Artificial Intelligence: An Introduction

Wei Wang(王伟)

Engineering Research Centre of Applied Technology on Machine Translation and Artificial Intelligence , Macao Polytechnic University

weiwang@mpu.edu.mo; 匯智樓 (WUI CHI)-5/F, N56

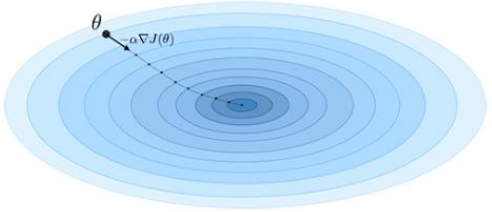
Mar. 09, 2026

AI Models Types

AI models refer to different approaches or frameworks that are used to represent and solve problems in the field of AI.

These models provide a structured way to understand and analyze complex systems and make intelligent decisions.

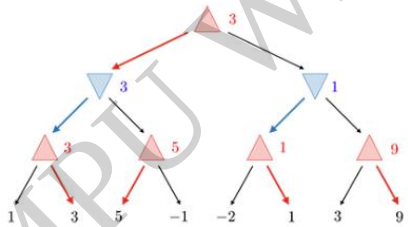
Reflex



- Feature vector, score, margin, residual
- Linear predictors, non-linear predictors
- Loss minimization, stochastic gradient descent, backpropagation, regularization

PDF Web

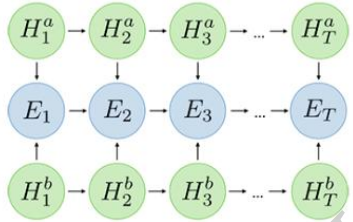
States



- Tree search, graph search, A star search
- Learning costs, relaxation
- Markov decision processes, game playing

PDF Web

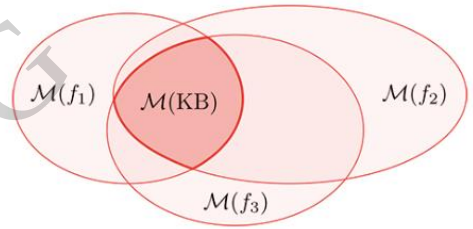
Variables



- Constraint satisfaction problem, backtracking search
- Beam search, Gibbs sampling
- Bayesian networks, probabilistic program, inference

PDF Web

Logic



- Notations, knowledge base, forward inference, rule properties
- Propositional logic, modus ponens, resolution rule, conjunctive normal form
- First-order logic, substitution, unificaion

PDF Web

Source: https://stanford.edu/~shervine/teaching/cs-221/

AI models define how an agent perceives, reasons, and acts.

Different models suit different environments and tasks.

AI Models Types

1. Logic-based models

- Symbolic representation of classes of objects.
- Deductive Reasoning.
- **Apps**: Question Answering Systems, Natural Language Understanding, Expert system
- **Options**: Propositional Logic , First-Order Logic, Knowledge Base.

2. States-based models

- Solutions are defined as a sequence of steps.
- Model a task as a graph of states and a solution as a path in the graph.
- A state captures all of the relevant information about the past in order to act in the future.
- **Apps**: Navigation and Games.
- **Options**: Tree Search (Breadth-first search, Depth-first search, and Iterative deepening), Graph search (Dynamic programming), Markov decision processes, Game playing

3. Variables-based models (Uncertainty)

- Solution in an assignment of values for a set of variables.
- **Apps**: Soduko, Speech Recognition, and Face Recognition.
- **Options**: Convolutional Neural Networks, Constraint Satisfaction, Bayesian Networks, Factor Graphs, and Dynamic Ordering.

4. Reflex-based models

- Given a set of <Input, Output> pairs of training data, learn a set of parameters that will map input to output for future data.
- **Apps**: Classification and Regression.
- **Options**: Artificial Neural Networks (ANN), Decision Trees, Support Vector Machines, Regression, Principal Component Analysis, K-Means Clustering, and K-Nearest Neighbor

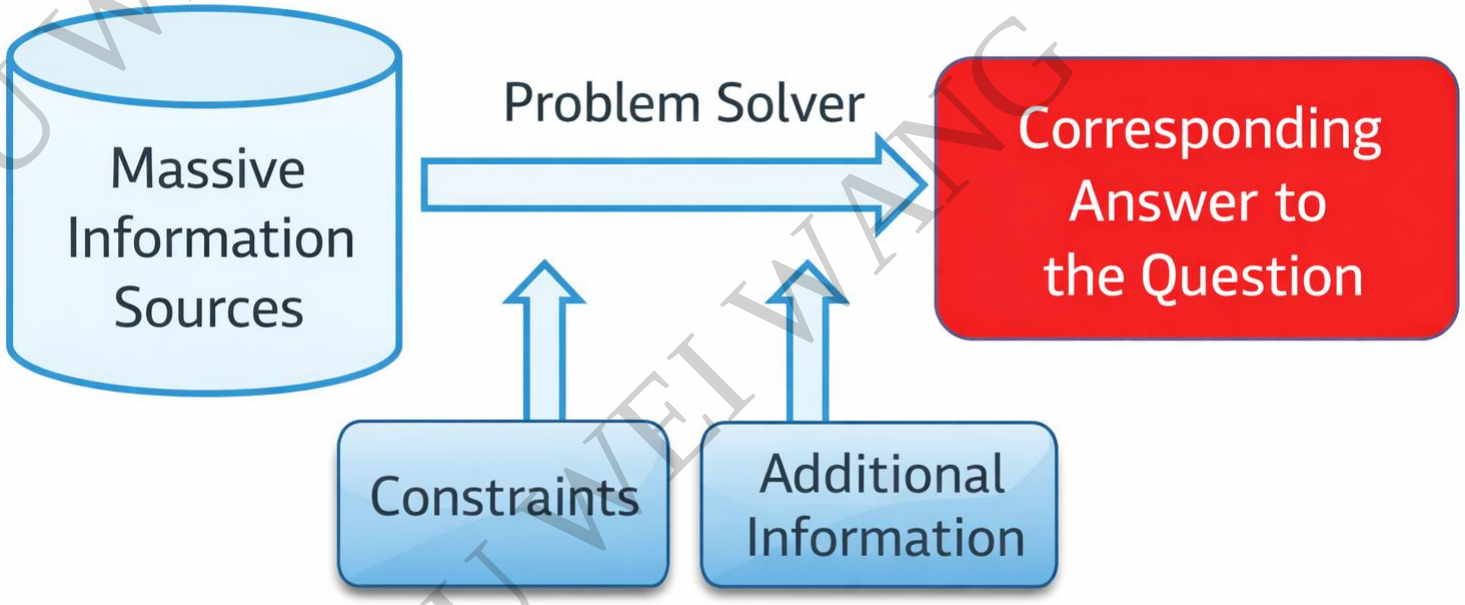
Outline

- States-based Models
- Search

Search

Whether you see me or not,
I am right there.
Neither sad nor joyful.

— Tashi Lhamo Dodo



Problem Solving

Each research area in artificial intelligence has its own characteristics and patterns, but from the perspective of problem-solving, they can all be abstracted into a single problem-solving process:

The solution process for some problems is actually a search.

Two important aspects of search problem solving: representation and search.

Representation is fundamental; search must be based on representation. Representation relates to search efficiency.

Knowledge Representation:
State space, problem reduction, predicate logic, production rules, ...

This chapter mainly discusses search-based problem solving:
States-based Models: Search

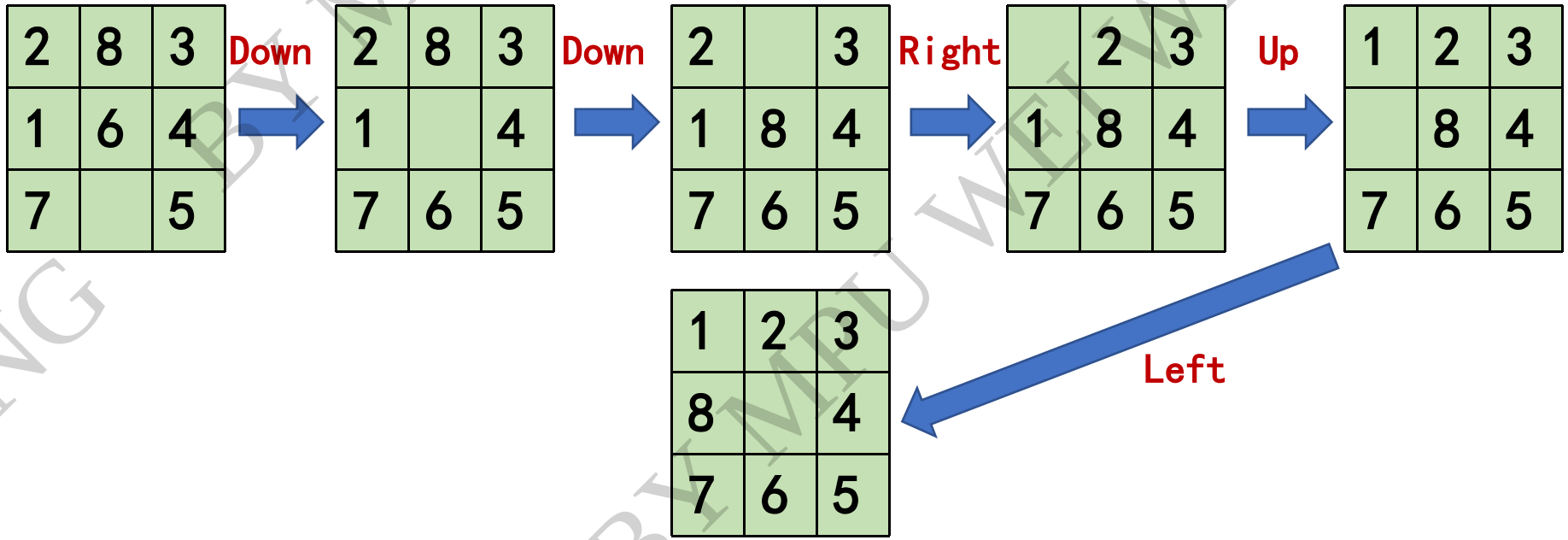
State-space method:

A problem representation and solution method based on a solution space. It uses states and operators to represent and solve problems.

(1) State: A data structure representing the state of the problem at each step in the solution process.

(2) Operator: A means of transforming a problem from one state to another.

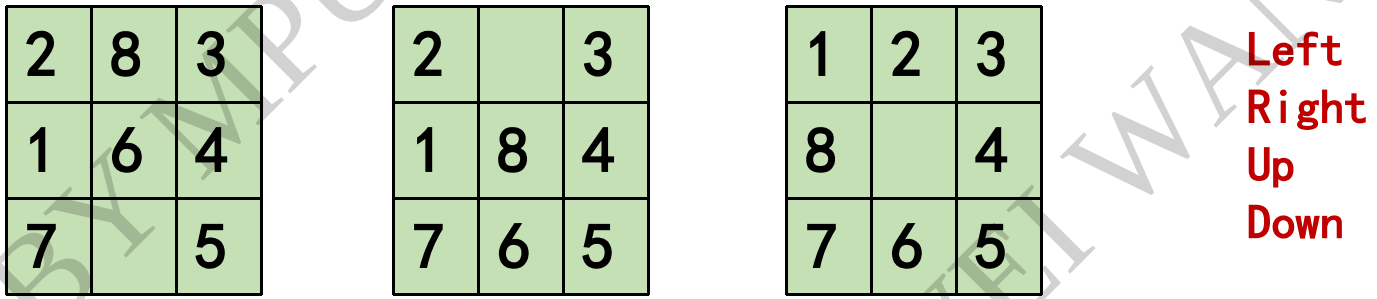
(3) State space: Represents all possible states of the problem and their relationships.



Definition of the State-Space Method:

The state-space method can be formally represented as a quadruple: **(S, O, s0, g)**

Where S is the set of states, O is the set of operators, s0 is the initial state, and g is the target state.



Down→Down →Right →Up →Left

The solution in the state space is the sequence of operators from the initial state s0 to the target state g.

Search

Search is a fundamental problem in artificial intelligence.

It is an integral part of reasoning

directly affects the performance and efficiency of intelligent systems.

Search: The process of continuously **seeking usable knowledge** based on the actual situation of the problem, constructing **a low-cost reasoning path** to achieve a satisfactory solution.

Why does artificial intelligence study search?

Problems that do not have direct solutions

require exploratory solutions.

Search Strategy Evaluation Criteria:

Completeness: If a solution exists, is the strategy guaranteed to find it?

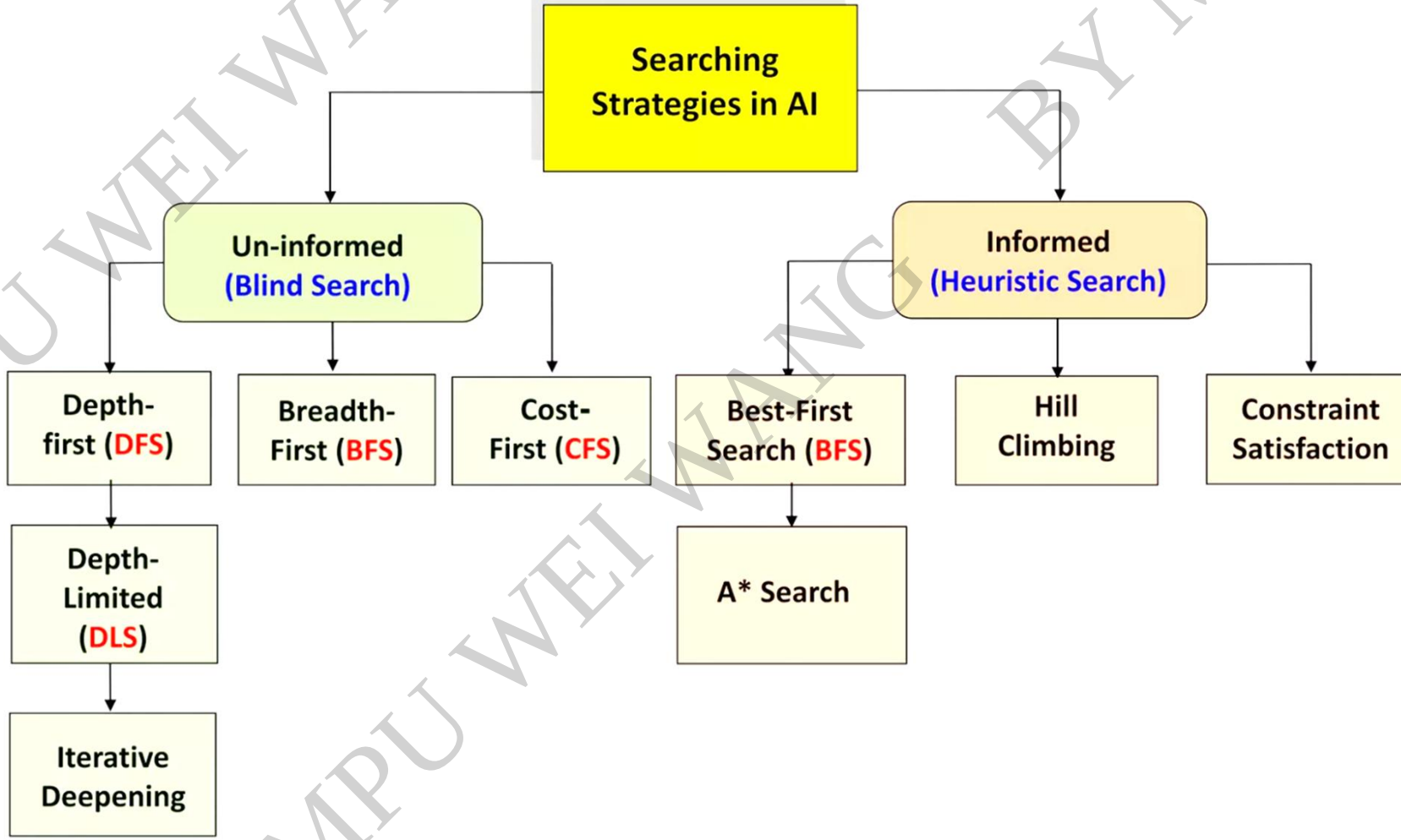
Optimality: If several distinct solutions exist, can the strategy find the highest quality solution?

