Artificial Intelligence

State-space search and toy problems

The notes for the lesson “State-space search and toy problems”: chapters 1.7 to 1.8, with their figures, worked examples, exam traps, quizzes and exam question bank, in one file.

2 chapters | Chapters 1.7 to 1.8

About these notes

These are the notes for the University free n easy video lesson **State-space search and toy problems** on Artificial Intelligence: chapters 1.7 to 1.8, with every figure, worked example, exam trap and quiz, and the long-answer exam questions on these chapters.

1.7 State-space search, control strategies and production systems

Once a problem is formulated, solving it means searching its state space for a path from the start to a goal. Classic AI organises that search as a **production system**: a set of if–then rules, a working memory holding the current state, and a **control strategy** that decides which rule to fire next.

You need: the five components of a problem from Chapter 1.6.

Searching a state space

Searching the state space builds a **search tree**: the root is the start state, the branches are paths tried so far. The same state can appear on many branches, so a careful search remembers visited states and skips repeats (graph search). Two numbers measure the size: the **branching factor** b, the number of successors of a state, and the **depth** d of the nearest goal. A tree explored to depth d has about b^d nodes at its deepest level.

Data-driven and goal-driven search

- **Data-driven (forward) search** starts from the given facts and applies rules to produce new states until it reaches a goal. Use it when all the data is given at the start, there are many possible goals, or a goal is hard to state in advance (configuring a computer, as R1/XCON did).
- **Goal-driven (backward) search** starts from the goal, finds rules that could produce it, makes their conditions subgoals, and works back to known facts. Use it when the goal is given (proving a theorem, checking "is it dengue?"), when forward branching is large, or when the goal tells you which data to gather.

The branching factors decide. To check whether you descend from an ancestor 10 generations back, searching forward through 3 children per family visits 3^{10} = 59,049 people; searching backward through 2 parents per person visits 2^{10} = 1,024, about 58 times fewer. Search in the direction with the smaller branching factor.

Control strategies

Rich and Knight set two requirements. A control strategy must **cause motion**: always firing the first rule in the list can loop for ever (fill a jug, empty it, fill it). It must be **systematic**: miss no part of the space and search none twice without need. Breadth-first search (level by level) and depth-first search (one path deep, then back up) are the basic systematic strategies (Unit 2).

Strategies are also **irrevocable** or **tentative**. An irrevocable strategy never reconsiders a rule once fired (hill climbing). A tentative one keeps the chance to return: **backtracking** remembers choice points and goes back to the last one on failure; **graph search** keeps many paths open at once. A **heuristic** uses knowledge of the domain to try the more promising rule first.

Production systems

A production system has three parts: a **rule base** of productions, each "condition → action"; a **working memory** (global database) holding the current state; and a **control system** (rule interpreter) that runs the **recognise–act cycle** of Figure 1.7.1 until the goal test succeeds or no rule applies. When several rules match, the set of matching rules is the **conflict set**, and **conflict resolution** picks one: by rule order, refraction (do not fire the same rule on the same data twice), recency (prefer rules matching the newest facts) or specificity (prefer the rule with the more detailed condition).

