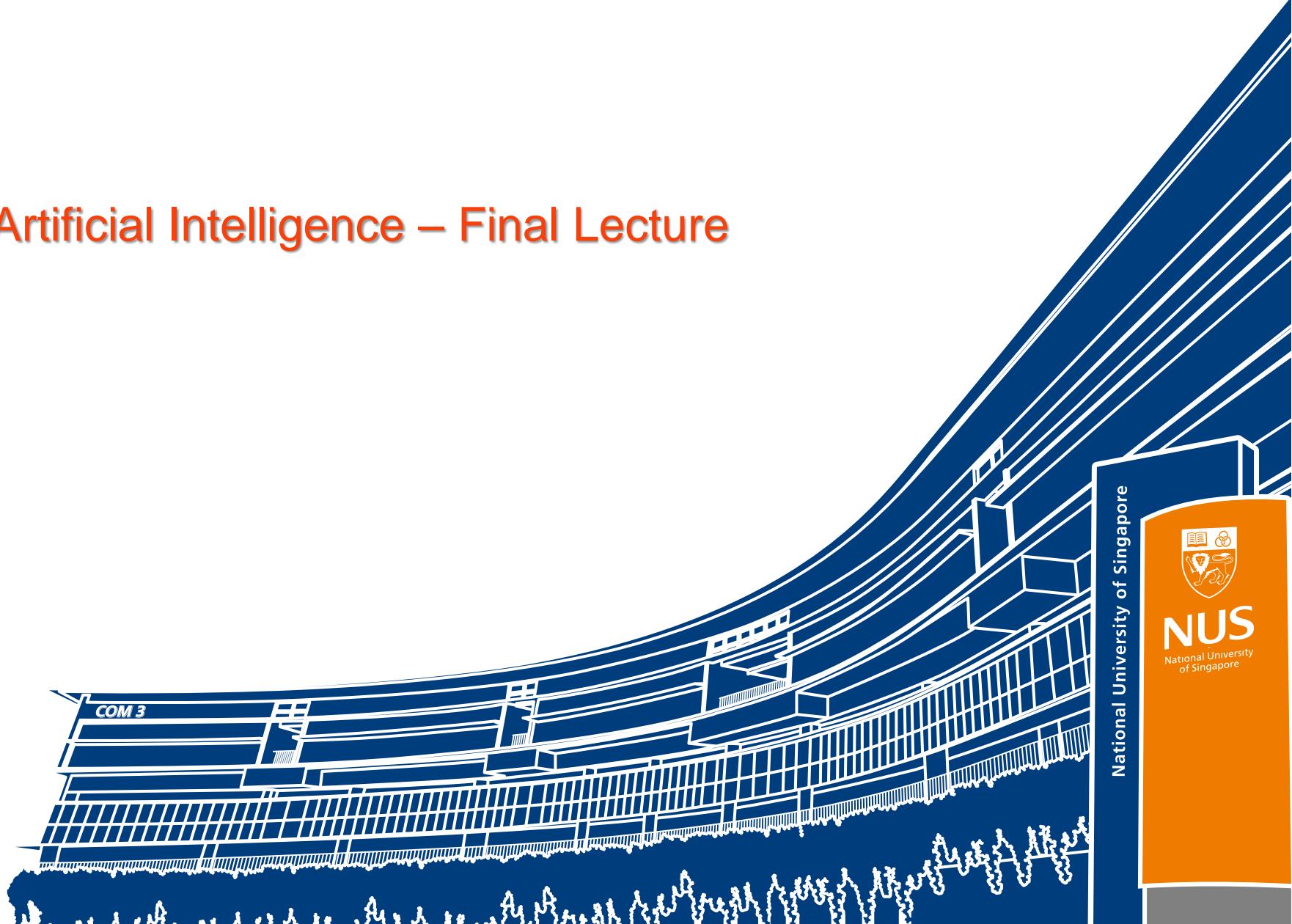


# Course Review

CS3243: Introduction to Artificial Intelligence – Final Lecture



# Contents

- Course Summary
- Assessment Report
- Administrative Matters

1

# Summary of CS3243

# AI as Problem Solving

- **Developing Intelligent Agents**
    - Mechanisms that **solve** problems
    - More generically applicable → more intelligent - i.e., **stronger AI**
  - **Desire to construct rational agents**
    - Optimal performance given:
      - Available information (data)
      - Performance measure (metrics)
  - **CS3243: Studying Symbolic AI**
    - Solutions inspired by the way we (humans) do things
      - Planning
      - Reasoning
      - Learning (over Uncertainty)
- See: [https://en.wikipedia.org/wiki/Symbolic\\_artificial\\_intelligence](https://en.wikipedia.org/wiki/Symbolic_artificial_intelligence)

## Artificial Intelligence (in a Nutshell)

- Intelligent mechanisms that solve problems to help humans
  - Programs concerned with **human actions / thinking**
  - Intelligence assessed based on the mechanism's generality and performance

### Generality

- More dynamic solutions → able to deal with
- Example
  - Google DeepMind's AlphaGo, AlphaZero, and <https://deepmind.com/research/case-studies>

### Performance

- Perform at least as well as humans (and probably better)
- Not necessarily in the same way as performance

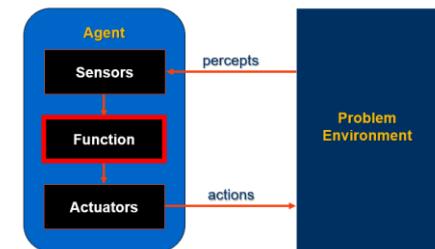
## Kinds of AI

- **Strong AI**
  - General problem solver
  - Very dynamic programs → able to solve many different problems
- **Weak / Narrow AI**
  - Less dynamic programs → solves fewer problems (typically just 1)
  - Corresponds to most AI work
- Usually, focus is on **Narrow AI**

## Rationality & Performance

- Desire a program that works well
  - At least better than humans; ideally optimal
  - Implies a **quantifiable objective** → **performance measure**
    - Are the objectives and performance measure aligned?
- **Rational agent (function)**,  $f : P \rightarrow a_t$ 
  - Given:
    - Percept sequence
    - Prior knowledge
    - Set of actions
    - Performance measure
  - Rational agent optimises performance measure

## Agent Framework



# Decomposition with Problem Environments

- **Taxonomy of Problems → Problem Environment Properties**

- Fully Observable or Partially Observable
- Deterministic or Stochastic
- Episodic or Sequential (assume sequential)
- Discrete or Continuous (assume discretised)
- Single Agent or Multi-Agent (assume single\*)  
\* except for one competing agent under Adversarial Search
- Known or Unknown (assume known)
- Static or Dynamic (assume static)

### Environment Properties

- Single vs multi-agent
  - Do other entities exist in within the environment that directly influence the performance
    - Chess → opponent is a competitor
    - Automated vehicles → other vehicles
- Known versus unknown
  - Refers to knowledge of the agent
  - Includes performance measure
- Static versus dynamic
  - Will the environment change while the agent acts?