Time Complexity: How long does it take to find the solution?

Space Complexity: How much storage space is required to perform the search?

Search

- Searching Overview
- Depth-First Search
- Breadth-First Search
- Uniform Cost Search
- A* Search
- Simulated Annealing



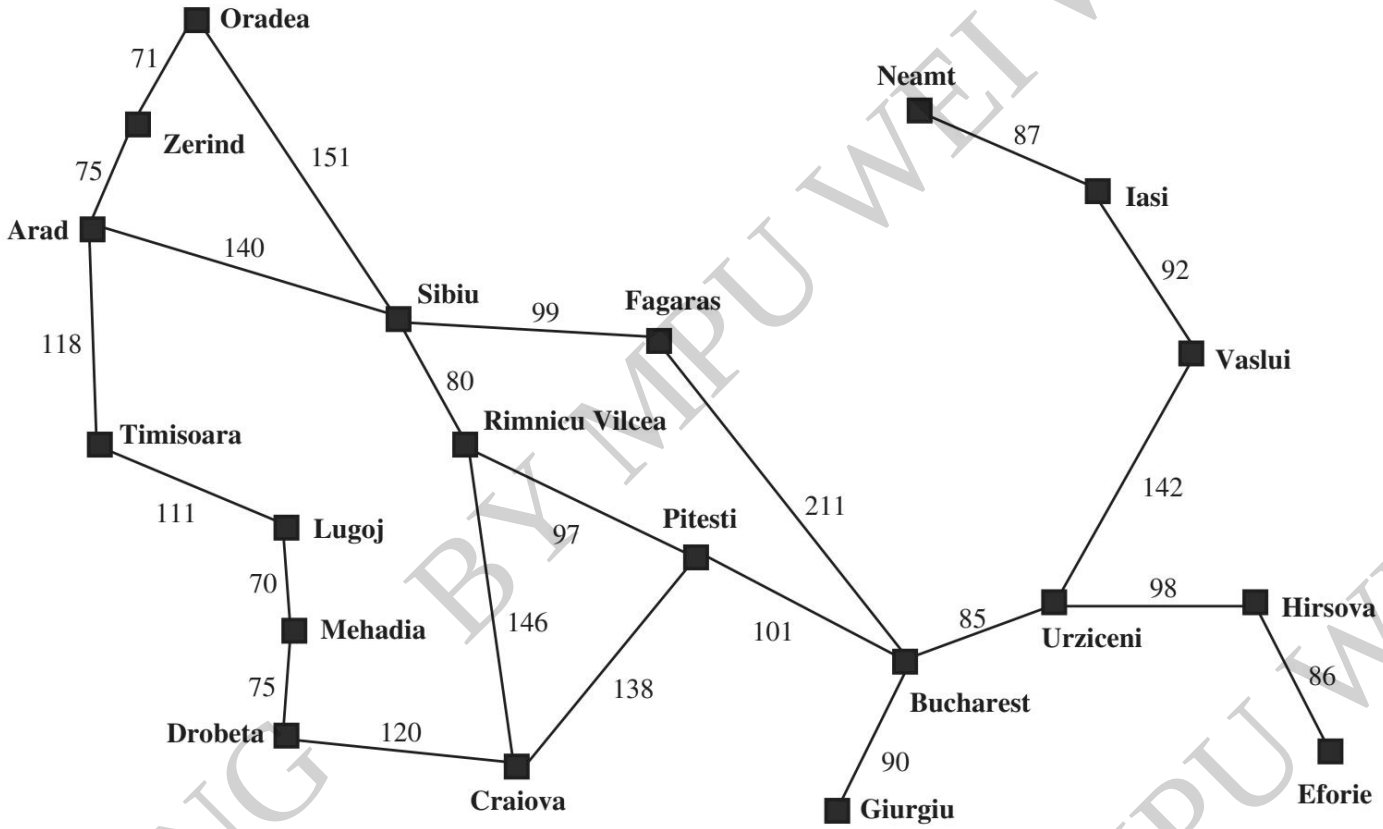
Search

- A search problem consists of:
  - A state space $\mathcal{S}$
  - An initial state s_0
  - Actions $\mathcal{A}(s)$ in each state
  - Transition model *Result*(s,a)
  - A goal test $G(s)$
    - S has no dots left
  - Action cost $c(s,a,s')$
    - +1 per step; -10 food; -500 win; +500 die; -200 eat ghost

- A solution is an action sequence that reaches a goal state
- An optimal solution has least cost among all solutions

Source: https://courses.cs.washington.edu/courses/csep573/24wi/slides/csep573wi24-Agents.pdf

Search Example: Traveling in Romania

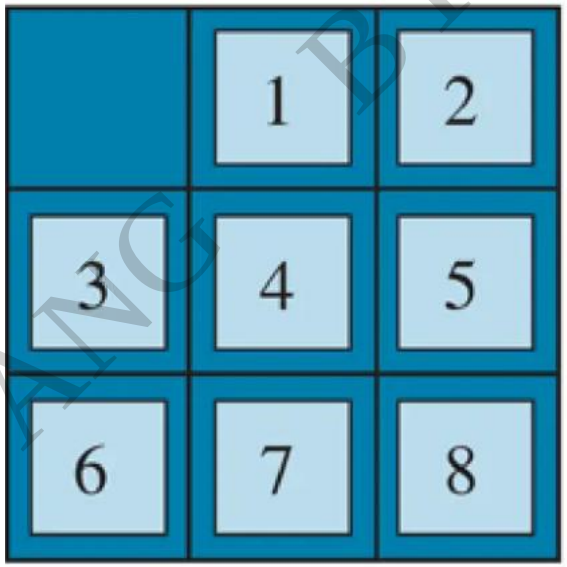


- State space:
  - Cities
- Initial state:
  - Arad
- Actions:
  - Go to adjacent city
- Transition model:
  - Reach adjacent city
- Goal test:
  - *s* = Bucharest?
- Action cost:
  - Road distance from *s* to *s'*
- Solution?

Search Example: 8-Puzzle

- States??
  - The location of each tile.
- Initial state??
  - Any state can be initial.
- Actions??
  - {Left, Right, Up, Down}.
- Goal test??
  - Check whether goal configuratio
- Path cost??
  - Number of actions to reach goal

- State space:
  - Cities
- Initial state:
  - Arad
- Actions:
  - Go to adjacent city
- Transition model:
  - Reach adjacent city
- Goal test:
  - *s* = Bucharest?
- Action cost:
  - Road distance from *s* to *s'*
- Solution?

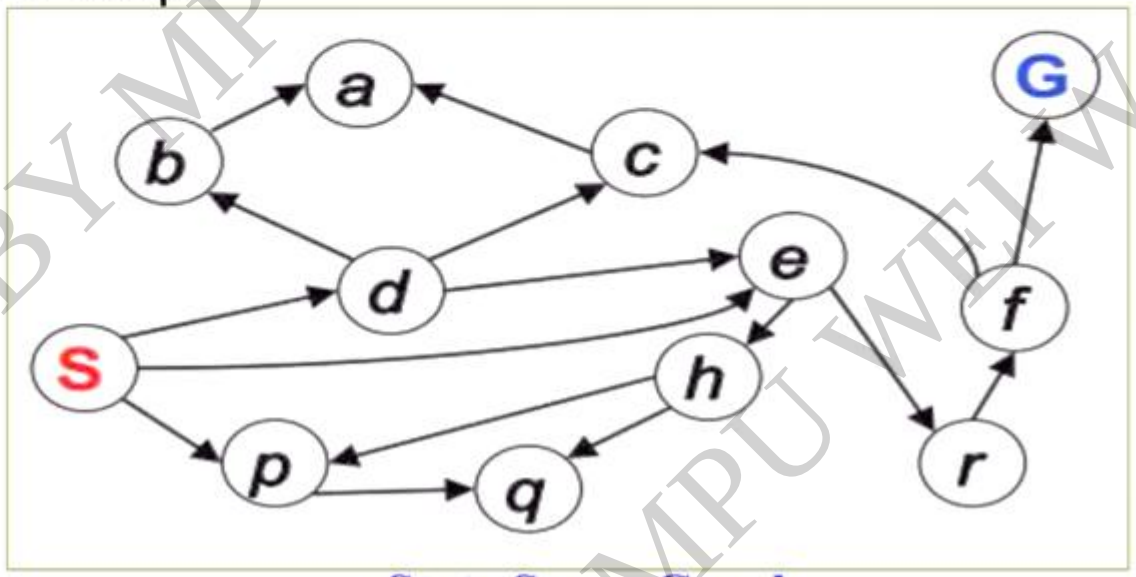


Goal State

State Space Graph versus Search Trees

We can rarely build this full graph in memory (it's too big), but it's a useful idea.

- State Space Graph
  - Graph of **states** with arrows pointing to successors.
  - State graph shows the possible states of a system.
  - A state is a node in which a system can be at any given time.
  - May contain a loop.

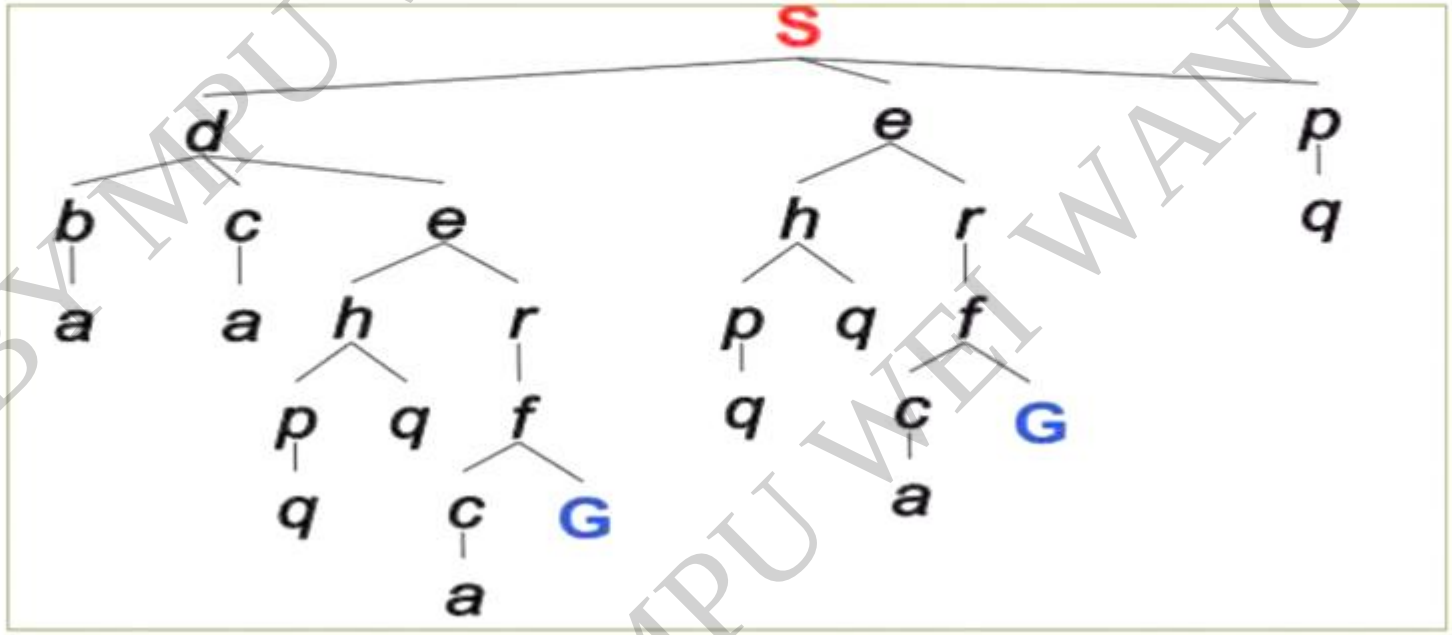


State Space Graph

Tiny state space graph for a tiny search problem

State Space Tree

- Tree is a special case of a graph.
- The **topmost node** in a tree is called the **root node**; at root node all operations on the tree begin.
- Each **NODE** in in the search tree is an entire **PATH** in the problem graph.
- Represent a **plan** (sequence of actions) which results in the node's state .

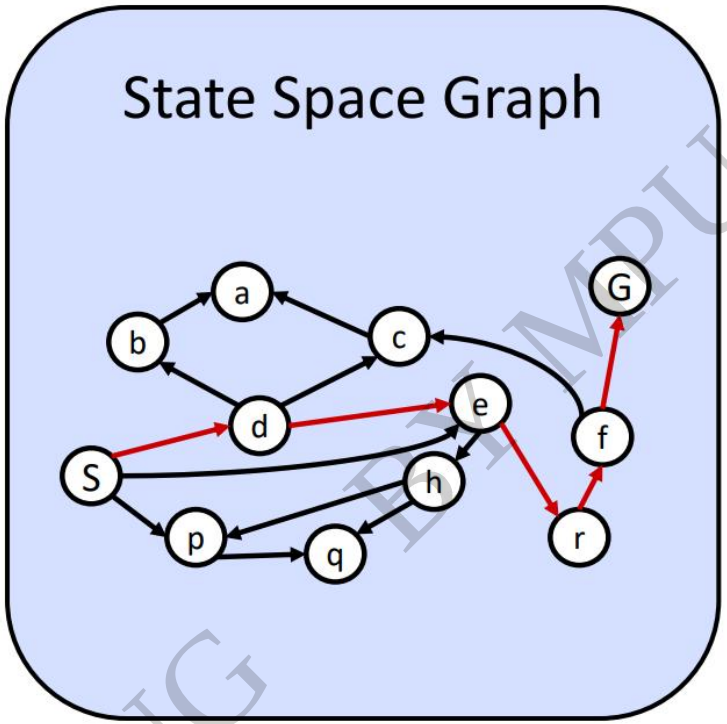


State Space Tree

Search

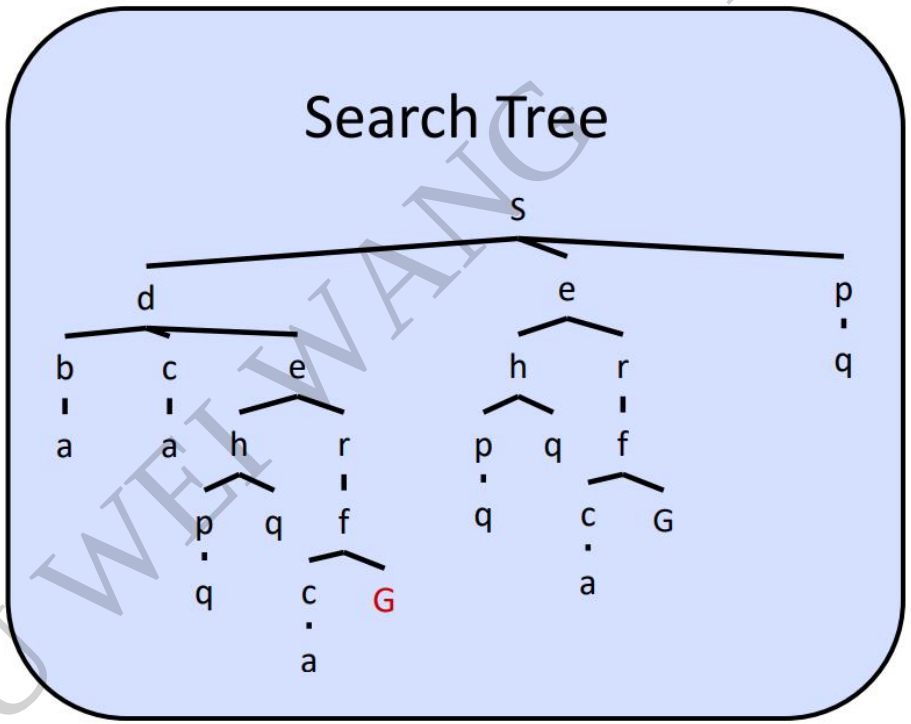
Aspect	State Space Graph	Search Tree
Definition	A graph that represents all possible states of a problem and the transitions between them.	A tree structure generated during search that shows how the algorithm explores the state space.
Nodes	Each node represents a unique state in the problem.	Each node represents a state instance in the search process (the same state may appear multiple times).
Structure	Usually represented as a graph (can contain cycles).	A tree, starting from the initial state as the root.
Repetition	Each state appears only once.	The same state may appear multiple times through different paths.
Purpose	Represents the complete problem space conceptually.	Represents the actual exploration process of a search algorithm.
Size	Typically very large and often not fully generated.	Generated incrementally during search.
Cycles	May contain loops or cycles.	Trees do not contain cycles by definition.

