

Cambridge AS & A Level · Computer Science 9618 · Notes

Artificial intelligence (AI) A2

Read online at pastopic.com/cambridge/computer-science-9618/notes/artificial-intelligence

Last checked 30 September 2026

© 2026 Pastopic. Free for personal study. Please share the link, not the file.

What you need to know

This is an A Level topic (Paper 3). By the end of it you should be able to:

1. **Explain how graphs help Artificial Intelligence (AI):** what a graph is for (recording relationships between things) and how it is built (nodes, edges and weights).
2. **Use Dijkstra's algorithm and the A* algorithm to search a graph** by hand: find shortest distances and shortest paths, and show your working. You will **not** be asked to write these algorithms in pseudocode or code, or to write code that sets up or reads a graph.
3. **Explain how artificial neural networks have helped machine learning.**
4. **Explain Machine Learning, Deep Learning and Reinforcement Learning, and why each one is used.**
This includes the two main categories of machine learning, **supervised** and **unsupervised** learning.
5. **Explain back propagation of errors and regression methods** in machine learning.

You only need to describe how these work; there are no formulas to learn beyond $f = g + h$ for A*.

Understand it

What AI and machine learning are

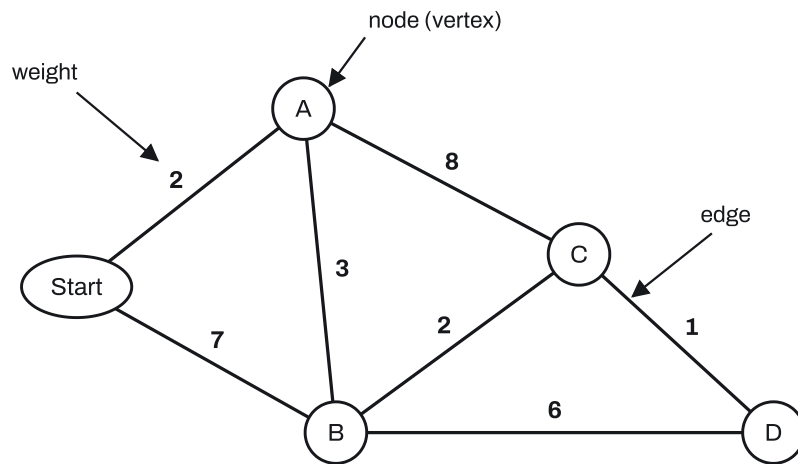
Artificial Intelligence is getting computers to do tasks that normally need human intelligence: recognising faces, understanding speech, planning a route, playing a game. **Machine learning** is the part of AI where the program is not given a rule for every case. Instead it is given data (or experience) and **improves its own performance** at the task from that data. A spam filter that has learned from thousands of emails can judge an email it has never seen, even though nobody wrote a rule for that email.

Graphs in AI

A **graph** is a way of recording **relationships between things**. It has two parts:

- **nodes** (also called **vertices**): the things themselves, such as towns, people, web pages or the positions in a game;
- **edges** (also called arcs): lines joining two nodes that are related, such as a road between two towns.

An edge can carry a **weight**, a number that means something in the problem: a distance, a travel time or a cost. A **path** is a list of nodes joined by edges, from one node to another. A **cycle** is a path that comes back to the node it started from.



Graphs help AI in several ways:

- **Many AI problems become "find a path in a graph".** A sat-nav stores places as nodes and roads as weighted edges, then searches for the shortest route. A puzzle or game can be stored the same way: each position is a node and each legal move is an edge.
- **Graph algorithms can then do the work.** Dijkstra's algorithm and the A* algorithm search a graph for the best path.
- **An artificial neural network is itself a graph:** its neurons are nodes and its weighted connections are edges (see below). Machine learning methods such as back propagation of errors work on this graph.

Dijkstra's algorithm

Purpose: find the **shortest distance** (or cheapest cost, or quickest time) from a start node to **every other node** in a weighted graph. The weights must not be negative.

The idea: keep a best-known distance for every node, and always settle the closest unsettled node next. Once a node is the closest one left, no other route can reach it more cheaply, so its distance is final.

1. **Set up.** Give the start node distance 0 and every other node ∞ (not reached yet).
2. **Visit** the unvisited node with the **smallest** distance. Mark it visited: its distance is now final.
3. **Update its neighbours.** For each unvisited neighbour, add the edge's weight to the visited node's distance. If that total is **smaller** than the neighbour's current distance, replace it. If not, leave it.
4. **Repeat** steps 2 and 3 until every node has been visited.

Here it is on the graph above. Each row is one visit; the bold value is the one that becomes final, and ✓ marks a node already visited.

