CS3243 Tutorial 1

Eric Han (TG4, TG5)

Aug 23, 2022

Introduction - Eric Han

Singaporean, Final Year PhD Student

- [Pioneer JC 2009-2010] Took 'A' levels and fell in love with Computing
  - H2 Computing, Interested in research
- [B.Com. NUS 2013-2018] Not so long ago I was in your seat
  - A*STAR Scholarship, Turing Programme
  - [University of Southern California, 2016] Student Exchange
- [NUS 2018-2023] And now, I am taking a PhD. in Com. Sci.
  - My research is in AI/Machine Learning regarding scaling and robustness.
  - Some of the courses I taught: CS3217(1), CS3243(1), CS3203(5), CS2030(1)
  - Teaching this course is coming full circle for me, to teach the next generation.

You are welcome to check my profile & research: https://eric-han.com.

Likely (highly) my last semester teaching; which will mean its going to be the best.

Expectations / Commitment

Expectations of you

1. Fill seats from the front.
2. Good students are always prepared.
   1. Attempt your Tutorial
   2. Review lecture content
   3. Be on time
3. Refrain from taking pictures of the slides.
   1. Learn to take good notes.
   2. Slides will be distributed, but delayed.

Commitment from me

1. Be avaliable for your learning as much as possible.
2. Strive to make the lessons interesting and fun.

Any comments or suggestions for the lessons welcome!

Administrative

- Plagiarism - Tutorial Assignments are **individual** work.
- Tutorial attendance will be recorded and factored into your Assignments grade.
- In case if you cannot make it for tutorial (for any valid reason); makeup:
  - Attend TG4/TG5 interchangeably (Don't need inform me; I teach both)
  - Attend other TG (Inform me; Let me know which)
- If you still cannot make it (Send me an email with valid reason with proof)
- Consultations are avaliable in 1hr slots (Telegram/Email me)
  - Tuesday 1-4pm
- Any questions always ask in our chat group first, then PM me.

Telegram me: `@Eric_Vader` ; chat about module, research, sch etc…

Email: `eric_han@nus.edu.sg`

Annoucements

Important admin:

1. Join TG4/5 Telegram Group
   - https://t.me/+q74TDVvov3tiMjZl
2. We will be taking attendance via telegram, so fill in this Google Form Survey!
   - https://forms.gle/4p9hdGST9LyoyqJe6

Figure 1: TG4/5 Telegram Group

Figure 2: Survey

Question 1 - Recap

- **Fully / Partially Observable**: Is the complete state of the environment accessible to the agent's sensors
- **Single / Multi-Agent**: Are there more than one actor in the environment? (competitive vs cooperative)
- **Deterministic / Stochastic**: Is the next state determined by the current state and action by the agent?
- **Episodic / Sequential**: Is the next episode dependent on the action taken previously?
- **Static / Dynamic**: Can the environment change while the agent is deliberating?
- **Discrete / Continuous**: Is the state of the environment discretized or varying continuously?
- [**Known / Unknown**] Are the rules of the game known to the agent?

Easiest: Fully, Single, Deterministic, Episodic, Static

Question 1a

Determine the properties of the above problem from the perspective of an intelligent agent planning a solution. Complete the table below.

Environment Characteristic	Sudoku Puzzle Generation
Fully / Partially Observable	
Single / Multi-Agent	
Deterministic / Stochastic	
Episodic / Sequential	
Static / Dynamic	
Discrete / Continuous	

Question 1a - Answer

Environment Characteristic	Sudoku Puzzle Generation
Fully / Partially Observable	Fully Observable
Single / Multi-Agent	Single agent
Deterministic / Stochastic	Deterministic
Episodic / Sequential	Episodic / Sequential
Static / Dynamic	Static
Discrete / Continuous	Discrete

Explaination on Episodic / Sequential: Depending on the environment formulation, the next episode is dependent on the previous action.

Question 1b

Define the search space for the problem of generating a Sudoku Puzzle by completing the following.

Recap

- What are the 5 parts of the environment formulation?

Question 1b - Answer

Sample Environment Formulation:

- **State Space**: $A \in \{0, \cdots, 9\}^{9\times 9}$ where 0 is blank
- **Initial State**: A valid Sudoku Puzzle, completely filled without 0.
- **Final State**: A goal state is $T \in \{0, 1, \cdots\}$ steps away from the inital state; Also goal check to make sure it can be solved in one way.
- **Action**: Removing the value $a \in \{1, \cdots, 9\}$ at (i, j) in A
- **Transition Model**: $A - E_{i,j}(a)$, where $E_{i,j}(a)$ is zeros everywhere but a at (i, j)

Sample: There is no right answer, there are several correct representations; as long as requirements are fulfilled. Some representations are better than others - Compute complexity of Transition, Number of states, etc…

Question 2a

Describe the difference between Tree Search and Graph Search algroithms.