State Space Graph versus Search Trees



Each NODE in in the search tree is an entire PATH in the state space graph.

We construct the tree on demand – and we construct as little as possible.

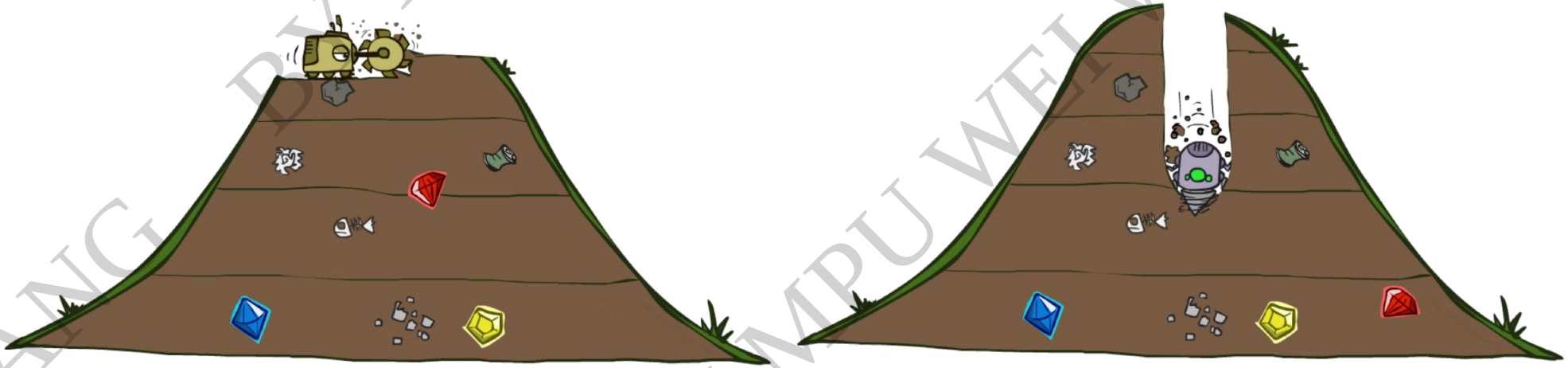


Graph search strategies are mainly divided into:

Uninformed Search and Heuristic Search

Uninformed Search: Also known as blind search, this involves searching solely according to a predetermined control strategy. Intermediate information obtained during the search process is not used to improve the control strategy.

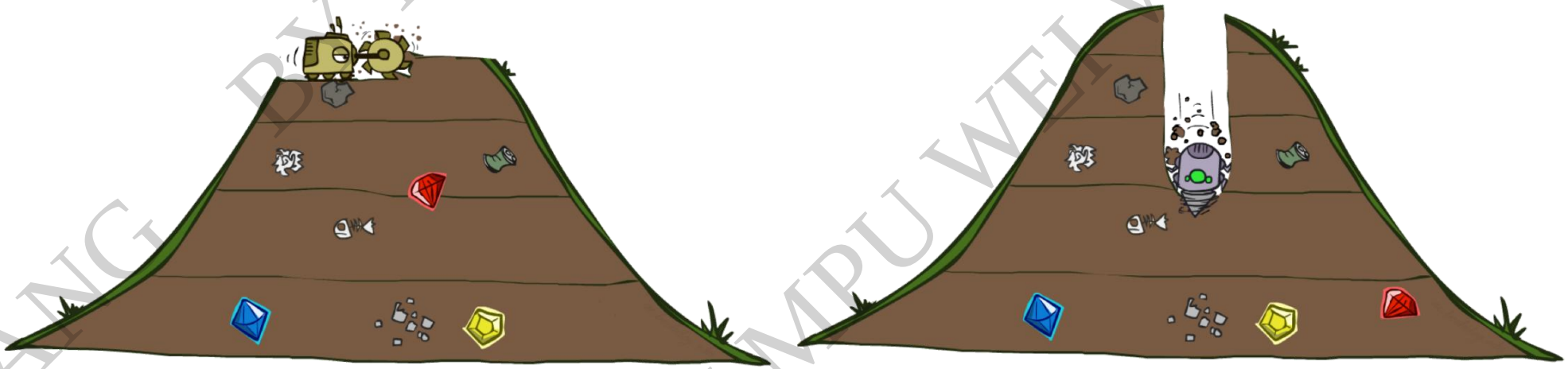
Heuristic Search: This incorporates problem-related heuristic information into the search, guiding the search in the most promising direction, accelerating the problem-solving process, and finding the optimal solution.



Graph search strategies are mainly divided into:

Uninformed Search and Heuristic Search

Uninformed Search: uninformation search typically follows a **predetermined search strategy**. Because this type of search always proceeds along a pre-defined route without considering the characteristics of the problem itself, it is highly unpredictable, inefficient, and only suitable for solving simple problems.



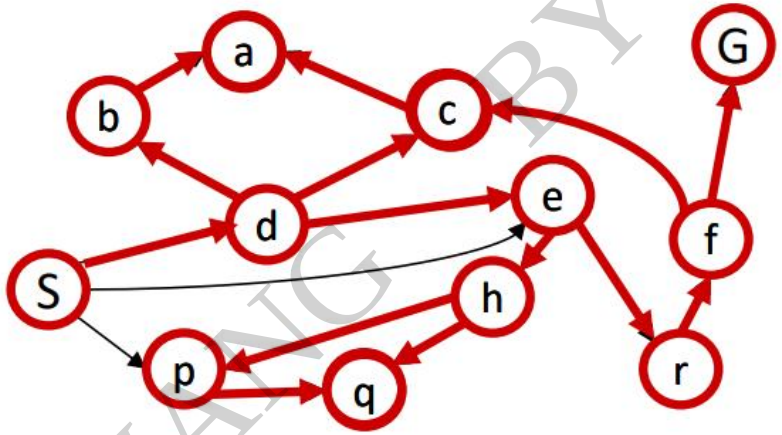
Depth-First Search

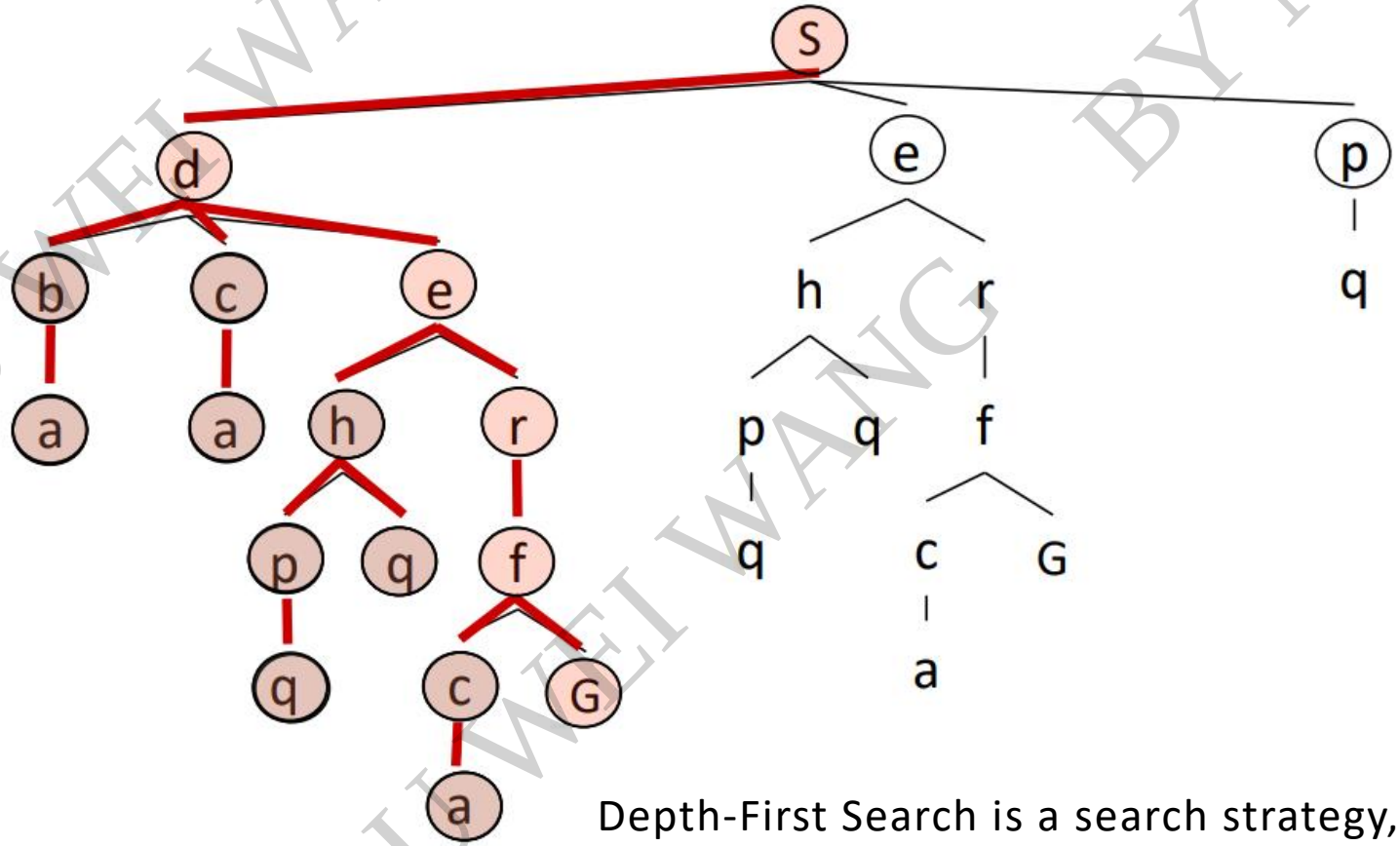
Depth-First Search (DFS)

Strategy: expand a deepest node first

Implementation:
Frontier is a LIFO stack

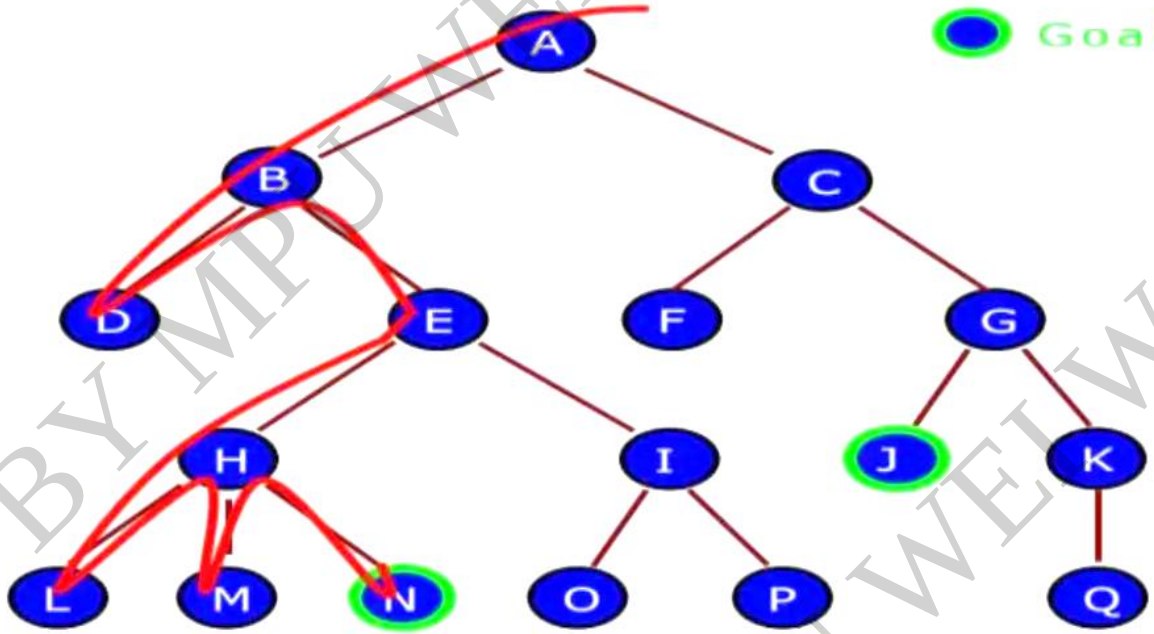
(last in first out)





Depth-First Search is a search strategy, that extends the current path as far as possible before backtracking to the last choice point and trying he next alternative path.

Depth-First Search (DFS)



Node are explored in the order :

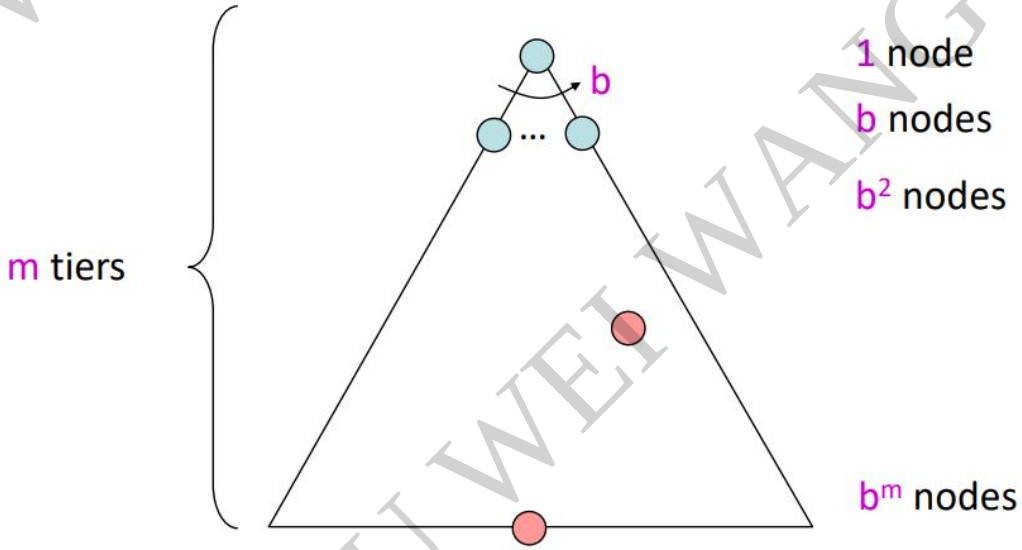
A B D E H L M N I O P C F G J K Q

- After searching node **A**, then **B**, then **D**, the search backtracks and tries another path from node **B** .
- The goal node **N** will be found before the goal node **J**.

