Computer Programming
(30509)

Dynamic Programming intro

Dynamic programming: overview

- Not an algorithm, but an algorithmic pattern (no solver+problem coding pattern here)
- The name doesn't really mean anything, it was just "marketing"... (by Richard E. Bellmann)
- Closely related to recursion (+memoization)
- Used to solve (discrete) combinatorial optimization problems in polynomial time
- Can't always be used. Depends on the problem
- When you can use it, it's often the best possible approach
  - Extremely powerful
  - Sometimes feels like magic

Dynamic programming: overview

- «Dynamic programming typically applies to optimization problems in which we make a set of choices in order to arrive at an optimal solution. As we make each choice, subproblems of the same form often arise. Dynamic programming is effective when a given subproblem may arise from more than one partial set of choices; the key technique is to store the solution to each such subproblem in case it should reappear. [...] This simple idea can sometimes transform exponential-time algorithms into polynomial-time algorithms.»

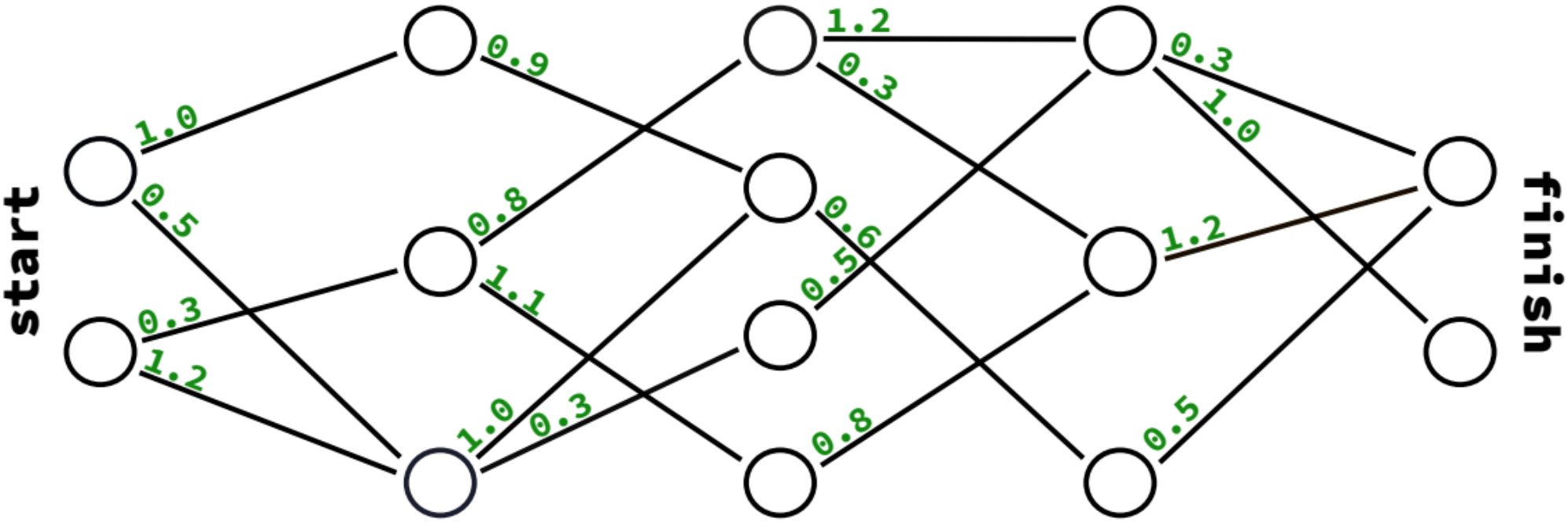
  Introduction to Algorithms, Cormen et al., Part IV, Introduction

- «Dynamic programming, like the divide-and-conquer method, solves problems by combining the solutions to subproblems. [...] Dynamic programming applies when the subproblems overlap – that is, when subproblems share subsubproblems. [...] A dynamic-programming algorithm solves each subsubproblem just once and then saves its answer in a table, thereby avoiding the work of recomputing the answer every time it solves each subsubproblem.»