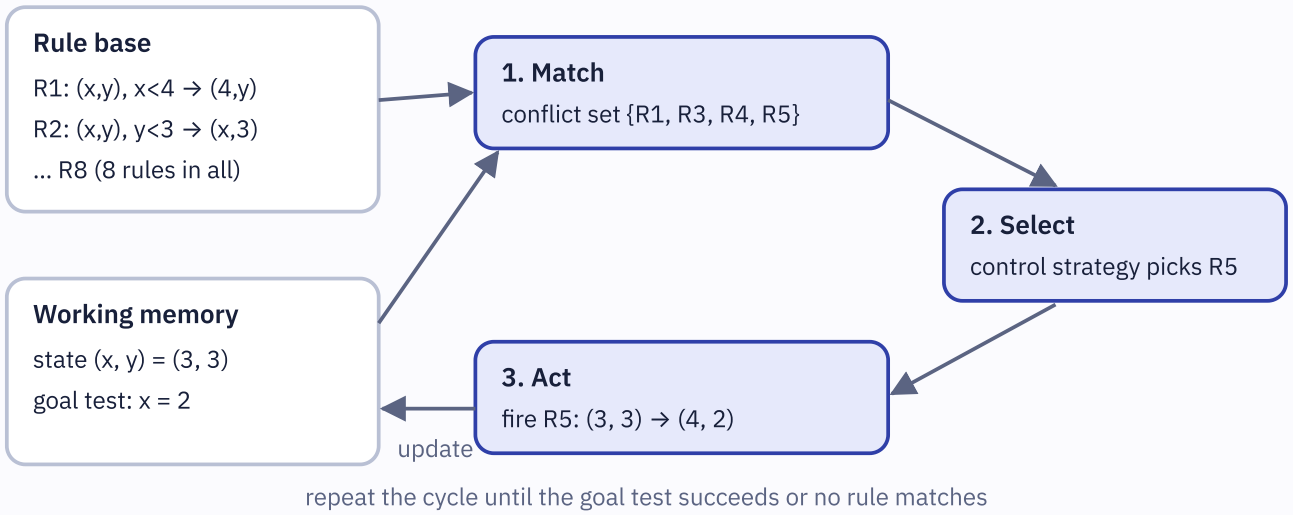


Figure 1.7.1 The recognise–act cycle at cycle 6 of the worked example: match builds the conflict set, the control strategy picks one rule, and firing it updates working memory. Notice that rules never call each other: all the sequencing lives in the control strategy.

Rich and Knight classify production systems by two properties. **Monotonic**: firing a rule never stops another rule that was also applicable at that time from being fired later. **Partially commutative**: if a sequence of rules turns state x into state y, every allowable reordering of those rules also turns x into y. A system with both is **commutative**.

	MONOTONIC	NON-MONOTONIC
Partially commutative	Theorem proving	Robot navigation
Not partially commutative	Chemical synthesis	Bridge (card game)

Rules are small and independent, so knowledge is easy to add or change; expert systems use them (Unit 6). The cost: control flow is hard to follow, and matching every rule each cycle is slow.

WORKED EXAMPLE: THE WATER-JUG PROBLEM

Question (12 marks). You have a 4-litre jug X and a 3-litre jug Y with no markings, and a tap. Get exactly 2 litres into X. (a) Write it as a production system. (b) Trace it with the control strategy "fire the first rule, in the order R1 to R8, that gives a state not seen before". (c) Is that solution the shortest?

(a) State (x, y): litres in X and Y, $0 \le x \le 4$, $0 \le y \le 3$. Start (0, 0). Goal: x = 2.

```
R1  x < 4                → (4, y)            fill X
R2  y < 3                → (x, 3)            fill Y
R3  x > 0                → (0, y)            empty X
R4  y > 0                → (x, 0)            empty Y
R5  x+y ≥ 4, y > 0, x < 4 → (4, y−(4−x))   pour Y into X until X is full
R6  x+y ≥ 3, x > 0, y < 3 → (x−(3−y), 3)   pour X into Y until Y is full
R7  x+y ≤ 4, y > 0       → (x+y, 0)          pour all of Y into X
R8  x+y ≤ 3, x > 0       → (0, x+y)          pour all of X into Y
```

(b) The trace:

CYCLE	WORKING MEMORY	CONFLICT SET	FIRED	NEW STATE
1	(0, 0)	R1, R2	R1	(4, 0)
2	(4, 0)	R2, R3, R6	R2	(4, 3)
3	(4, 3)	R3, R4	R3	(0, 3)
4	(0, 3)	R1, R4, R7	R7	(3, 0)
5	(3, 0)	R1, R2, R3, R6, R8	R2	(3, 3)
6	(3, 3)	R1, R3, R4, R5	R5	(4, 2)
7	(4, 2)	R2, R3, R4, R6	R3	(0, 2)
8	(0, 2)	R1, R2, R4, R7	R7	(2, 0) goal

In cycles 4 to 8 an earlier rule was skipped because it led to a state already seen, such as R1 from (0, 3) back to (4, 3).

Answer. (b) 8 rule firings reach (2, 0). (c) No. Only 14 of the 20 pairs (x, y) can be reached, and breadth-first search over them (Figure 1.7.2) finds two shortest solutions of 6 firings: R2, R7, R2, R5, R3, R7 ending at (2, 0), and R1, R6, R4, R8, R1, R6 ending at (2, 3). The strategy wasted

cycles 2 and 3 on a detour through (4, 3): it avoided loops, but like depth-first search it does not guarantee the shortest path.