Depth-First Search (DFS) Properties

- **Complete:** Guaranteed to find a solution if one exists?
- **Optimal:** Guaranteed to find the least cost path?
- Time complexity?
- Space complexity?

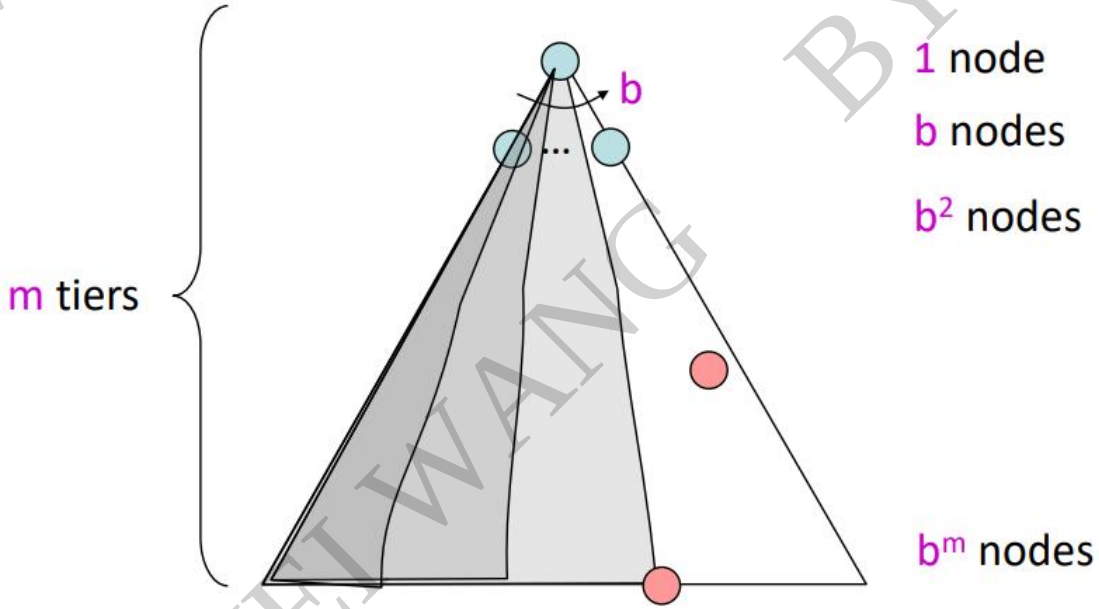
- Cartoon of search tree:
  - *b* is the branching factor
  - *m* is the maximum depth
  - solutions at various depths



- Number of nodes in entire tree?
  - $1 + b + b^2 + ... b^m = O(b^m)$

Remember O(..) is the upper bound of the function

- What nodes does DFS expand?
  - Some left prefix of the tree down to depth *m*.
  - Could process the whole tree!
  - If m is finite, takes time $O(b^m)$
- How much space does the frontier take?
  - Only has siblings on path to root, so $O(bm)$
- Is it complete?
  - *m* could be infinite
  - preventing cycles may help (more later)
- Is it optimal?
  - No, it finds the "leftmost" solution, regardless of depth or cost



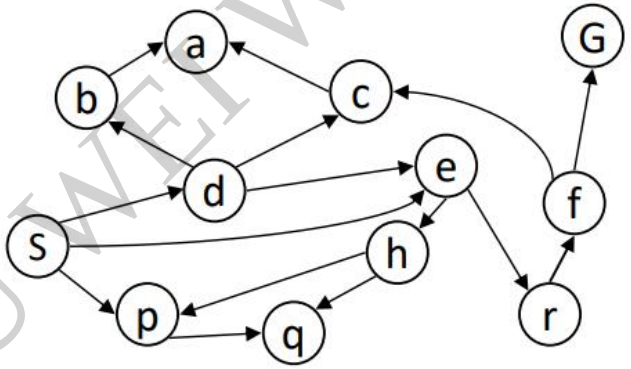
- No: fails in infinite-depth spaces, spaces with loops
  Modify to avoid repeated spaces along path
- Yes: in finite spaces

Breadth-First Search

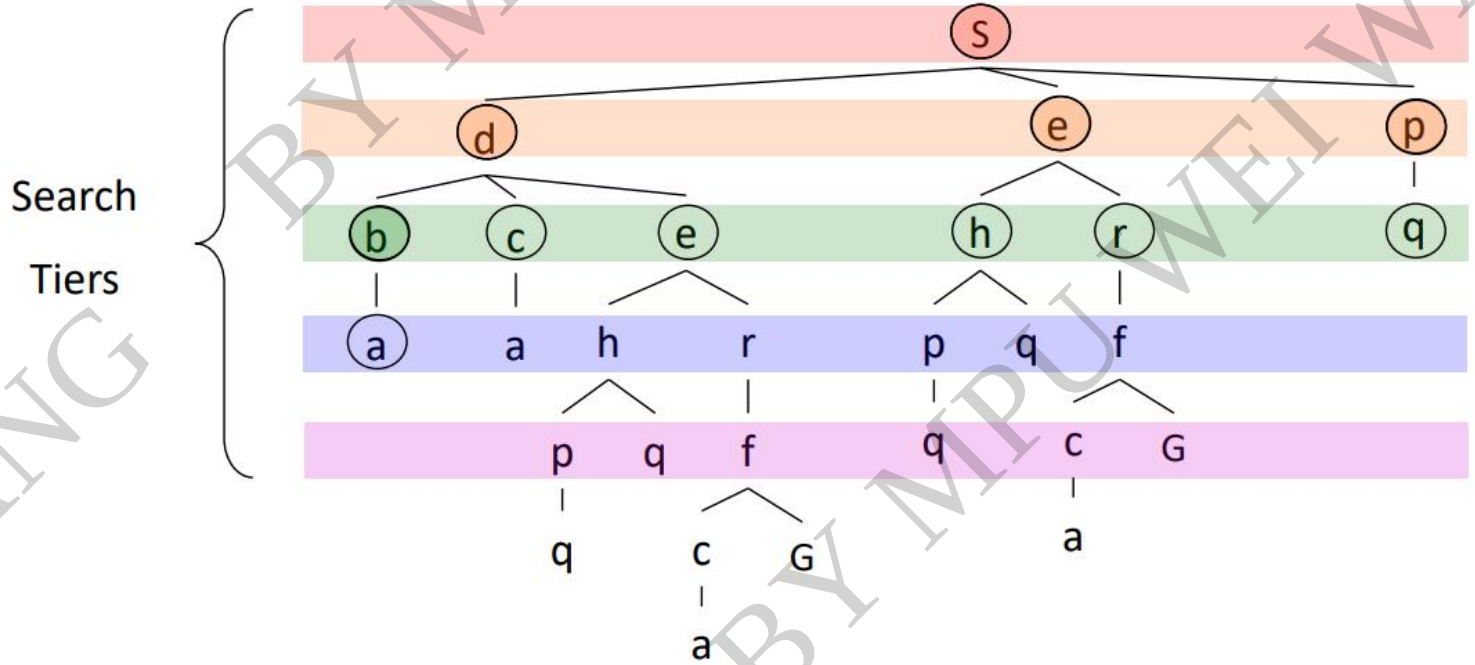
Strategy: expand a shallowest node first

Implementation: Frontier is a FIFO queue

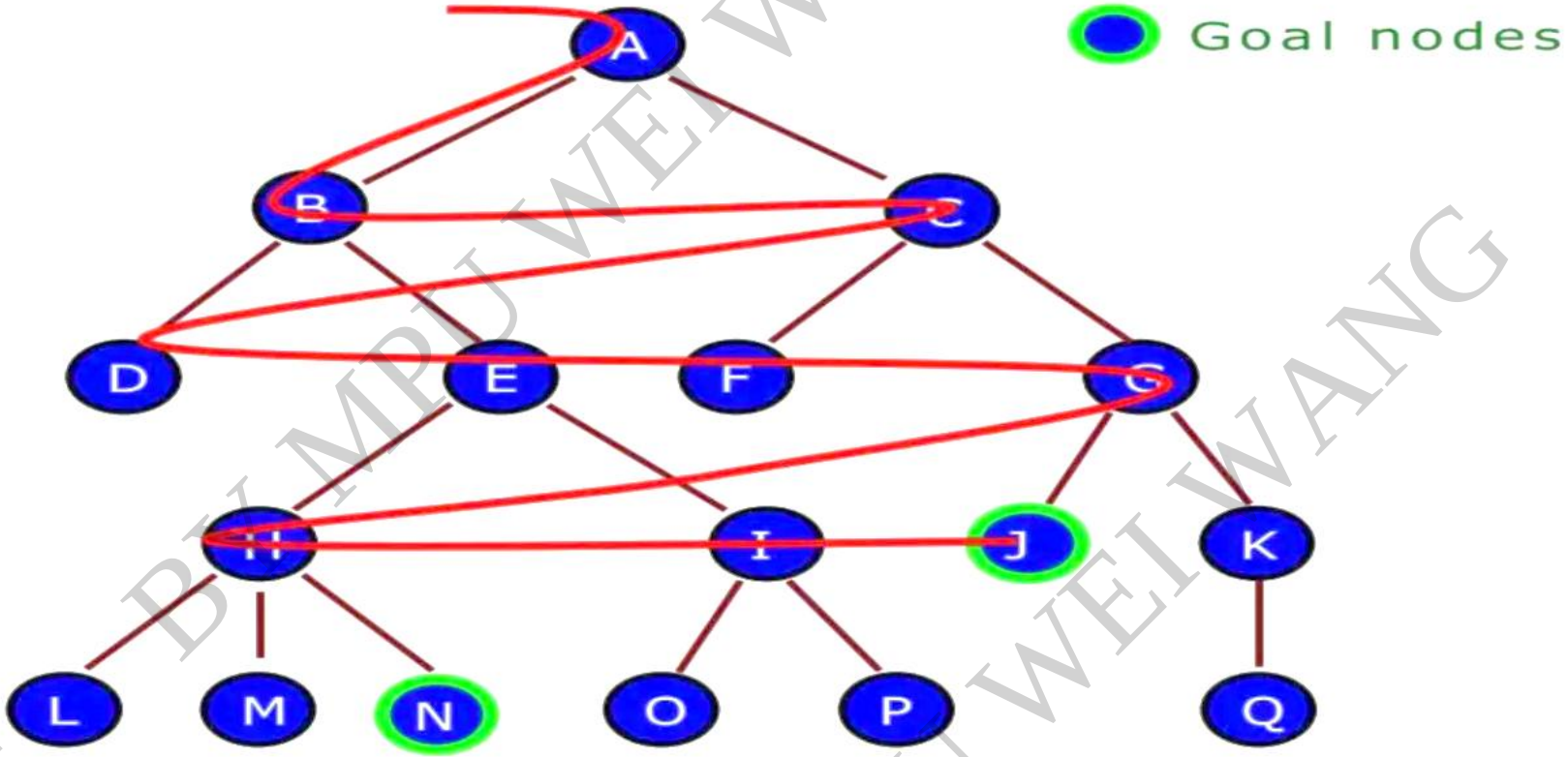
(first in first out)



Breadth-First Search is a search strategy, in which the highest layer of a decision tree is searched completely before processing to the next layer.



Breadth-First Search



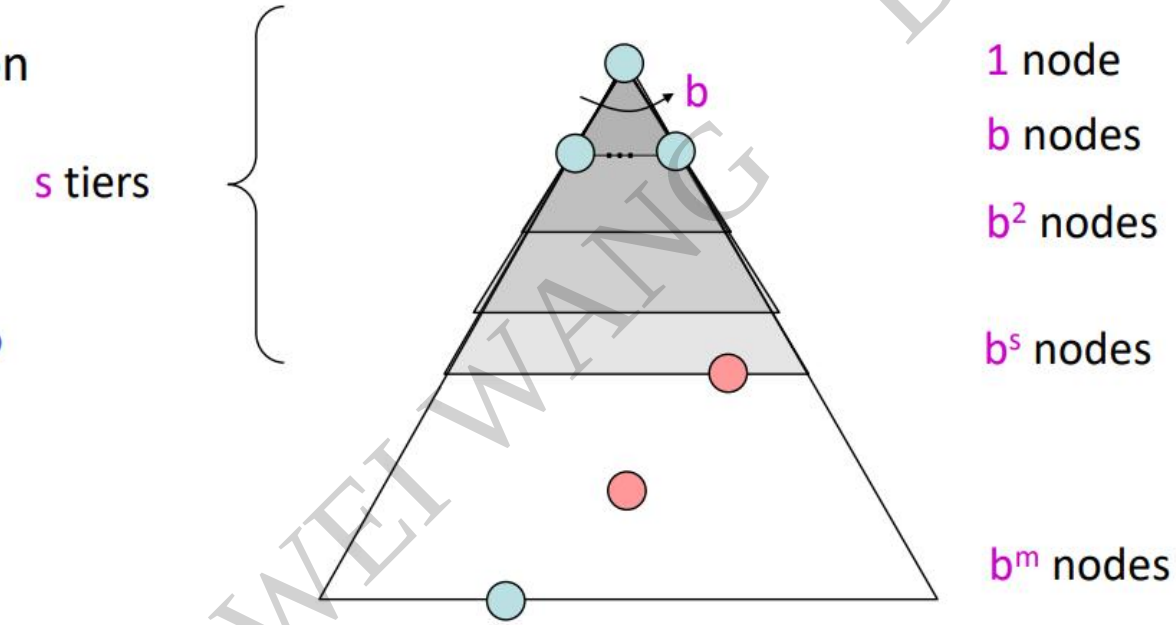
Node are explored in the order :

A B C D E F G H I J K L M N O P Q

- After searching **A**, then **B**, then **C**, the search proceeds with **D,E, F, G**, . .
- The goal node **J** will be found before the goal node **N**.