  Introduction to Algorithms, Cormen et al., Chapter 15

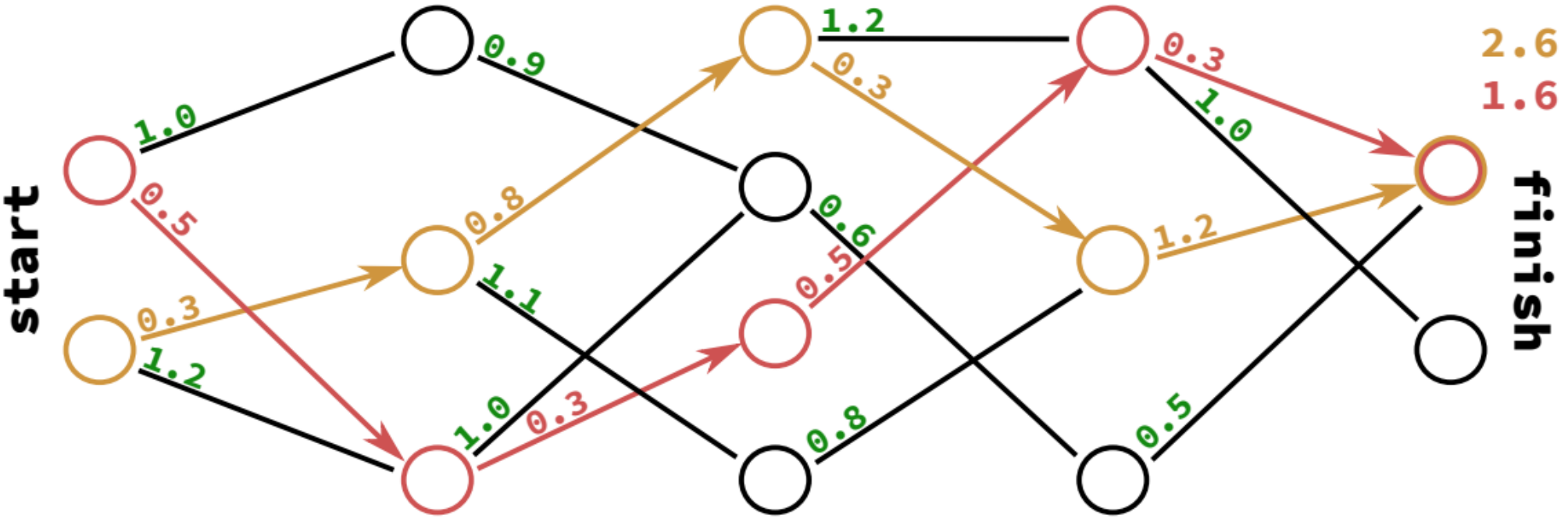
Part I
The optimal left-to-right path in a layered directed acyclic graph, revisited

Example: optimal left-to-right path



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Example: optimal left-to-right path



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

What makes this a good example?

- All possible configurations = all possible paths = can scale exponentially

  → brute-force search unfeasible

What makes this a good example?

- All possible configurations = all possible paths = can scale exponentially

  → brute-force search unfeasible

- A configuration is the result of a sequence of decisions (at each layer, pick a node)

What makes this a good example?

- All possible configurations = all possible paths = can scale exponentially

  → brute-force search unfeasible

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost

What makes this a good example?

- All possible configurations = all possible paths = can scale exponentially

  → brute-force search unfeasible

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost
- Incomplete decisions can trap you into sub-optimal configurations
  - The greedy approach (2nd kind) fails

