NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

Introduction to Data Science and Artificial Intelligence

Solving Problems by Search: Uninformed Search and Informed Search *(Review Lecture)*

ONG Chin Ann

https://dr.ntu.edu.sg/cris/rp/rp02123
Email: chinann.ong@ntu.edu.sg
Office: N4-02c-108

Lesson Outline

- Recap on Problem Formulation
- Search Algorithm Overview
- Uninformed search strategies
- Informed search strategies
  - Greedy search
  - A * search
- Quick Briefing on Lab Exercise 7

Review: Well-Defined Formulation

Definition of a problem	The information used by an agent to decide what to do

Specification	• Initial state
	• Action set, i.e. available actions (successor functions)
	• State space, i.e. states reachable from the initial state
	    • Solution path: sequence of actions from one state to another
	• Goal test predicate
	    • Single state, enumerated list of states, abstract properties
	• Cost function
	    • Path cost *g(n),* sum of all (action) step costs along the path

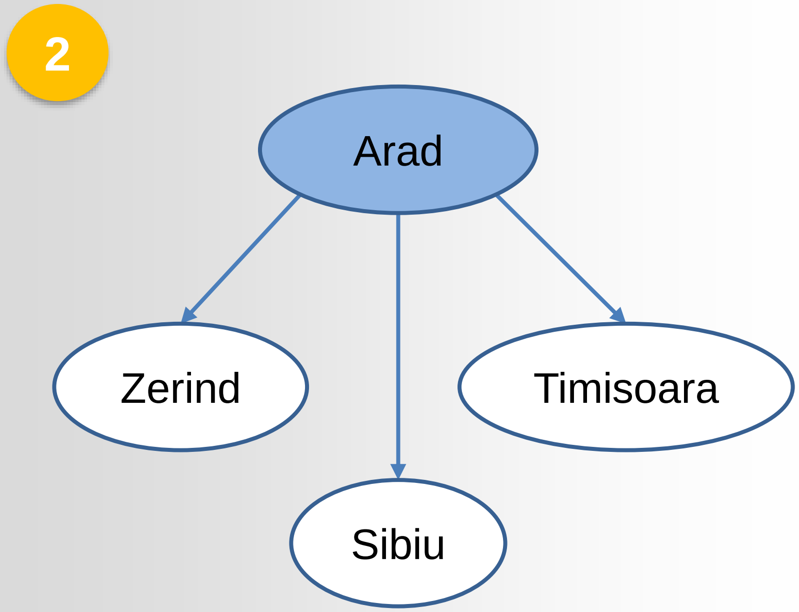
Solution	A path (a sequence of operators leading) from the Initial-State to a state that satisfies the Goal-Test

Search Algorithms

- Exploration of state space by generating successors of already-explored states
  - Frontier: candidate nodes for expansion
  - Explored set

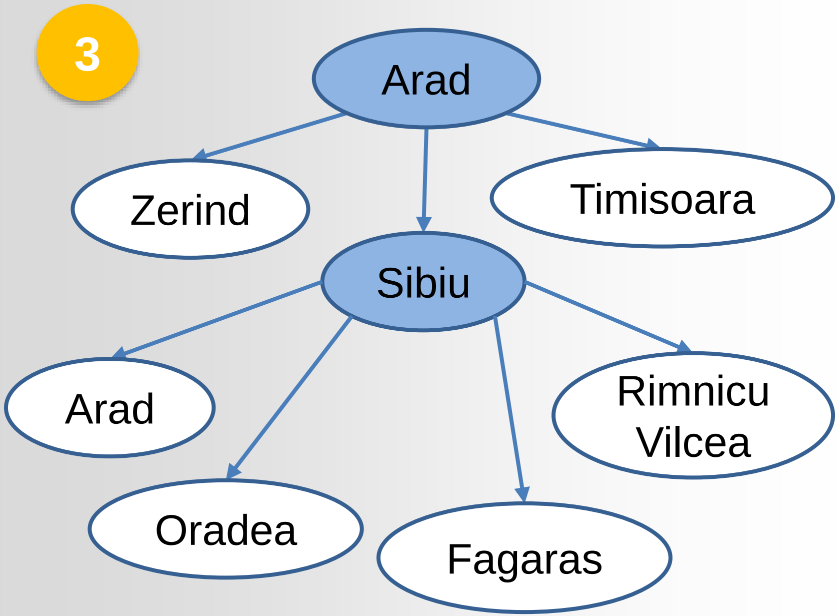
Search Algorithms

- Exploration of state space by generating successors of already-explored states
  - Frontier: candidate nodes for expansion
  - Explored set



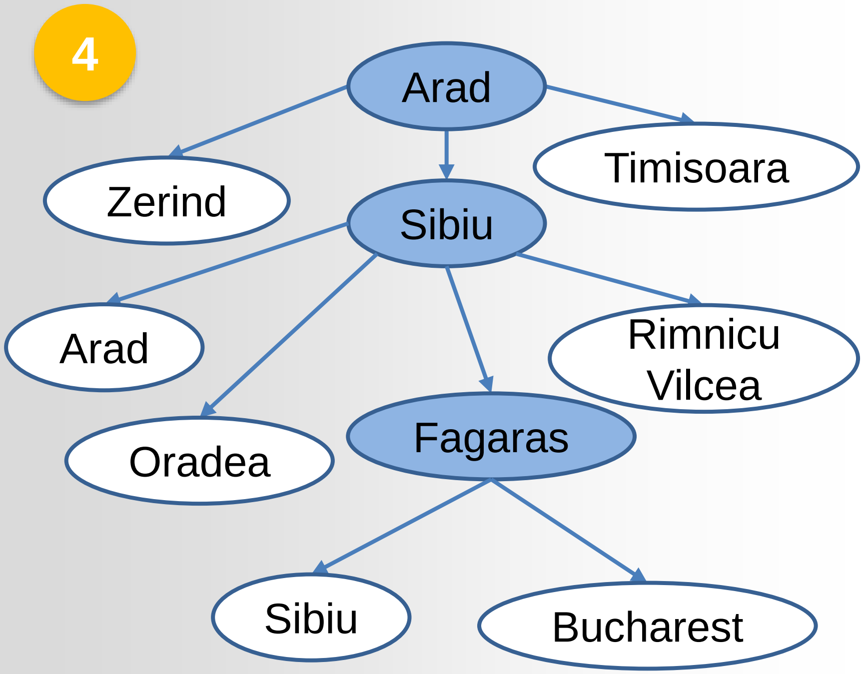
Search Algorithms

- Exploration of state space by generating successors of already-explored states
  - Frontier: candidate nodes for expansion
  - Explored set



Search Algorithms

- Exploration of state space by generating successors of already-explored states
  - Frontier: candidate nodes for expansion
  - Explored set



Search Strategies

- A strategy is defined by picking the order of node expansion.
- Strategies are evaluated along the following dimensions:

Completeness	Does it always find a solution if one exists?
Time Complexity	How long does it take to find a solution: the number of nodes generated
Space Complexity	Maximum number of nodes in memory
Optimality	Does it always find the best (least-cost) solution?

Search Strategies

- Branching factor
  - Maximum number of successors of any node
  - Or average branching factor

Uninformed vs Informed

Uninformed search strategies

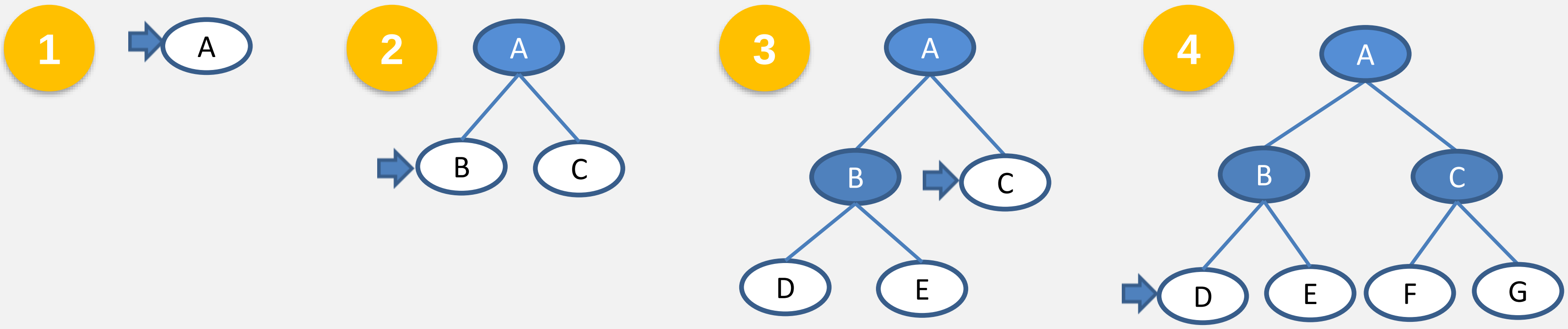
- Use only the information available in the problem definition
  1. Breadth-first search
  2. Uniform-cost search
  3. Depth-first search
  4. Depth-limited search
  5. Iterative deepening search