### Environment Properties

- Episodic versus sequential
  - Episodic → actions only impact the current state (not those beyond)
  - Sequential → an action may impact all future decisions

### Environment Properties

- Discrete versus continuous
  - Refers to state information, time, percepts, actions

### Environment Properties

- Deterministic versus stochastic
  - Stochastic → intermediate state cannot be determined based on action taken at a given state
  - Handling uncertainty typically required

**Environment Properties**

- Fully observable versus partially observable
  - Agent cannot access all information as some cannot be sensed
  - Requires inference (and/or, handling uncertainty)

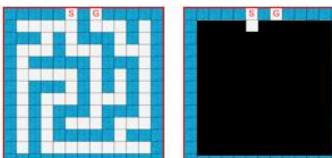


Image source: Wikipedia

 NUS | Computing  
National University of Singapore

5

# Planning Topics

- Assuming complete information → Path Planning or Goal Search
  - Fully Observable and Deterministic
    1. Model problem environment as a graph and solve by finding goal path/state
      - Uninformed Search – e.g., BFS, DFS, DLS, IDS, UCS (i.e., Dijkstra's)
      - Informed Search – e.g., best-first greedy search, A\* search
      - Local Search – e.g., hill-climbing
    2. Model problem environment as a constraint problem and solve by finding goal state
      - Constraint Satisfaction Problems – e.g., backtracking CSP
- Including an opposing agent → Adversarial Search
  - Fully Observable, Deterministic, and Multi-agent (i.e., one competing agent)
  - 3. Model problem environment as a game tree and solve by finding strategy with optimal utility
    - Adversarial Search – e.g., Minimax, α-β Pruning

# Reasoning & Learning Topics

- Assuming incomplete information → Reasoning via Inference
  - Partially Observable and Deterministic
    1. Model problem environment as a knowledge base, then use inference engine to solve by finding goal path or path with optimal utility
      - Logical Agent is non-planning – i.e., directly interacting with problem-environment
      - Inference Algorithms – e.g., Truth Table Enumeration, Resolution  
Note: inference algorithms may be applied in any agent to prove things
- Assuming stochastic problem → Uncertainty handled via Bayesian Network
  - Stochastic (with Fully or Partially Observable)
  - 3. Model stochastic components via Bayesian Networks
    - Decision-Theoretic Agent makes decisions by optimising expected utility
    - Bayesian Algorithms - e.g., Naive Bayes, Bayesian Network
      - Note: Bayesian reasoning is categorised with Symbolic Machine Learning to handle uncertainty

# Summarising the Key Ideas

## Generalising Problems & Rational Agents

- Standardised **problem modelling** (graphs of states and actions) and design of **agents**
- Strive for rational agents – optimise **performance** based on **data/info** (knowledge)

Generally, assume:  
**Known** and **Static** Environments

## Definition of Basic Problem Paradigms

- **Path planning problems** (Informed & Uninformed)
- **Goal composition** (Local & CSPs)
- Game-playing **strategies** (Adversarial)

Additionally, assume:  
**Full Observable**, **Deterministic**,  
and **Discrete** Environments

## Dealing with Large Search Trees

- **Heuristics** – quantifying better actions
- **Pruning** – eliminating sub-trees we know we don't need (CSP Inference,  $\alpha$ - $\beta$  Pruning)

Note: Adversarial Search is  
the only topic with a  
**Multi-Agent** Environment

## Dealing with Partial Information

- **Building a KB** (Logic)
- **Growing a KB** by making **inferences** (Truth-Table Enumeration, Resolution)

Assuming:  
**Partially Observable**  
Environments

## Dealing with Uncertainty

- Making **inferences on likelihood** in the face of **uncertainty** (Bayesian Networks)

Assuming:  
**Stochastic** Environments

# Modelling Search Problems

- **Representation**
  - Problem → States + Actions + Transition → Search Tree
  - Problem → Goal + Costs → Performance Measure

Midterm Q1 focuses on this learning objective  
Note: this is not directly assessed in the Final; but may be indirectly assessed via the formulation of other solutions (e.g., CSP, Adversarial, Logical Agent, Bayesian Network models)
- **Analysis of representation**
  - Correctness as the determination of a plan or legal goal
    - Planning: action sequence from initial state to goal state
    - Legal goal: a valid goal state that is reachable from initial state
  - Complexity or tree size in terms of  $b$ ,  $m$ 
    - $b$ : branching factor
    - $m$ : maximum depth

Midterm Q1 also focused on these learning objectives; at most MCQs on search tree complexities in the Final

# Solving Search Problems – Uninformed Search

- Applying uninformed search algorithms – BFS / DFS / DLS / IDS (only frontier differs)

- Systematically traverse search tree to find path to goal or legal goal
  - Complexity – given the search tree properties
  - Completeness – will the algorithm guarantee a solution
  - Optimality – will the solution have minimal cost

Assumes problems are:

- Fully Observable
- Deterministic
- Single-Agent

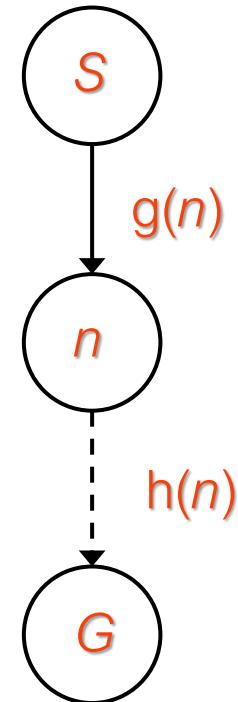
- Algorithm properties

- Complexities
  - Time: ALL exponential
  - Space: BFS exponential; DFS/DLS/IDS linear
- Implementations
  - BFS with early goal test reduces complexity
  - DFS with backtracking improves space complexity
  - Note Tree versus Graph variants
- Completeness
  - BFS/IDS: if goal exists;  $b$  finite
  - DLS:  $d \leq$  depth limit
  - DFS:  $m$  finite
- Optimality
  - BFS/DFS/DLS/IDS are generally, NOT optimal

Midterm Q2 focused on these learning objectives; at most MCQs on these in the Final

# Solving Search Problems – Informed Search

- Best-first Search Algorithm – frontier as priority queue
  - Evaluation function used to order nodes on Frontier
    - UCS ( $g(n)$  priority): explore paths in order of path cost (uninformed)
    - Greedy ( $h(n)$  priority): explore paths by following states closest to goal
    - A\* ( $g(n)+h(n)$  priority): explore paths in order of approximate cost from initial to goal state
  - Greedy and A\* are informed search algorithms – require heuristic  $h$
- Idea: use domain knowledge to determine good heuristics ( $h$ )
  - Relax problem constraints to define (admissible) heuristics
  - Complexity of  $h$  should be low (e.g., constant or linear)
- Algorithm properties
  - Exponential complexities – dependent heuristic accuracy (effective branching factor)
  - UCS is complete and optimal; Greedy is neither
  - A\* depends on heuristic properties (i.e., admissible, consistent)



Midterm Q2 & Q3 focused on these learning objectives;  
expect a Q on these in the Final!