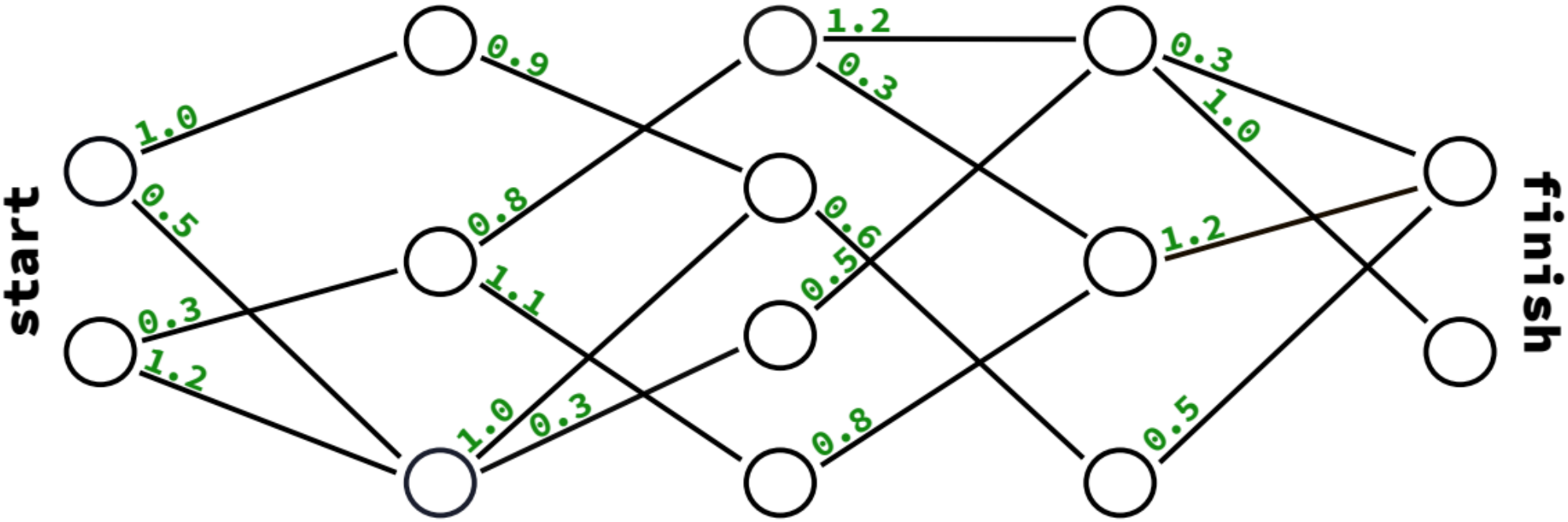
What makes this a good example?

- All possible configurations = all possible paths = can scale exponentially

  → brute-force search unfeasible

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost
- Incomplete decisions can trap you into sub-optimal configurations
  - The greedy approach (2nd kind) fails
- But: the problem has a particular structure: the optimal decision can be built from optimal partial decisions...

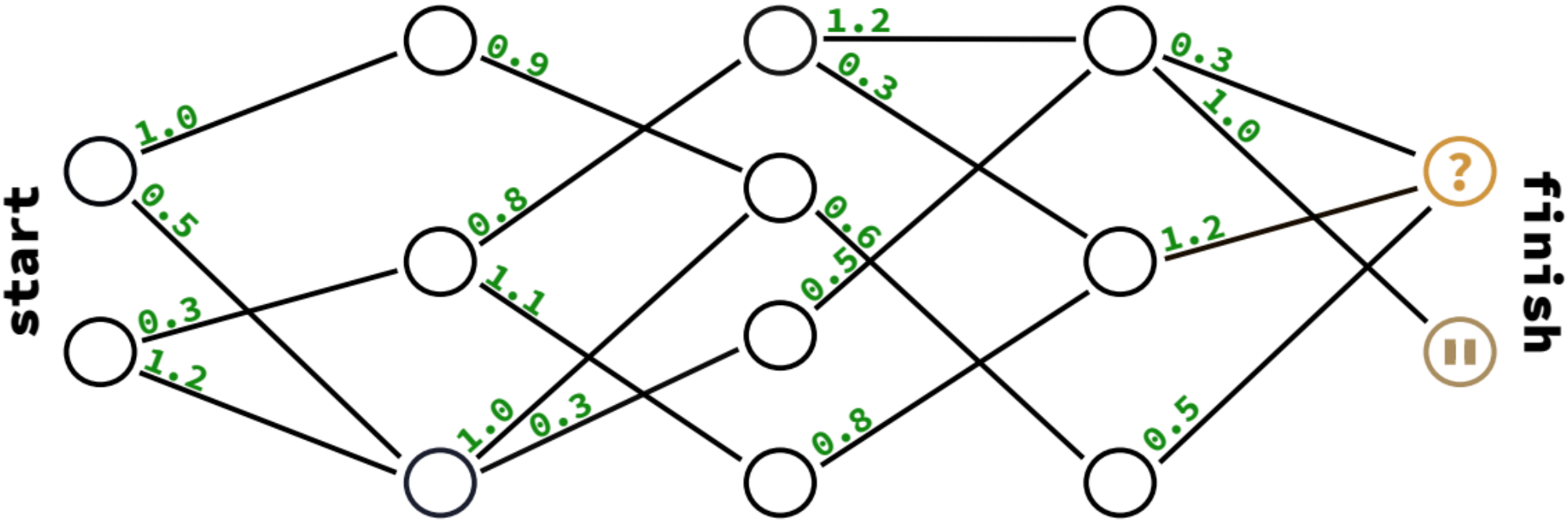
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

