CS2109S Tutorial 1

Problem Formulation and Uninformed Search

(AY 24/25 Semester 2)

January 31, 2025

(Prepared by Benson)

Join our telegram group!
https://t.me/+Q1NeQ2BSIchhMzll

Happy Chinese New Year and...

Welcome to the year of

Contents

Admin Info

- Complete (and submit) the pre-survey form (10/10).
- Join the telegram group (10/10).
  - Polls in telegram will be considered for EXP.
- Slides will be uploaded after every tutorial on the website. EXP will be disbursed on Coursemology (please check).
  - No recording except for today.

Introduction

YEUNG Man Tsung (Benson)

- Year 3 Computer Science
- **Email**: mtyeung@u.nus.edu (for official matters, e.g. absence for tutorial)
  mtyeung@comp.nus.edu.sg
- **Discord**: @mtyeung
- **Telegram**: @mtyeung

- Teaching CS2109S and CS3230 tutorials this semester.
- Consultations: Tuesday 1-3pm and Friday 3-5pm at COM1-0215 (please drop me a message before you come).

Introduction

YEUNG Man Tsung (Benson)

My Experience with AI/ML:

- Took CS2109S with the 28-hour practical final exam.
  Achievement: Still alive! (slept for 9 hours)
- Did a (research) internship on machine learning systems (the intersection of "machine learning" and "(distributed) systems").
  Achievement: Published a paper!
- Active Copilot (a code completion tool) user.
  Achievement: Saved lots of time! (highly recommended)

About CS2109S

Tutorial 1-3:
Classical AI

- Search algorithms (CS2040S++) in **state spaces**.
- Some optimality proofs (CS1231S comes in handy).

Tutorial 4-10:
Modern ML

- Lots of math (linear algebra and calculus).
- Lectures & Tutorials: Helps you understand "what you are doing"
  + Problem Sets: Implementation (libraries).
- Some expectations:
  - We'll not teach you how to implement/debug.
  - Math will be tested in finals.

About CS2109S Tutorials

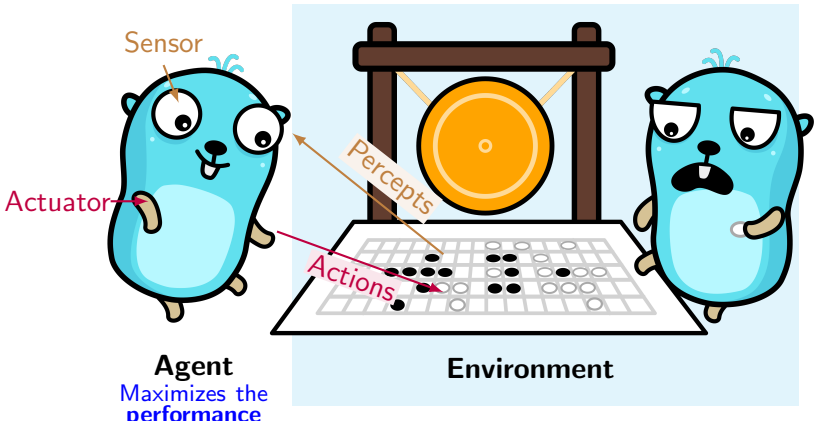
Format:

- We will discuss the tutorial questions. **You are expected to attempt them beforehand** *(to participate in class)*.
- We'll also do lecture recap & (non-anonymous) polls along the way.

EXP Policy:

EXP (Max 500)	Item	
300	Attendance	} Free
+200	Participation & Polls	} Tiny differentiation
+100	Bonus Question (if any)	} *(You'll get 0 / 100 / 200)*

Recap: PEAS


Sensor
Actuator
Percepts
Actions
Agent
Maximizes the
performance
Environment

Q1. PEAS for AI Chess Opponent

Determine the PEAS (Performance measure, Environment, Actuators, Sensors) for an AI chess opponent in a mobile chess application.

Performance Measure:
Make "human" moves (ability to emulate human players with certain rankings (ELO)).
Win, draw, lose the game.

Environment:
Chess board, position of game pieces, rules of the game, mobile operating system, human user's ranking (ELO), human user's playing style.

Actuators:
Visual output to move a chess piece, or resign.

Sensors:
Touch screen interactions to detect the change in position of pieces.

Recap: Different Types of Search

Why learn everything again?