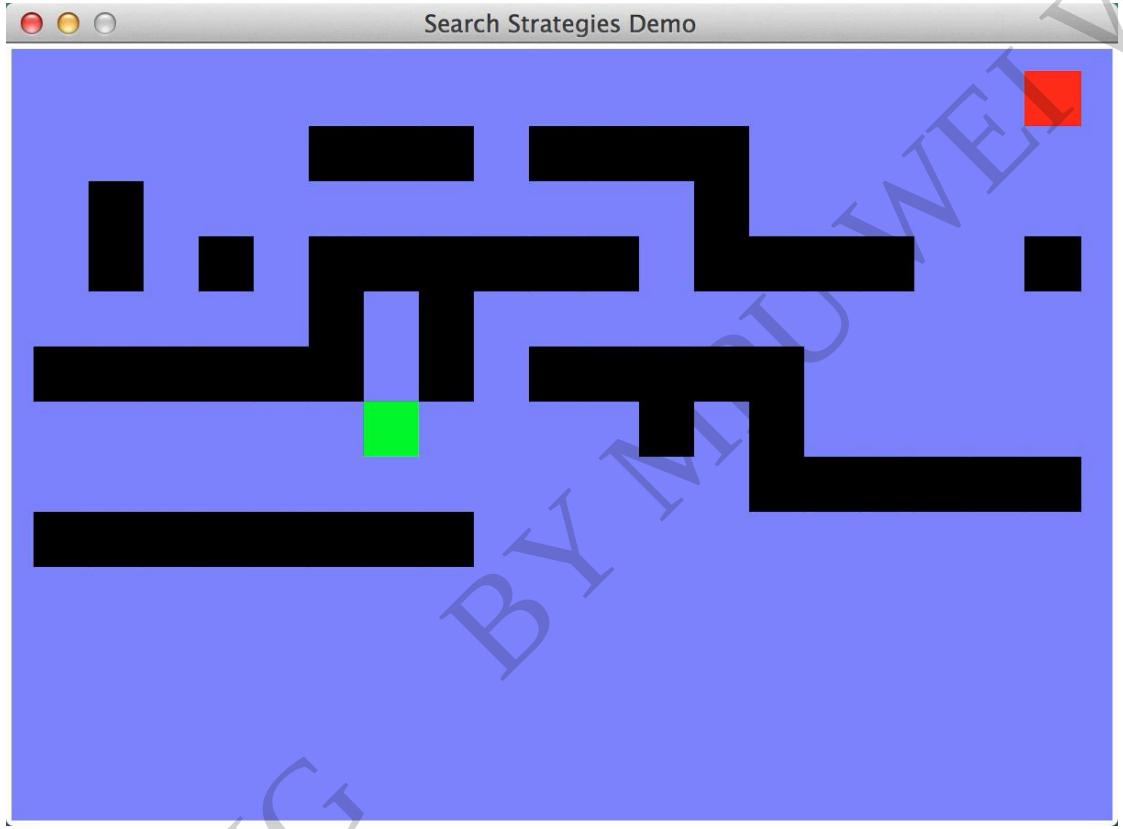
Breadth-First Search (BFS) Properties

- What nodes does BFS expand?
  - Processes all nodes above shallowest solution
  - Let depth of shallowest solution be *s*
  - Search takes time $O(b^s)$
- How much space does the frontier take?
  - Has roughly the last tier, so $O(b^s)$
- Is it complete?
  - *s* must be finite if a solution exists, so yes!
- Is it optimal?
  - If costs are equal (e.g., 1)



- Time: $1 + b + b^2 + \ldots + b^s + b(b^s-1) = O(b^{s+1})$
  exponential in s
- Space: $O(b^{s+1})$
  Keeps every node in memory

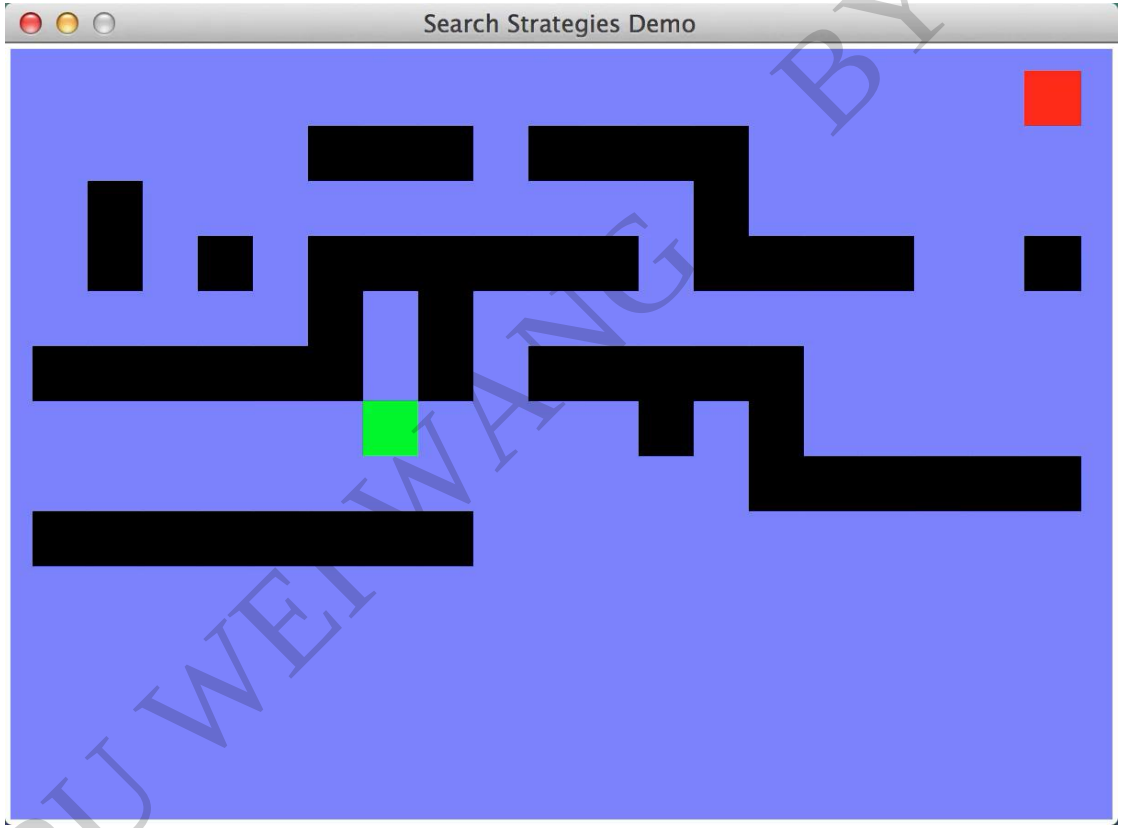
Example: Maze Water DFS or BFS



Green nodes spread gradually across nearby areas.

Exploration expands layer by layer around the start.

This behavior matches Breadth-First Search (BFS).



Green nodes follow a long path moving away from the start, exploring deeply first.

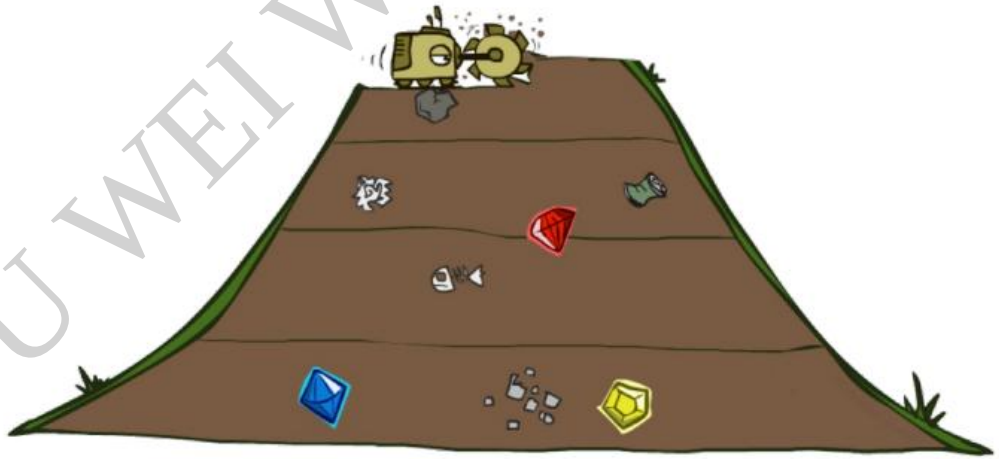
Large unexplored regions remain nearby.

This pattern matches Depth-First Search (DFS).

DFS vs BFS

(In terms of S, the depth of the shallowest solution and M, the maximum depth)

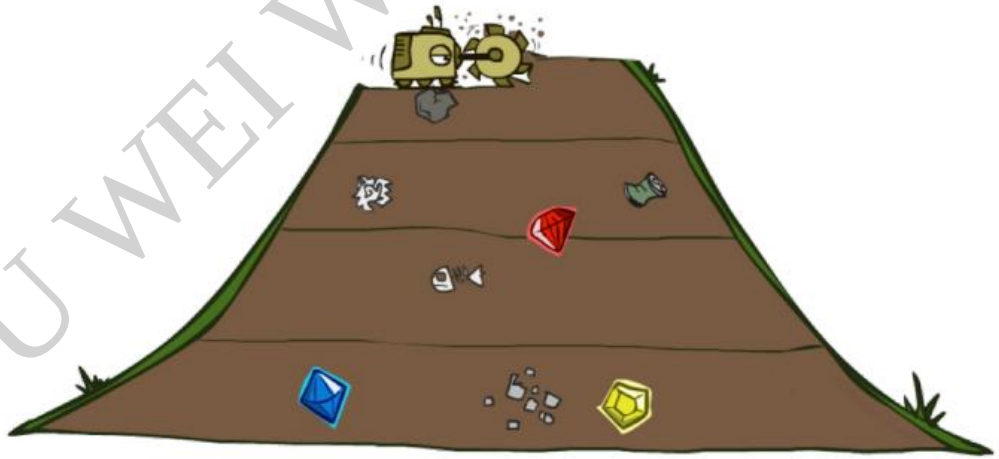
- When will BFS outperform DFS?
- When will DFS outperform BFS?



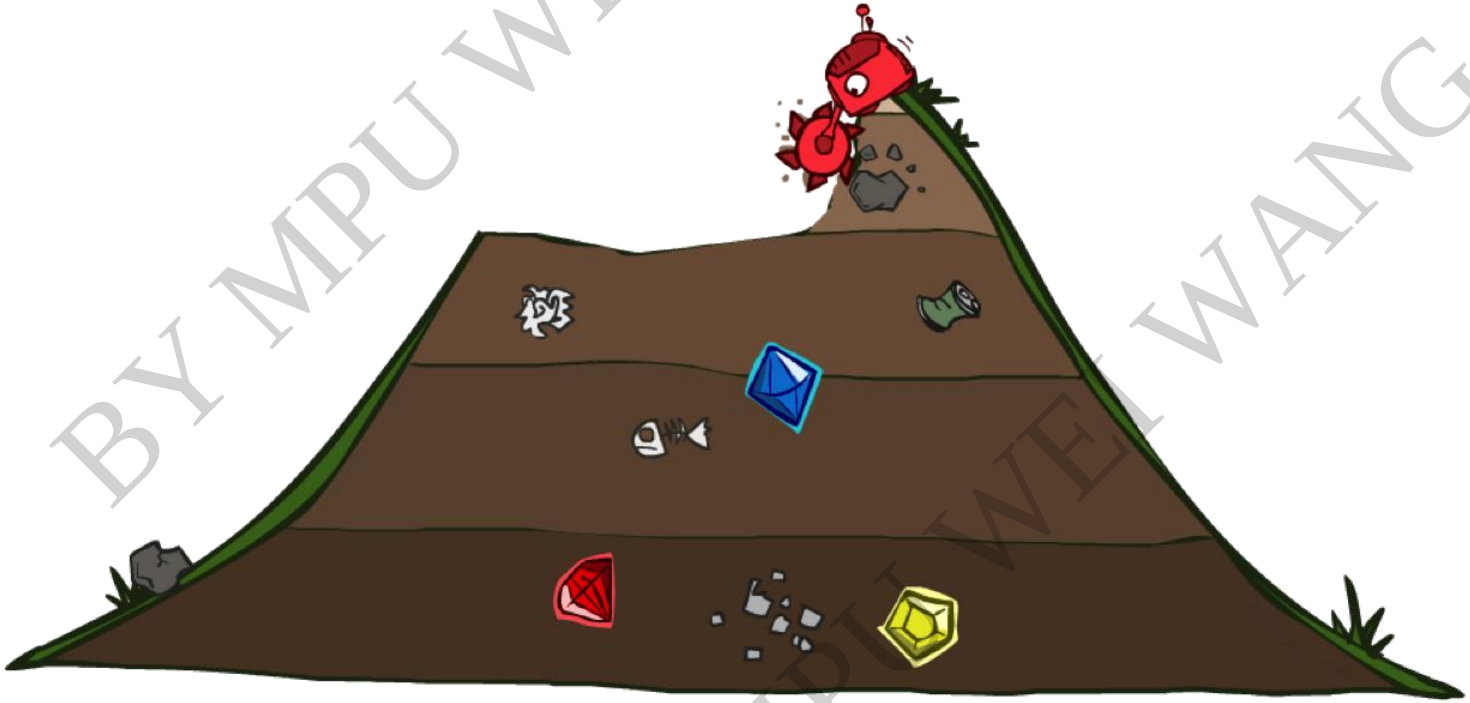
DFS vs BFS

(In terms of S, the depth of the shallowest solution and M, the maximum depth)

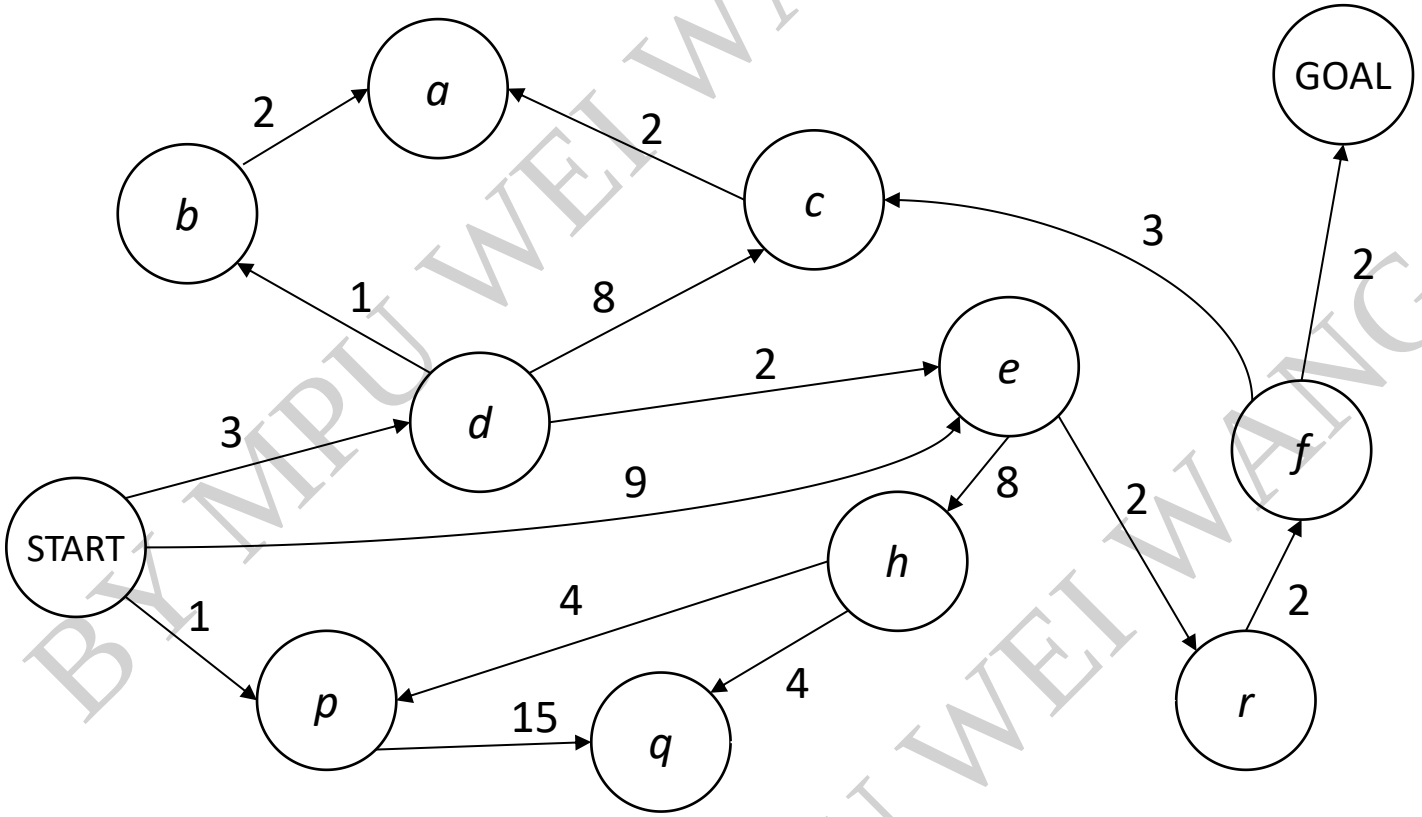
- When will BFS outperform DFS? **S << M**
- When will DFS outperform BFS? **S ~= M**



Uniform Cost Search



Cost-Sensitive Search



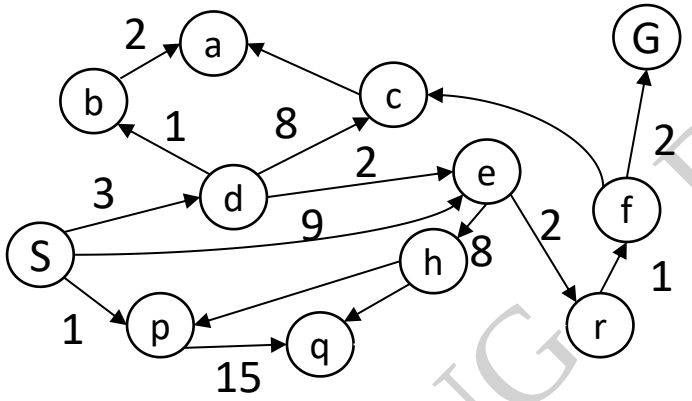
BFS finds the shortest path in terms of number of actions.
It does not find the least-cost path. We will now cover
a similar algorithm which does find the least-cost path.