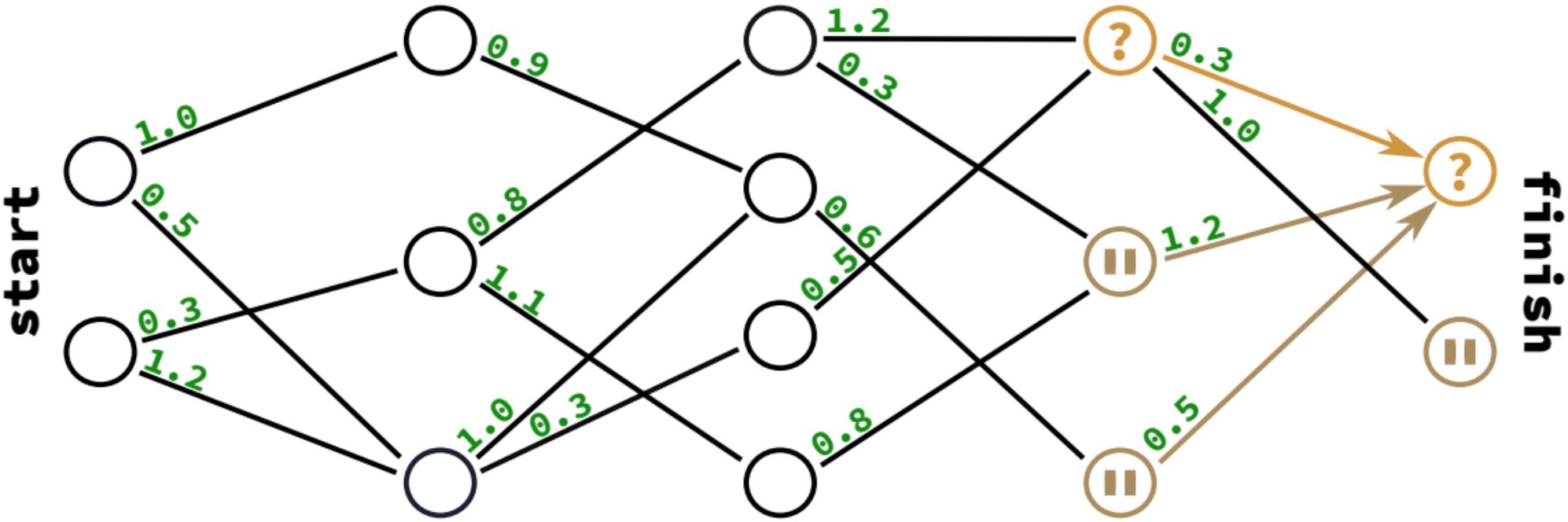
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