Search in CS2040(C/S)

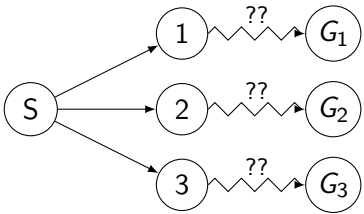
- Searching in *graphs*.
- Entire graph given as input (usually "small").
- Finds the shortest path to all vertices (single-source / all-pairs).

Search in CS2109S

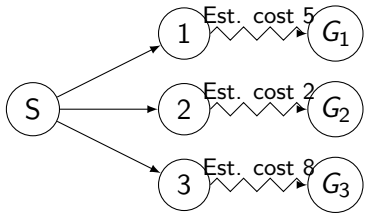
- Searching in **state spaces**, i.e. modeling real life problems with states (vertices) & transitions (edges).
- We don't know the size of the graph (can be infinite). *Lazily* evaluated.
- Only interested in reaching the **goal states**. (We don't know which.)
- Sometimes, we are satisfied with "good enough" solutions.

Recap: Different Types of Search

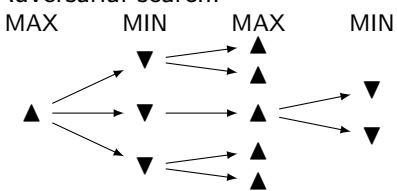
Uninformed search:



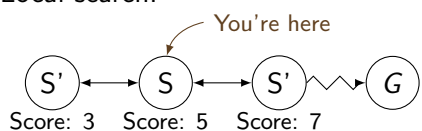
Informed search:



Adversarial search:



Local search:



Q2. Tower of Hanoi

The objective is to move all the disks from L to R in the *minimum number of moves*.

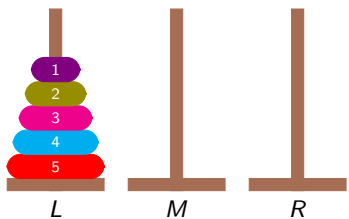
(a) Give the state representation.

(b) Describe the representation invariant of your state. Compare the total number of possible configurations in the actual problem to that of your chosen state representation, with and without the representation invariant.

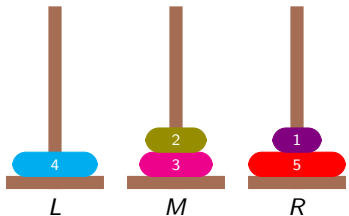
(c) Specify the initial and goal states.

(d) Define the actions.

(e) State the transition function T.



Q2. Tower of Hanoi



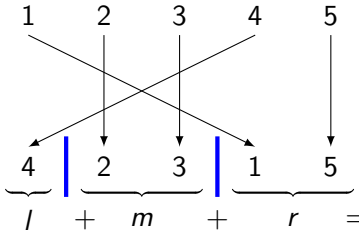
State Representation:

- Three tuples (L, M, R): $((4), (2, 3), (1, 5))$
  (Invariant: Each tuple must be in *ascending order*)
- The assignment for each disk: (R, M, M, L, R)

Q2. Tower of Hanoi

How many possible states are there?

- Three tuples (L, M, R): $((4), (2, 3), (1, 5))$ **without the invariant**.



$n!$ ways

CS1231/S Refresher: Number of **non-negative** integer solutions to $x_1 + x_2 + \ldots + x_k = n$ is $\binom{n+k-1}{k-1}$.

$\binom{n+2}{2}$ ways

  - Number of states $= n! \times \binom{n+2}{2} = \frac{(n+2)!}{2}$.

- Three tuples **with the invariant**: Each tuple must be in *ascending order*.

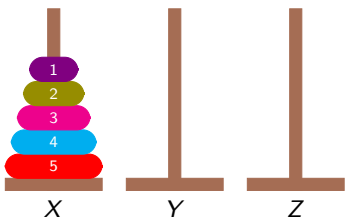
$1_R \quad 2_M \quad 3_M \quad 4_L \quad 5_R$ $\qquad 3^n$ ways

  - Number of states $= 3^n$.

Q2. Tower of Hanoi

What are the initial and goal states?