Step	Visit	Start	A	B	C	D
0	(set up)	0	∞	∞	∞	∞
1	Start (0)	0	2	7	∞	∞
2	A (2)	✓	2	5	10	∞
3	B (5)	✓	✓	5	7	11
4	C (7)	✓	✓	✓	7	8
5	D (8)	✓	✓	✓	✓	8

Watch the updates: B starts at 7 (direct from Start) but drops to $2 + 3 = 5$ through A. C drops from 10 to $5 + 2 = 7$, and D from 11 to $7 + 1 = 8$. The shortest distances are A 2, B 5, C 7, D 8, and the shortest route to D is $\text{Start} \rightarrow \text{A} \rightarrow \text{B} \rightarrow \text{C} \rightarrow \text{D}$. Notice that the direct-looking routes ($\text{Start} \rightarrow \text{B}$, costing 7) are not always the shortest.

The A* algorithm

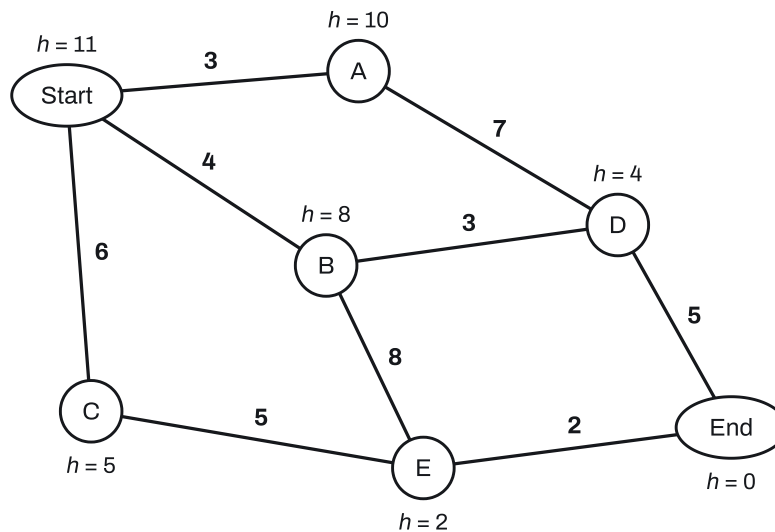
Purpose: find the **shortest path between two particular nodes**, a start and a goal, while exploring as few nodes as possible.

Dijkstra's algorithm spreads out evenly in every direction. A* is aimed: it uses a **heuristic**, an estimate of how far each node still is from the goal (for example the straight-line distance on a map). For each node it works out

$$f = g + h$$

- g : the **actual cost** of the best path found so far from the start to this node (add up the edge weights);
- h : the **heuristic**, the estimated cost from this node to the goal (it is given in the question, usually written beside each node);
- f : the estimated **total** cost of a path through this node.

A* always expands the node with the **lowest f** next, so it tries the most promising routes first and ignores nodes that are clearly heading the wrong way. If the heuristic never overestimates the real remaining distance, the first time the goal has the lowest f the path found is the shortest.



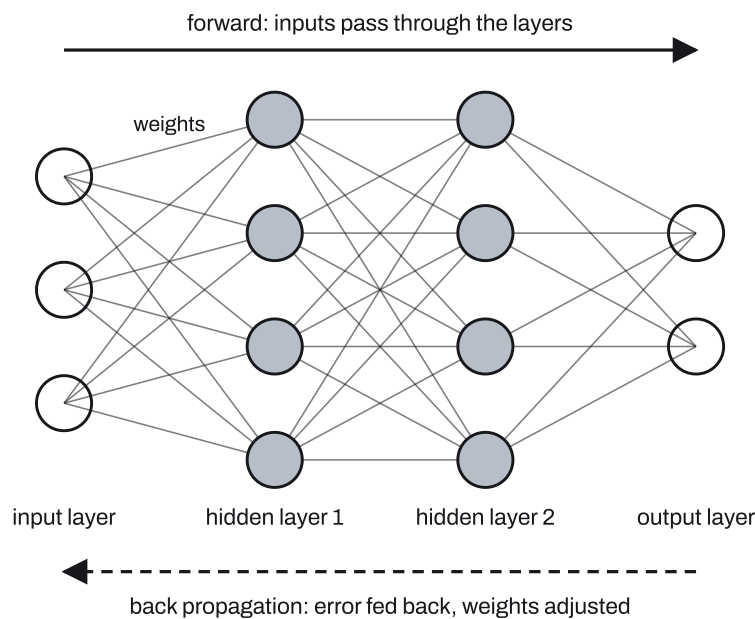
On this graph, the three nodes next to Start get f values: $A \ 3 + 10 = 13$, $B \ 4 + 8 = 12$ and $C \ 6 + 5 = 11$. C has the lowest f , so A* expands C first, even though B turns out to be on the best path. Example 4 finishes the search and shows how A* corrects itself.

How they differ: Dijkstra's algorithm uses only the real distances and explores every route outwards from the start, finding the shortest distance to all nodes. A* adds a heuristic that gives priority to nodes that seem closer to the goal, so it usually reaches the goal after exploring fewer nodes. If every h were 0, A* would behave just like Dijkstra's.