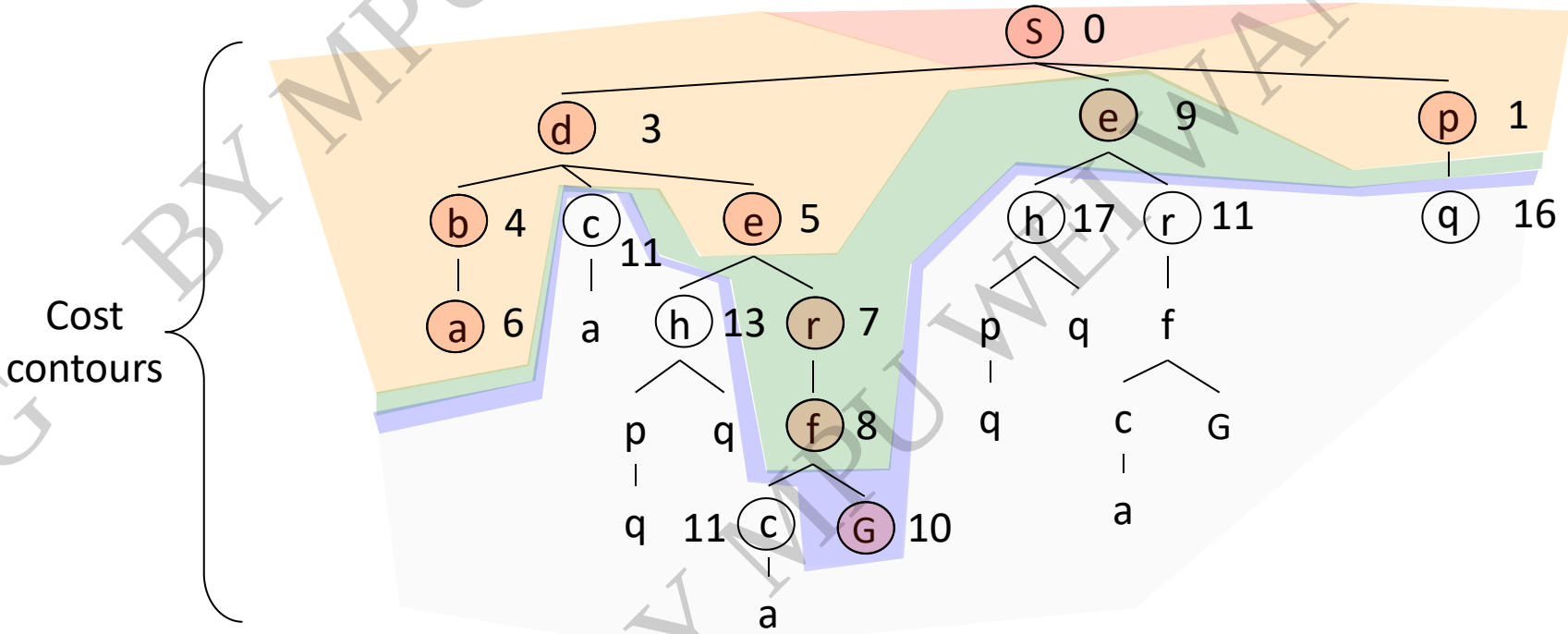
Uniform Cost Search

g(n) = cost from root to n

Strategy: expand lowest g(n)

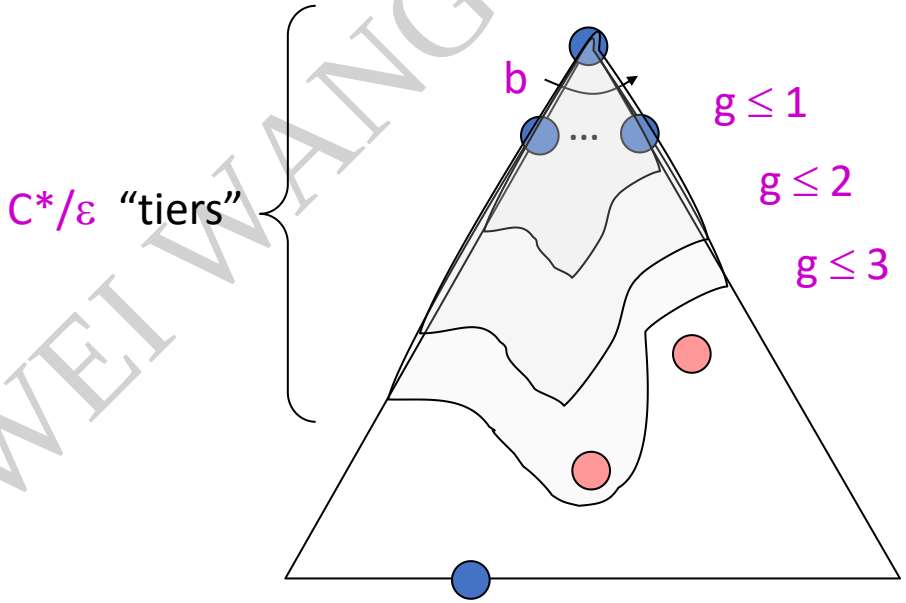
Frontier is a priority queue sorted by g(n)





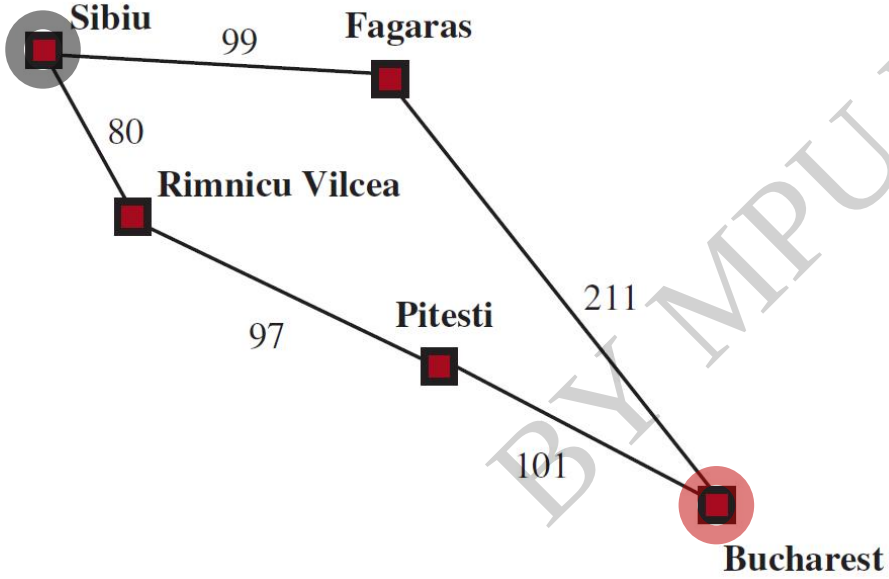
Uniform Cost Search (UCS) Properties

- What nodes does UCS expand?
  - Processes all nodes with cost less than cheapest solution!
  - If solution costs C* and arcs cost at least ε, then C^*/ε is effective depth (upper bound on depth of solution)
  - Takes time $O(b^{C^*/\varepsilon})$ (exponential in effective depth)
- How much space does the frontier take?
  - Has roughly the last tier, so $O(b^{C^*/\varepsilon})$
- Is it complete?
  - Assuming C* is finite and $\varepsilon > 0$, yes!
- Is it optimal?
  - Yes!

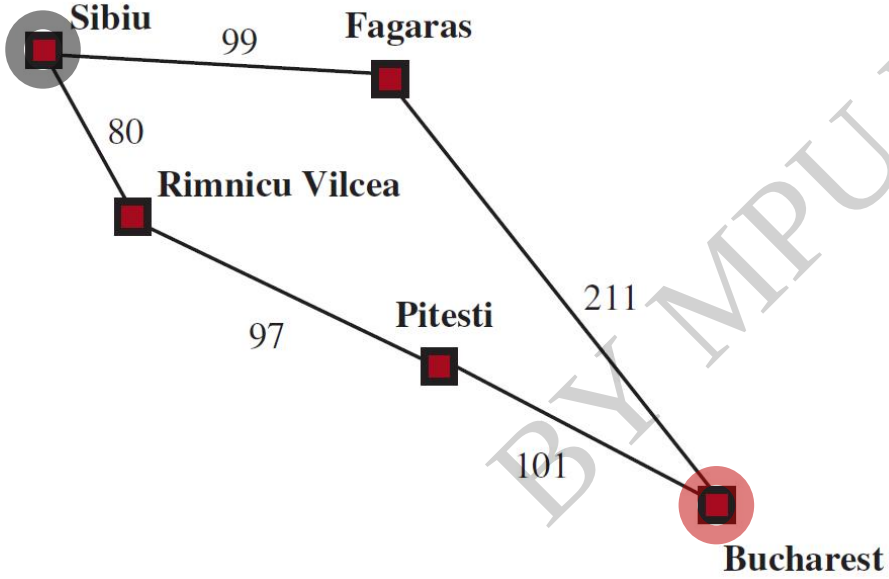


Uniform Cost Search (UCS)

0: Sibiu(Sibiu) = 0



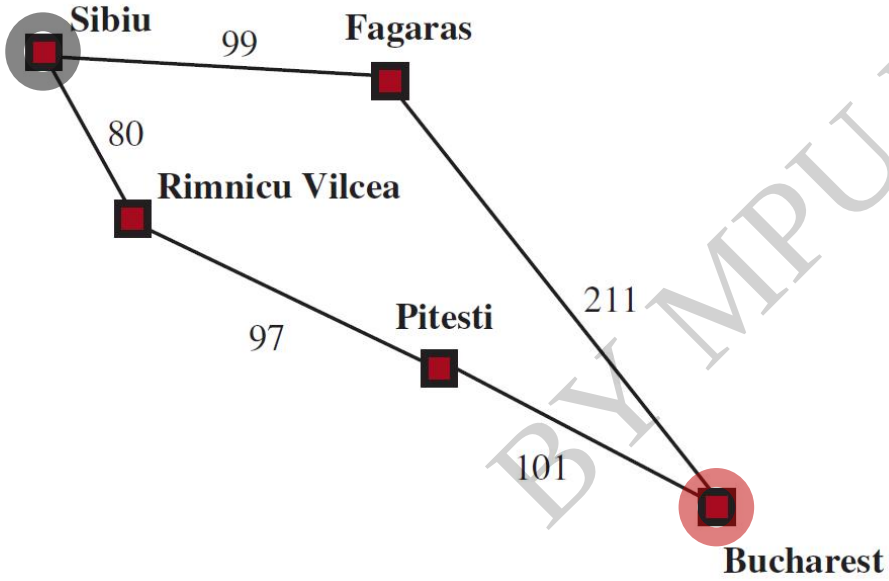
Uniform Cost Search (UCS)



0: Sibiu(Sibiu) = 0

1: Fagaras(Sibiu)=0+99,
Rimnicu Vilcea(Sibiu)=0+80

Uniform Cost Search (UCS)

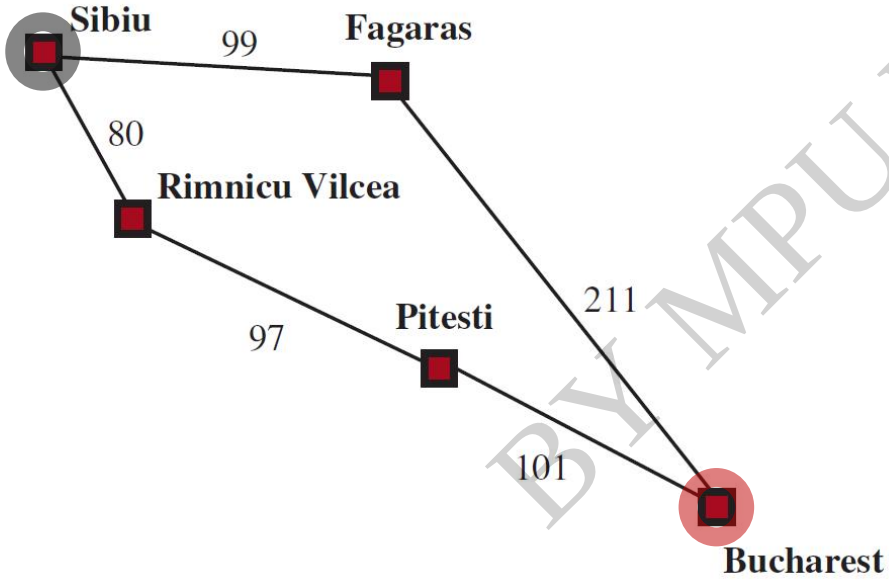


0: Sibiu(Sibiu) = 0

1: Fagaras(Sibiu)=0+99,

Rimnicu Vilcea(Sibiu)=0+80 => 99 > 80

Uniform Cost Search (UCS)

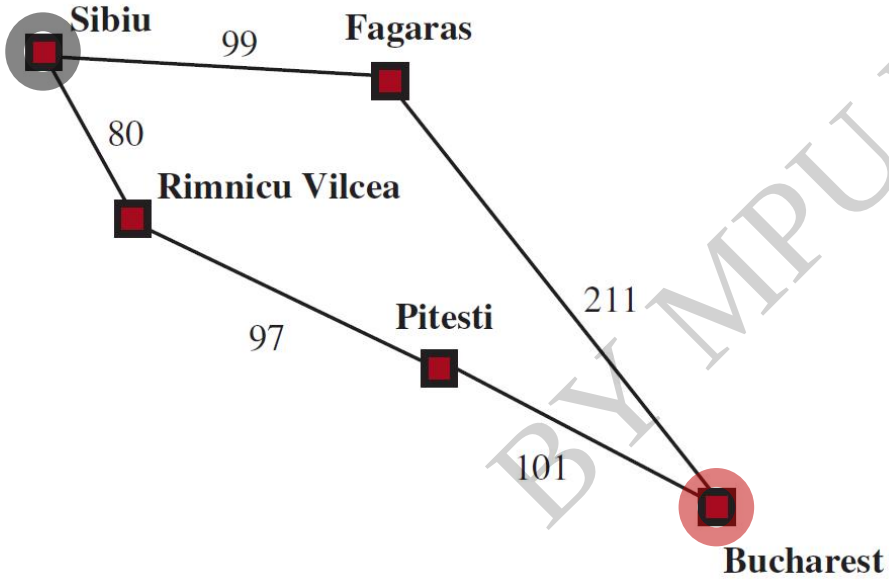


0: Sibiu(Sibiu) = 0

1: Fagaras(Sibiu)=0+99,
Rimnicu Vilcea(Sibiu)=0+80 => 99 > 80

2: Fagaras(Sibiu)=0+99,
Pitesti(Rimnicu Vilcea)=80+97

Uniform Cost Search (UCS)