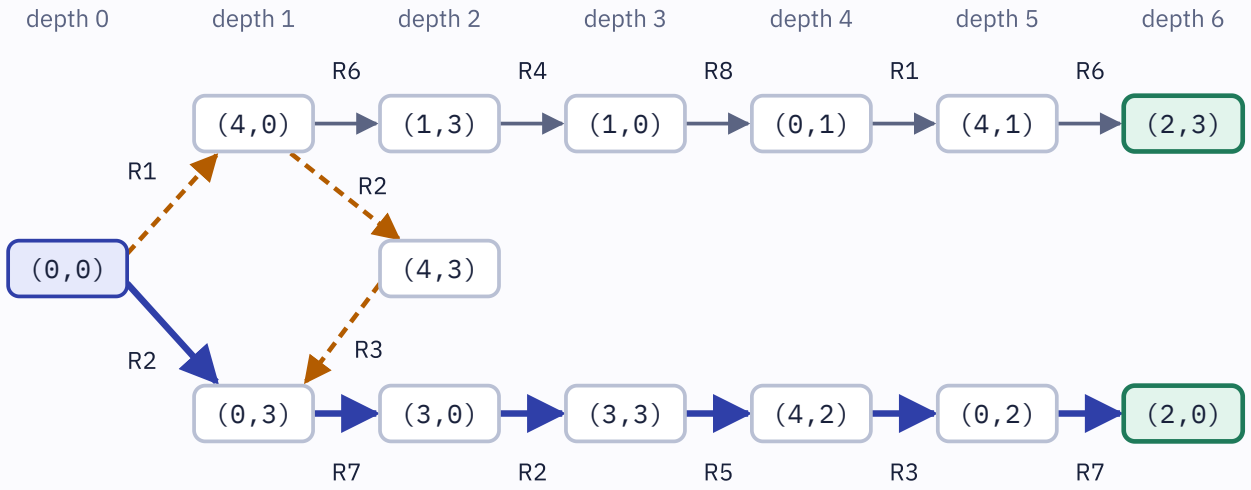


Figure 1.7.2 All 14 reachable states of the water-jug problem, placed by their depth (fewest firings from the start). The two green states are goals at depth 6. Blue arrows: a shortest solution, the bottom row. Orange dashed arrows: the detour taken by the first-new-state strategy before it joined the bottom row. Notice that every reachable state has a jug full or empty; states such as (2,2) never occur.

EXAM TRAPS

- Data-driven = forward from the facts; goal-driven = backward from the goal. Justify the choice by branching factor.
- A control strategy must "cause motion" and "be systematic"; give an example of failing each.
- The conflict set is all matching rules; conflict resolution picks one. Do not call the chosen rule the conflict set.
- In water-jug answers, write every state as an ordered pair and name each rule fired; the trace earns the marks.

CHECK YOURSELF

– `REMEMBER` **Name the three parts of a production system.** `2 marks`

A rule base of condition → action rules, a working memory (global database), and a control system that chooses and fires rules.

– REMEMBER **State the two requirements of a good control strategy.** 2 marks

It must cause motion (make progress, not loop) and it must be systematic (cover the space without needless repeats).

– UNDERSTAND **When is goal-driven search better than data-driven search?** 4 marks

When the goal is given, when many rules match the facts (large forward branching), and when data must be gathered and the goal guides which data to collect, as in diagnosis.

– APPLY **Using the rules of the worked example, give the state after each step of R1, R6, R4, R8, R1, R6 from (0, 0).** 3 marks

(4,0) → (1,3) → (1,0) → (0,1) → (4,1) → (2,3). The last state has x = 2, so it is a goal.

– APPLY **Is the water-jug production system monotonic? Justify with a state.** 3 marks

No. At (3, 0) both R6 and R8 apply. Firing R6 gives (0, 3), where R8 no longer applies because x = 0. Firing one rule stopped another that was applicable, so it is non-monotonic.

– ANALYZE **A rule base has 50 facts and 10,000 rules; most rules lead towards conclusions nobody asked about, and the question is "is the pump faulty?". Which search direction would you choose, and why?** 4 marks

Goal-driven. The goal is given, and forward chaining through 10,000 rules would fire many rules towards irrelevant conclusions. Backward chaining examines only rules that conclude "pump faulty" and their conditions.

BEYOND THE SYLLABUS

This is not for the exam. Rule engines such as Drools and the CLIPS shell match rules quickly with the Rete algorithm; cognitive architectures such as Soar and ACT-R model human thinking as production rules. Search next for "Rete algorithm", "CLIPS rule engine" and "Soar cognitive architecture".