Artificial neural networks

An **artificial neural network (ANN)** is a program built to work in a similar way to a **human brain**. It is a large number of simple processing units (artificial **neurons**, or nodes) connected together and arranged in **layers**:

- the **input layer** receives the data (for example the pixels of an image);
- one or more **hidden layers** process it; each extracts features from the layer before;
- the **output layer** gives the result (for example "cat" or "dog").



Every connection has a **weight** that says how strongly one node's output affects the next node. A node combines the weighted values arriving at it and passes a result on to the next layer. The network's "knowledge" is its set of weights.

How ANNs made machine learning possible. Nobody writes rules into the network. It is **trained**: data is fed in at the input layer and passes through the hidden layers to the output layer, the output is checked, and the **weights are adjusted** to make the output more accurate. This is repeated many times with many examples. The network ends up making decisions it was never specifically programmed to make, it can solve problems that are too hard or too slow for humans (such as recognising speech), and it gets better as more data becomes available. More hidden layers let it learn more complex patterns and give more accurate results.

Back propagation of errors

Back propagation is how a neural network is trained, that is, how it works out which weights to change and by how much.

1. **Forward pass:** training data goes in and the network produces its outputs.
2. **Compare:** the actual outputs are compared with the **expected** (correct) outputs; the difference is the **error**.
3. **Find the error gradient:** calculus is used to work out how much each weight contributed to the error.
4. **Feed back:** the error is sent **backwards** through the network, from the output layer through the hidden layers towards the input layer.
5. **Adjust:** each weight is changed a little in the direction that **reduces the error**.
6. **Repeat** with more data, many times, until the outputs are accurate enough.

Machine learning categories

	Supervised learning	Unsupervised learning	Reinforcement learning
Data	Labelled: each input comes with its known correct output	Unlabelled: inputs only, no known outputs	No data set: an agent acts in an environment
How it learns	Learns to map inputs to outputs from example input–output pairs, then predicts the output for new inputs	Finds hidden patterns, groupings (clusters), similarities and differences in the data by itself	Trial and error: a good action earns a reward , a bad action a penalty ; it learns to maximise its reward
Human input	Needed to label the training data	None needed	None needed beyond setting the rewards
Example	Spam filter trained on emails marked "spam" or "not spam"; predicting house prices	Grouping customers by their shopping habits; spotting unusual transactions	Learning to play a game; a robot learning to walk

Why use each one:

- **Supervised:** when you have past data with the right answers, and you want to **predict future outcomes** from it.
- **Unsupervised:** when the data has no labels (most real data doesn't), or you want to **discover structure nobody knew about**; it needs no human to label anything.
- **Reinforcement:** when there is no data set of right answers, but the program can **try actions and see the results**, such as in games, robotics and self-driving; it keeps improving from feedback.

Deep learning

Deep learning is a specialised form of machine learning that uses artificial neural networks with **many hidden layers** (that is what "deep" means). Each layer builds on the one before, so the network extracts features step by step, from simple ones (edges in an image) to complex ones (a face). It can use supervised, unsupervised or reinforcement learning, and it is trained with back propagation of errors.

Why use deep learning:

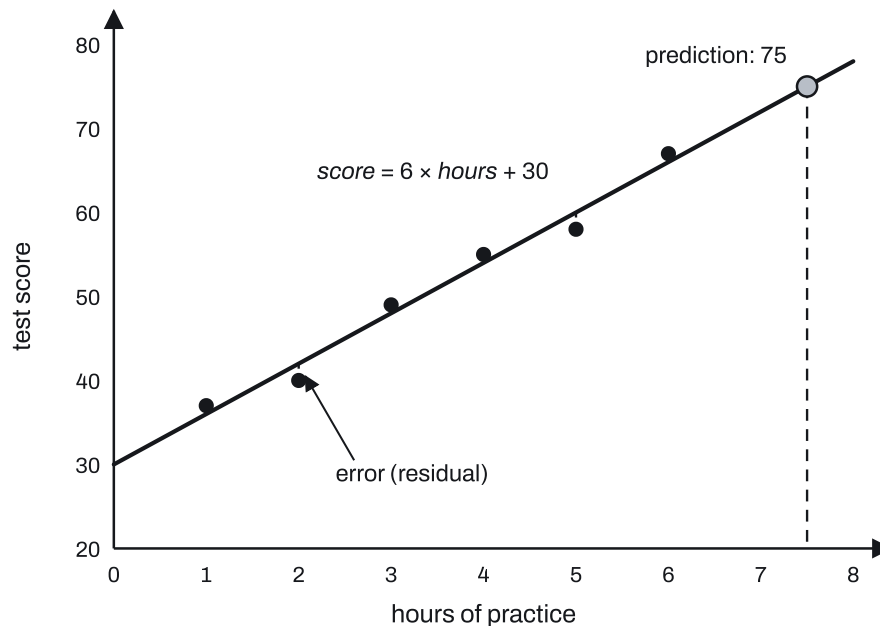
- it works on **unstructured data** such as images, sound and text, which is hard to describe with rules;
- it finds **hidden patterns** that humans can't see, or that would be too complex or too slow for humans to find;
- it **outperforms other methods when the data set is very large**;
- it handles **non-linear** relationships, and adding more hidden layers (with enough data) makes it more accurate.

