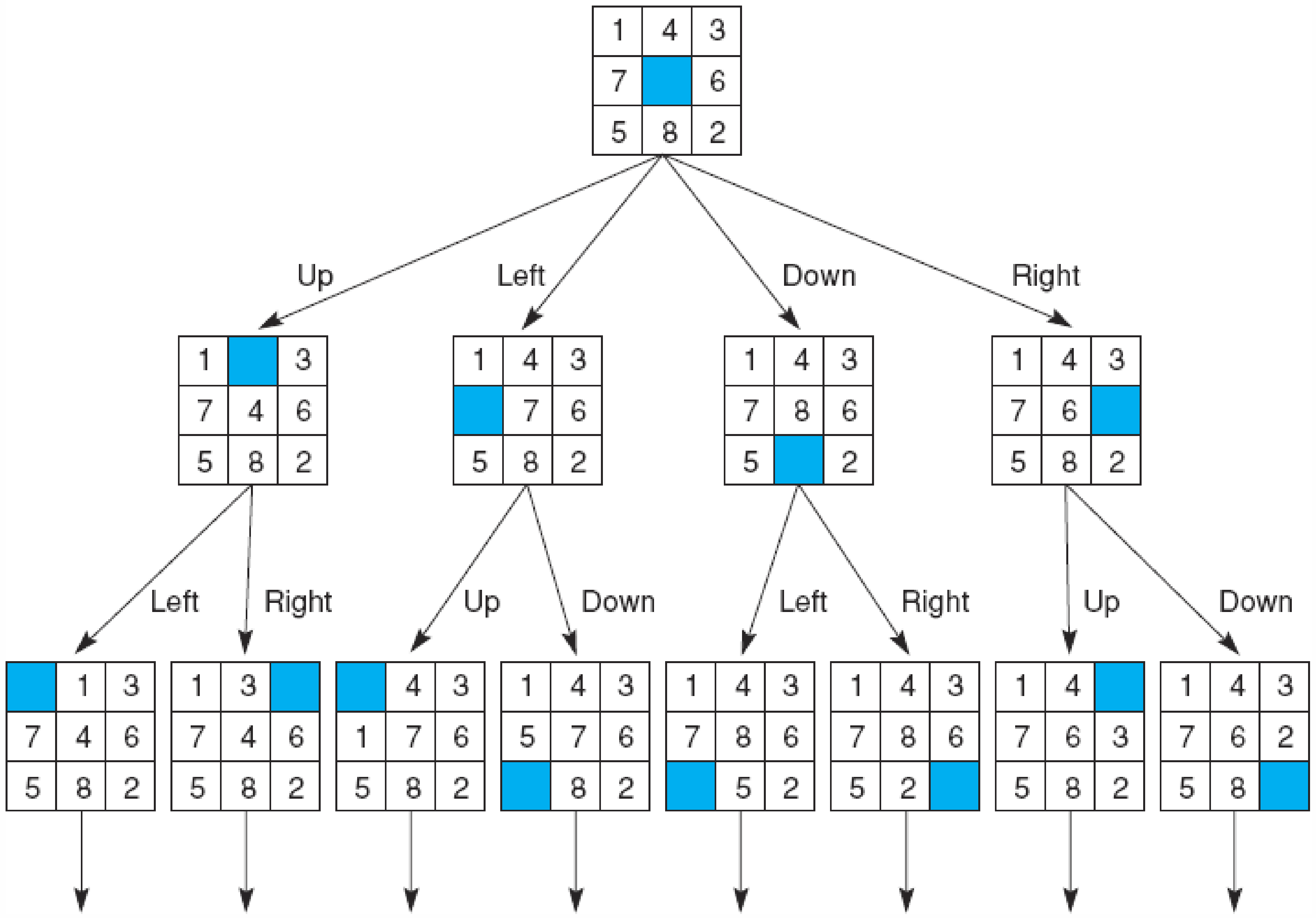
$$\mathrm{goal}\left(\begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline 4 & 5 & 6 \\ \hline 7 & 8 & \blacksquare \\ \hline\end{array}\right) = \top$$

$$\mathrm{goal}\left(\begin{array}{|c|c|c|}\hline 8 & \blacksquare & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = \bot$$

$$\mathrm{goal}\left(\begin{array}{|c|c|c|}\hline \blacksquare & 8 & 7 \\ \hline 6 & 5 & 4 \\ \hline 3 & 2 & 1 \\ \hline\end{array}\right) = \bot$$