0: Sibiu(Sibiu) = 0

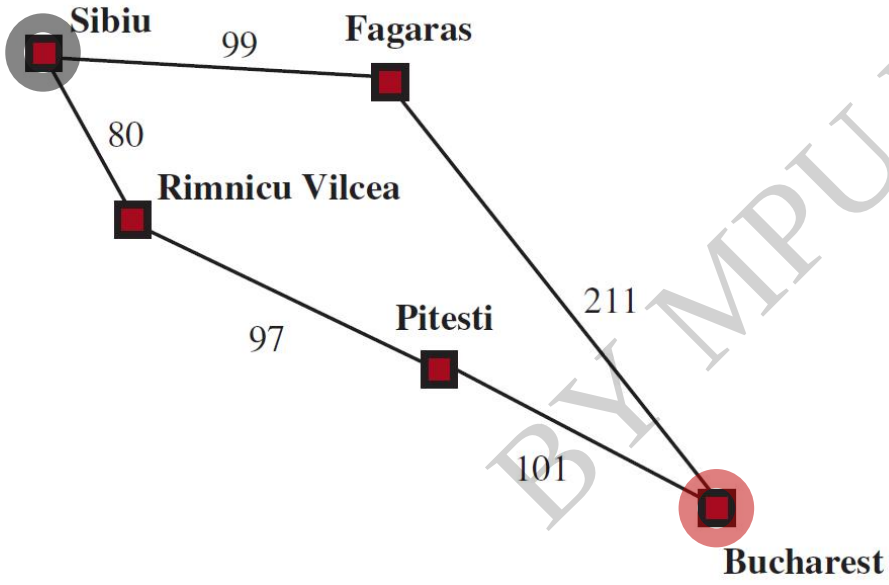
1: Fagaras(Sibiu)=0+99,

Rimnicu Vilcea(Sibiu)=0+80 => 99 > 80

2: Fagaras(Sibiu)=0+99,

Pitesti(Rimnicu Vilcea)=80+97 => 99 < 177

Uniform Cost Search (UCS)



0: Sibiu(Sibiu) = 0

1: Fagaras(Sibiu)=0+99,

Rimnicu Vilcea(Sibiu)=0+80 => 99 > 80

2: Fagaras(Sibiu)=0+99,

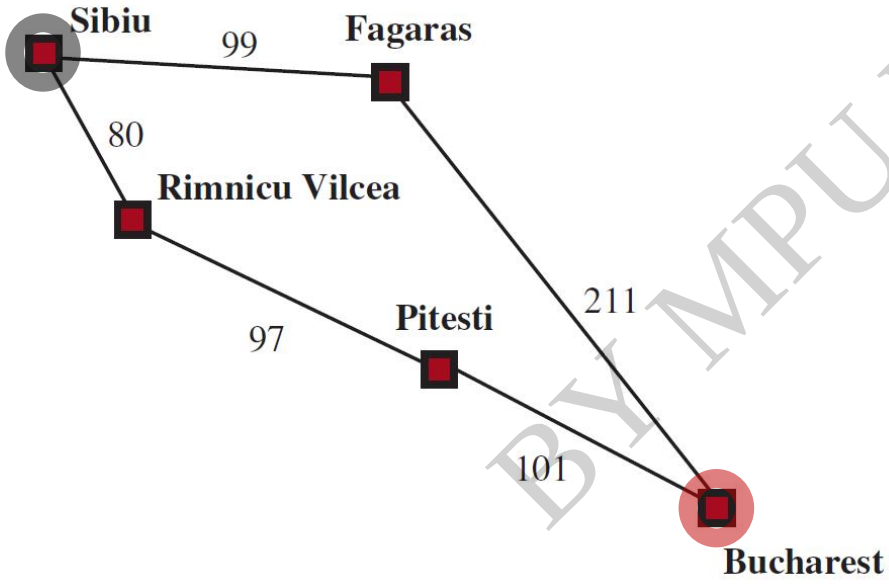
Pitesti(Rimnicu Vilcea)=80+97 => 99 < 177

3: Bucharest(Fagaras)=99+211,

Pitesti(Rimnicu Vilcea)==80+97

Will it stop?

Uniform Cost Search (UCS)



0: Sibiu(Sibiu) = 0

1: Fagaras(Sibiu)=0+99,

Rimnicu Vilcea(Sibiu)=0+80 => 99 > 80

2: Fagaras(Sibiu)=0+99,

Pitesti(Rimnicu Vilcea)=80+97 => 99 < 177

3: Bucharest(Fagaras)=99+211,

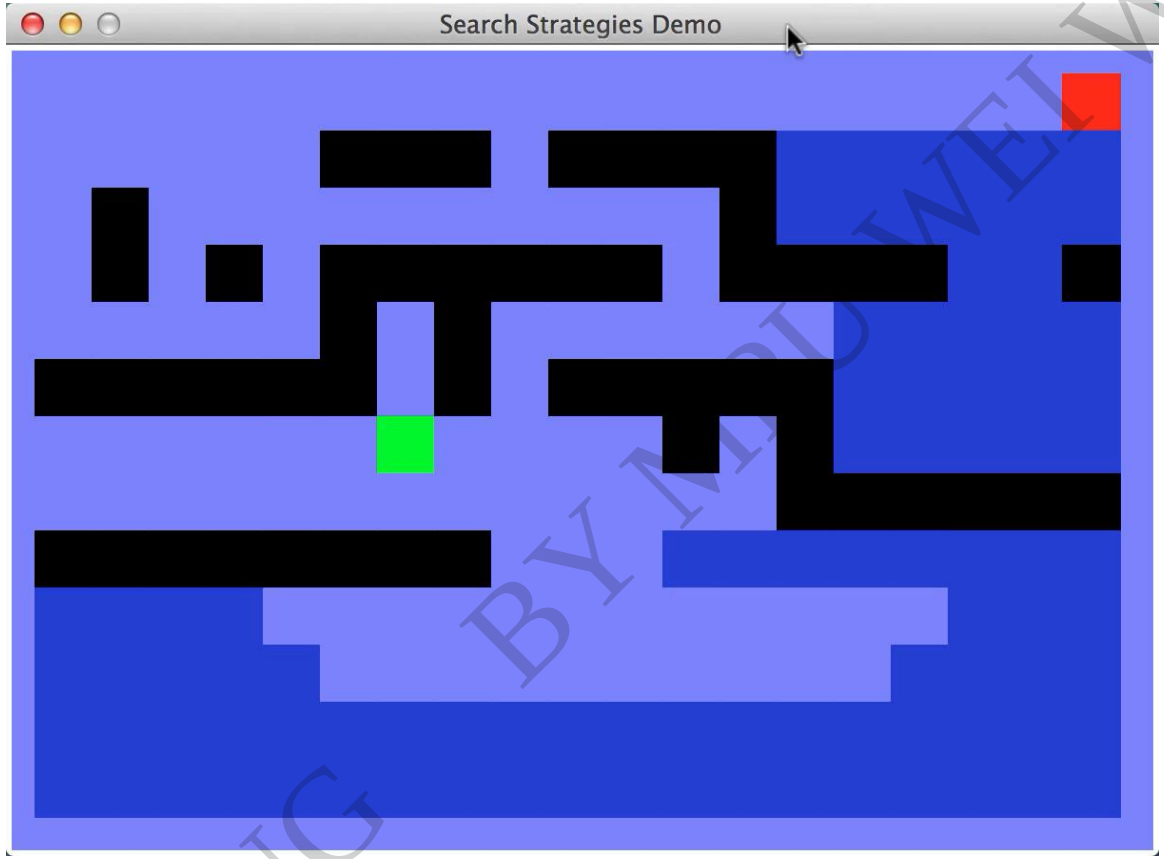
Pitesti(Rimnicu Vilcea)==80+97

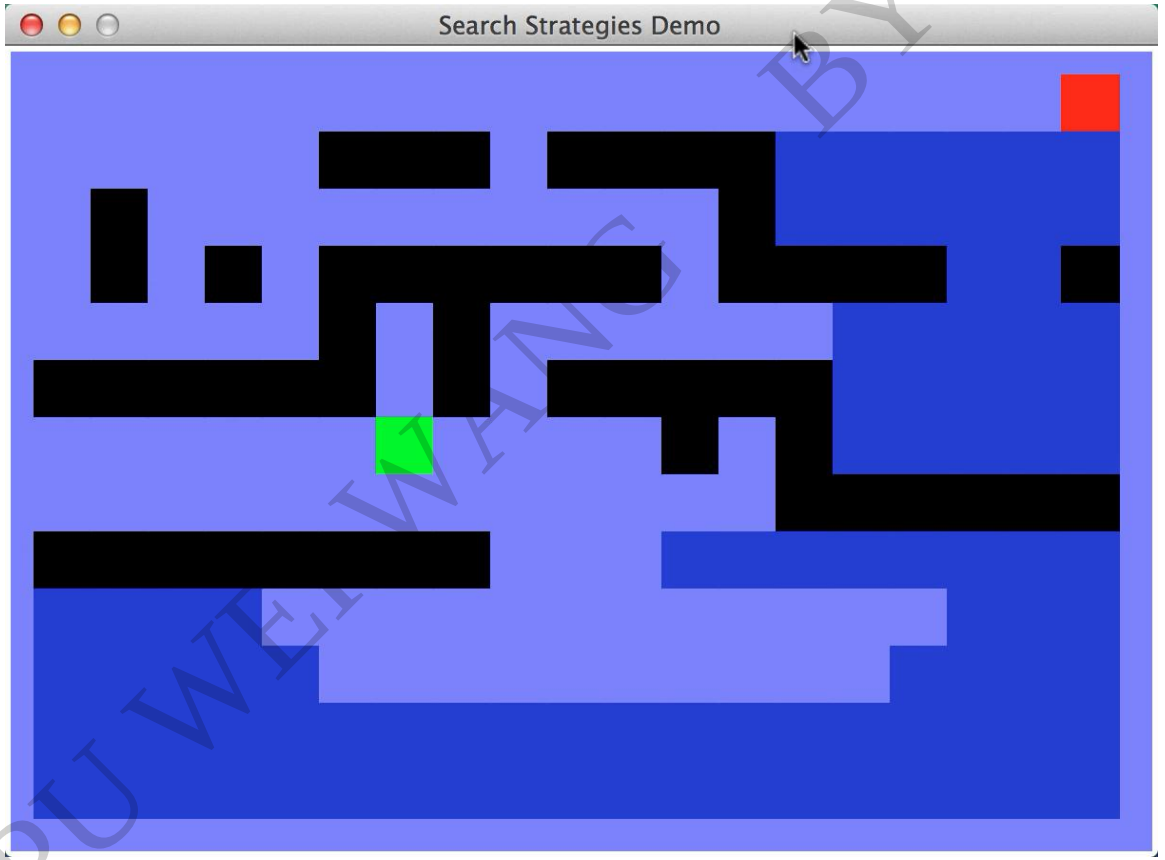
Will it stop?

Bucharest(Fagaras)=99+211=310

Bucharest(Pitesti)=80+97+101=278 => **310 > 278**

Example: Maze Water BFS or UCS





https://www.youtube.com/watch?v=cS-198wtfj0

或许并非所有应用都会消失，而是有一部分会转化为API——本质上，这些应用正在迅速转型，转而面向AI智能体提供服务。

Well, maybe not all apps will disappear. Maybe some will transform into APIs—basically apps that rapidly transform to being agent-facing.

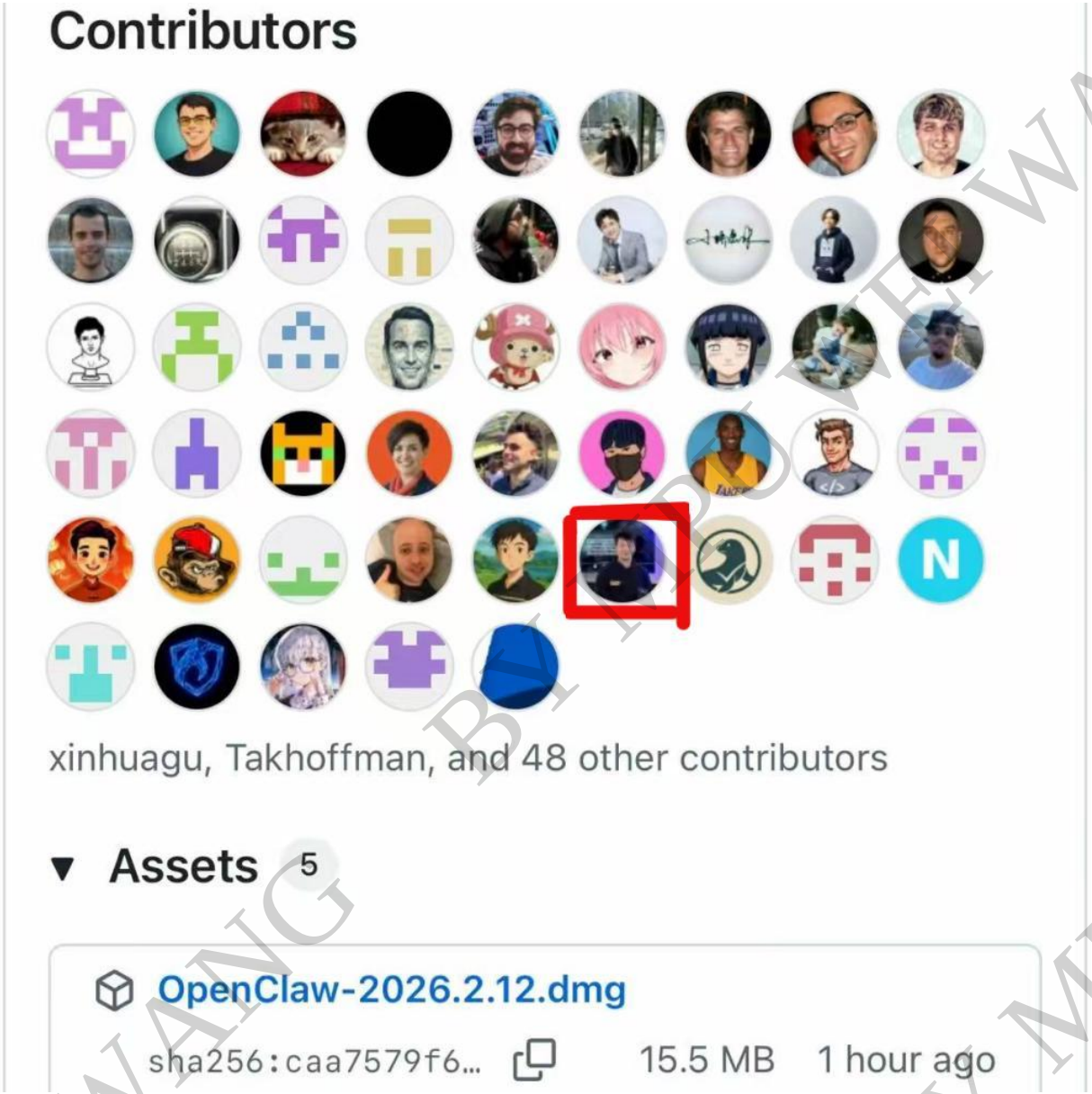
OPENCLAW

OpenClaw is an open-source AI agent project. You can think of it as a **digital employee with hands and brain**, which is the core difference between it and traditional AI tools like ChatGPT and Doubao. Traditional **AI** is more like a chat tool or a "strategist," while OpenClaw can actually perform tasks.