Where it is used: image and face recognition, speech recognition and virtual assistants, natural language processing and chatbots (including large language models), machine translation, self-driving cars, fraud detection, medical image analysis, weather forecasting and game playing.

Regression methods

Regression is a machine learning method for **predicting a number** (a continuous value) from one or more inputs, such as a house price from its size, or a test score from hours of practice. It is supervised: it is trained on data where the correct outputs are known.

The simplest kind, **linear regression**, fits a straight line through the data points. The **error** (residual) of each point is the gap between its actual value and the line's prediction; the line chosen is the one that makes these errors as small as possible overall (by least squares, the smallest total of squared errors). The model then uses the line to predict values for new inputs.



Here the line of best fit is $\text{score} = 6 \times \text{hours} + 30$, so for 7.5 hours the model predicts $6 \times 7.5 + 30 = 75$. The point at 2 hours has score 40 where the line predicts 42: an error of -2 . Machine learning systems use regression to **find the relationship between variables** and **make predictions**, and the same idea (adjust the model's numbers to reduce the error) is what back propagation does inside a neural network.

Key facts and formulas

There is no formula list in the 9618 exams, so every fact here is **Learn it**.

Formula	Status
Graph: nodes (vertices) joined by edges ; edges can be weighted (distance, time, cost)	Learn it
Dijkstra's: shortest distance from the start to every node; set start = 0, others = ∞ ; visit the smallest unvisited; update a neighbour only if the new total is smaller	Learn it
A*: $f = g + h$, where g = actual cost from the start, h = heuristic (estimated cost to the goal); expand the lowest f	Learn it

Formula	Status
A* uses a heuristic to find the shortest path between two nodes, exploring fewer nodes; Dijkstra's explores all routes	Learn it
ANN: input layer → hidden layers → output layer; weighted connections; weights adjusted by training	Learn it
Back propagation: compare actual with expected output → error gradient (calculus) → feed error back → adjust weights → repeat	Learn it
Supervised = labelled data, predicts outputs; unsupervised = unlabelled data, finds hidden patterns/clusters; reinforcement = rewards and penalties , trial and error	Learn it
Deep learning = machine learning with ANNs that have many hidden layers ; best with large data sets	Learn it
Regression: predicts a numerical value by fitting a line or curve that minimises the errors	Learn it

How to do it

In the Paper 3 questions we have tagged for this topic (23 questions, 2021–2025, with repeats on the three variants of a paper counted on each), the types came up this often. 13 of the 23 questions mix types, so the counts add up to more than 23. Regression methods have not been asked yet, but they are in the syllabus.

Question type	Questions
Machine learning categories	7
Deep learning: what it is and why it is used	6
Dijkstra's algorithm: shortest distances	5
Graphs in AI: purpose and structure	5
Artificial neural networks and machine learning	5
Purpose of A* and Dijkstra's, and how they differ	4
A* algorithm: shortest path table	3
Back propagation of errors	1

1. Machine learning categories (7 questions)

- 1. Name the data first.** Supervised: **labelled** data (inputs with known outputs). Unsupervised: **unlabelled** data. Reinforcement: no data set, an agent getting **rewards or penalties**.
- 2. Say how it learns.** Supervised maps inputs to outputs from example pairs and **predicts future outcomes** from past data. Unsupervised **finds hidden patterns, clusters, similarities and differences** without human help. Reinforcement learns by **trial and error**, getting positive feedback for good actions and

negative feedback for bad ones.

3. **Add the human role** if it fits: supervised needs humans to label the training data; unsupervised and reinforcement don't.

Variations you will meet:

- **"State one category of machine learning"**: supervised, unsupervised, reinforcement or deep learning. One word is enough.
- **"Explain how supervised and unsupervised learning differ"**: write matched pairs in one sentence each, such as "supervised uses labelled data, **whereas** unsupervised uses unlabelled data", then "supervised needs human input to label the data, whereas unsupervised needs none", then "in unsupervised learning the outcomes are not known, so the AI searches for hidden patterns or clusters in the data".
- **"Describe supervised learning and unsupervised learning"** (two answer spaces): give two or three separate points for each. The examiners said fully correct answers were rare, usually because only the labelled/unlabelled point was made.
- **"Explain how unsupervised learning takes place"**: algorithms analyse and **cluster unlabelled** data sets; they **discover hidden patterns or groupings** with no human intervention; they spot **similarities and differences** in the data.
- **"Explain Reinforcement Learning"**: an **agent** learns how to behave in an **environment** by **performing actions and seeing the results**; each good action earns a **reward** and each bad one a **penalty**; it learns automatically from this feedback, **without labelled data** or specific instructions, improving its performance at similar tasks.
- **Matching categories to descriptions**: supervised = learns by **mapping inputs to outputs from example input–output pairs**; reinforcement = learns in an interactive environment by **trial and error from its own experiences**; deep learning = **imitates the brain's data processing** to make decisions; unsupervised = **discovers previously undetected patterns on its own**. A description about **passing error information back** through the network is back propagation, which is not a category, so it is left unmatched.