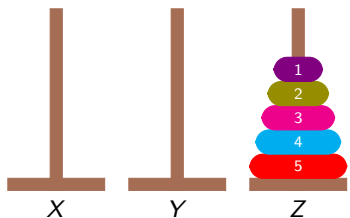
Initial state:



$((1, 2, \ldots, n), (), ())$

$(X, X, \ldots, X)$

Goal state:



$((), (), (1, 2, \ldots, n))$

$(Z, Z, \ldots, Z)$

Q2. Tower of Hanoi

Branching factor = 3

Actions and transition functions:

- Transfer a disk from L to M:

$$T(((l_1,\ldots),(m_1,\ldots),(\ldots)),a_1) \to ((\ldots),(l_1,m_1,\ldots),(\ldots)) \quad l_1 < m_1$$
$$T(((l_1,\ldots),(),(\ldots)),a_1) \to ((\ldots),(l_1),(\ldots))$$

- Transfer a disk from M to L:

$$T(((l_1,\ldots),(m_1,\ldots),(\ldots)),a_2) \to ((m_1,l_1,\ldots),(\ldots),(\ldots)) \quad m_1 < l_1$$
$$T((),(m_1,\ldots),(\ldots)),a_2) \to ((m_1),(\ldots),(\ldots))$$

} 1 valid action

- Transfer a disk from L to R:

$$T(((l_1,\ldots),(\ldots),(r_1,\ldots)),a_3) \to ((\ldots),(\ldots),(l_1,r_1,\ldots)) \quad l_1 < r_1$$
$$T(((l_1,\ldots),(\ldots),()),a_3) \to ((\ldots),(),(l_1))$$

- Transfer a disk from R to L:

$$T(((l_1,\ldots),(\ldots),(r_1,\ldots)),a_4) \to ((r_1,l_1,\ldots),(\ldots),(\ldots)) \quad r_1 < l_1$$
$$T((),(\ldots),(r_1,\ldots)),a_4) \to ((r_1),(\ldots),(\ldots))$$

} 1 valid action

- Transfer a disk from M to R:

$$T(((\ldots),(m_1,\ldots),(r_1,\ldots)),a_5) \to ((\ldots),(\ldots),(m_1,r_1,\ldots)) \quad m_1 < r_1$$
$$T(((\ldots),(m_1,\ldots),()),a_5) \to ((\ldots),(),(m_1))$$

- Transfer a disk from R to M:

$$T(((\ldots),(m_1,\ldots),(r_1,\ldots)),a_6) \to ((\ldots),(r_1,m_1,\ldots),(\ldots)) \quad r_1 < m_1$$
$$T(((\ldots),(),(r_1,\ldots)),a_6) \to ((\ldots),(r_1),(\ldots))$$

} 1 valid action

Q2. Tower of Hanoi

Bonus: Which search algorithm would be most appropriate for finding the solution?

- Branching factor $b = 3$.
- Depth $d = 2^n - 1$. $(T(n) = T(n-1) + 1 + T(n-1))$

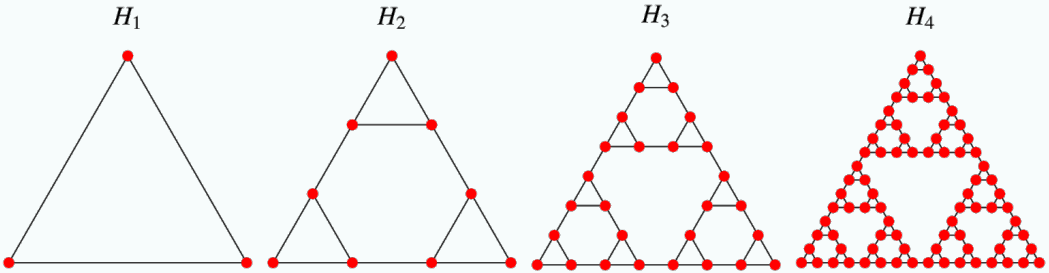
Search	Time	Space
Breadth-First Search (BFS)	$O(b^d) = O(3^{2^n-1})$	$O(b^d) = O(3^{2^n-1})$
Uniform Cost Search (UCS)	Same as BFS, all costs are 1.	
Depth-First Search (DFS)	∞, can get trapped.	
Depth-Limited Search (DLS)	$O(b^d) = O(3^{2^n-1})$	$O(bd) = O(2^n)$
Iterative Deepening Search (IDS)	We already know the depth of the solution d.	

DLS is the most suitable algorithm.