# Solving Search Problems – Local, CSP, & Adversarial

- Goal search – focus on finding legal goal state; assumes path inconsequential
  - Local Search
    - Tends to use complete-state formulation – i.e., no incomplete states that preclude goal
    - Restrict frontier size to 1 (hill-climbing) or  $k$  (beam search) – many variants
    - Requires heuristic  $h$  to choose among successors to explore
  - Constraint Satisfaction Problem (CSP)
    - Generalised goal-search solution
    - Formulation → Variables + Domains + Constraints (with constraint graph representation)
    - Uses backtracking search (DFS)
- Adversarial search – assumes 2-player, zero-sum game
  - Formulation additionally requires: Player + Terminal + Utility
  - Assume an opponent strategy to plan
  - Minimax algorithm (based on backtracking) – assumes optimal opponent (given utility)

Adversarial search assumes problems are multi-agent

Midterm up to Local Search;  
Finals may include MCQs  
local search (recall that only  
up to 20% of Final on  
Midterm topics)

Finals WILL include a CSP  
modelling question! No need  
to focus on trace/proof Qs!

No question on problem modelling or  
evaluation function modelling since these  
were covered in Project 3;  
instead, focus on the Minimax and  $\alpha$ - $\beta$   
Pruning algorithms – i.e., trace/proof Qs!

# Heuristics & How They Help

- Estimating path cost of goal (to generate/expand fewer nodes)
  - Informed search algorithms use them to direct search
  - Local search uses them to estimate proximity to a solution
  - Domain-specific
- As evaluation functions (for limited depth traversal in games)
  - To estimate utility at non-terminal game states
  - Domain-specific
- Determining value/variable order (remove subtrees in CSPs)
  - Strategies to find better orderings, which in turn allow more pruning
  - Domain-generic

Less emphasis on evaluation function formulation; for heuristic design, focus on informed and local search

Make sure that you are able to trace the various CSP heuristics!

# Shrinking Search Spaces

- **$\alpha$ - $\beta$  Pruning**
  - Do not explore sequences of moves that would never be taken
  - Optimal ordering →  $O(b^m)$  to  $O(b^{m/2})$
  - Random Ordering →  $O(b^m)$  to  $O(b^{3m/4})$
- **Inference in CSPs via Forward Checking or AC-3**
  - Can we infer if a state is terminal?
  - Shrinking domains (fewer actions to try)
  - Can be expensive
    - Is there really a saving?
    - Just pre-processing?

Similar to Minimax algorithm: focus on the  $\alpha$ - $\beta$  Pruning algorithm – trace/proof Qs

Make sure you also focus on the Forward Checking and AC-3 algorithms!

# Knowledge Representation

- Representing Knowledge
  - Logic
  - Others?
- Growing knowledge via Inference Engine  
(inference via automated theorem proving)
  - Truth-table enumeration
  - Resolution – proof by contradiction (over CNF KB)
- Logical agents and queries
  - Use domain knowledge to formulate queries
    - Answering queries directs decision making

Now assumes problems are:

- Partially Observable

Finals WILL include a Logical Agent modelling question! Including how to determine queries for IE

Can you formulate KB from context? What about CNF KB? (ref. past-sem. papers)

# Handling Uncertainty

- **Use Bayesian Networks (BN)**
  - Infer what is most probable to make decisions
  - Only store/reference some of the joint probability distribution table by defining beliefs about variable relationships
  - Assume conditional independence between non-connected variables
    - Reduces number of probabilities required

Now assumes problems are:

- Stochastic

Includes Naïve Bayes algorithm; make sure you can apply from tabular data (ref. part-sem. papers)

No need to model BN from context; just apply BN to determine probabilities

# Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?
- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question



Invitation Link (Use NUS Email --- starts with E)  
<https://archipelago.rocks/app/resend-invite/92727263219>