2. Deep learning: what it is and why it is used (6 questions)

1. **Define it**: deep learning is a **branch (specialised form) of machine learning** that uses **artificial neural networks**, modelled on the **human brain**.
2. **Structure**: many interconnected layers, an **input layer**, **many hidden layers** and an **output layer**; the input layer feeds the first hidden layer and the last hidden layer feeds the output layer.
3. **How it learns**: the layers **progressively extract features** and find **hidden patterns** in the data; it needs **large quantities of data**; it uses **back propagation of errors** to adjust its weights and improve its accuracy; more hidden layers give more accurate results.

Variations you will meet:

- **"Outline the reasons for using deep learning"**: give **reasons**, not a description. It copes with images, sound and text that don't fit into tables; it beats simpler methods once there is a lot of data; handles non-linear data; finds patterns too complex or time-consuming for humans; more accurate with more hidden layers. The examiners saw many answers that described deep learning here instead.
- **"State one example where deep learning is used"**: one application, such as image or facial recognition, a chatbot, a self-driving car or machine translation.
- **"Identify how deep learning can be made more effective"**: **add more hidden layers**.
- **"State the reason for having multiple hidden layers"**: they enable deep learning, handle more complex problems, and improve the accuracy of the result.

3. Dijkstra's algorithm: shortest distances (5 questions)

The question gives a weighted graph and a table to fill in with the shortest distance from the start to each node. It is worth 5 marks: up to 3 for **working** and 2 for the **answers**.

1. **Initialise, in writing**: $\text{start} = 0$, every other node $= \infty$. This earns marks on its own, so show it (on the graph beside each node, or in the working space).
2. **Visit the node with the smallest distance** that is not yet visited, and **mark it visited** (tick it, circle it or write it in a "visited" list).
3. **Update each unvisited neighbour**: write the sum, such as $13 + 5 = 18$, and if it beats the current value, **cross out the old value** and write the new one.
4. **Repeat** until every node is visited, then copy each final value into the answer table.

A trace table like the one in "Understand it" shows every mark point at once: the set-up row, updated values, visited nodes, and at least one node with more than one route calculated.

Variations you will meet:

- **Named towns instead of letters** ("Base", "Town 1" ...): the method is the same.
- **A node reachable by several routes**: the answer is the smallest total, and the working should show the other totals you rejected (for example 19 via one route and 18 via another).
- **Check with the answer table**: every node in the table needs a value. Four or five correct out of six still earns 1 of the 2 answer marks.

4. Graphs in AI: purpose and structure (5 questions)

1. **Purpose**: a graph **records relationships between entities**, using **nodes (vertices) and edges**; for example, places on a map and the distances between them, so that the **shortest route** can be found.
2. **Structure**: **nodes** represent entities (places, people); **edges** connect related nodes and show the possible paths between them; nodes and edges can be **labelled or weighted** (distance, cost, time) for the application; a **path** is a list of nodes joined by edges; a **cycle** is a path that returns to its starting node.
3. **"Explain the use of graphs to aid AI"** (3 marks): graphs provide a **structure for relationships**; AI problems can be solved as **finding a path in a graph**; graphs can be searched by algorithms such as **A*** and **Dijkstra's**; **artificial neural networks can be represented as graphs**, used in machine learning with

methods such as **back propagation of errors** and **regression**.

5. Artificial neural networks and machine learning (5 questions)

1. **What it is:** an artificial neural network is designed to **simulate the way the human brain works**; it is a key component of machine learning.
2. **Structure:** a **large number of connected processing units (nodes)**, arranged in **layers** (input, hidden, output) that work together; many of the layers are hidden.
3. **Weights:** a **weight** is assigned to each connection between nodes.
4. **Learning:** data is **input at the input layer, analysed at each hidden layer** where features are extracted, and the **output layer gives the result; weights are adjusted through training** (back propagation of errors) to make the results more accurate, and this is **repeated many times**.
5. **Why it matters:** it has **self-learning** capabilities, makes decisions **without being specifically programmed** for them, solves problems too hard for humans, and gives better results with more data and **more hidden layers**.