Q2. Tower of Hanoi

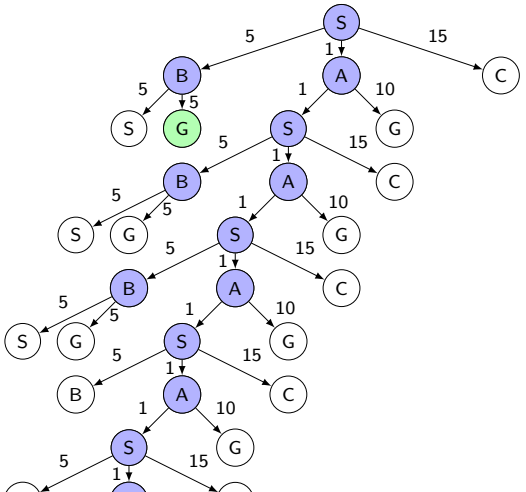
Extra Slide

The state graph corresponding to the Tower of Hanoi puzzle is called the **Hanoi graph**, which has a beautiful recursive pattern.

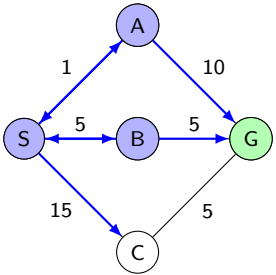


Q3. Uniform-Cost Search Analysis

(a) Give a trace of uniform-cost search using `SEARCH` (without visited memory).



frontier
S(0)
A(1)
S(2)
A(3)
S(4)
A(5)
B(5)
S(6)
A(7)
B(7)
S(8)
A(9)
B(9)
G(10)
S(10)
S(10)
B(11)
G(11)
G(12)
S(12)

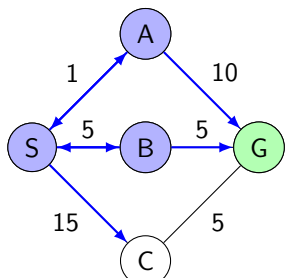


Q3. Uniform-Cost Search Analysis

(b) Give a trace of uniform-cost search using `SEARCH WITH VISITED MEMORY`.

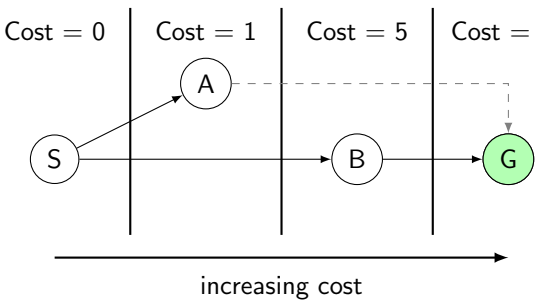
frontier
S(0)
A(1)
B(5)
G(10)
G(11)
C(15)

- Push S into frontier.
- Visit S. Push A, B, C into the frontier.
- Visit A. S has been visited, thus we only push G into the frontier.
- Visit B. S has been visited. The distance to G via B is $5 + 5 = 10 < 11$, so we update the distance to G to 10.
- Visit G. We're done!



Q3. Uniform-Cost Search Analysis

(c) When A generates G, the goal state, with a path cost of 11 in (b), why doesn't the algorithm halt and return the search result since the goal has been found?
With your observation, discuss how uniform-cost search ensures that the shortest path solution is selected.



✔ From states with cost < 10: already considered

$\Rightarrow$ no more shorter paths

✔ From states with cost ≥ 10:

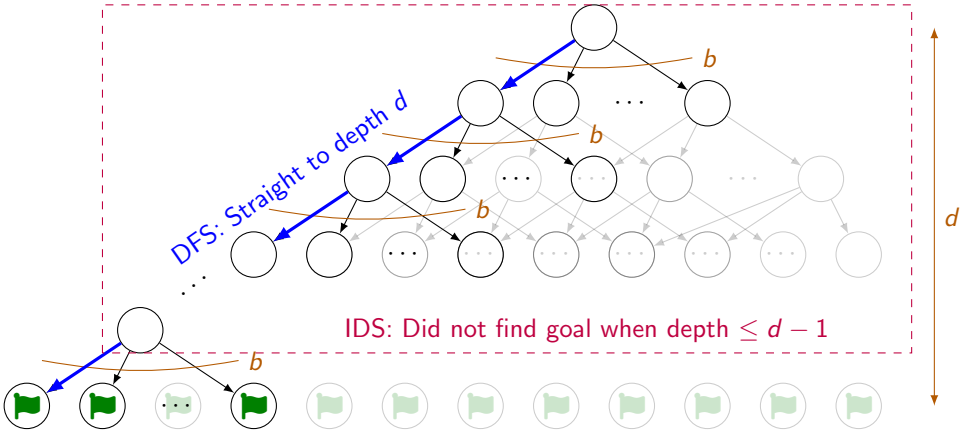
Cost $= (\geq 10) + (\geq 0) \geq 10$

assume non-negative costs

no better

Q4. Comparison of Search Strategies

(a) Describe a problem in which iterative deepening search performs much worse than depth-first search.



