CS2109S Tutorial 1

Uninformed Search and Informed Search

(AY 25/26 Semester 2)

January 29, 2026

(Prepared by Benson)

Join our telegram group!
https://t.me/+tsc9QW6Xy00xYTU1

Contents

Admin Info

- Complete (and submit) the pre-survey form (9/14).
- Join the telegram group (9/14).
  - 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).
  - Recordings will be delayed.
- Bonus Lab Logistics (Already published at the GitHub Repo):
  - Submit the bonus lab at most one week after the corresponding tutorial (e.g. bonus lab 1 before week 4 tutorial).
  - For each accepted submission, I will upgrade 800 $\Rightarrow$ 1000 EXP for one future week.
  - **Submission**: Send a telegram direct message to Suhail with relevant screenshots & your writeup (if required).

Introduction

YEUNG Man Tsung (Benson)

- Third time teaching CS2109S!
  - Also doing some course admin...
- Year 4 Computer Science
- **Email**: mtyeung@u.nus.edu
- **Discord**: @mtyeung
- **Telegram**: @mtyeung
- **Consultation**: By appointment.

^Me in MLSys 2025.

Introduction

Suhail Loya

- He will be your problem set grader!
  - Reach him if your question is *completely* problem set related (grading, need help, etc.).
- Year 3 Computer Science
- **Email**: suhail.loya@u.nus.edu
- **Telegram**: @suhail_loya
- Or simply leave a comment under your Coursemology submissions.

About CS2109S

"My job is being taken over by ~~AI~~."

a stupid autoregressive machine
a weighted average of the internet
a stack of matrix multiplications
probabilities pretending to be thoughts

About CS2109S

Tutorial 1-2:
Classical AI

- Search algorithms (CS2040S++) in **state spaces**.
- What makes a *good* practical solution? (More in the Capstone Project!)

Tutorial 3-10:
Modern ML

- What do ML scientists care about? What do HPC scientists care about?
  - We unveil this by covering classical & modern ML techniques.
- Lots of math (linear algebra and calculus).
- Lectures & Tutorials: Helps you understand "what you are doing"
  + Problem Sets: Implementation (libraries).

About CS2109S Tutorials

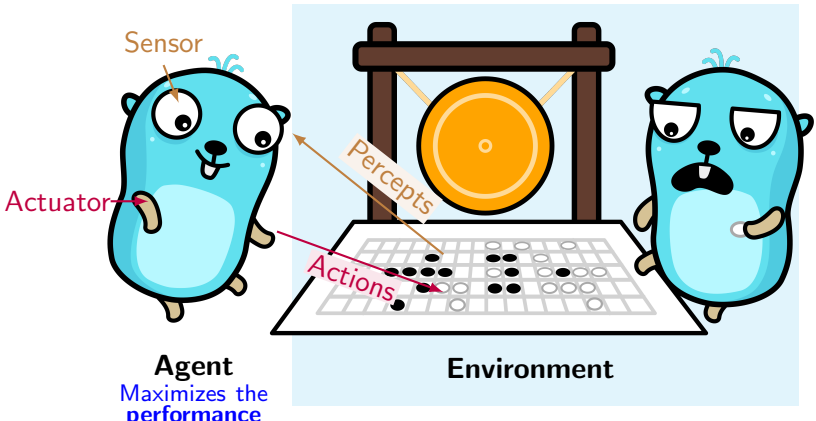
Format:

- 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	Item	
800	Attendance	} Free
1000	Active Participation / Solving Bonus Lab / Poll Bonus	} Tiny differentiation

Recap: PEAS


Sensor
Actuator
Percepts
Actions
Agent
Maximizes the
performance
Environment

Q1. PEAS for AI Fitness Coach

Determine the PEAS (Performance measure, Environment, Actuators, Sensors) for an AI-powered personal fitness coach.

Performance Measure:
Improvement in user's health metrics
User consistency in the training programme
Minimise user injury and maximise fitness

Environment:
User health status
Preferred workout setting
The user's lifestyle constraints
The device on which the user is using the app
Internet connectivity

Actuators:
App interface where workout plans and progress are shown
Notifications regarding workouts and meals
Voice guidance during workout

Sensors:
Camera for workout posture monitoring
Keyboard for user feedback
Movement sensors for tracking user movement
Wearable sensors for measuring heart rate, step count, sleep pattern etc.