Variations you will meet:

- **"Explain how artificial neural networks enable machine learning":** describe the **training process** (points 3 and 4). The examiners said that just describing supervised and unsupervised learning did not answer this.
- **"Explain the use of artificial neural networks in deep learning":** brain-like design; they give the **architecture and algorithms for learning from data**; many connected nodes in layers; deep learning models **learn by adjusting the weights** of the connections; **multiple hidden layers extract complex features** and make predictions.

6. Purpose of A* and Dijkstra's, and how they differ (4 questions)

1. **Purpose (both):** to find the **optimal (shortest, cheapest or quickest) path between two nodes on a graph**, based on distance, cost or time.
2. **Difference:** A* uses a **heuristic** function that estimates the distance to the goal and **gives priority to nodes that seem better** (closer to the goal), so it tries the most promising routes first; **Dijkstra's explores all possible routes** using only the actual distances.

Variations you will meet:

- **"Identify one AI algorithm to find the shortest distance between two points on a graph":** "the A* algorithm" or "Dijkstra's algorithm".
- **"Outline the difference"** (2 marks): one point about A* and its heuristic, then the matching point about Dijkstra's, joined with "whereas".

7. A* algorithm: shortest path table (3 questions)

The question gives a graph with a heuristic h beside each node and a table with columns for the start node, the destination node, g , h and $f = g + h$. The first rows are filled in. It is worth 5 marks: up to 4 for correct rows and 1 for the final path.

1. **Expand the start node:** for each neighbour write a row with g = the edge weight, its h , and $f = g + h$.
2. **Choose the open node with the lowest f** (a node you have reached but not yet expanded).
3. **Expand it:** for each neighbour not yet expanded, g = the chosen node's g + the edge weight. Write it as a sum, such as $5 + 2 = 7$. Write a row if the neighbour is new, or if this g is **lower** than its current g .
4. **Repeat** steps 2 and 3. **Stop when the goal node has the lowest f** of all the open nodes.
5. **Final path:** trace back from the goal through the "start node" column of the rows that gave each node its best g , and write the path forwards, such as Start $\rightarrow$ B $\rightarrow$ D $\rightarrow$ End.

Variations you will meet:

- **The same node reached twice:** keep the lower g . A second row for a node with a lower g (for example B $1 + 3 = 4$ after B 5) earns a mark.
- **The goal reached, but not with the lowest f yet:** keep going. A row for the goal with $f = 14$ is not the end if another open node has $f = 12$; that node may lead to the goal more cheaply.
- **Extra correct rows** for nodes that don't turn out to matter do no harm; missing the rows on the way to the goal loses marks.

8. Back propagation of errors (1 question)

Write the steps in order (the six steps in "Understand it"): the **actual outputs are compared with the expected outputs**; **calculus finds the error gradient**; the **error is fed back** through the network; the **weights of each node are adjusted** to reduce the difference between actual and expected outputs; the process **repeats until the results are accurate enough**.

Worked examples

The bold codes show where marks are earned, as in Cambridge mark schemes. The number is how many marks.

- **B1** mark for a correct result on its own, with no method needed.
-

Example 1

A music app wants to use machine learning.

(a) The app could learn from songs its staff have already sorted into genres, or from listening data that nobody has sorted. Compare these two ways of training it. [4]

(b) The developers also build a program that teaches itself to play a rhythm game by trying moves and scoring points. Name this kind of machine learning and explain how the program learns. [3]

(a) Supervised learning uses **labelled** data, whereas unsupervised learning uses **unlabelled** data **B1**.

Labelled data means each input comes with its known correct output, such as songs already tagged with their genre, so the AI can learn to predict the genre of new songs **B1**. Supervised learning needs humans to provide the labels for training, whereas unsupervised learning needs no human input **B1**. In unsupervised learning the outcomes are not known, so the AI searches the data for hidden patterns or clusters, such as groups of listeners with similar tastes **B1**.

(b) This is **reinforcement learning**, which is based on **feedback: rewards and penalties B1**. An **agent** (the program) learns how to behave in an **environment** (the game) by **performing actions and seeing their results B1**. Each good action, such as hitting a note in time, earns a reward and each bad action a penalty, so it learns automatically to improve, without labelled data **B1**.

Example 2

A photo app uses an artificial neural network to recognise which pet (cat, dog or rabbit) is in a photo.