1.8 Toy problems: tic-tac-toe, missionaries and cannibals, TSP

A **toy problem** is a small, exactly stated puzzle used to explain and compare problem-solving methods; a **real-world problem** is one people need solved, such as routing delivery

vans. This short chapter formulates three classic ones, tic-tac-toe, missionaries and cannibals and the travelling salesman problem, in the form of Chapter 1.6.

Tic-tac-toe

State: a 3 × 3 board, each cell X, O or empty, plus whose turn it is. Initial state: empty board, X to move. Actions: put your mark in an empty cell. Terminal test: three in a row, or a full board. Utility: +1 win, 0 draw, −1 loss. There are 3^9 = 19,683 ways to fill the board, but only 5,478 are legal positions, and 255,168 different games can be played. Rich and Knight's first program stored a reply for each of the 19,683 boards: fast but rigid. Better programs look ahead with minimax (Unit 3).

Missionaries and cannibals

Three missionaries and three cannibals must cross a river in a boat holding one or two. On neither bank may cannibals outnumber missionaries who are present. State (m, c, b): missionaries and cannibals on the left bank, and the boat's side. Start (3, 3, L); goal (0, 0, R). Actions: carry 1M, 2M, 1C, 2C or 1M+1C. A state is safe when (m = 0 or m ≥ c) and (3 − m = 0 or 3 − m ≥ 3 − c). The shortest solution takes 11 crossings.

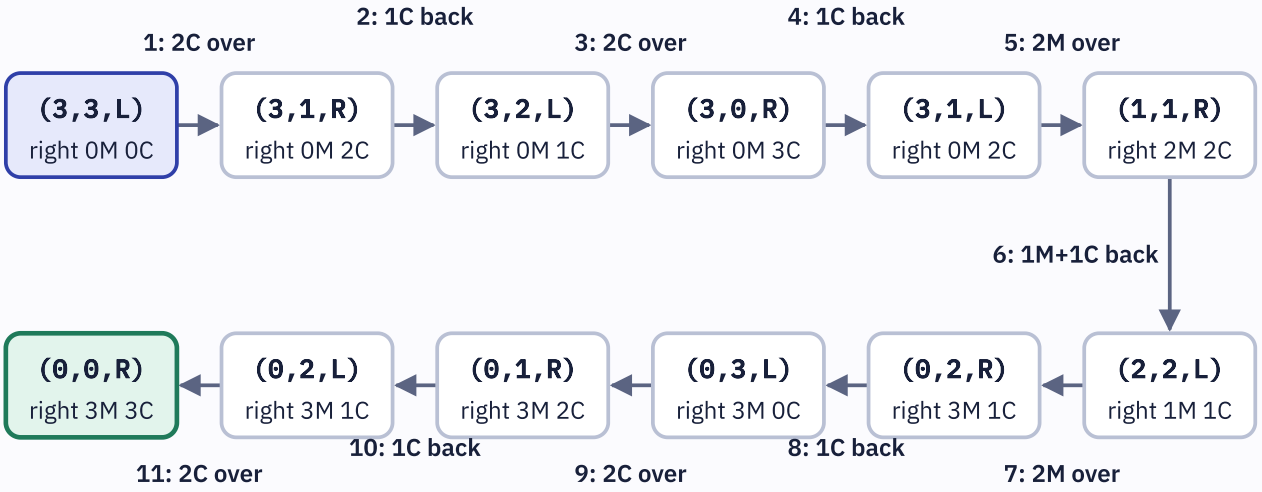


Figure 1.8.1 The 11-crossing solution: each box gives the state and, below, who is on the right bank. Read the top row left to right, then the bottom row right to left. Notice move 6: a missionary and a cannibal come back together, the step most students miss.

Travelling salesman problem (TSP)

Visit each of n cities exactly once and return to the start with the least total distance. State: the cities visited so far and the current city. Actions: go to an unvisited city. Goal test: all visited and back at the start. Path cost: total distance. With symmetric distances there are (n − 1)!/2

different tours: 12 for 5 cities, 181,440 for 10, about 6.1×10^{16} for 20. TSP is NP-hard, so large cases use heuristics such as **nearest neighbour**: always go to the closest unvisited city. Russell and Norvig count TSP as a real-world problem (circuit-board drilling, delivery routes).

WORKED EXAMPLE

Question (6 marks). A delivery rider starts at restaurant R and must visit flats A, B and C and return. Distances in km: R–A 2, R–B 3, R–C 6, A–B 4, A–C 5, B–C 7. (a) How many different tours are there? (b) Find the optimal tour. (c) Find the nearest-neighbour tour from R and compare.