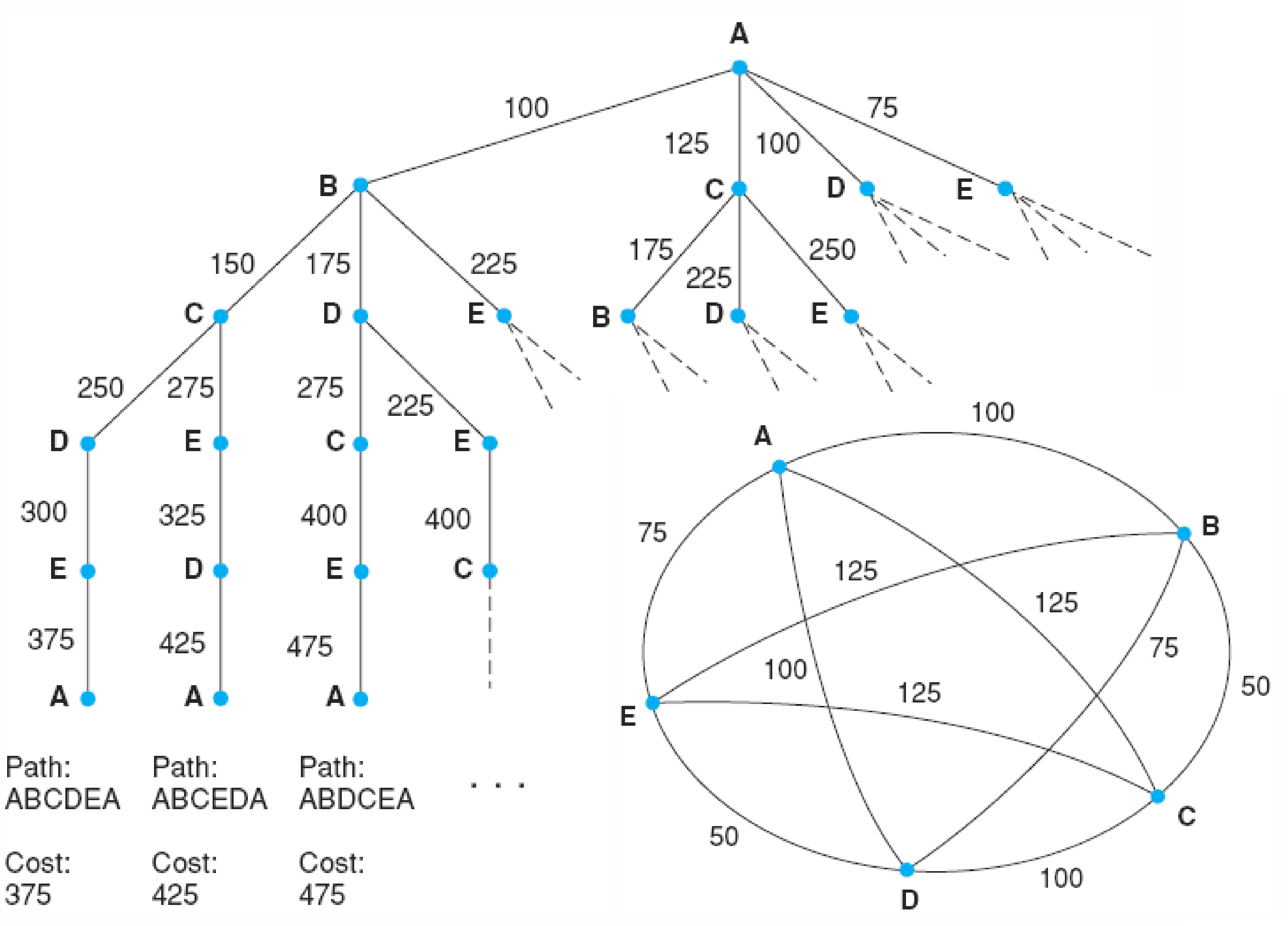
$$\vdots$$

# 8 Puzzle




The 8-puzzle search space consists of 8! states

# Traveling Salesman Problem



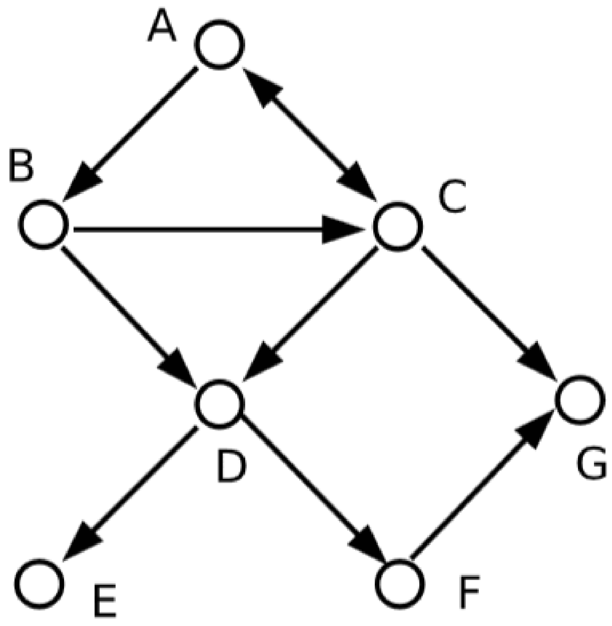
# Search Tree

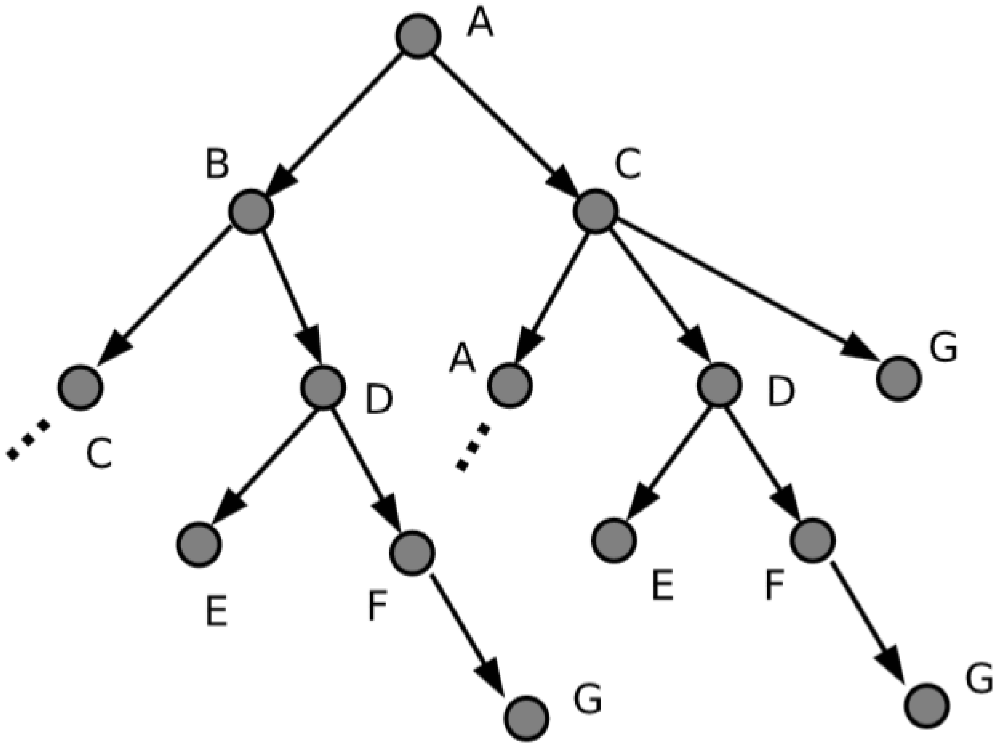
- By searching through a directed graph, we gradually construct a search tree!
- We do this by expanding one node after the other: we use the successor function to generate the descendants of each node
- **Open nodes** or "the front": nodes that have been generated, but have not yet been expanded
- **Closed nodes**: already expanded nodes