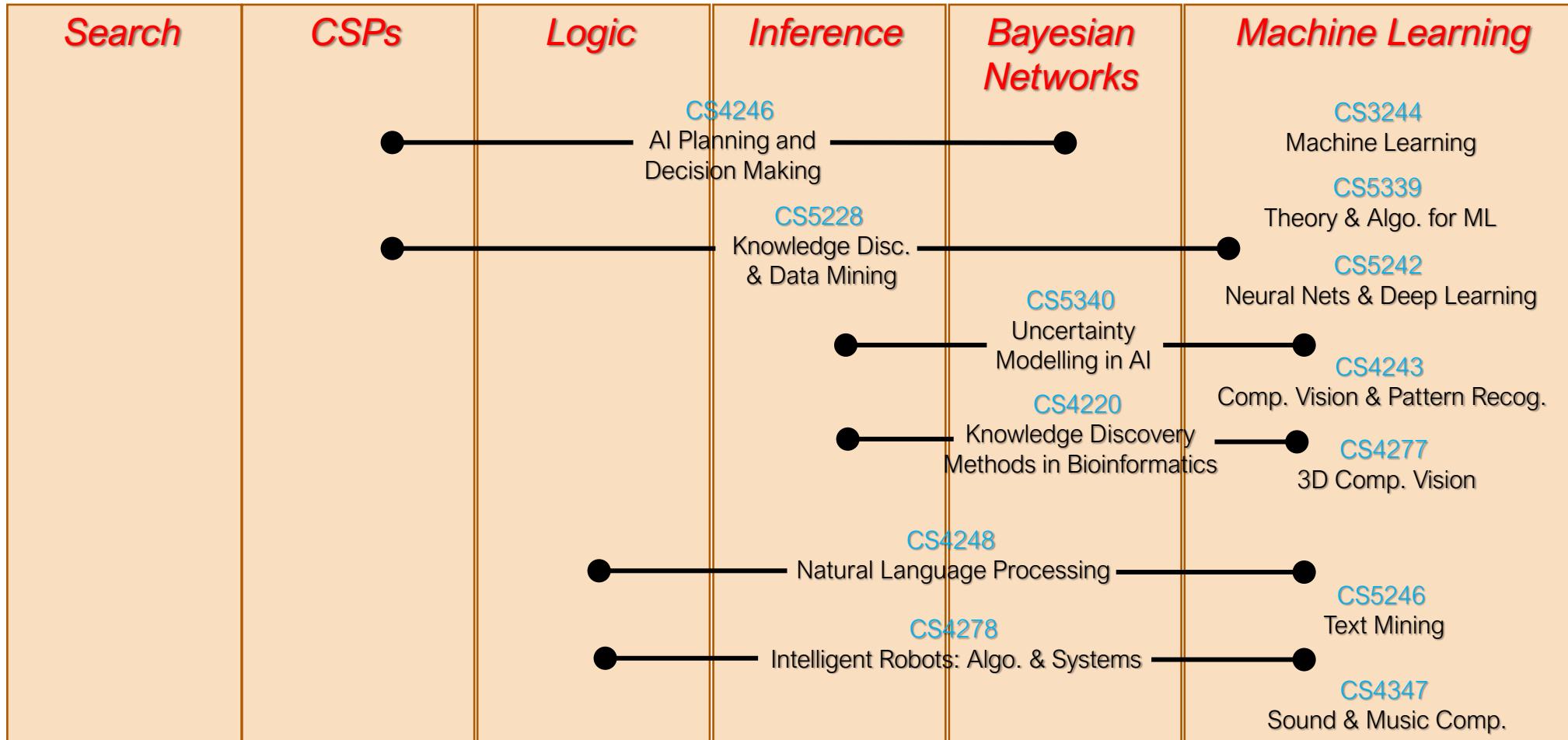
2

## Possible Next Steps in AI

# Learning More About AI

- **Theory:** more on algorithms
  - Domain expertise = algorithms
  - Problem categories
    - (more) planning and search
    - (more) reasoning with logic and knowledge representation
    - (machine) learning
    - etc.
- **Practice:** more on formulation and representation
  - Domain expertise = problem domains
    - Data-specific – e.g., text, images, video, etc.
    - Problem-specific – e.g., medicine, games, etc.

# Further CS Courses in AI



# Research Horizon: Synthesis & Back to Symbolic AI

- Going back towards Strong(er) AI
    - Driven by successes in deep learning
  - Requires synthesis of various AI fields
    - Neuro-symbolic AI
      - Integrates neural and symbolic AI architectures to overcome weaknesses of each
      - Providing robust AI capable of reasoning, learning, and cognitive modelling
    - Dual-cognition systems
      - System 1: fast, reflexive, intuitive, and unconscious
      - System 2: handles planning, deduction, and deliberation
      - Both are required for a robust, reliable AI that can facilitate humans to accept advice and answer questions

The following AAAI Presidential Address by Bart Selman in 2022 describes this further:

[https://aaai-2022.virtualchair.net/plenary\\_2.html](https://aaai-2022.virtualchair.net/plenary_2.html)

(scroll down and watch the video from the 1h 40m mark)

Introduction | Research | Industry | Politics | Safety | Predictions

#stateofai | 20

## Are neuro-symbolic systems making a comeback?

► Deficiencies in both reasoning capabilities and training data mean that AI systems have frequently fallen short on math and geometry problems. With AlphaGeometry, a symbolic deduction engine comes to the rescue.

- A Google DeepMind/NYU team generated millions of synthetic theorems and proofs using symbolic engines, using them to train a language model from scratch.
- AlphaGeometry alternates between the language model proposing new constructions and symbolic engines performing deductions until a solution is found.
- Impressively, it solved 25 out of 30 on a benchmark of Olympiad-level geometry problems, nearing human International Mathematical Olympiad gold medalist performance. The next best AI performance scored only 10.
- It also demonstrated generalisation capabilities - for example, finding that a specific detail in a 2004 IMO problem was unnecessary for the proof.

IMO 2015 P3

Given ABC is an acute-angled triangle. Let O(ABC) be its circumcircle. An altitude is drawn from C to the base AB, meeting it at D. Line CD is extended to meet O at E. Line DE is drawn on O, meeting O again at F. Prove that the circumcircles O(DEF) and O(BCF) touch each other at point E.

**Solution**

AlphaGeometry

```

 $\text{Let } \angle BAC = \alpha, \angle ABC = \beta, \angle ACB = \gamma$ 
 $\text{Angle sum property of triangle } ABC \Rightarrow \alpha + \beta + \gamma = 180^\circ$ 
 $\text{As } O(ABC) \text{ is the circumcircle of } \triangle ABC$ 
 $\therefore \angle AOC = 2\alpha, \angle BOC = 2\beta, \angle AOB = 2\gamma$ 
 $\text{In } \triangle OED, \angle OED = 180^\circ - \angle ODE - \angle OED$ 
 $\angle OED = 180^\circ - 90^\circ - \angle ODE$ 
 $\angle OED = 90^\circ - \angle ODE$ 
 $\angle OED = 90^\circ - \frac{1}{2}(\angle BOC)$ 
 $\angle OED = 90^\circ - \frac{1}{2}(2\beta)$ 
 $\angle OED = 90^\circ - \beta$ 
 $\angle OED = \beta$ 
 $\angle OED = \angle OBC$ 
 $\therefore O(DEF) \text{ and } O(BCF) \text{ touch each other at point } E$ 

```

stateof.ai 2024

## **1. AI beats humans on some tasks, but not on all.**

AI has surpassed human performance on several benchmarks, including some in image classification, visual reasoning, and English understanding. Yet it trails behind on more complex tasks like competition-level mathematics, visual commonsense reasoning and planning.