1. (a) n = 4 places: (4 − 1)!/2 = 6/2 = 3 tours.
2. (b) R-A-B-C-R = 2 + 4 + 7 + 6 = 19 km. R-A-C-B-R = 2 + 5 + 7 + 3 = 17 km. R-B-A-C-R = 3 + 4 + 5 + 6 = 18 km.
3. (c) From R the nearest is A (2 < 3 < 6). From A the nearest unvisited is B (4 < 5). Then C (7), then back to R (6): 2 + 4 + 7 + 6 = 19 km.

Answer. (a) 3 tours. (b) R → A → C → B → R, 17 km. (c) Nearest neighbour gives 19 km, the worst tour, 12% longer (19/17 ≈ 1.12): each greedy step looked cheap but left the long B–C road for later.

EXAM TRAPS

- 3^9 = 19,683 counts board fillings, not legal positions (5,478) or games.
- The missionaries-and-cannibals state must include the boat's side; check safety on both banks.
- Use (n − 1)!/2 for symmetric TSP; (n − 1)! counts each tour twice, once in each direction.

CHECK YOURSELF

– REMEMBER **Give the state, start and goal for missionaries and cannibals.** 2 marks

State (m, c, b): missionaries and cannibals on the left bank and boat side. Start (3, 3, L); goal (0, 0, R).

– UNDERSTAND **Why must the boat's position be part of the state?** 2 marks

The same people on each bank allow different actions depending on the boat's side, so without it the transition model is undefined.

– APPLY **Which of these states are safe: (1, 2, R), (2, 2, L), (0, 1, L)?** 3 marks

(1, 2, R): unsafe (1M with 2C on the left). (2, 2, L): safe (2M 2C left, 1M 1C right). (0, 1, L): safe (no M left; 3M 2C right).

– APPLY **How many different tours does a symmetric TSP with 6 cities have?** 1 mark

$(6 - 1)!/2 = 120/2 = 60$.

– ANALYZE **Compare the three problems by number of agents and by what a solution is.** 4 marks

Tic-tac-toe is two-player and adversarial; its solution is a strategy (a reply to every move). Missionaries and cannibals is single-agent; its solution is a path. TSP is single-agent; its solution is the cheapest tour, a relative (best-path) solution. All three are fully observable, deterministic and discrete.

BEYOND THE SYLLABUS

This is not for the exam. Exact solvers such as Concorde prove optimal tours for very large TSPs, and delivery apps solve vehicle-routing problems with tools like Google OR-Tools. Tic-tac-toe is solved: perfect play always draws. Search next for "Concorde TSP solver" and "OR-Tools vehicle routing".

Key terms of this lesson

TERM	MEANING OR FORMULA	CHAPTER
Production system	Rule base + working memory + control strategy (match, select, act)	1.7
Search direction	Forward (data-driven) or backward (goal-driven): pick the smaller branching factor	1.7
Symmetric TSP tours	$(n - 1)!/2$	1.8

Exam question bank

Long-answer exam questions on these chapters, each with an outline of the points an examiner looks for. Answer one in full on paper before you read its outline.

Apply

- **Q8. Formulate the water-jug problem (4-litre and 3-litre jugs, get 2 litres in the 4-litre jug) as a production system. Give a solution and explain the role of the control strategy. (12 marks)**

  Outline: state (x, y), start (0, 0), goal x = 2; the eight rules; a shortest solution of 6 firings, e.g. (0,0) → (0,3) → (3,0) → (3,3) → (4,2) → (0,2) → (2,0); the recognise–act cycle; a first-new-state strategy reaches the goal in 8 firings; requirements of motion and systematicity.

- **Q9. Formulate the missionaries and cannibals problem, state how many states are safe, and give a solution. (12 marks)**

  Outline: state (m, c, b); five actions; safety condition on both banks; 10 safe (m, c) pairs × 2 boat sides = 20 safe states out of 32; the 11-crossing solution of Figure 1.8.1 with the move at each step.

- **Q10. Derive the number of distinct tours in a symmetric TSP and evaluate it for 5, 10 and 20 cities. Apply the nearest-neighbour heuristic to the delivery example of Chapter 1.8 and compare with the optimum. (10 marks)**

  Outline: fix the start, $(n - 1)!$ orders, divide by 2 for direction; 12, 181,440, about 6.1×10^{16}; nearest neighbour R → A → B → C → R = 19 km versus optimal R → A → C → B → R = 17 km; heuristics are fast but give no guarantee.

