NUS | Computing
National University of Singapore

Eric Han
eric_han@nus.edu.sg
https://eric-han.com

Computer Science

T01 – 29 Aug 2024

Tutorial 1

CS2109s TG35,36

Content I

Section 1: **Introduction**

Your Tutor Dr Eric Han

- [Pioneer JC 2009-2010] Took ‘A’ levels and fell in love with Computing
  - H2 Computing, Interested in research and in AI.
- [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
- [PhD. NUS 2018-2023] PhD in AI/ML
  - My research is in AI/Machine Learning regarding scaling and robustness.
  - Some of the courses I taught: CS2109S(2), CS3217(1), CS3243(2), CS3203(5), CS2030(1)
  - Teaching this course is coming full circle for me, to teach the next generation.
- Applied for jobs and got 2 - Industry and another in NUS SoC as Lecturer
  - Teaching this semester: CS1010(1), CS2109S(1), CS3244(1)

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

Expectations / Commitment

Expectations of you

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

Commitment from me

1. Be available for your learning as much as possible.
2. Strive to make the lessons interesting and fun.
3. Pass on a good foundation in AI/ML (not just the `A+`).

Any comments or suggestions for the lessons welcome!

Administrative

- Plagiarism - Passing work or ideas as your own w/o full ack/consent.
  - Tutorial / Problem Sets are **individual** work.
  - About ChatGPT - Use ChatGPT responsibly: Acceptable/Not Acceptable.
- Absent for tutorial - Fill in https://forms.gle/EaqM6D8iEmtGg4UT7.
- Consultations are available in 1hr slots, Arrange on Telegram
- Keep slides/notes within this class: https://www.eric-han.com/teaching
- Bonus / [@] Questions
  - Show / Answer to me in class
  - Select `Yes` to the qns when filling in the Buddies/Attendance Form.
- Questions?
  - Ask ChatGPT to teach you
  - Ask on our group
  - Telegram `@Eric_Vader` or Email `eric_han@nus.edu.sg`

Transparent Grading

Tutorial EXP Rubrics (Adapted from Module Policy)

EXP	Item
+200	Attendance
+50	Attempted Tutorial (Not all Qns)
+50	Completed Tutorial (All Qns - attempt is good enough)
+50	Active Discussion (Contribute to Group/Buddy [G] Discussion)
+50	Active Participation (Contribute to Class Discussion)
+50	Bonus (Completed Bonus Qn, Awarded by Eric)
+50	Answer a [@] question in class fully (Awarded by Eric)

How it works:

- $s = \min(\text{buddy}, \text{self})$: Minimum of your declaration, buddy declaration.
- Round s to the closest $\{200, 300, 400, 450, 500\}$ EXP.

Comments from students: Other tutors simply give out 400 EXP.

Asking for Help

- ChatGPT: Appropriate use of LLMs is *encouraged* but you must declare it:
  - OK as a search engine: What is np.reduce?
  - Ok as a personal tutor: How can I debug...
  - Not OK: Code a function that (1) Do not use numpy...
  - Not OK: Help me debug...
- You can consult your buddy for help:
  - OK: Hey do u know what the qn means by dont use iterative?
  - Not OK: Yo can send me your code?
- You can ask on our Tutorial chat
  - OK: Hey anyone got problem with PS1.1?
  - Not OK: Can someone help me debug...
  - Not OK: This is my answer...
- You can PM me
  - OK: Hey Eric, I am having problems with np.reduce, how ...
  - Not OK: Hey, this is my code...

Buddy System

- Everyone will be paired with a buddy (in pairs)
  - He/She will be your buddy for this class/sem
  - If there are odd numbers then we will have one group of 3
- Buddy/You will be responsible for taking your attendance/rating
  - Rate yourself
  - Rate your buddy
  - Help your buddy
- Learning is **social** and I hope that we are able to build friendships in this class.
- The buddy must declare cheating and plagrism when there is suspect of that
  - Please email me at `eric_han@nus.edu.sg` with proof
  - I place a heavy penalty on Plagiarism/Cheating
  - I will recommend max penalty to the school

Annoucements

Important admin:

1. Join our Telegram Group: https://t.me/+jWEjTd9W5Ug0NmE1
2. Welcome Survey (also attendance): https://forms.gle/cAQSeYvqi4yzrfkD6
3. Take Attendance for your buddy: Will be flashed at the end.

Figure 1: Our Telegram Group

Figure 2: Survey

Section 2: **PEAS for SIRI Function**

Question 1

Determine the PEAS for the SIRI function of iPhones.

Recap

- Explain PEAS and why is it used?

Question 1

Determine the PEAS for the SIRI function of iPhones.

Recap

- Explain PEAS and why is it used?