Recap: Different Types of Search

Why learn everything again?

Search in CS2040(C/S)

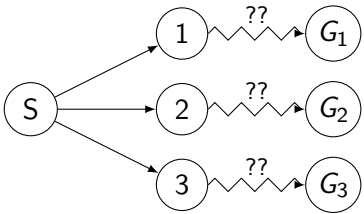
- Searching in *graphs*.
- Entire graph given as input (usually "small").
- Finds the shortest path to all vertices (single-source / all-pairs).

Search in CS2109S

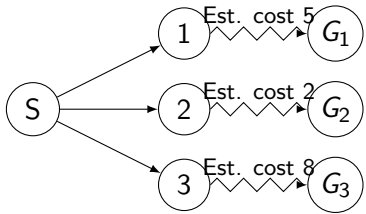
- Searching in **state spaces**, i.e. modeling real life problems with states (vertices) & transitions (edges).
- We don't know the size of the graph (can be infinite). *Lazily* evaluated.
- Only interested in reaching the **goal states**. (We don't know which.)
- Sometimes, we are satisfied with "good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

Recap: Different Types of Search

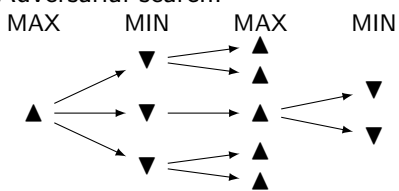
Uninformed search:



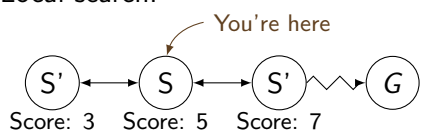
Informed search:



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*.

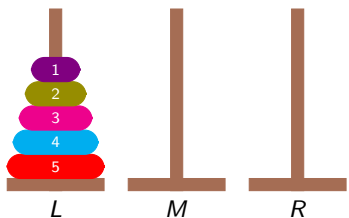
(a) Give the state representation.

(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.

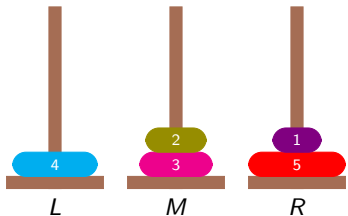
(c) Specify the initial and goal states.

(d) Define the actions.

(e) State the transition function T.



Q2. Tower of Hanoi



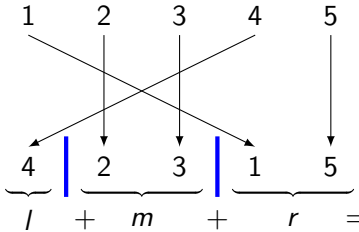
State Representation:

- 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*.

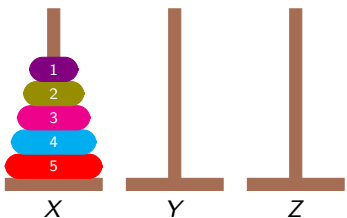
$1_R \quad 2_M \quad 3_M \quad 4_L \quad 5_R$ $\qquad 3^n$ ways

  - Number of states $= 3^n$.

Q2. Tower of Hanoi

What are the initial and goal states?

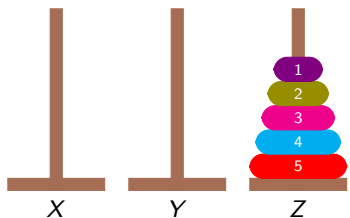
Initial state:



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

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

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 $d = 2^n - 1$. $(T(n) = T(n-1) + 1 + T(n-1))$