Analyze and evaluate

- **Q12. Compare data-driven and goal-driven search. Which would you use for medical diagnosis and which for configuring a computer system? Justify with branching factors. (10 marks)**

  Outline: definitions; when to use each; the $3^{10} = 59{,}049$ versus $2^{10} = 1{,}024$ ancestor example; diagnosis: goal-driven (hypothesis given, tests chosen by goal); configuration: data-driven (all requirements known, many possible final configurations).

- **Q13. Explain state-space search and control strategies. Compare irrevocable and tentative strategies, and explain the classes of production systems with examples. (12 marks)**

  Outline: state space as a graph, search tree, repeated states; requirements of a control strategy; BFS and DFS as systematic strategies; irrevocable (hill climbing) versus tentative (backtracking, graph search); monotonic, partially commutative and commutative systems with the four examples; water jug is non-monotonic (R6 versus R8 at (3, 0)).

- **Q15. Using the seven problem characteristics, explain why tic-tac-toe, the 8-puzzle and TSP need different solution methods. Give each formulation and the control strategy you would use. (12 marks; spans Chapters 1.6–1.8)**

  Outline: tic-tac-toe: irrecoverable, unpredictable opponent, solution is a strategy, game search (Unit 3). 8-puzzle: recoverable, predictable, solution is a path, systematic search with a heuristic, 181,440 reachable states. TSP: relative (best-path) solution, (n – 1)!/2 tours, exact search for small n, heuristics such as nearest neighbour for large n.

Glossary (terms used in this lesson)

Agent	Anything that perceives its environment through sensors and acts on it through actuators.
Branching factor	The number of successors of a state; with depth d it sets the search size, about b^d.
Conflict set	All rules whose conditions match working memory in one cycle.
Control strategy	The method that decides which rule or state to try next; it must cause motion and be systematic.
Data-driven search	Forward search from the given facts towards a goal.
Goal-driven search	Backward search from the goal through subgoals to known facts.
Heuristic	Domain knowledge used to choose the more promising option first, with no guarantee.
Path cost	The sum of the step costs along a path.
Production system	A rule base, a working memory and a control strategy running the recognise–act cycle.
State space	All states reachable from the initial state, as a graph of states and actions.
Transition model	Result(s, a): the state that results from doing action a in state s.

Sources and further reading

Textbooks

- Stuart Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th edition, Pearson, 2020. Chapters 1–3 cover this unit: introduction and history, intelligent agents, problem formulation (from memory; check).
- Elaine Rich, Kevin Knight and Shivashankar B. Nair, *Artificial Intelligence*, 3rd edition, Tata McGraw-Hill, 2009. Early chapters: AI problems and techniques, problem spaces, production systems, problem characteristics (from memory; check).
- George F. Luger, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*, 6th edition, Pearson, 2009. State-space search, data-driven and goal-driven search (from memory; check).
- Saroj Kaushik, *Artificial Intelligence*, Cengage Learning. An Indian textbook covering sub-areas of AI, state-space search and control strategies (from memory; check).
- Nils J. Nilsson, *The Quest for Artificial Intelligence*, 2010. A readable history of the field (from memory; check).

Open resources

- AIMA code repository, the textbook's algorithms in Python and Java: github.com/aimacode (from memory; not opened).
- Alan Turing, "Computing Machinery and Intelligence", *Mind*, 1950, and the 1955 Dartmouth proposal by McCarthy, Minsky, Rochester and Shannon: both are widely reprinted online (from memory; not opened).
- NPTEL, free video courses on artificial intelligence: nptel.ac.in (from memory; not opened).
- Our World in Data, charts on AI progress: ourworldindata.org/artificial-intelligence (from memory; not opened).
- ACM/IEEE-CS/AAAI Computer Science Curricula 2023 (CS2023): csed.acm.org (from memory; not opened).

The web was not reachable while this book was written, so the links above were not opened; the numbers in the worked examples were computed and checked by program.

Syllabus: the University free n easy universal syllabus, built from top universities' syllabi.

Built for the University free n easy YouTube channel from the universal syllabus of Artificial Intelligence (Oct 2026). Written with top universities' syllabi and standard textbooks; check a formula against your

prescribed book before an exam, and report mistakes on the channel.