(a) Describe how the network's layers process a photo to reach an answer. [3]

(b) The developers plan to add more hidden layers. Give one benefit of doing this. [1]

(c) During training, the network labels a photo of a cat as "dog". Explain how the network's weights are corrected after this wrong answer. [3]

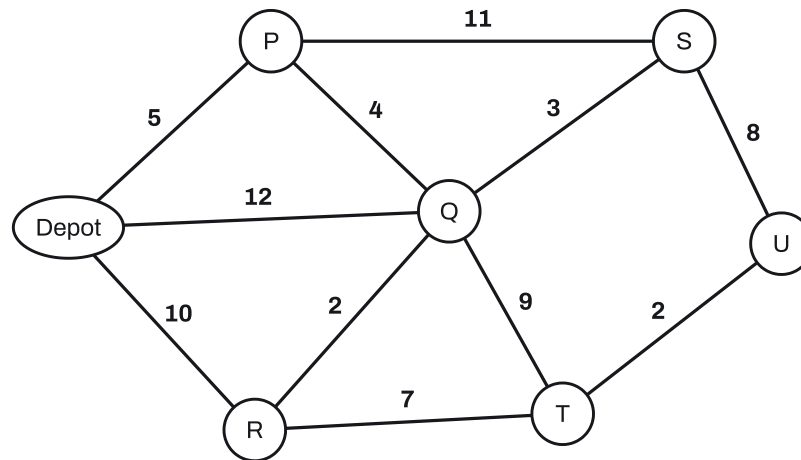
(a) The photo's pixel values enter the network at the **input layer B1**. Each **hidden layer** combines the values from the layer before, using the **weights** on the connections, and picks out features such as edges, then ears and fur patterns **B1**. The **output layer** gives a score for each pet, and the pet with the highest score is the network's answer **B1**.

(b) Extra hidden layers can pick out more complex features, so the app **recognises pets more accurately B1**.

(c) The network's output ("dog") is **compared with the expected output** ("cat") to measure the error **B1**. The error is **passed back through the network**, from the output layer towards the input layer, and each **weight is changed** so that the error would be smaller next time **B1**. This is **repeated** over many labelled photos until the network's answers are accurate enough **B1**.

Example 3

The graph shows the roads between a delivery depot and six shops, P to U. The weights are journey times in minutes.



Use Dijkstra's algorithm to find the shortest time from the Depot to each shop. Show your working. [5]

Working:

Step	Visit	Depot	P	Q	R	S	T	U
0	(set up)	0	∞	∞	∞	∞	∞	∞
1	Depot (0)	0	5	12	10	∞	∞	∞
2	P (5)	✓	5	9	10	16	∞	∞
3	Q (9)	✓	✓	9	10	12	18	∞
4	R (10)	✓	✓	✓	10	12	17	∞
5	S (12)	✓	✓	✓	✓	12	17	20
6	T (17)	✓	✓	✓	✓	✓	17	19
7	U (19)	✓	✓	✓	✓	✓	✓	19

- Row 0 sets the Depot to 0 and every shop to ∞ **B1**.
- Values are updated when a shorter route is found: Q drops from 12 to $5 + 4 = 9$; S from $5 + 11 = 16$ to $9 + 3 = 12$; T from $9 + 9 = 18$ to $10 + 7 = 17$; U from $12 + 8 = 20$ to $17 + 2 = 19$ **B1**.
- The visited nodes are ticked, and more than one route is worked out for Q, S, T and U **B1**. Visiting R gives $10 + 2 = 12$ for Q, which is **not** better than 9, so Q is not changed.

Answers:

P	Q	R	S	T	U
5	9	10	12	17	19

All six correct **B2** (four or five correct would earn **B1**).

Example 4

Use the second graph in "Understand it" (heuristics: Start 11, A 10, B 8, C 5, D 4, E 2, End 0).

(a) Use the A* algorithm to find the shortest path from Start to End. Show your working in a table. **[5]**

(b) Outline **one** difference between the A* algorithm and Dijkstra's algorithm. **[2]**

(a)

Start node	Destination node	Cost from start (g)	Heuristic (h)	Total ($f = g + h$)
Start	Start	0	11	11
Start	A	3	10	13
Start	B	4	8	12
Start	C	6	5	11
C	E	$6 + 5 = 11$	2	13
B	D	$4 + 3 = 7$	4	11
D	End	$7 + 5 = 12$	0	12

- Rows for A, B and C from Start **B1**. C has the lowest f (11), so it is expanded first.
- Row for E from C ($f = 13$) **B1**. Now B has the lowest f (12) of the open nodes A 13, B 12, E 13, so B is expanded. Its route to E costs $4 + 8 = 12$, worse than 11, so no row is written for it.
- Row for D from B ($f = 11$) **B1**, then D has the lowest f and is expanded.
- Row for End from D ($f = 12$) **B1**. End now has the lowest f of the open nodes (A 13, E 13, End 12), so the search stops.

Final path: **Start** → **B** → **D** → **End**, total cost 12 **B1**.

The search went to C first because C's heuristic made it look promising, then turned back to B when C's route turned out dearer. A* never expanded A or E.

(b) A* uses a **heuristic** function to give priority to nodes that are estimated to be closer to the goal **B1**, whereas Dijkstra's algorithm uses only the actual distances and **explores all possible routes B1**. (Dijkstra's on this graph also finds 12, but only after visiting all seven nodes.)