Informed search strategies

- Use problem-specific knowledge to guide the search
- Usually more efficient

Breadth-First Search

Expand shallowest unexpanded node which can be implemented by a First-In-First-Out (FIFO) queue



Denote
- b: maximum branching factor of the search tree
- d: depth of the least-cost solution
- Complete: Yes
- Optimal: Yes when all step costs equally

Complexity of BFS

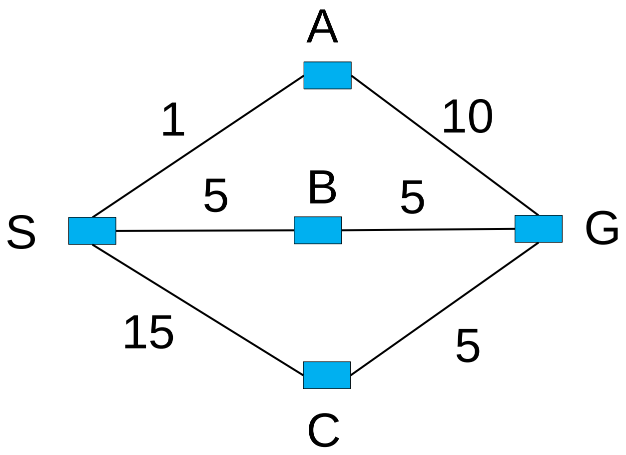
- Hypothetical state-space, where every node can be expanded into *b* new nodes, solution of path-length *d*
- Time: $1 + b + b^2 + b^3 + \ldots + b^d = O(b^d)$
- Space: (keeps every node in memory) $O(b^d)$ are equal

Depth	Nodes	Time		Memory	
0	1	1	millisecond	100	bytes
2	111	0.1	seconds	11	kilobytes
4	11111	11	seconds	1	kilobytes
6	10^6	18	minutes	111	megabyte
8	10^8	31	hours	11	gigabytes
10	10^{10}	128	days	1	terabyte
12	10^{12}	35	years	111	terabytes
14	10^{14}	3500	years	11111	terabytes

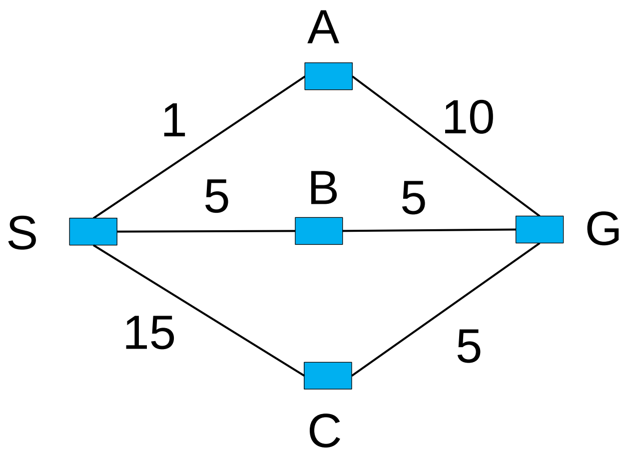
Uniform-Cost Search

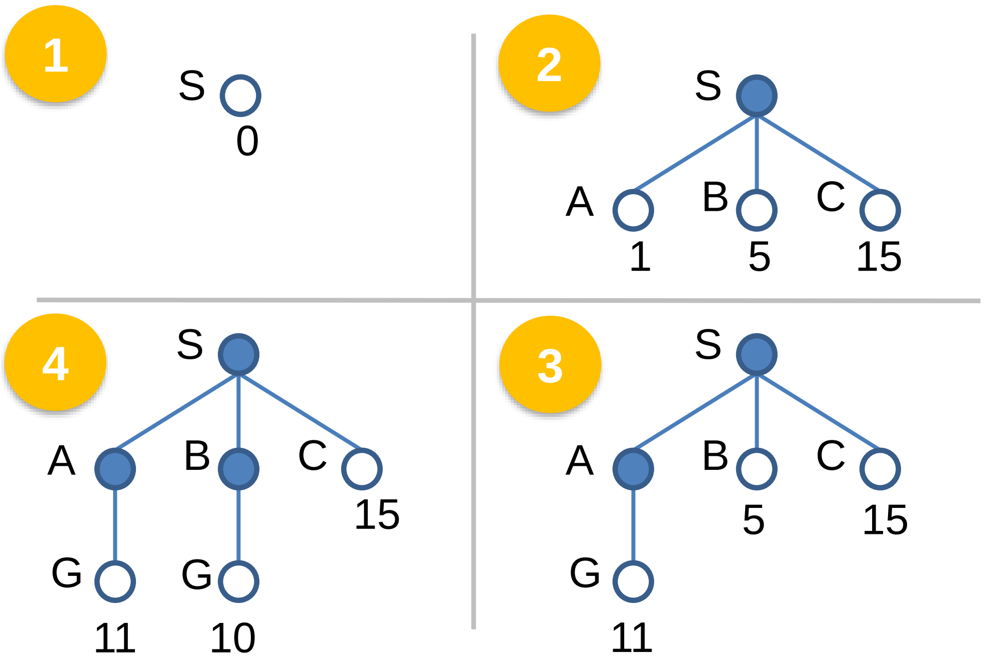
To consider edge costs, expand unexpanded node with the least path cost *g*

- Modification of breath-first search
- Instead of First-In-First-Out (FIFO) queue, using a priority queue with path cost *g(n)* to order the elements
- BFS = UCS with *g(n)* = Depth(n)



Uniform-Cost Search





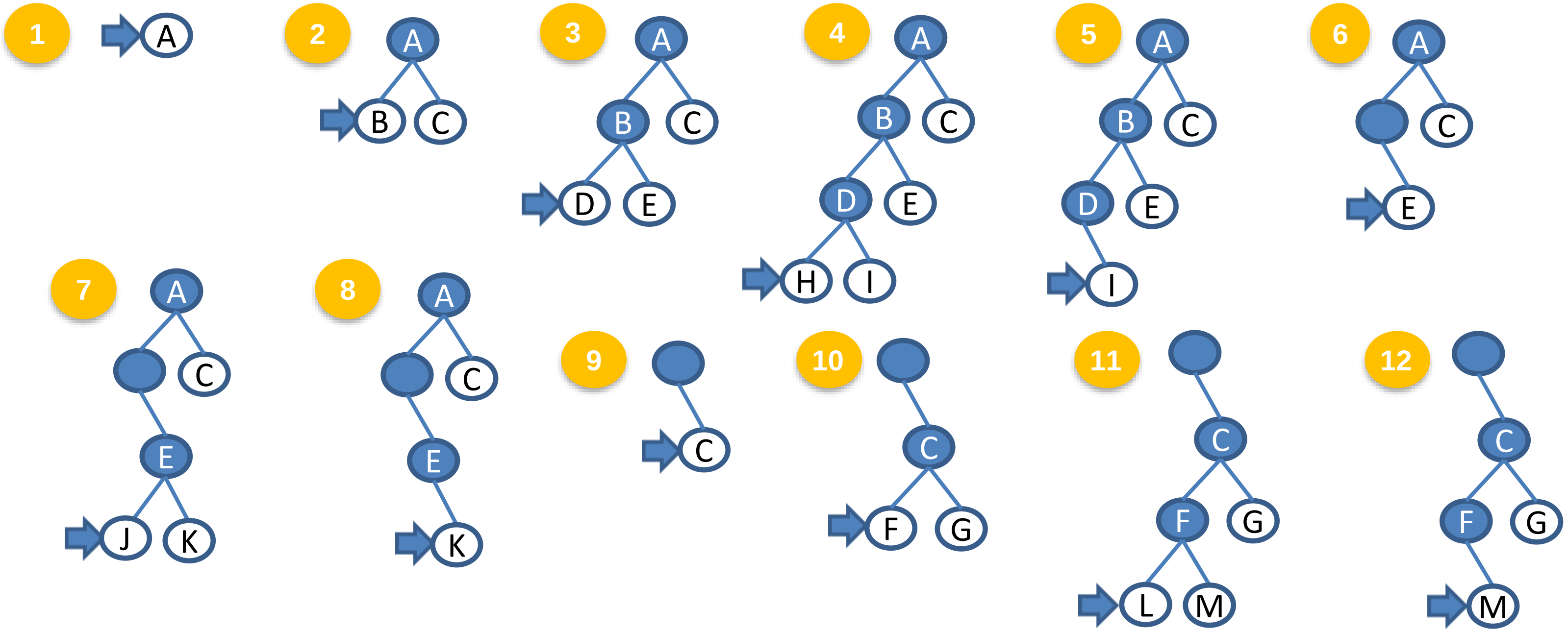
Here we do not expand notes that have been expanded.