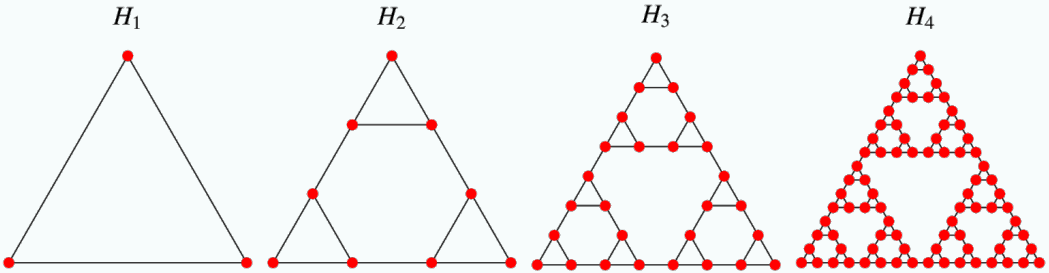
Search	Time	Space
Breadth-First Search (BFS)	$O(b^d) = O(3^{2^n-1})$	$O(b^d) = O(3^{2^n-1})$
Uniform Cost Search (UCS)	Same as BFS, all costs are 1.	
Depth-First Search (DFS)	∞, can get trapped.	
Depth-Limited Search (DLS)	$O(b^d) = O(3^{2^n-1})$	$O(bd) = O(2^n)$
Iterative Deepening Search (IDS)	We already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

DLS is the most suitable algorithm.

Q2. Tower of Hanoi

Extra Slide

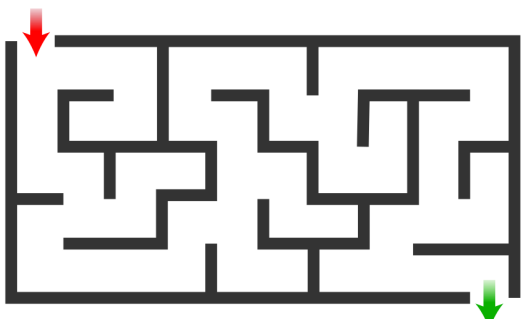
The state graph corresponding to the Tower of Hanoi puzzle is called the **Hanoi graph**, which has a beautiful recursive pattern.



Q3. Search Algorithm for the Maze Problem

(a) Suppose the maze is extremely large and has very few walls. Which search algorithm *with visited memory* would you use to find any escape path?

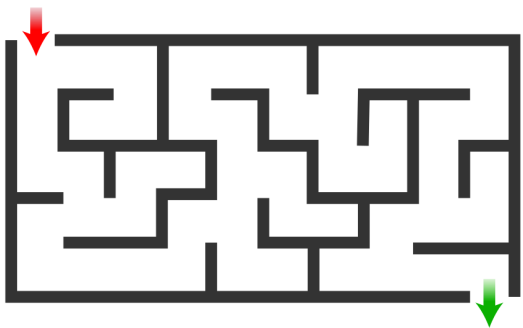
- We don't care about the optimal path $\Rightarrow$ DFS is sufficient.
- Pros: Less memory usage.



Q3. Search Algorithm for the Maze Problem

(b) Suppose the longest escape path without revisiting cells has 100 moves, but most escape paths are around 10 moves. Which search algorithm *without visited memory* would you use to find the shortest escape path?

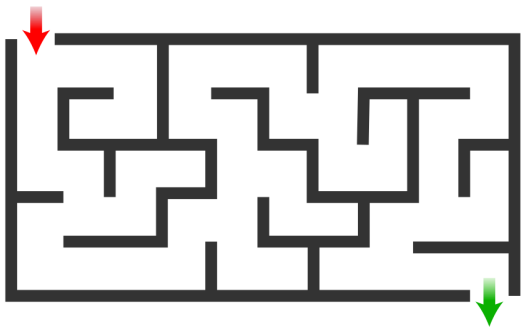
- We want the search to terminate as soon as possible $\Rightarrow$ IDS.
- Note that there are no strong depth limit guarantees.



Q3. Search Algorithm for the Maze Problem

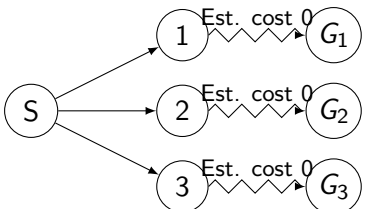
(c) Suppose the maze is now infinite and escape is not guaranteed. Any escape path that exists will be no longer than 50 moves. If no escape path is found within 50 moves, then escape should be considered impossible. Which search algorithm *without visited memory* would you use to attempt to find the shortest escape path?

- BFS with depth limit 50.