Stanford AI Index Report 2024:  
<https://aiindex.stanford.edu/report/>

3

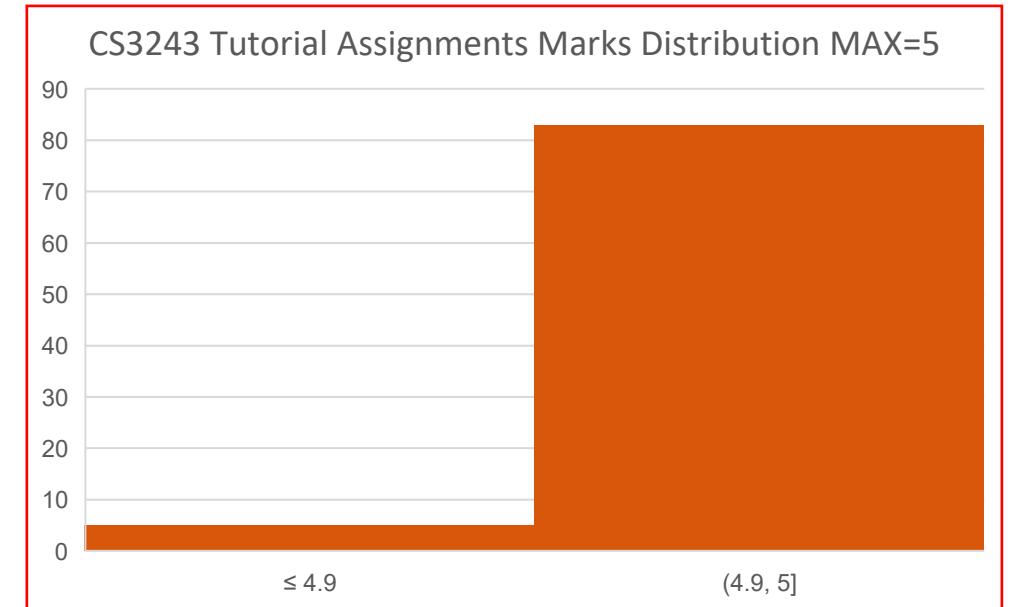
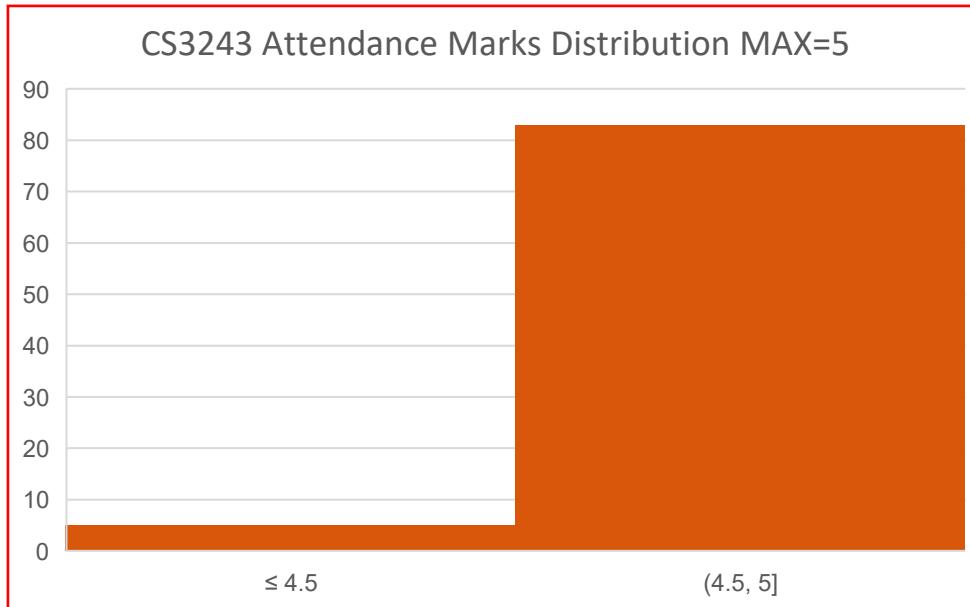
## Report on Assessments

# Assessments & Course Weights

- Course Participation (11)
  - Total 5%
  - Tutorial Attendance
- Tutorial Assignments (9)
  - Total 5%
  - Each worth 1% (best 5 taken)
- Midterm Examination (1)
  - Total 20%
  - Closed Book + Cheat Sheet (1 x A4 Sheet)
  - In-person + Written
- Python Projects (5)
  - Total 30%
    - Projects 1.1 + 1.2: 10%
    - Projects 2.1 + 2.2: 10%
    - Project 3: 10%
  - Individual
  - Bonus marks available (covering projects)
- Final Examination (1)
  - Total 40%
  - Closed Book + Cheat Sheet (1 x A4 Sheet)
  - In-person + Written

# Summary of Assessments

- **Attendance (AT – MAX 5%)**
  - Mean: **4.85**
  - Median: **5**
- **Tutorial Assignments (TA – MAX 5%)**
  - Mean: **4.85**
  - Median: **5**



# Summary of Assessments

- Why Attendance and TAs (and DQs)?