Uniform-Cost Search

Complete	Yes
Time	# of nodes with path cost g <= cost of optimal solution (eqv. # of nodes pop out from the priority queue)
Space	# of nodes with path cost g <= cost of optimal solution
Optimal	Yes

Depth-First Search

Expand deepest unexpanded node which can be implemented by a Last-In-First-Out (LIFO) stack, Backtrack only when no more expansion



Depth-First Search

Denote

- m: maximum depth of the state space

Complete	• infinite-depth spaces: No • finite-depth spaces with loops: No • with repeated-state checking: Yes • finite-depth spaces without loops: Yes
Time	$O(b^m)$ If solutions are dense, may be much faster than breadth-first
Space	$O(bm)$
Optimal	No

Depth-Limited Search

To avoid infinite searching, Depth-first search with a cutoff on the max depth l of a path

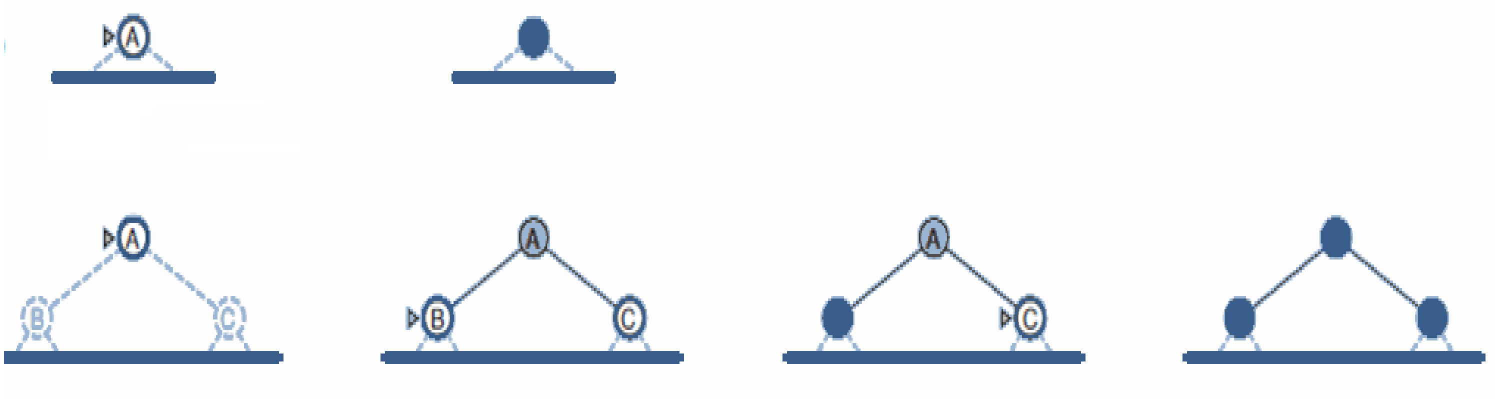
Complete	Yes, if $l \geq d$
Time	$O(b^l)$
Space	$O(bl)$
Optimal	No

Iterative Deepening Search

Iteratively estimate the max depth *l* of DLS one-by-one

Limit = 0

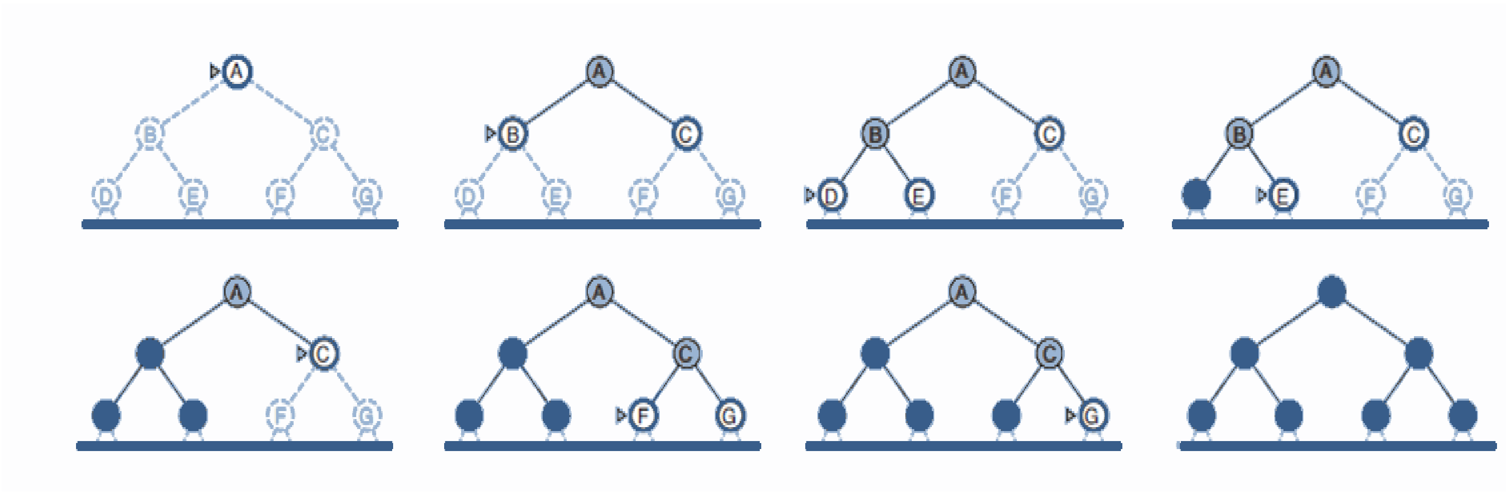
Limit = 1



Iterative Deepening Search

Iteratively estimate the max depth *l* of DLS one-by-one

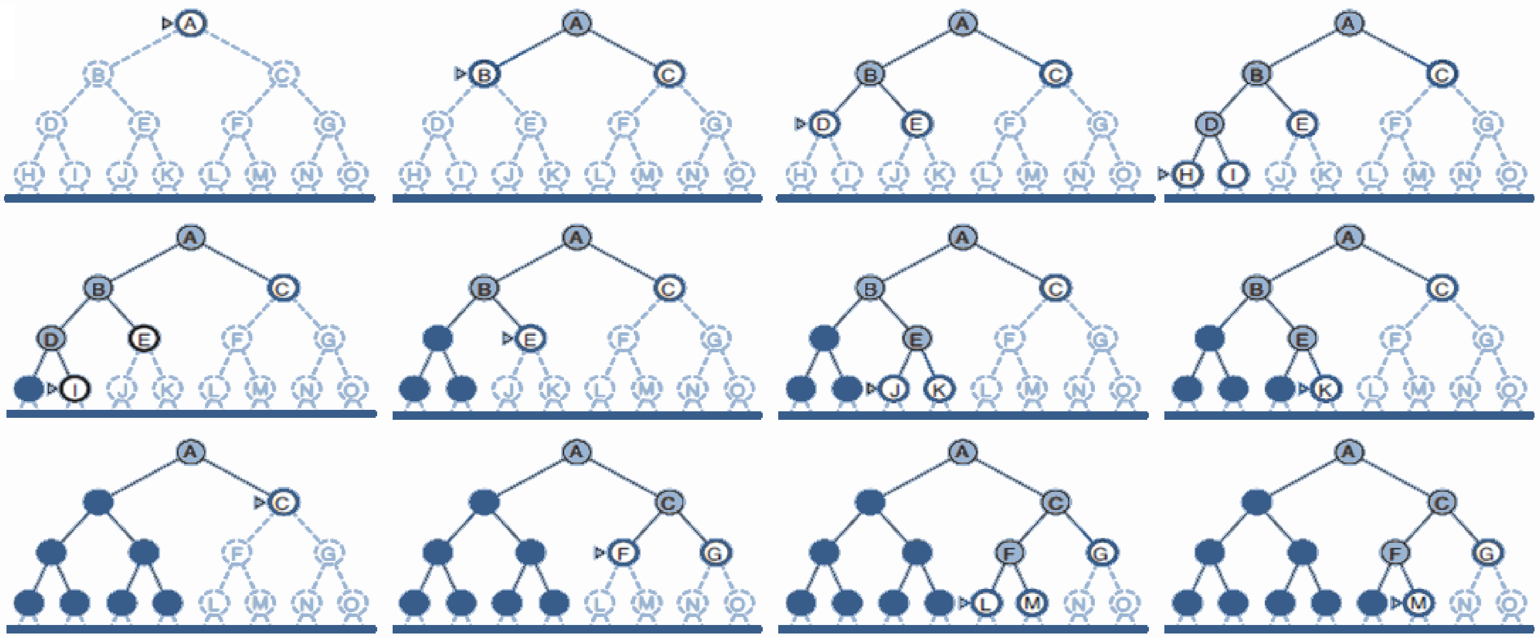
Limit = 2



Iterative Deepening Search

Iteratively estimate the max depth *l* of DLS one-by-one

Limit = 3



Iterative Deepening Search...

```
Function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution sequence
    inputs: problem, a problem

    for depth   0 to ∞ do
     if DEPTH-LIMITED-SEARCH(problem, depth) succeeds then return its result
    end
    return failure
```

Complete	Yes
Time	$O(b^d)$
Space	$O(bd)$
Optimal	Yes

Summary (we make assumptions for optimality)

Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

Question to think:

- If a search strategy is optimal, is it also complete?