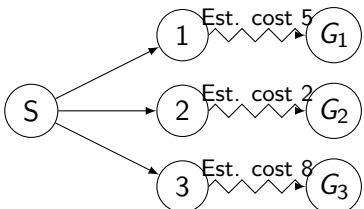


Recap: Informed Search

Searching with a **heuristic** (*estimates* the cost to the goal).



Degrades to uninformed search



A good one?

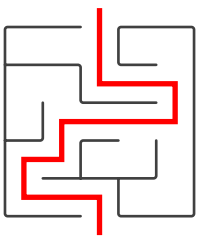
Qn: Why would we prefer informed search over uninformed search?

Recap: Informed Search

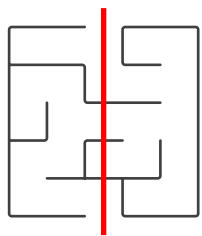
Admissible

$h(v) \leq h^*(v)$

$\Rightarrow$ Never overestimates the true cost of the **path**



True cost



Admissible heuristic

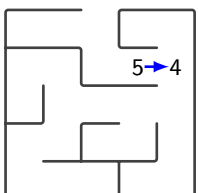
(**Relaxed problem**: We can go through walls.)

Recap: Informed Search

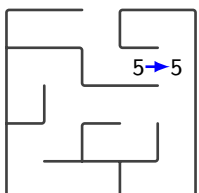
Consistent

$h(v) \leq c(v, a, u) + h(u)$ i.e. $h(v) - h(u) \leq c(v, a, u)$

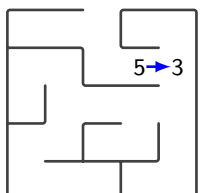
$\Rightarrow$ Never overestimates the cost of an **action**



OK



OK



Inconsistent

Recap: Informed Search

Admissible

$h(v) \leq h^*(v)$

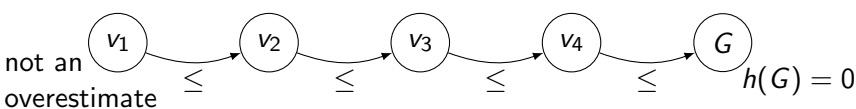
$\Rightarrow$ Never overestimate cost of path

Consistent

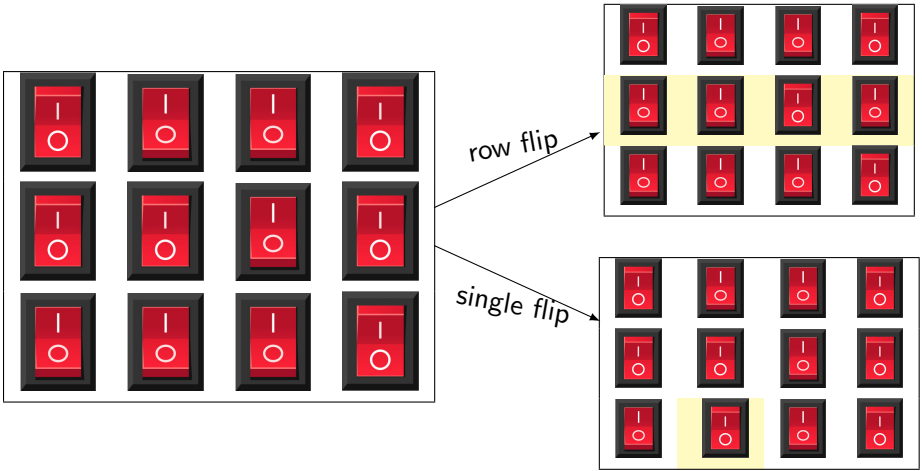
$h(v) \leq c(v, a, u) + h(u)$

$\Rightarrow$ Never overestimate cost of action

Intuitive: Consistent $\Rightarrow$ Admissible



Q4. Escape Room Control Panel

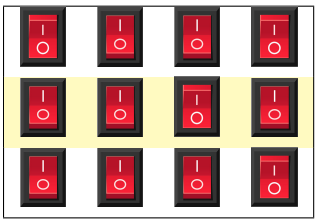


Q4. Escape Room Control Panel

- $h_1(n)$ = minimum Hamming distance from n to any state in $\mathcal{G}$.

Admissible

$h(v) \leq h^*(v)$



- $h_1(n) = 4$.
- $h^*(n) = 1$.