PEAS:

- **Performance measure**: How good or bad?
- **Environment**: External context
- **Actuators**: Output/Actions
- **Sensors**: Input

Answer

PEAS	Description
Performance measure	Correctly processes users' voice into text based on different language with possible accents. Make correct action with acceptable reaction speed.
Environment	iOS device with a functional audio input system. Internet connection and supported language voice.
Actuators	Speaker or/ and visual output to show answers to user.
Sensors	Speech Listener/ Microphone. Touchscreen Interactions.

Section 3: **Tower of Hanoi Problem**

Question 1 [G]

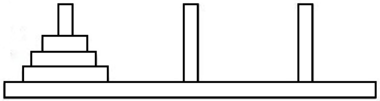

Figure 3: Tower of Hanoi with $n = 3$ disks

Tower of Hanoi is a game with 3 (L,M,R) pegs and a set of n disks with ascending (different) sizes. The aim is to move all disks from the L to the R peg with the minimum moves, considering:

- Move one disk at a time, from one peg to the other.
- Larger disks cannot be placed ontop of smaller disks.

Discuss its environment formulation.

Recap: What are the 5 parts of the environment formulation?

Question 1 [G]

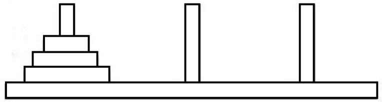
Figure 3: Tower of Hanoi with $n = 3$ disks

Tower of Hanoi is a game with 3 (L,M,R) pegs and a set of n disks with ascending (different) sizes. The aim is to move all disks from the L to the R peg with the minimum moves, considering:

- Move one disk at a time, from one peg to the other.
- Larger disks cannot be placed ontop of smaller disks.

Discuss its environment formulation.

Recap: What are the 5 parts of the environment formulation?

State Space, Initial State, Final State, Action, Transition Model.

Answer 1

Sample (no right answer) Environment Formulation:

- **State Space**:
  - Each disk has an index from 1 to n corresponding to its size,
  - 3 tuples for the pegs $[L, M, R]$
  - **Invariant**: each tuple must be in ascending order
- **Initial State**: $\left[(1, 2, \cdots, n), (), ()\right]$
- **Final State**: $\left[(), (), (1, 2, \cdots, n)\right]$
- **Action**: 6 actions - L>M, L>R, M>L, M>R, R>L, R>M
- **Transition Model**: (Incomplete; Need to describe the rest of the 5)
  - Action a_1: L peg to M peg
    - $T\Big([(l_1, ...), (m_1, ...), (...)], a_1\Big) \rightarrow [(...), (l_1, m_1, ...), (...)], l_1 < m_1$
    - $T\Big([(l_1, ...), (), (...)\Big), a_1] \rightarrow [(...), (l_1), (...)]$

Some representations are better than others - Compute complexity, No of states, etc...

Quality of Representation

Surjection must be formed between the set of all possible valid state representations: For any possible real world config, there is a corresponding state representation.

Invariant

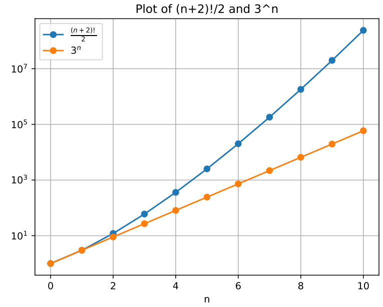

Figure 4: With/Without invariant: Indices in tuple must be ascending.

Without invariant: $\frac{(n+2)!}{2}$: Shuffle then split the items into partitions
With invariant: 3^n: Consider the state tree of placing largest to smallest disks

Question 2 [G]

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

Recap

What are the search algorithms that we have learnt?

Question 2 [G]

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

Recap

What are the search algorithms that we have learnt?

- BFS: Searches row by row down the depth of the tree.
  - UCS: Searches rows at the boundary where the cost is uniform/same.
- DFS: Searches column by column of the tree.
  - DLS: Searches columns until a certain depth.
- IDS: Searches row by row by traversing column by column of increasing depth.

Answer 2

We know that:

- Minimum number of moves to solve Tower of Hanoi is $l = 2^n - 1$
- Number of actions b at each state is:
  - maximally: 6 (from Q1)
  - considering only legal moves: 3 (smaller disk on-top)
  - hierarchal: 2
- There is may be no maximum depth m: Loops.

.	Time	Space
BFS	$O(b^d)$	$O(b^d)$
UCS	$O(b^d)$	$O(b^d)$
DFS	$O(b^m)$	$O(bm)$
DLS*	$O(b^l)$	$O(bl)$
IDS	$O(b^d)$	$O(bd)$

Section 4: **Uniform-Cost Search Analysis**

Question 1 [G]

Let S be the initial state and G be the goal state.

1. Trace uniform-cost search using *tree*
2. Trace uniform-cost search using *graph*
3. Why doesn't the algorithm halt and return the search result since the goal has been found?
4. What's the difference between uniform-cost search (using graph search) and Dijkstra's algorithm?

Recap

- What is the difference between Tree Search and Graph Search?
- AIMA Chapter 3, on graph and tree-like search.