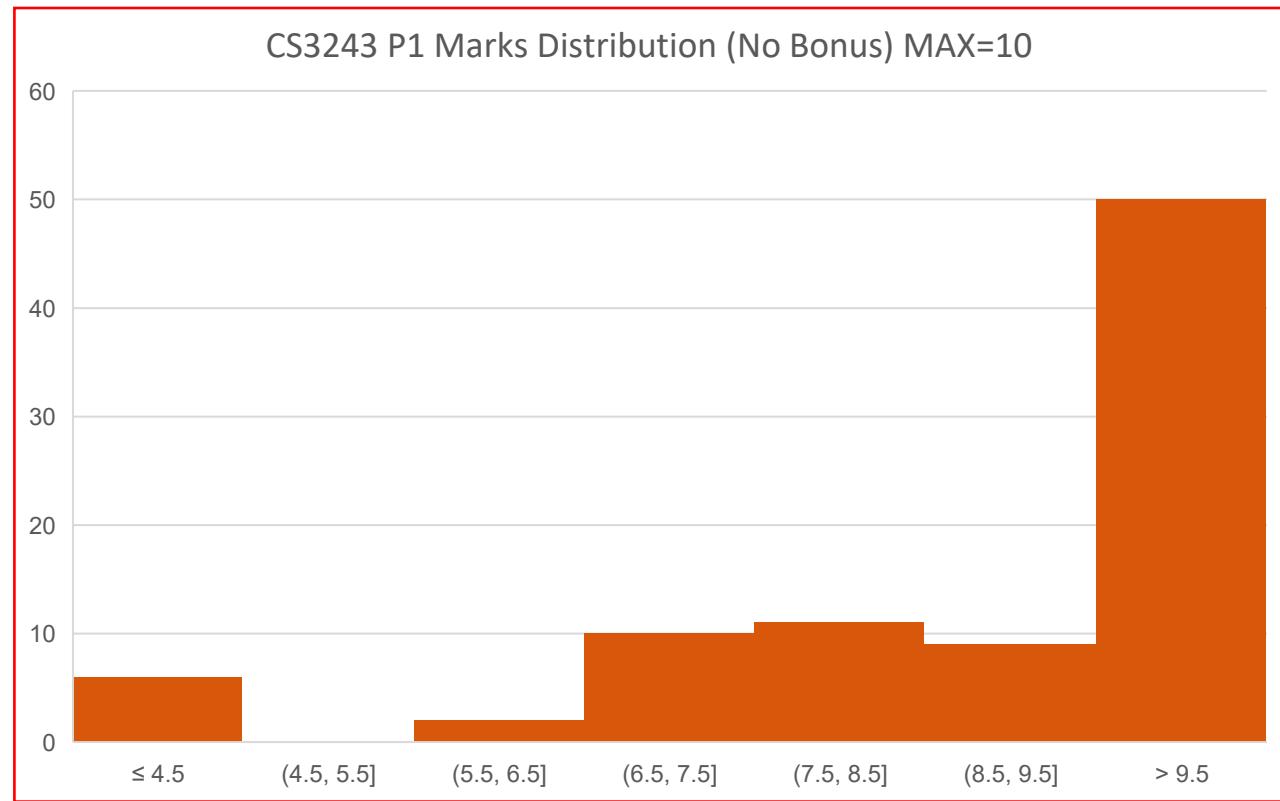
Motivate Consistent Effort

Highlight Importance of Tutorials

Foster Higher Self-Efficacy

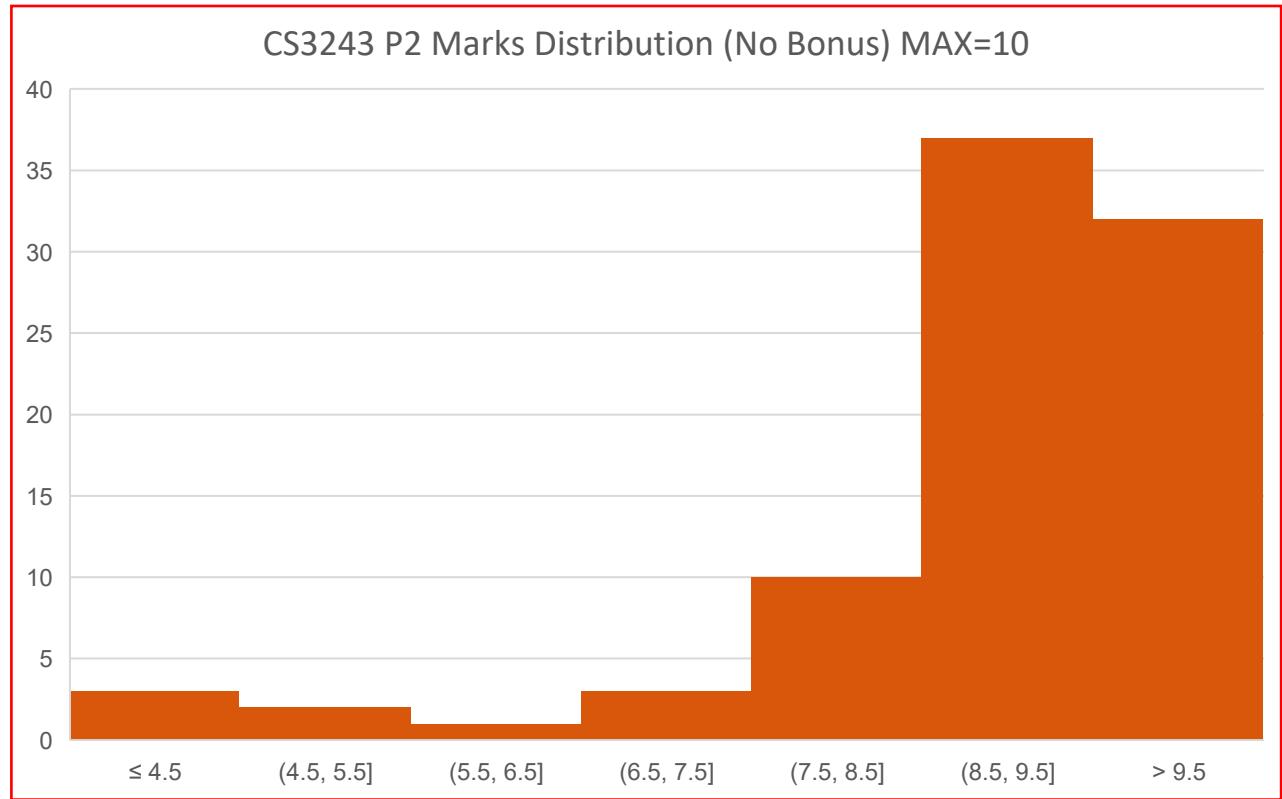
# Summary of Assessments

- Project 1.1 (MAX = 3)
  - Mean: 2.89
  - Median: 3
- Project 1.2 (MAX = 7)
  - Mean: 5.83
  - Median: 6.89



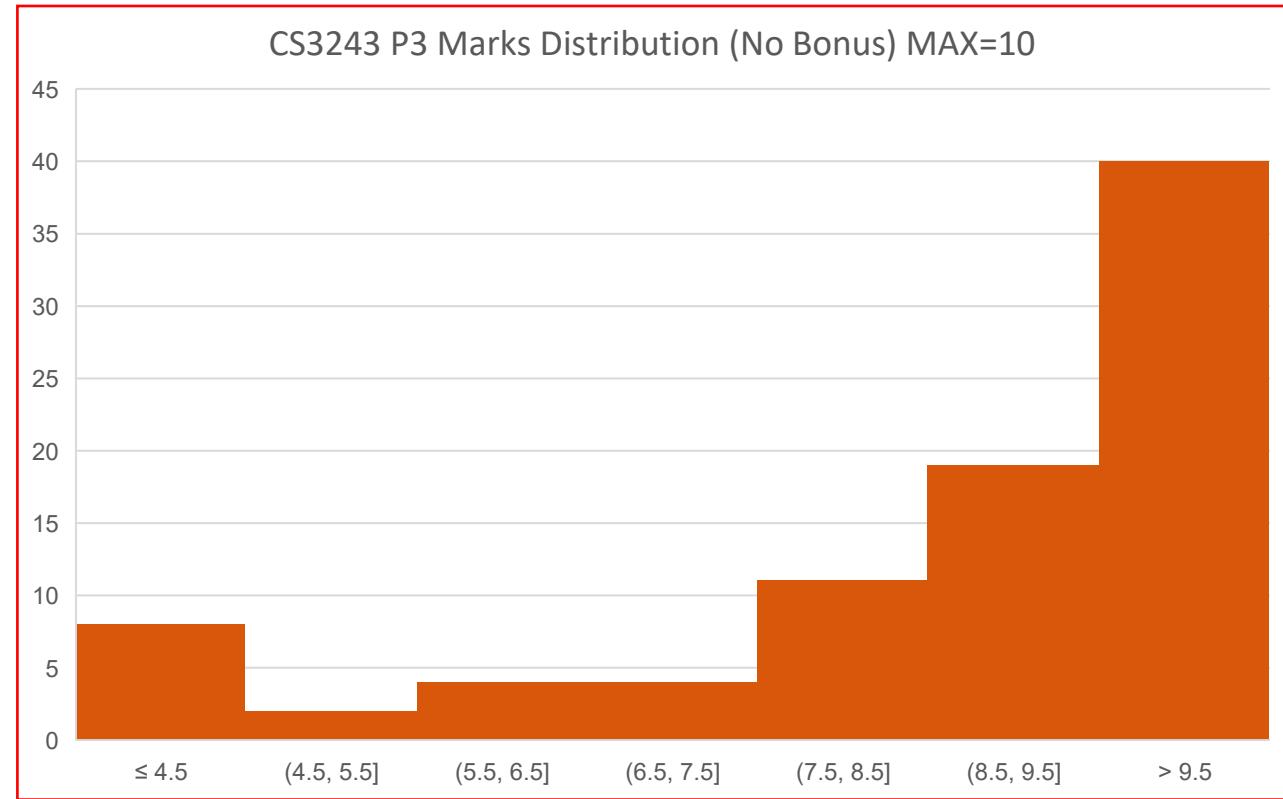
# Summary of Assessments

- Project 2.1 (MAX = 3)
  - Mean: 2.73
  - Median: 2.8
- Project 2.2 (MAX = 7)
  - Mean: 6.12
  - Median: 6.5
- Project 2 Bonus (MAX = 3)
  - Mean: 1.63
  - Median: 2.0