Recap

- AIMA Chapter 3, on graph search and tree-like search.

. . .

Answer

Graph search will not explore redundant paths, only exploring:

1. unvisited states
2. visited states, but via less than optimal paths

Tree search will explore all paths, including redundant paths.

Question 2b

i. Depth-First Search with tree-based implementation
ii. Depth-First Search with graph-based implementation
iii. Breadth-First Search with tree-based implementation
iv. Breadth-First Search with graph-based implementation

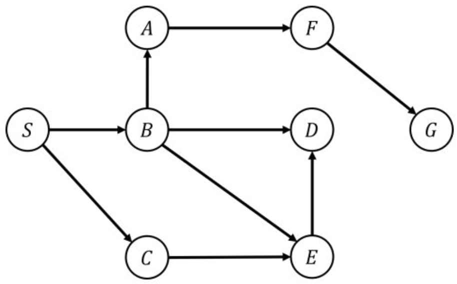

Figure 3: Graph for Q2b

Question 2b

Answer

i. `S-C-E-D-B-E-D-D-A-F-G`
ii. `S-C-E-D-B-A-F-G`
iii. `S-B-C-A-D-E-E-F-D-D-G`
iv. `S-B-C-A-D-E-F-G`

Bonus Qn: Which implementation is typically used in practice?

- BFS: tree or graph?
- DFS: tree or graph?

Question 3

Prove that the **Uniform-Cost Search** algorithm is optimal as long as each action cost exceeds some small positive constant ϵ.

Recap

- AIMA Chapter 3, on uniform search
- How does UCS differ from BFS?
- Proof by Induction
- What is an invariant condition?

Question 3 - Answer

Intuition: For each visited node, $g(u)$ is the least cost from the source s to u.

Given some small positive cost $\ell(a, b) > \epsilon$ between 2 nodes a and b,

- **Base**: When there is one visited node, its tivially true.
- **Inductive**: Assuming for all $u \in U$ visited nodes, $g(u)$ is with the least cost.
  - For any V frontier nodes that is unvisited, we choose v where $\ell(u, v)$ is smallest
  - The cost to v is $g(v) = g(u) + \ell(u, v)$; we proof that it is the least.
    - Assume that $g(v)$ is not the least cost,
    - then there will be a shorter path through some other node w.
    - If w is visited then $g(u)$ must go through w, which contradicts.
    - If w is unvisited[1], then $\ell(u, w) < \ell(u, v)$ which contradicts as w should be chosen first.

Hence, if cost $\ell(a, b) > \epsilon$, $g(.)$ is the optimal cost to the source.

Question 4

Formulate the above as a search problem. More specifically, define the following:

- State representation
- Initial state

[1] Also consider the case where w is not immediately reachable; highly similar to the case here.

- Actions
- Transition model
- Step cost
- Goal test

Recap

- Question 1b,
- but in some different words.

Question 4 - Answer

There is no right answer, but minimally:

- **State Space (Representation)**: describes how pieces are connected to the neighbours
- **Initial State**: representation varies but must not have any inital connections.
- **Final State**: checks must consider pieces with 2/3/4 sides are correctly connected.
- **Action**: considers that legal connections, ie. you cannot simply take any two puzzle pieces.
- **Transition Model**:
  - Correctly mutates the current state that maintains semantics.
  - **Step cost**: step cost is any positive, non-zero number.

Absolute positioning is acceptable, but the solution *must* consider orientation.

Question 5

Assignment Question, due on Sunday.

Bonus Question - Work for Snack

To help you further your understanding, not compulsory.

Tasks

1. Fork the repository https://github.com/eric-vader/CS3243-2223s1-bonus
2. We will be first solving Question 2b using code; DFS is already implemented, so
   1. Implement BFS, both tree and graph variants.
3. Now we explore the difference between late and early goal test; For early goal test:
   1. Implement DFS, both tree and graph variants.
   2. Implement BFS, both tree and graph variants.

To claim your snack, show me your forked repository and your code's output.

Recap

- Early Goal Test - Goal test on pushing to frontier instead of popping from frontier.
- AIMA Python Implementation - https://github.com/aimacode/aima-python.