Consistent

$h(v) \leq c(v, a, u) + h(u)$

Since Consistent $\Rightarrow$ Admissible, we have Not Admissible $\Rightarrow$ Not Consistent.

CS1231/S Refresher: $X \Rightarrow Y$ is equivalent to $\sim X \Rightarrow \sim Y$ (contrapositive).

Follow Up: Can you propose a fix?

Q4. Escape Room Control Panel

- $h_2(n) = \min_{g \in \mathcal{G}} \sum_{i=1}^{3}$ number of actions needed for row i of n to match g.

Admissible

$h(v) \leq h^*(v)$

This is the exact cost to the closest goal!

Consistent

$h(v) \leq c(v, a, u) + h(u)$

After an action on row i:

- "number of actions needed for row i" decreases by ≤ 1.
- "number of actions needed for row $j \neq i$" remains unchanged.
- For any g, the summation decreases by ≤ 1.

Q4. Escape Room Control Panel

- $h_3(n) = \min_{g \in \mathcal{G}} \sum_{i=1}^{3} \begin{cases} 2 & \text{if } n \text{ and } g \text{ differs by 2 states on row } i \\ 0 & \text{otherwise} \end{cases}$.

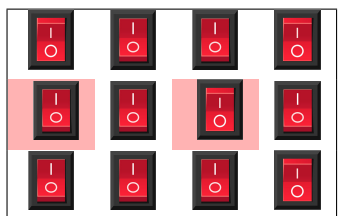
Admissible

$h(v) \leq h^*(v)$

Relaxed problem: We accept the arrangement as long as there is no row where n and g differs by 2 states.

Consistent

$h(v) \leq c(v, a, u) + h(u)$



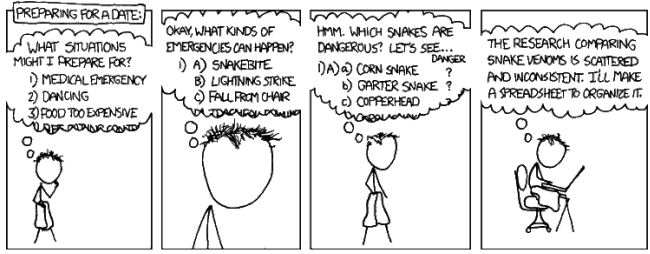
After toggling one of the switches, $h_3(n)$ decreases from 2 to 0.

That's it!

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.

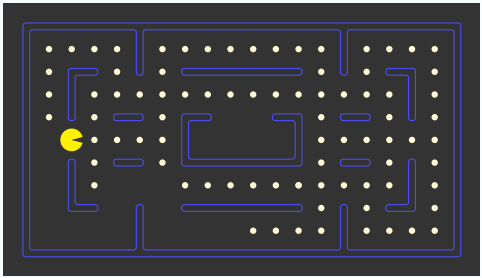


Extra Practice

Q5. Pacman

Extra Slide

Pacman has to eat all the pellets in the maze while executing the least amount of (4-directional) moves. Devise a non-trivial admissible heuristic for this problem.



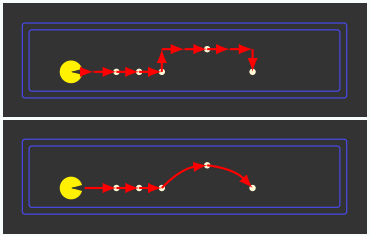
Q5. Pacman

Extra Slide

- $h_1(n)$ = The number of pellets left.

Admissible

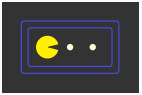
$h(v) \leq h^*(v)$



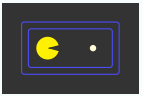
Relaxed problem: Pacman can jump to any cell in one move.

Consistent

$h(v) \leq c(v, a, u) + h(u)$



$\Delta h(n) = -1$



$\Delta h(n) = 0$

In both cases, the cost of the action (1) is not overestimated.

Q5. Pacman

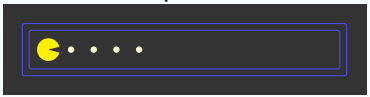
Extra Slide

- $h_2(n)$ = The **sum** of distances to all pellets.