Summary (we make assumptions for optimality)

Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

General Search

Uninformed search strategies

- Systematic generation of new states (→Goal Test)
- Inefficient (exponential space and time complexity)

Informed search strategies

- Use problem-specific knowledge
  - To decide the order of node expansion
- Best First Search: expand the most desirable unexpanded node
  - Use an evaluation function to estimate the “desirability" of each node

Evaluation function

- Path-cost function $g(n)$
  - Cost from initial state to current state (search-node) n
  - No information on the cost toward the goal
- Need to estimate cost to the closest goal
- “Heuristic” function $h(n)$
  - Estimated cost of the cheapest path from n to a goal state $h(n)$
    - Exact cost cannot be determined
  - depends only on the state at that node
  - $h(n)$ is not larger than the real cost (admissible)

Greedy Search

Expands the node that appears to be closest to goal

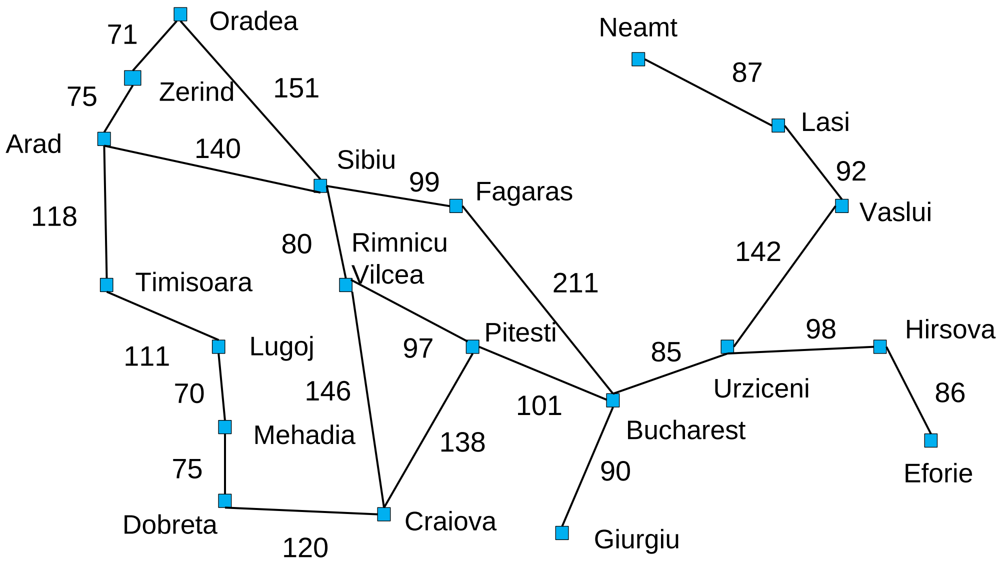
- Evaluation function $h(n)$:estimate of cost from n to $goal$
- Function Greedy-Search(problem) returns solution
  - Return Best-First-Search(problem, h) // $h(goal) = 0$

Question: How to estimation the cost from n to goal?

Answer: Recall that we want to use problem-specific knowledge

Example: Route-finding from Arad to Bucharest

$h(n)$ = straight-line distance from n to Bucharest



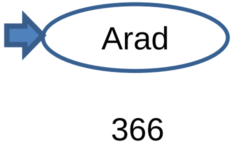
- Useful but potentially fallible (heuristic)
- Heuristic functions are problem-specific

Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example

a) The initial state

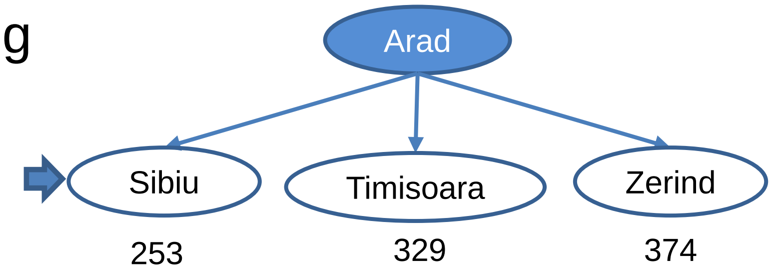


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example

b) After expanding Arad

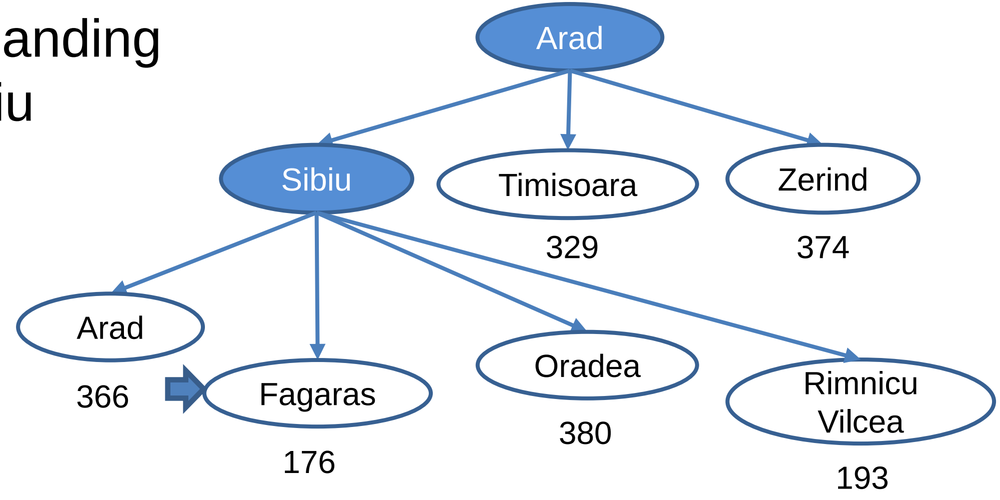


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example

c) After expanding Sibiu

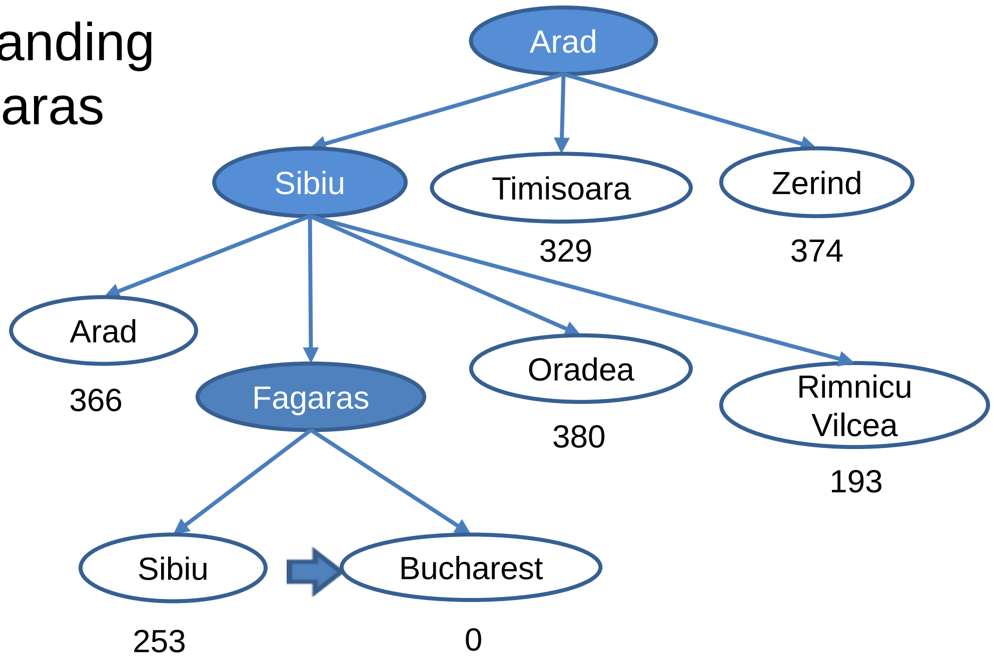


Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Example

d) After expanding Fagaras



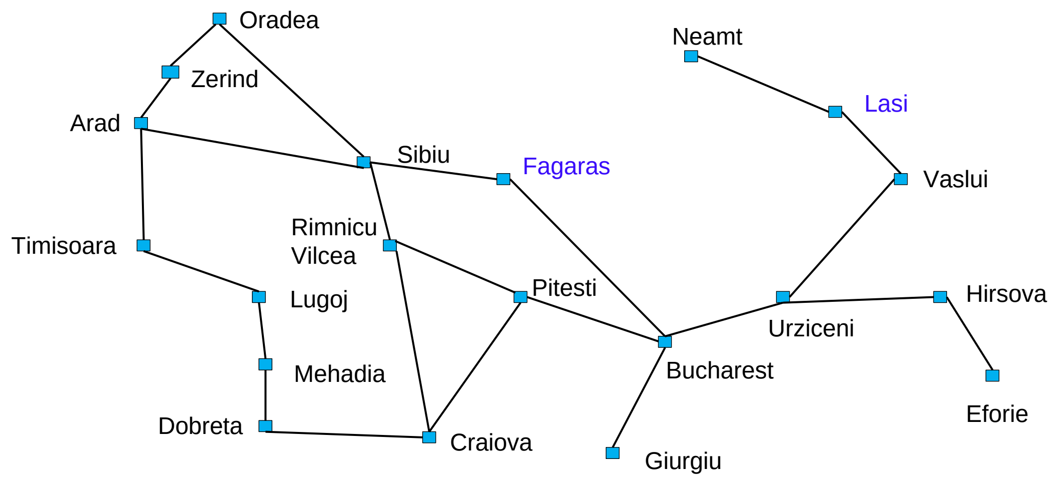
Straight-line distance to Bucharest

Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Efoire	161
Fagaras	176
Giurgiu	77
Hirsova	151
Lasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Complete?