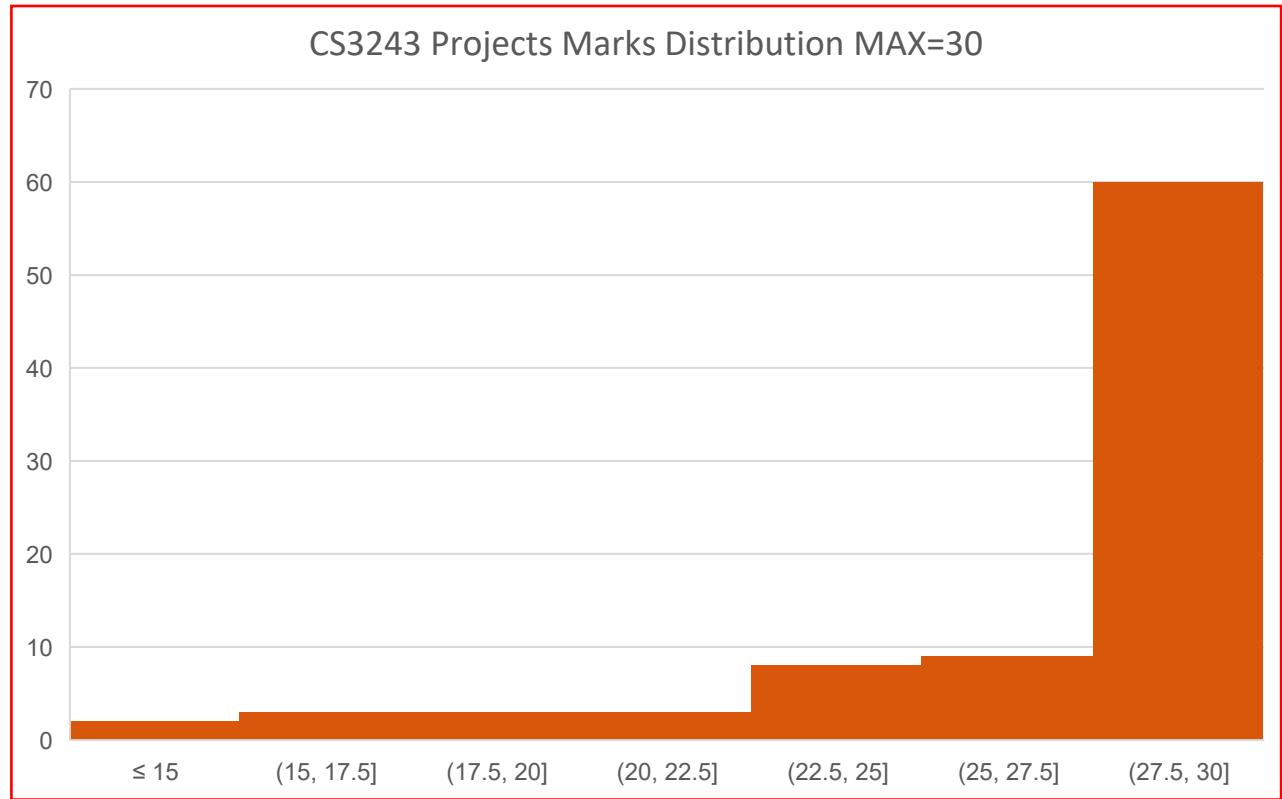
# Summary of Assessments

- Project 3 (MAX = 10)
  - Mean: 8.81
  - Median: 9.5
- Project 3 Bonus (MAX = 2)
  - Mean: 1.15
  - Median: 1



# Summary of Assessments

- Overall Projects (P – MAX = 30)
  - Mean: 27.34
  - Median: 30



# Summary of Assessments

- Why did I have to implement a representation, algorithms, etc.?