Q4. Comparison of Search Strategies

How many search nodes are generated by DFS?

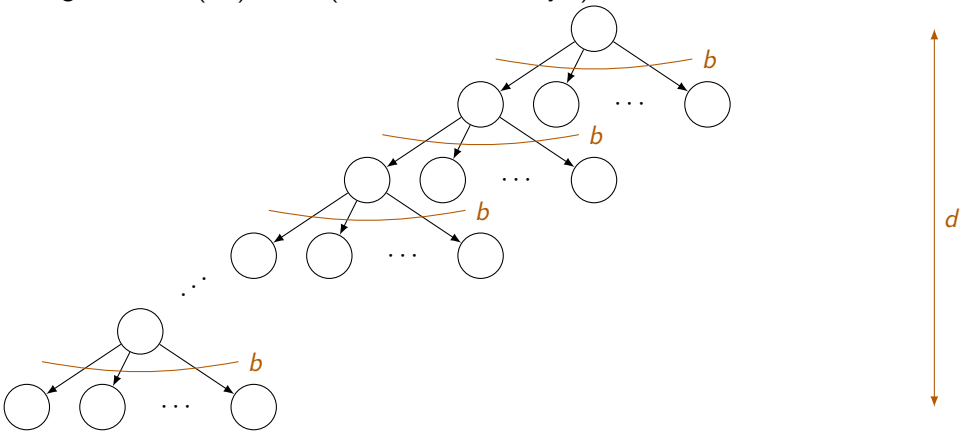
A. $O(b)$
B. $O(d)$
C. $O(bd)$
D. $O(b^{d-1})$
E. $O(b^d)$

How many search nodes are generated by IDS?

A. $O(b)$
B. $O(d)$
C. $O(bd)$
D. $O(b^{d-1})$
E. $O(b^d)$

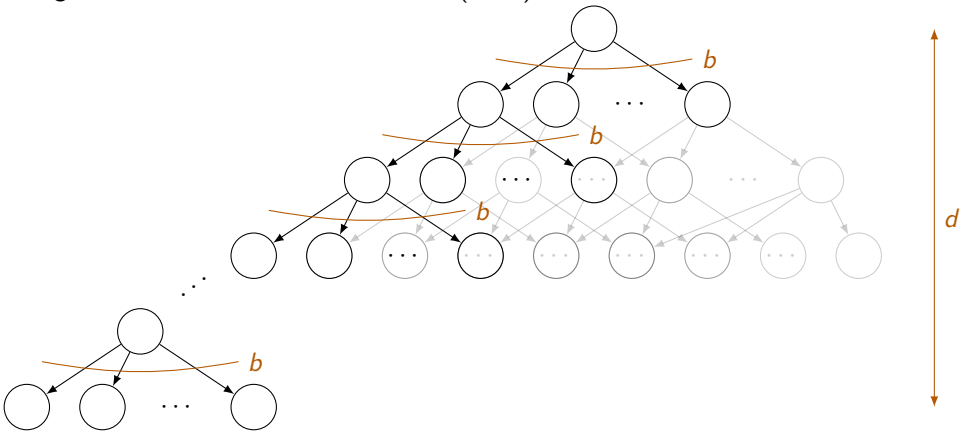
Q4. Comparison of Search Strategies

DFS generates $O(bd)$ nodes (b nodes on each layer).



Q4. Comparison of Search Strategies

IDS generates $b^0 + b^1 + \cdots + b^{d-1} = O(b^{d-1})$ nodes.



Q4. Comparison of Search Strategies

(b) Describe a problem where `SEARCH` performs much better than `SEARCH WITH VISITED MEMORY`.

- When do we not need visited memory? When there are no repeated states!
- Memory usage of `SEARCH`: $O(bd)$.
- Memory usage of `SEARCH WITH VISITED MEMORY`: $O(b^d)$.

} For DFS/DLS

Feedback

Anonymous Feedback (throughout the semester):

https://forms.gle/4ayi7Wmg9AmzYvbaA

The link to the tutorial slides will be posted in telegram.
Please do not share the slides with other classes before all tutorial classes in the week are over.