Question: Is this approach complete?

Example: Find a path from Lasi to Fagaras



Answer: No

Greedy Search...

- m: maximum depth of the search space

Complete	No
Time	$O(b^m)$
Space	$O(b^m)$(keeps all nodes in memory)
Optimal	No

Question to think:

- Is it possible to combine functions g(n) and h(n) in one search strategy?

Yes, **A* Search** is the answer

A * Search

- Uniform-cost search
  - *g(n):* cost to reach n (Past Experience)
  - optimal and complete, but can be very inefficient
- Greedy search
  - *h(n):* cost from n to goal (Future Prediction)
  - neither optimal nor complete, but cuts search space considerably

A * Search

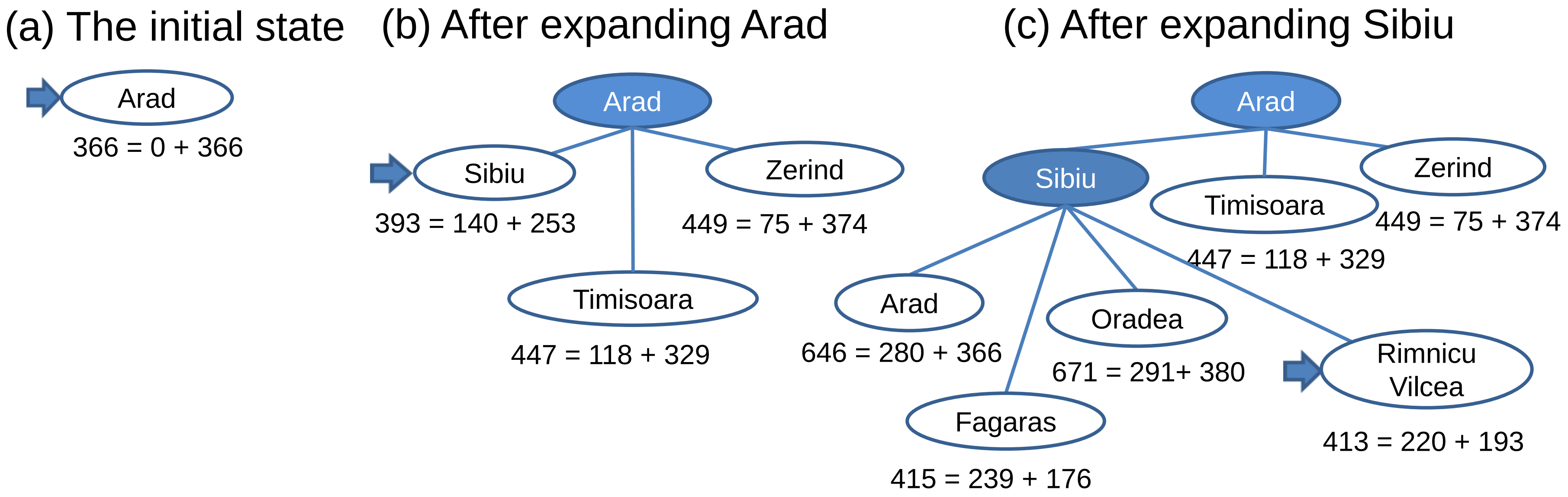
Idea: Combine Greedy search with Uniform-Cost search

Evaluation function: $f(n) = g(n) + h(n)$

- *f (n):* estimated total cost of path through *n* to goal (Whole Life)
- If g = 0 → greedy search; If h = 0 → uniform-cost search
- Function A* Search(problem) returns solution
  - Return Best-First-Search(problem, $g + h$)

Example: Route-finding from Arad to Bucharest

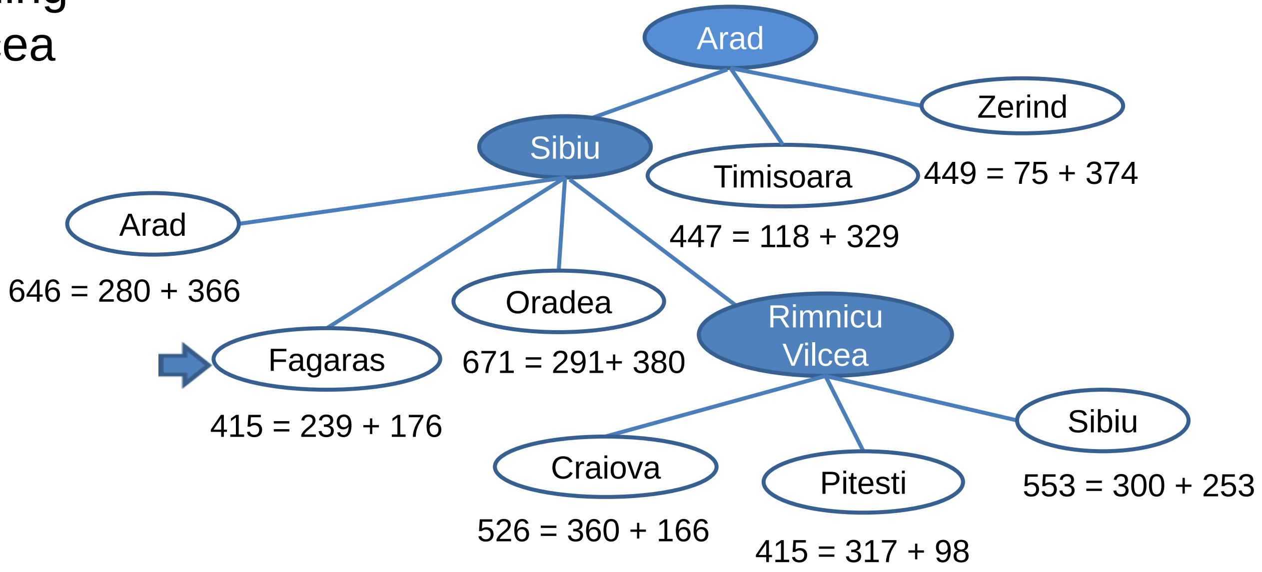
Best-first-search with evaluation function $g + h$