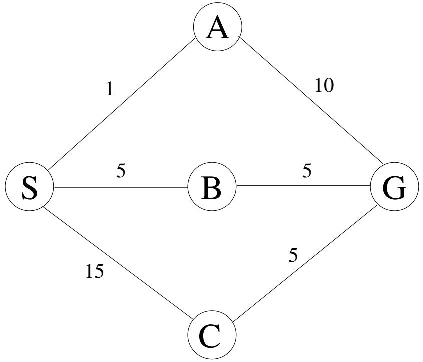

Figure 5: Graph with cost.

Answer 1

```
S(0)
A(1) B(5) C(15)
S(2) B(5) G(11) C(15)
A(3) B(5) B(7) G(11) C(15) C(17)
S(4) B(5) B(7) G(11) G(13) C(15) C(17)
A(5) B(5) B(7) B(9) G(11) G(13) C(15) C(17) C(19)
B(5) S(6) B(7) B(9) G(11) G(13) C(15) G(15) C(17) C(19)
S(6) B(7) B(9) G(10) S(10) G(11) G(13) C(15) G(15) C(17) C(19)
A(7) B(7) B(9) G(10) S(10) B(11) G(11) G(13) C(15) G(15) C(17) C(19)
    C(21)
```

```
B(7) S(8) B(9) G(10) S(10) B(11) G(11) G(13) C(15) G(15) C(17) G(17)
    C(19) C(21)
S(8) B(9) G(10) S(10) B(11) G(11) G(12) S(12) G(13) C(15) G(15) C(17)
    G(17) C(19) C(21)
A(9) B(9) G(10) S(10) B(11) G(11) G(12) S(12) B(13) G(13) C(15) G(15)
    C(17) G(17) C(19) C(21) C(23)
B(9) G(10) S(10) S(10) B(11) G(11) G(12) S(12) B(13) G(13) C(15) G(15)
    C(17) G(17) C(19) G(19) C(21) C(23)
G(10) S(10) S(10) B(11) G(11) G(12) S(12) B(13) G(13) G(14) S(14) C(15)
    G(15) C(17) G(17) C(19) G(19) C(21) C(23)
```

Answer 2

frontier	explored
S(0-)	
A(1-S) B(5-S) C(15-S)	S
B(5-S) G(11-SA) C(15-S)	S A
G(10-SB) C(15-S)	S A B

Answer 3

- Goal check is done on `frontier.pop()`, this means that only when the goal is the cheapest item then the algorithm terminates.
- If the algorithm terminates on discovery, we may miss the node with smaller cost.
- The algorithm works by always selecting the node with the least path cost for expansion and it ensures that the first goal node selected for expansion is an optimal (i.e., shortest path) solution.

Answer 4

The algorithms are the same, with minor differences, ie.:

- Dijkstra finds the shortest path to every node from a single source
- UCS concerns itself with the shortest path to the goal states.

Section 5: **Comparison of Search Strategies**

Question 1 [G]

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

Answer

Consider a problem with branching factor b such that all nodes at depth d are solutions and all nodes at shallower depths are not solutions.

- Number of search nodes expanded by DFS (graph) $= O(d)$
- Number of search nodes expanded by IDS $= O(b^{d-1})$

Remark

We are looking at nodes expanded, not generated (in that case it will be $O(bd)$)

Question 2 [G]

Describe a problem where tree search performs much better than graph search.

Recap

What is the difference between tree search and graph search?

Question 2 [G]

Describe a problem where tree search performs much better than graph search.

Recap

What is the difference between tree search and graph search?

- Tree: Visits repeated states
- Graph: Has memory to store visited states

Answer 2

Consider a problem with a perfect binary tree structure, with no cycles:

- Since no repeated states, graph search performs exactly like tree search:
  - same time complexity,
  - same space complexity for frontier, ie. $O(bm)$ for DFS
- Additional memory usage for graph search to store visited, $O(b^d)$ at depth d

Bonus Question

To help you further your understanding, not compulsory; Work for Snack/EXP!

Tasks

1. Fork the repository https://github.com/eric-vader/CS2109s-2425s1-bonus
2. We will be first solving tracing using code, `uniform_cost_search(graph, inital_node, goal_test, ...)` that returns the path found:
   a. Able to solve 4a via `is_tree=True,is_update=False`
   b. Able to solve 4b via `is_tree=False,is_update=True`
3. Some code have been implemented for you, including the priority queue.
4. You should print the frontier and explored at the beginning of the loop.

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

Useful Links

- AIMA Python Implementation - https://github.com/aimacode/aima-python.

Buddy Attendance Taking

Important admin:

1. Join our Telegram Group: https://t.me/+jWEjTd9W5Ug0NmE1
2. Welcome Survey: https://forms.gle/cAQSeYvqi4yzrfkD6
3. Take Attendance for your buddy: https://forms.gle/BejdiR451K9ZhRa17

Figure 6: Telegram Group

Figure 7: Survey

Figure 8: Buddy Attendance

References I