**Search Strategy :**

- The search strategy is defined by the **order** in which the nodes are expanded.
- Different orders yield different strategies

# State Space vs. Search tree




- Search tree is **created** by searching through the state space
- Search tree can be infinite even if the state space is finite.
- **Note**: state space contains cycles ⇒ search tree is infinite

# Comparing Problems and Algorithms

**Problem properties:**

|S| – number of states

b – search tree branching factor

d – depth of the optimal solution in the search tree

m – maximum depth of the search tree

**Algorithm properties:**

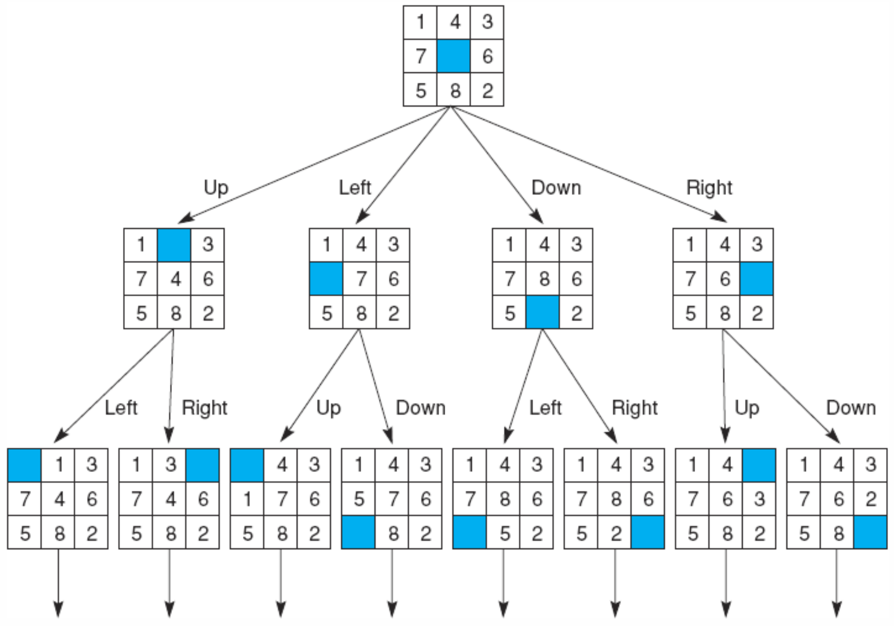
- *Completeness* – an algorithm is complete iff it finds a solution whenever the solution exists
- *Optimality* – an algorithm is optimal iff the solution it finds is optimal (*has the smallest cost*)
- *Time complexity* (number of generated nodes, how long?)
- *Space complexity* (number of stored nodes, how much memory?)

# Discussion: 8-puzzle problem properties

## Exercise:

Think about the 8-puzzle problem as a search problem. We wish to characterize the difficulty of this problem. We do this by considering the problem properties:

- Try to figure out the number of states |S|
- What is the minimal and the maximal branching factor?
- Calculate the average branching factor.
- Write down your answers



# Search Strategies

There are two types of strategies:

- Blind (uninformed) search
- Heuristic (directed, informed) search

# Search Strategies

- **Blind search** → traversing the search space until the goal nodes is found (might be doing exhaustive search).

- **Techniques** :
  - Breadth-first search (BFS)
  - Uniform cost search
  - Depth-first search (DFS)
  - Depth-limited search
  - Iterative deepening search

- **Heuristic search** → search process takes place by traversing search space with applied rules (information).

- **Techniques:**
  - Greedy Best First Search,
  - A* Algorithm
  - Local Search Algorithms
    - Hill-climbing search
    - Gradient Descent
    - Simulated annealing (suited for either local or global search)
  - Global Search Algorithms
    - Genetic Algorithm

Blind and heuristic search will be covered in the coming weeks ..

# Any questions?