Example: Route-finding from Arad to Bucharest

Best-first-search with evaluation function $g + h$

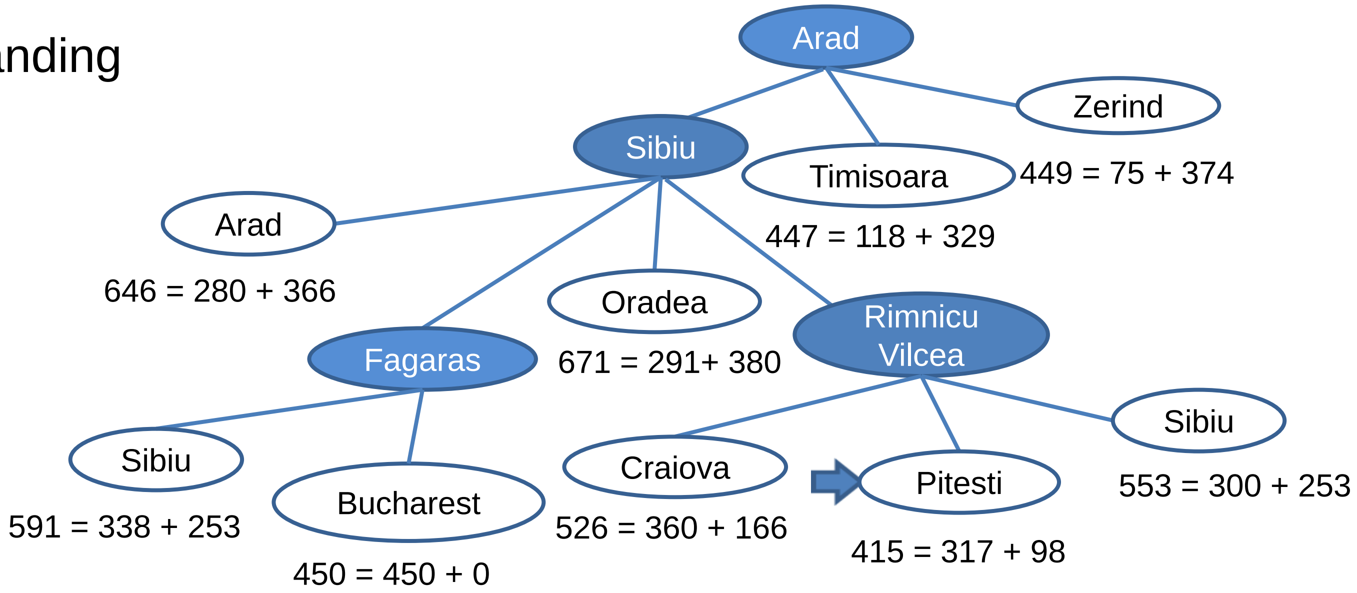
(d) After expanding
Rimnicu Vilcea



Example: Route-finding from Arad to Bucharest

Best-first-search with evaluation function $g + h$

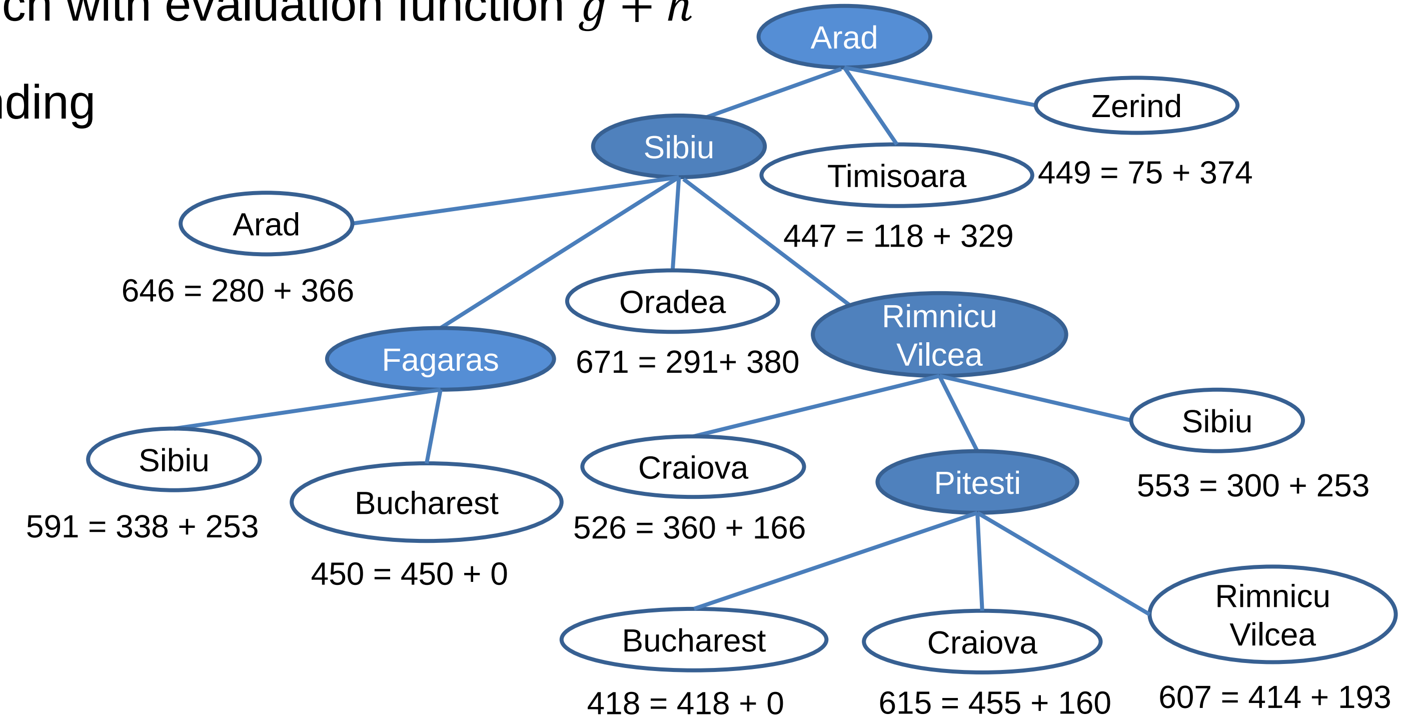
(e) After expanding Fagaras



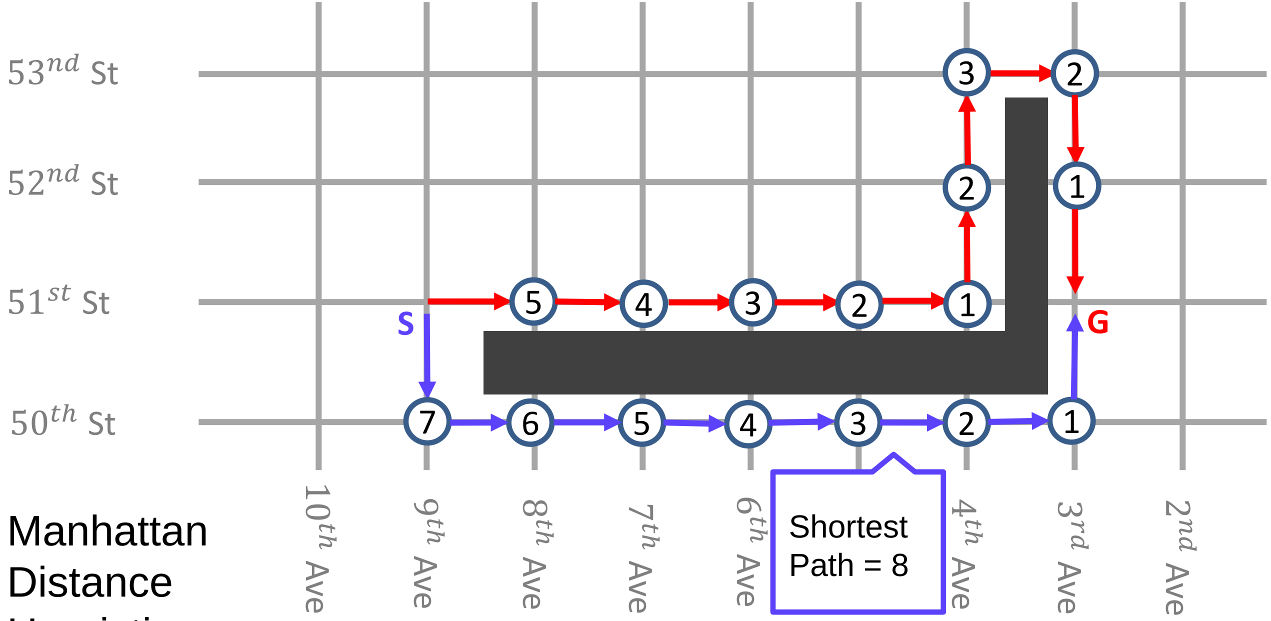
Example: Route-finding from Arad to Bucharest