Its biggest features are **local operation, full-privilege execution, and proactive service**. It supports systems like Mac, Windows, and Linux. After deployment, it has deep access to the computer's file system and various applications, and can connect to chat software like WhatsApp, Telegram, DingTalk, and Lark. Just send a message, and it can automatically complete the task, without requiring human intervention at the computer.

As a task execution tool, OpenClaw is officially labeled as an "**AI assistant with a soul**." Clearing inboxes, automatically sending emails, managing calendars, checking in for flights, writing code, organizing documents, performing data analysis… it can basically handle all the repetitive tasks on a computer, truly freeing people from tedious, repetitive work.

Talk is cheap, show me the code



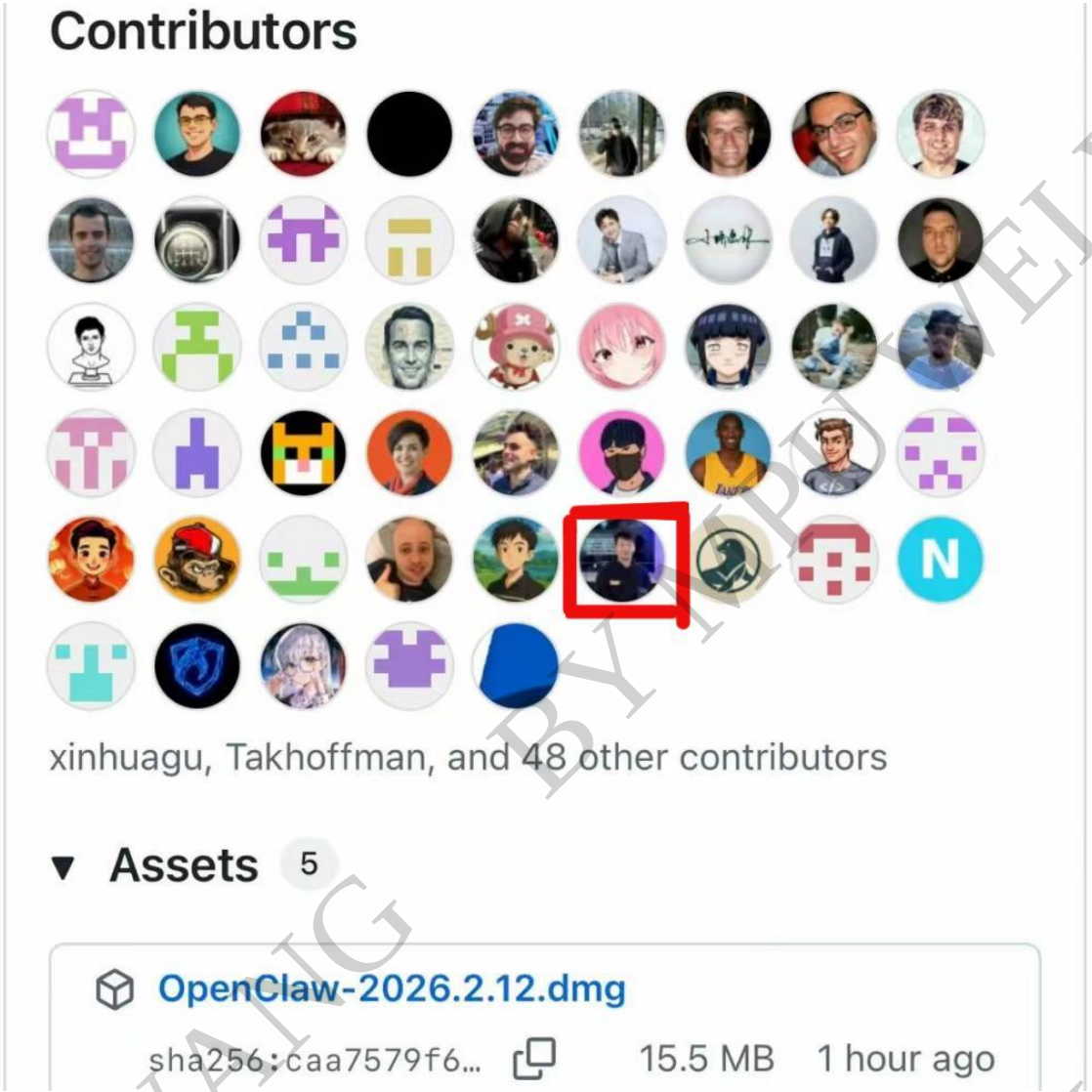
Tianrun Yang

The ticket to the new world of AI:
Curiosity, Imagination, and Courage.

The first level treats **AI as a tool**. You tell the AI every detail: font size, color depth, how to write the code.

The second level treats **AI as an employee**. You start assigning tasks, but can't resist micromanaging, telling it which technical path to take and what architecture to use. Both of these downgrade the AI, locking its capabilities to your level.

The third level treats **AI as a master**, not teaching it how to do things. You tell the AI, "You are one of the top ten engineers in the world, possessing the best aesthetic sense and architectural skills." In your view, "Since it's a top expert, what right do you have to tell it the path to achieve its goals?"



Tianrun Yang

First, he's **results-oriented**. He never tells the AI "to fix this bug" or "to write this function." He only sets the strategic goal: "I want to be in the top 20 of the contribution leaderboard within a week." How to achieve that? Whether it's modifying documentation, fixing bugs, or optimizing code? That's the AI's problem to solve.

Second, he **minimizes interference in the process**, which is the most difficult part. Humans always want to micromanage, teaching AI how to do things. But Tianrun forces himself to be a "hands-off manager." As long as the result is correct, he completely ignores how the AI writes code, calls libraries, or takes detours. He's found that human intervention often disrupts the AI's logical loop, actually reducing efficiency.

Finally, and most counterintuitively and boldly—**within manageable risk, grant it the highest level of control**. He opens up all permissions, tools, and context to it. Let it experiment, crash, and fix itself. You'll be surprised to find that its self-healing ability is far superior to yours.

Some Important Dates

Class Observation: March 16 （Pls Come）　　听课: 3月16号

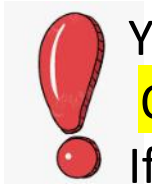
Company Career Talk: March 23 (16:30-17:30), March 30 (15:00-16:30) Sign Attendee
企业宣讲：3月23号(16:30-17:30)，3月30号 (15:00-16:30)　　需要签到

Midterm Test : April 11， 14:30-16:00　　期中测试：4月11号， 14:30-16:00

Assignment 2 Presentation: April 13, 20, 27(Q&A)　　作业2演示：4月13号，20号，27号(答疑)

Final Examination: April 30， 19:00 - 22:00　　期末考试：4月30号， 19:00 - 22:00

Classwork



You can sign out at any time after **completing this assignment**.

OR

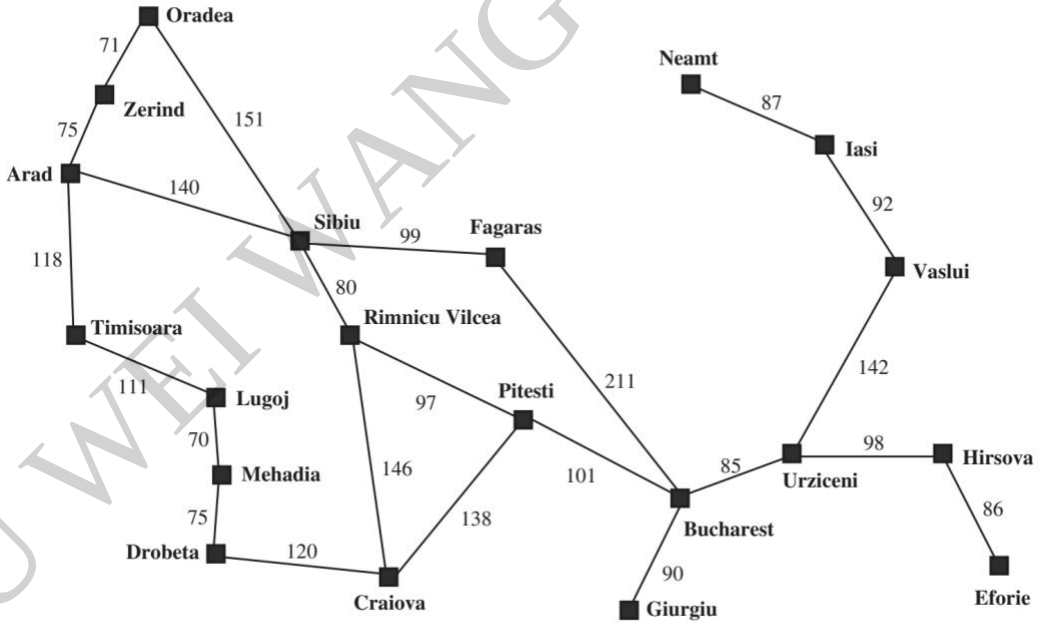
If you do not complete it, you can sign out **at 5 PM**.

Using Romania map on the rignt. Your task is to find a route from Arad to Bucharest using two search algorithms:

Breadth-First Search (BFS); Uniform Cost Search (UCS)

You must implement both algorithms and output:

The path from the start city to the goal city
The total path cost
The order of visited nodes



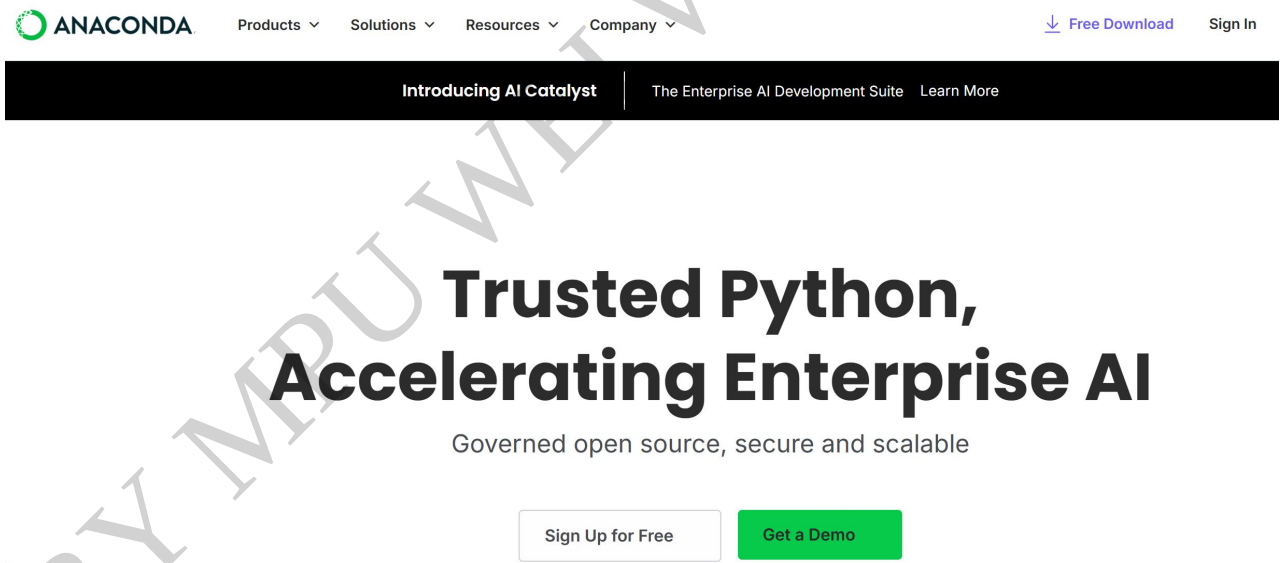
This assignment has nothing to do with the grade!

Run the Logistic Regression of Iris

https://mirror.tuna.tsinghua.edu.cn/help/anaconda/

1. What is Anaconda?
Anaconda is a Python data science distribution, essentially an 'integrated toolkit'. Its core components include:
- **Python interpreter**: The foundational program for directly executing Python code.
- **Hundreds of pre-installed libraries**: Covering common tools for data processing, numerical computation, visualisation, machine learning, and more.
Conda package manager: for installing, updating, and uninstalling libraries, offering greater capabilities than Python's built-in pip (manages non-Python dependencies); Environment management tool: enables creation of multiple independent virtual environments (e.g., one environment using Python 3.8, another using 3.10), preventing dependency conflicts between different projects. 2. How to install Anaconda
1. Download from the official website
2. Download via mirror sites

Anaconda Install



Run the Logistic Regression of Iris

https://www.jetbrains.com/pycharm/

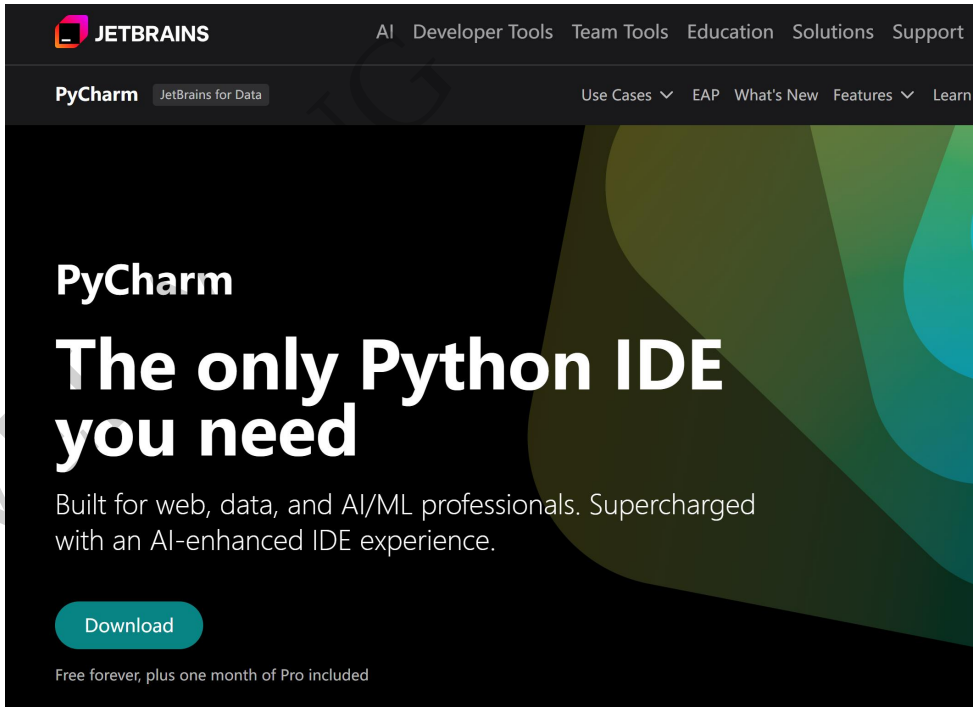
What is PyCharm?

PyCharm is a professional Python integrated development environment (IDE) developed by JetBrains, specifically designed for the Python programming language.
It offers extensive features and tools aimed at enhancing developers' programming efficiency and code quality.

It is widely used across various Python development domains, including web development, data analysis, artificial intelligence, and scientific computing.

It is available in two editions:
Community Edition and **Professional Edition**.

Opt for a **more stable cracked version** during PyCharm installation.



Pycharm Install

Thank you!

Innovating into the Future