Common mistakes

- **Leaving out the set-up in Dijkstra's working.** Writing "Start = 0, all others = ∞ " earns marks by itself, yet many candidates only wrote the final answers. Show the set-up, the updates, the visited nodes and at least one node reached by two routes.
- **Taking the first route found as final.** A node's value only becomes final when it is visited (the smallest unvisited value). Q in Example 3 is 12 direct, but 9 through P.

- **Adding the wrong way in A*.** g is the total from the **start**, not just the last edge: from D with $g = 7$, End has $g = 7 + 5 = 12$, not 5. Write each g as a sum.
- **Stopping A* too early.** Reaching the goal is not enough; stop only when the goal has the lowest f of all open nodes.
- **Giving a final path with no working,** or working with no final path. The strongest answers had both.
- **Describing deep learning when asked for reasons.** "Outline the reasons for using deep learning" wants benefits (large data sets, unstructured data, hidden patterns, accuracy), not what it is.
- **Answering "how do ANNs enable machine learning" with supervised and unsupervised learning.** Explain the network: weighted connections, data through the layers, weights adjusted in training, repeated many times, back propagation.
- **Stopping at "labelled versus unlabelled".** It's one mark. Add the human input needed, predicting outcomes from past data, and finding hidden patterns or clusters.
- **Vague answers about graphs** such as "graphs show data". Name the parts (nodes, edges, weights) and what they represent, and give a use (finding the shortest route).
- **Confusing back propagation with a learning category.** It is the training method inside a neural network, used in supervised learning and deep learning; it is not a category alongside supervised, unsupervised and reinforcement.

How to get the marks

- **Each mark is a separate point.** Count the marks and make at least that many different points, one per sentence. Don't repeat the same idea in new words.
- **Use the key words** the mark schemes look for: labelled/unlabelled, hidden patterns, clusters, reward/penalty, agent, environment, hidden layers, weights, heuristic, nodes, edges, error, expected output.
- **"State" or "Identify"** need a short answer only: one category, one algorithm, one use. **"Describe"** needs the features; **"Explain"** needs the reasons or how it works; **"Outline"** needs brief points.
- **"Explain how X and Y differ":** write matched pairs with "whereas". A description of only one of them gets few marks.
- **Calculate with Dijkstra's:** marks for working are separate from marks for answers, so show the working even if you are sure of the answers. Fill every cell of the answer table.
- **Calculate with A*:** fill the table row by row in the order you work it out, with each g as a sum, and **always write the final path** (it has its own mark).
- **Examples help.** When a question is about how or why AI is used, one short example (a sat-nav finding a route, a spam filter, face recognition) can make a point clear.

Quick check

1. A graph has edges Start–J 3, Start–K 8, J–K 4, J–L 9, K–L 2, K–M 7 and L–M 3. Use Dijkstra's algorithm to find the shortest distance from Start to J, K, L and M.

2. During an A* search the open nodes are X ($g = 6, h = 5$), Y ($g = 4, h = 9$) and Z ($g = 8, h = 2$). Which node is expanded next? Show why.
3. Name the category of machine learning in each case: (a) a program learns to sort photos into "cat" and "dog" from thousands of photos already labelled; (b) a shop's program sorts customers into groups nobody defined in advance; (c) a drone learns to land by being rewarded for smooth landings.
4. A regression model predicts a taxi's journey time as $t = 4d + 10$ minutes for a journey of d km. (a) Predict the time for a 6.5 km journey. (b) A 6.5 km journey actually took 39 minutes. Find the error.
5. Describe back propagation of errors in four steps.

Answers

1. Start 0, others ∞ . Visit Start: J 3, K 8. Visit J (3): K $\min(8, 3 + 4) = 7$, L $3 + 9 = 12$. Visit K (7): L $\min(12, 7 + 2) = 9$, M $7 + 7 = 14$. Visit L (9): M $\min(14, 9 + 3) = 12$. Visit M. **J 3, K 7, L 9, M 12.**
2. $f = g + h$: X $6 + 5 = 11$, Y $4 + 9 = 13$, Z $8 + 2 = 10$. **Z** has the lowest f , so it is expanded next (even though Y has the lowest g).
3. (a) **Supervised** (labelled data). (b) **Unsupervised** (unlabelled data, finding clusters). (c) **Reinforcement** (rewards from trial and error).
4. (a) $t = 4 \times 6.5 + 10 = 36$ minutes. (b) Error = $39 - 36 = 3$ minutes (the model predicted 3 minutes too little).
5. The network's actual output is compared with the expected output to find the error; the error gradient is found and the error is fed back through the network from the output layer; each weight is adjusted to reduce the error; this is repeated many times until the outputs are accurate enough.

Past questions to try

- [9618/32/O/N/23 Q8](#): shortest distances to six nodes with Dijkstra's algorithm.
- [9618/32/O/N/21 Q9](#): hidden layers, how neural networks learn, and an A* table.
- [9618/33/O/N/22 Q9](#): what deep learning is and the reasons for using it.
- [9618/32/M/J/24 Q11](#): what Reinforcement Learning is.