Best-first-search with evaluation function $g + h$

(f) After expanding Pitesti

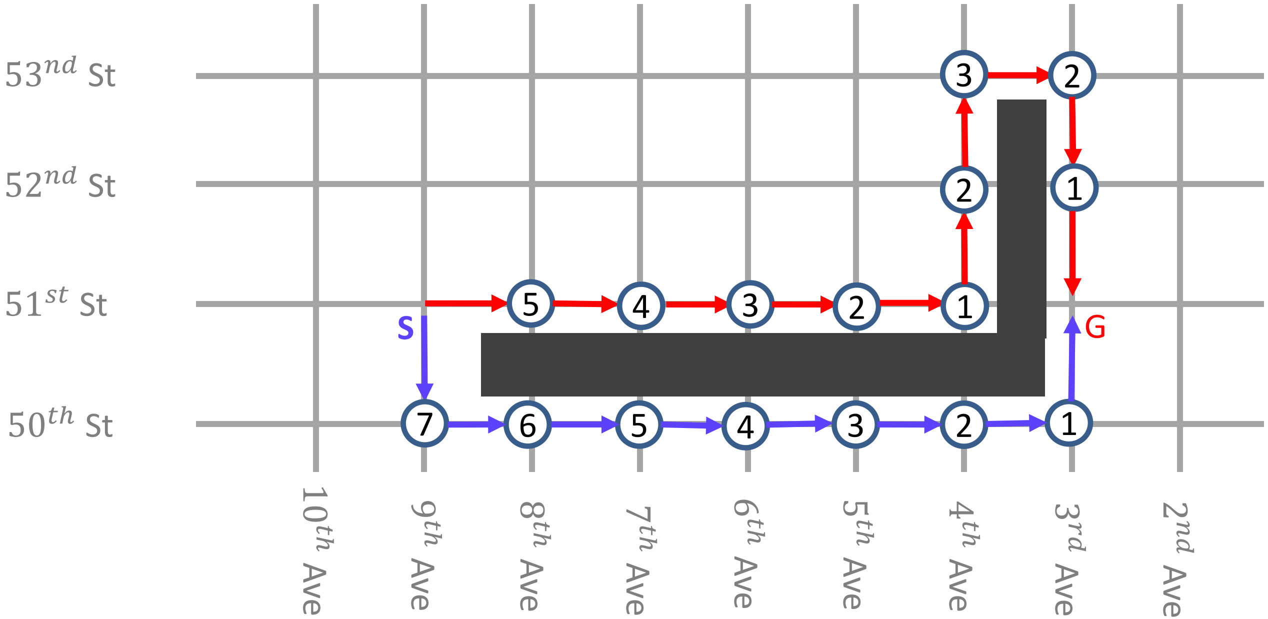


Example: Route-finding in Manhattan



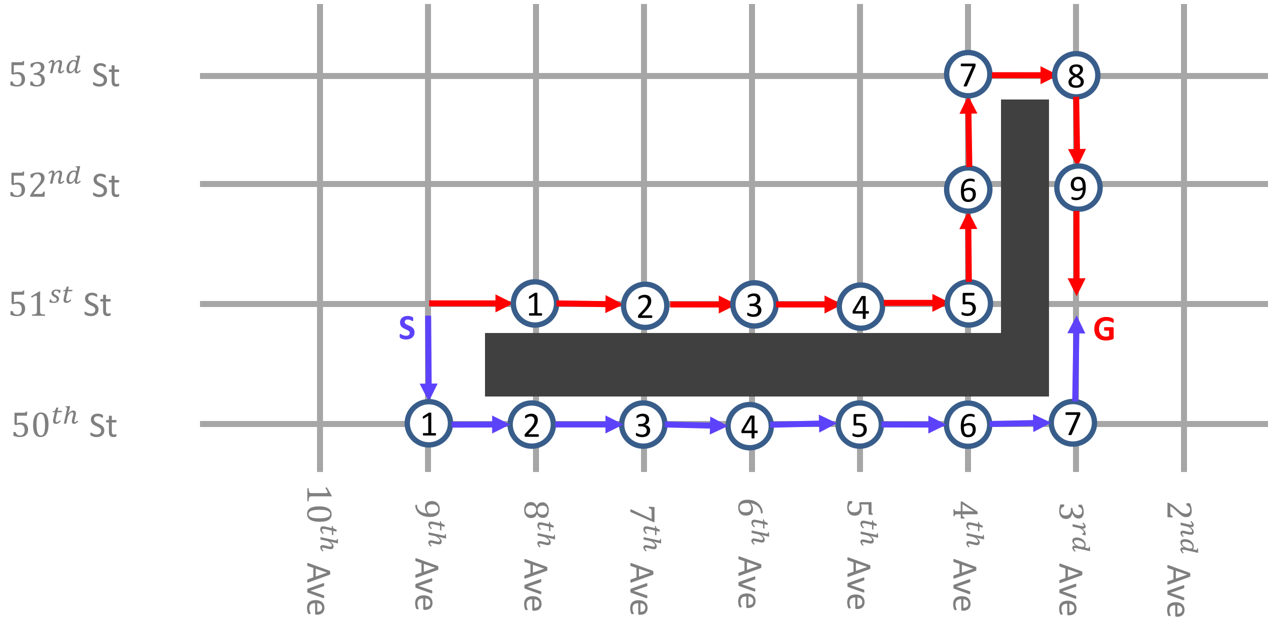
Manhattan Distance Heuristic

Example: Route-finding in Manhattan (Greedy)

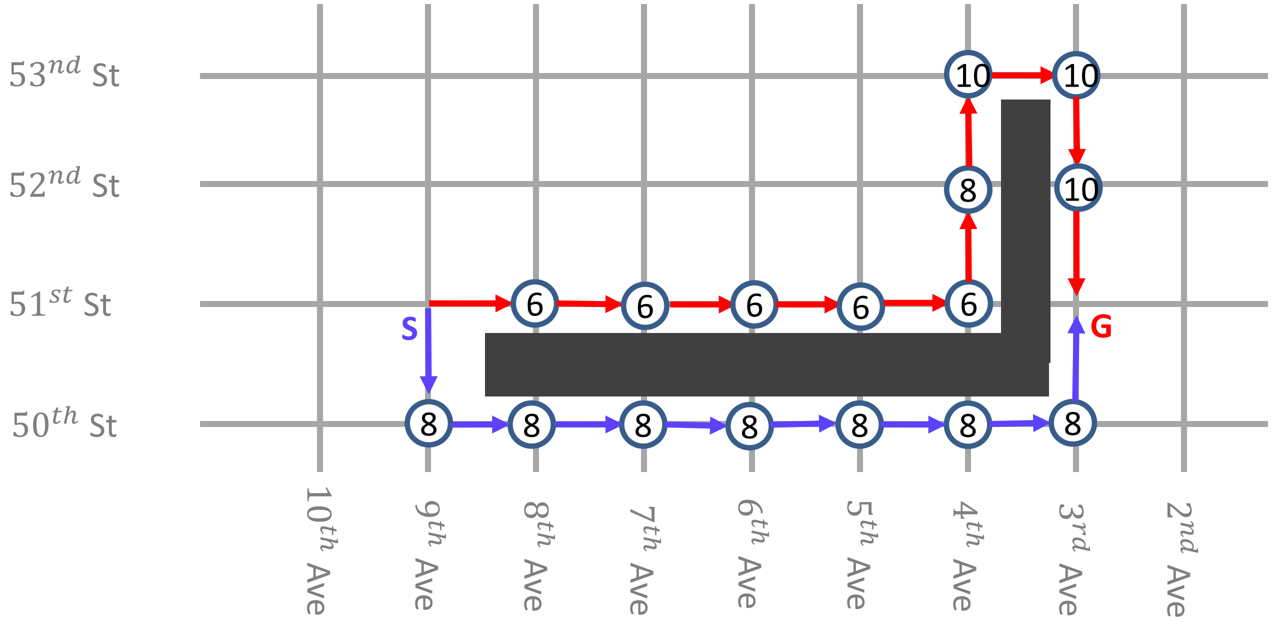


Here we do not expand notes that have been expanded.

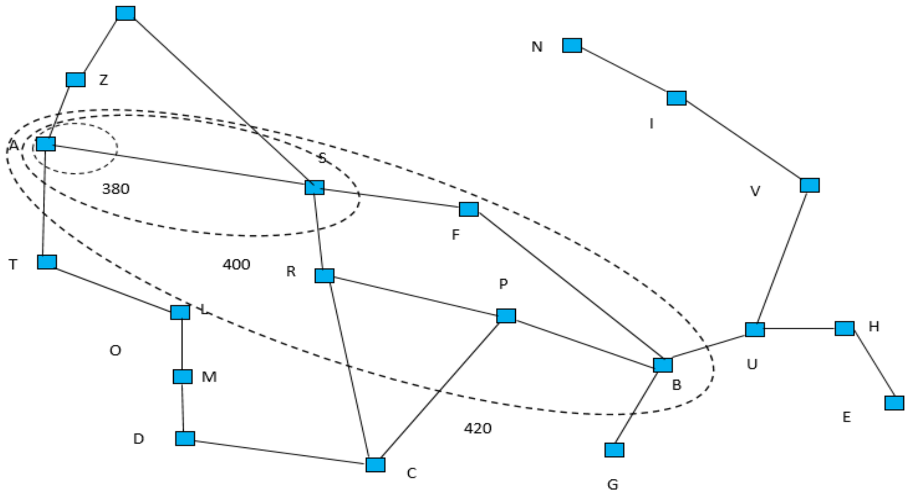
Example: Route-finding in Manhattan (UCS)



Example: Route-finding in Manhattan (A*)



Complexity of A*



Time	Exponential in length of solution
Space	(all generated nodes are kept in memory) Exponential in length of solution

With a good heuristic, significant savings are still possible compared to uninformed search methods

Summary (Recap)

Uninformed search strategies

- Use only the information available in the problem definition
  1. Breadth-first search
  2. Uniform-cost search
  3. Depth-first search
  4. Depth-limited search
  5. Iterative deepening search

Informed search strategies

- Use problem-specific knowledge to guide the search
- Usually more efficient
  1. Greedy Search
  2. A* Search

Summary (Recap)
- Uninformed Search Complexity

Criterion	Breadth-first	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Time	b^d	b^d	b^m	b^l	b^d	$b^{d/2}$
Space	b^d	b^d	bm	bl	bd	$b^{d/2}$
Optimal	Yes	Yes	No	No	Yes	Yes
Complete	Yes	Yes	No	Yes, if $l \geq d$	Yes	Yes

Summary (Recap)
- Informed Search Complexity

Criterion	Greedy Search	A* Search
Time	$O(b^m)$	Exponential in length of solution
Space	$O(b^m)$ (*keeps all nodes in memory*)	Exponential in length of solution (*all generated nodes are kept in memory*)
Optimal	No	Yes
Complete	No	Yes

Quick Briefing

LAB EXERCISE 7

Lab Exercise 7 (5%)

Exercise 7 : Problem & Environment Formulation (Group)

Attached Files: Lab Exercise 7 Sample.pdf (274.85 KB)

Submission is required for this Exercise. It is worth 5% of your grade.
Please check with your Lab Instructor or TA on how to submit your Solution.