Admissible

$h(v) \leq h^*(v)$

Counterexample:



- $h(v) = 1 + 2 + 3 + 4 = 10$
- $h^*(v) = 4$

Consistent

$h(v) \leq c(v, a, u) + h(u)$

Since Consistent $\Rightarrow$ Admissible, we have Not Admissible $\Rightarrow$ Not Consistent.

CS1231/S Refresher: $X \Rightarrow Y$ is equivalent to $\sim X \Rightarrow \sim Y$ (contrapositive).

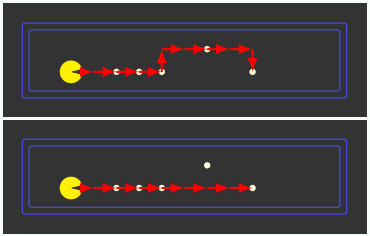
Q5. Pacman

Extra Slide

- $h_3(n)$ = The **maximum** distance to any pellet.

Admissible

$h(v) \leq h^*(v)$



Relaxed problem: Pacman only needs to eat the farthest pellet.

Consistent

$h(v) \leq c(v, a, u) + h(u)$

The maximum distance decreases by at most 1 after a move.

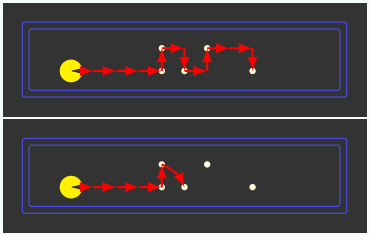
Q5. Pacman

Extra Slide

$h_4(n)$ = The distance to the closest pellet + number of pellets adjacent to that pellet. (If there are multiple closest pellets, take one with the most adjacent pellets.)

Admissible

$h(v) \leq h^*(v)$

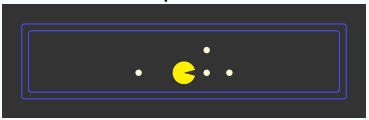


Relaxed problem: Pacman only needs to eat any pellet and its neighbours (by "jumping").

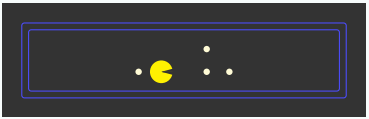
Consistent

$h(v) \leq c(v, a, u) + h(u)$

Counterexample:



- $h(v) = 1 + 2 = 3$



- $h(u) = 1 + 0 = 1$

Q5. Pacman

Extra Slide

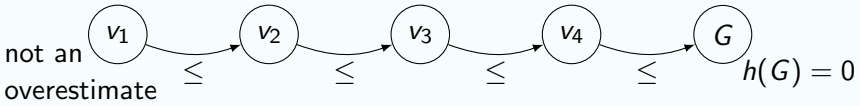
- Which heuristic would you choose for A* search?
  - $h_1(n)$: Distance from Pacman to the nearest pellet.
  - $h_2(n)$: Distance from Pacman to the farthest pellet.
  - $h_3(n) = h_1(n) + h_2(n)$

Note that h_2 **dominates** h_1 and h_3 **dominates** h_2.
However, h_3 is **NOT** admissible. We should choose heuristic h_2.

Q6. Consistent $\Rightarrow$ Admissible

Extra Slide

(a) Given that a heuristic h is such that $h(G) = 0$, where G is any goal state, prove that if h is consistent, then it must be admissible.



- Let $v_1 \to v_2 \to \cdots \to v_n$ be the optimal path from v_1 to G (with minimum cost).
- We have $h(v_{i-1}) - h(v_i) \leq c(v_{i-1}, a_{i-1}, v_i)$ by consistency of h.
- Summing up over all i, we have

cancels out

$$(h(v_1) - h(v_2)) + (h(v_2) - h(v_3)) + \cdots + (h(v_{n-1}) - h(v_n)) \leq \sum c(v_{i-1}, a_{i-1}, v_i)$$

sum of all costs

$$h(v_1) - h(v_n) \leq h^*(v_1)$$

- Since $h(G) = 0$, we have $h(v_1) \leq h^*(v_1) \Rightarrow h$ is admissible.

Q6. Consistent $\Rightarrow$ Admissible

Extra Slide

(b) Give an example of an admissible heurstic function that is not consistent.