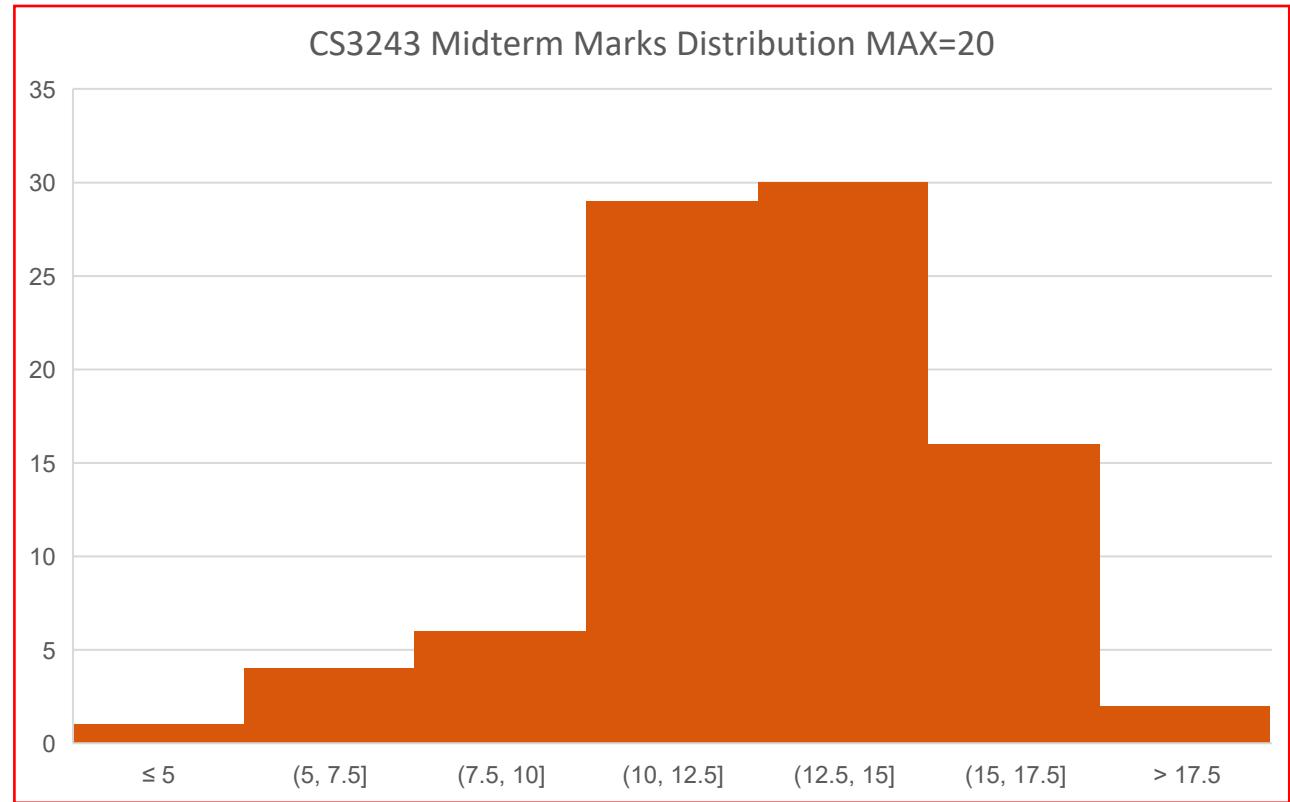
Enhances Algo. Understanding

Representation is Vital

Integration of Agent within  
Environment

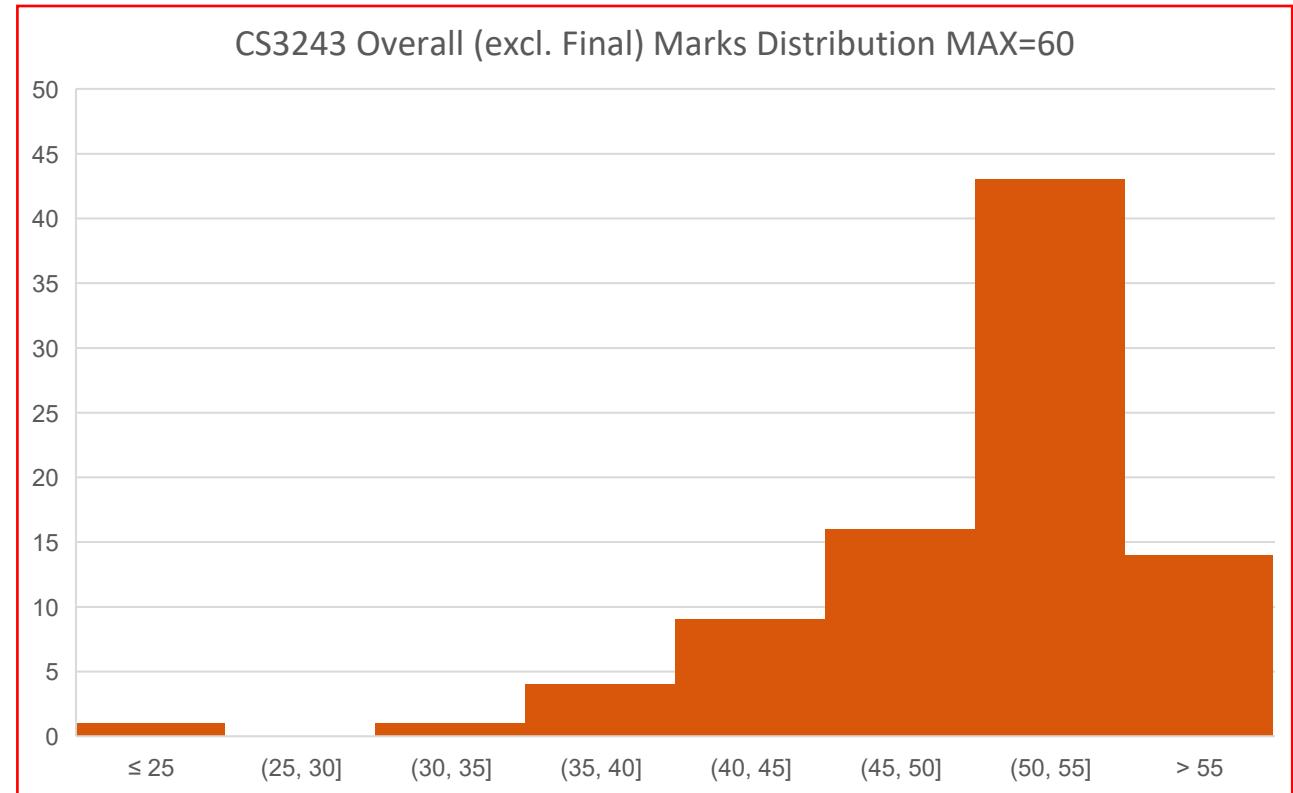
# Summary of Assessments

- **Midterm Q1 (%)**
  - Mean: **32.33**
  - Median: **30.0**
- **Midterm Q2 (%)**
  - Mean: **76.16**
  - Median: **80.0**
- **Moderated Midterm (MAX = 20)**
  - Mean: **12.67**
  - Median: **12.8**



# Summary of Assessments

- Overall  
**(AT+TA+P+MT – MAX = 60)**
  - Mean: **49.71**
  - Median: **51.6**



**4**

# Administrative Matters

# Appeals on Assessments

- Tutorial Attendance & Assignments
  - Deadline: **Next Monday (18 NOV), 2359 hrs**
- Project 3
  - Deadline: **Next Monday (18 NOV), 2359 hrs**
- Submitting Appeals
  - Email me: [dler@comp.nus.edu.sg](mailto:dler@comp.nus.edu.sg)

Check all marks on Canvas:  
Canvas > CS3243 > Grades

# Final Assessment

- **Schedule & Venue**
  - Tuesday (3 DEC), 0900-1100 hrs
  - MPSH 1A
- **Format (similar to the Midterm)**
  - Duration: **2 hours**
  - Total: **60 marks**
  - Closed-book + Cheat Sheet (1 × Double-sided A4 Sheet)
- **Topics**
  - All topics
  - About 75% of the paper will focus on topics not covered in the Midterm
- **Practice Papers**
  - [Canvas > CS3243 > Files > Past Papers > Final Papers](#)

You **MUST** confirm your Finals  
**Schedule on the Student Portal**  
(including time and venues)

# Student Feedback Exercise



## Your Voice Matters!



### Be Constructive

Comments on your learning experience increase the value of your feedback.



### Be Specific

Provide examples of how you think your teacher or the way the module is organised have helped (or not helped!) your learning.



### Be Considerate

Improper language or personal comments are highly inappropriate, and undermine your feedback. Abusive comments are unacceptable.



### Your feedback counts

Your constructive feedback helps professors to improve their modules and is one source of evidence for the university's appraisal decisions.



### It's confidential

Your professors will never see your name. They will only get an aggregate report after the exam results have been released.



### It's quick

Complete your module feedback on campus, at home, or on the go! It is easy to use and mobile compatible.



Provide your feedback now >>

<https://blue.nus.edu.sg/blue/>



# CS3243 Teaching Team Positions

- Work on projects during the Break
  - Applications open from now ...
  - ... or until roles filled
- TA for AY24/25 Semester 2
  - Apply before Week 0 next semester
  - The sooner the better ...
- Contact me if interested
  - Email: [dler@comp.nus.edu.sg](mailto:dler@comp.nus.edu.sg)

# Questions about the Lecture?

- Was anything unclear?
- Do you need to clarify anything?
- Ask on Archipelago
  - Specify a question
  - Upvote someone else's question



Invitation Link (Use NUS Email --- starts with E)  
<https://archipelago.rocks/app/resend-invite/92727263219>