Objectives:

In this exercise, you will formulate and design an agent of your choice by selecting a trivial environment such as a game or puzzle. Upon completion of this exercise, you should be able to :

1. Describe types of agents
2. Describe and differentiate key dimensions and types of environment for agent
3. Design a problem-solving agent by formulating the problem and its environment.

Exercise Problem: (Graded-5%)

- Prepare a report (.pdf) for the Design of Problem-Solving Agent with the outline below:
  - Provide an introduction for the selected topic (background)
  - Describe the type of agent you are modeling.
  - Identify and describe the type of environment you are modeling.
  - Define the selected problem.
  - Specify the key properties of the problem you are formulating.
  - Define the solution.
  - Provide a list of references.
- The report should not be more than **2 pages**.
- A sample of **Maze Runner/Solver** scenarios is provided for your reference.

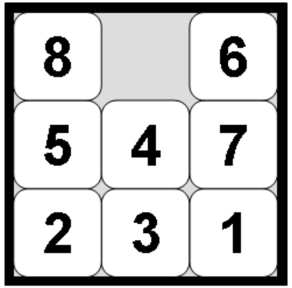
Lab Exercise 7 (5%)

IMPORTANT NOTE ON THIS LAB EXERCISE

1. This lab exercise is to be completed **in groups** (same members as the mini-project). - one submission per group
2. The deadline for this lab is **3 days** after the respective lab session (**including weekend & public holiday**).
3. Your group should not select topics such as **Maze Runner/Solver** as a sample has been provided.
4. Avoid **Tic-Tac-Toe** topic as we will discuss on that in Lab Exercise 8
5. The topic should not be too simple nor overly complicated as you will implement the agent with search tree in **Lab Exercise 8** later. The tree will typically represent the states of the agent's environment
6. Some guides/points for topic selection:
   - Look for some online mini-game websites to explore and select a game/puzzle.
   - the number of environment states should be between 20 - 120 states which can be constructed as a tree.
   - Choose a single-player game/puzzle rather than two or multi-player game/puzzle.
   - The action set should not be more than or around 10.
7. You **must attend the Graded Lab session (Lab Exercise 7)** to be eligible to earn the marks for your group work. If you are absent with valid reason, your team member will have to acknowledge that you have participated/contributed to the work and subsequently give consent to TA for them to award you the mark. There will be no make up lab.

Lab Exercise 7 (5%)

- Examples of General Topics



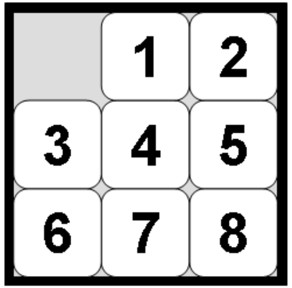


Image source: https://www.aiai.ed.ac.uk/~gwickler/eightpuzzle-uninf.html

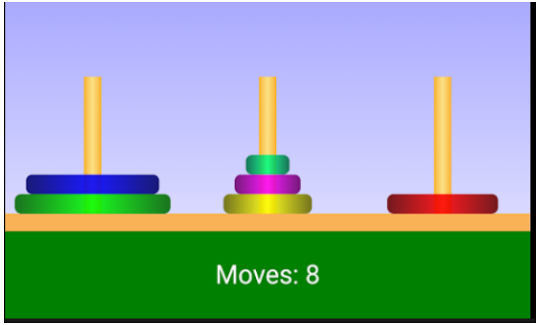


Image source: https://play.google.com/store/apps/details?id=johan.moller.towerofhanoi&hl=en_ZA

Lab Exercise 7 (5%)

- Examples of Slightly Advance Topics

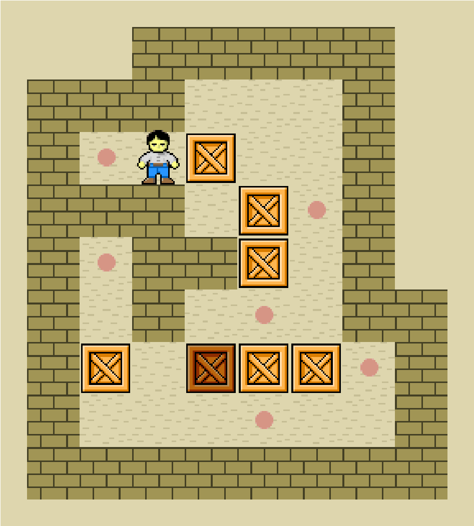
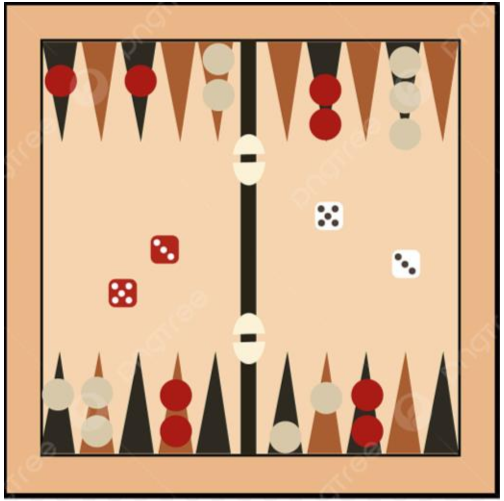


Image source: https://en.wikipedia.org/wiki/Sokoban



Note:
This is 2 players game

Image source: https://pngtree.com/freepng/cartoon-backgammon-board-element_5568324.html

Lab Exercise 7 (5%)

- Examples of Advance Topics

Image source:
https://www.instructables.com/Super-Mario-Odyssey-Question-Block-Box/

Will let your team to decide

Lab Exercise 7 (5%)

Criteria's	Need Improvement	Meet Expectation	Good	Excellent
Introduction / Background (0.5%)	With one line and a very brief introduction, the scenario is not clear.	general or common topic. scope might be too broad or overly simple	interesting topic (rare) with good references. Clear and precise/specific.	**N/A**
Type of Agent (0.5%)	Correct type of agent, but no further elaboration	Correct type of agent with some elaboration.	Correct type of agent with precise and detailed elaboration	**N/A**
Type of Environment (1%)	Correctly identify some but not all dimensions of the environment	Correctly identify all dimensions of environment but no descriptions provided	Correctly identify all dimensions of the environment but some descriptions are unclear/can be improved	Correctly identify all dimensions of the environment with clear descriptions/examples
Definition of selected problem (1%)	The problem definition is vague	problem definition is clear but the scope might be too broad or overly simple	Clear and precise/specific. Not too simple nor too complex	Clear and precise/specific Not too simple nor too complex With good and appropriate illustration
Specification (1.5%)	Correctly identify some but not all specification	Correctly describe all specifications without much elaboration.	Correctly describe all specifications but some parts are unclear (a bit hard to understand).	Correctly describe all specifications, clear, and effective.
Definition of Solution (0.5%)	The solution definition is vague	problem definition is acceptable/moderate but the scope might be too broad or overly simple	Clear and precise/specific. Not too simple nor too complex	**N/A**

Caution

- Try to minimize the reliance on ChatGPT when looking for ideas/solution
- Be kind to your teammates.
- Be responsible for your own learning.
- Tap on all available opportunities for your learning.
- Self-reflection is equally important along with marks.

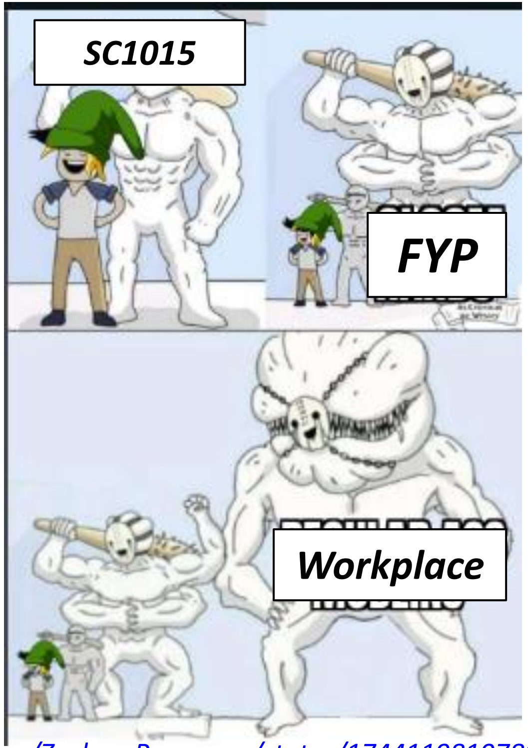


Image source: https://twitter.com/ZacharyPopowcer/status/1744119810789679281