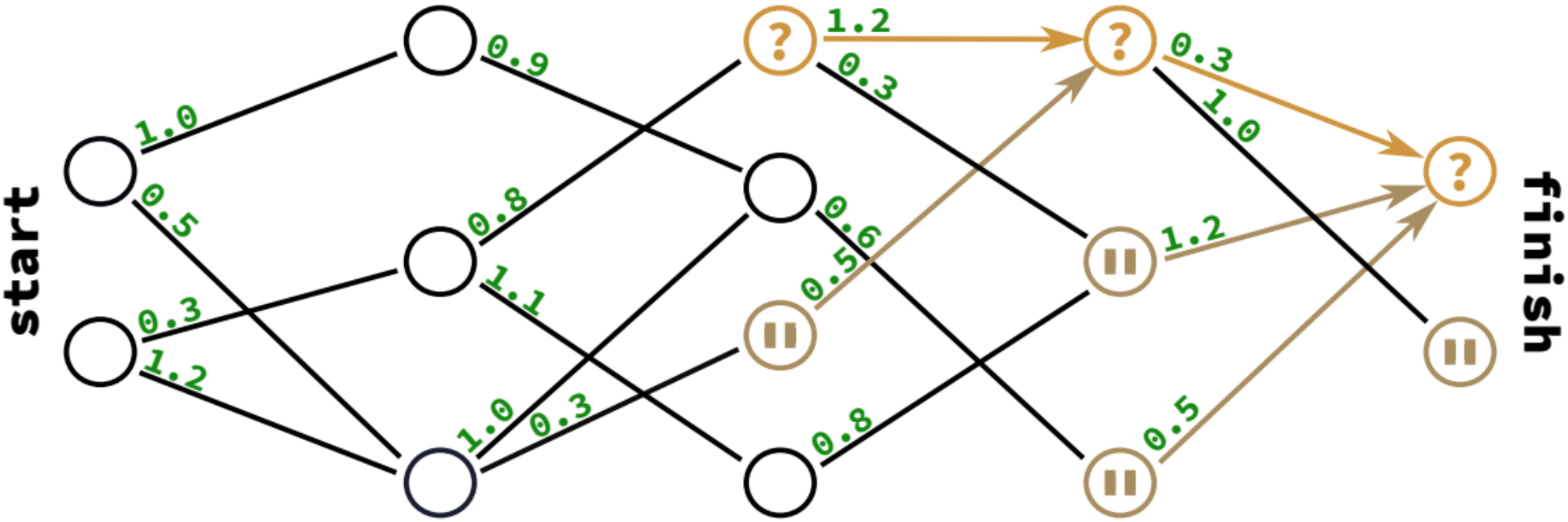
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

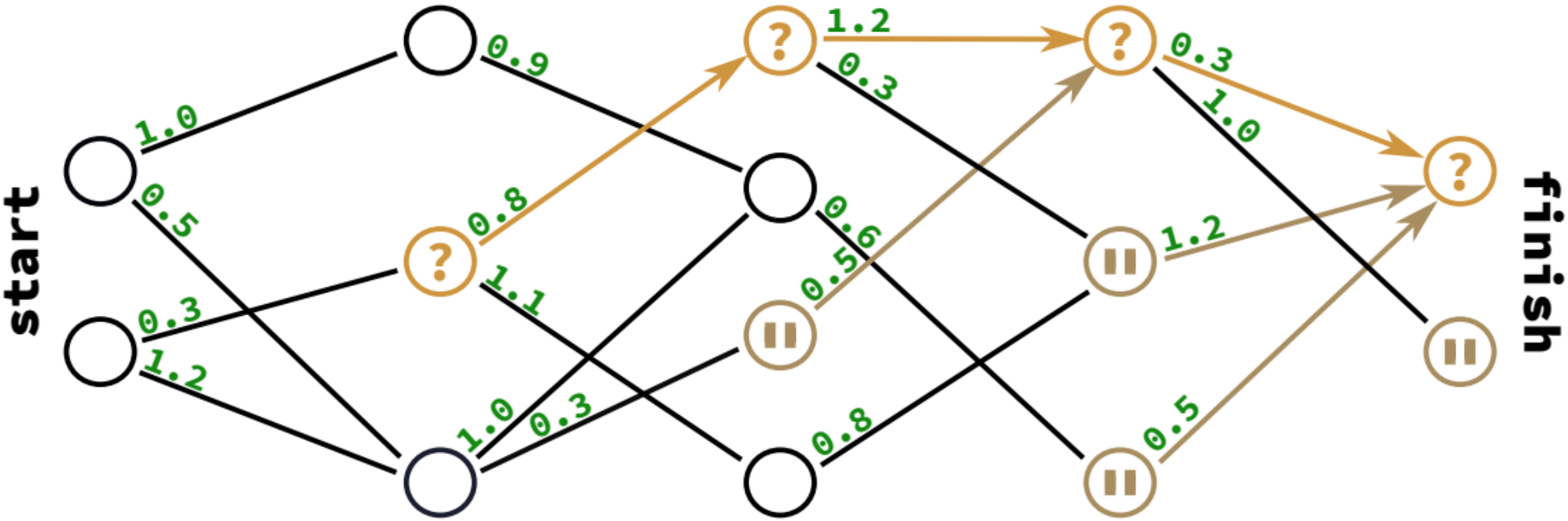
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

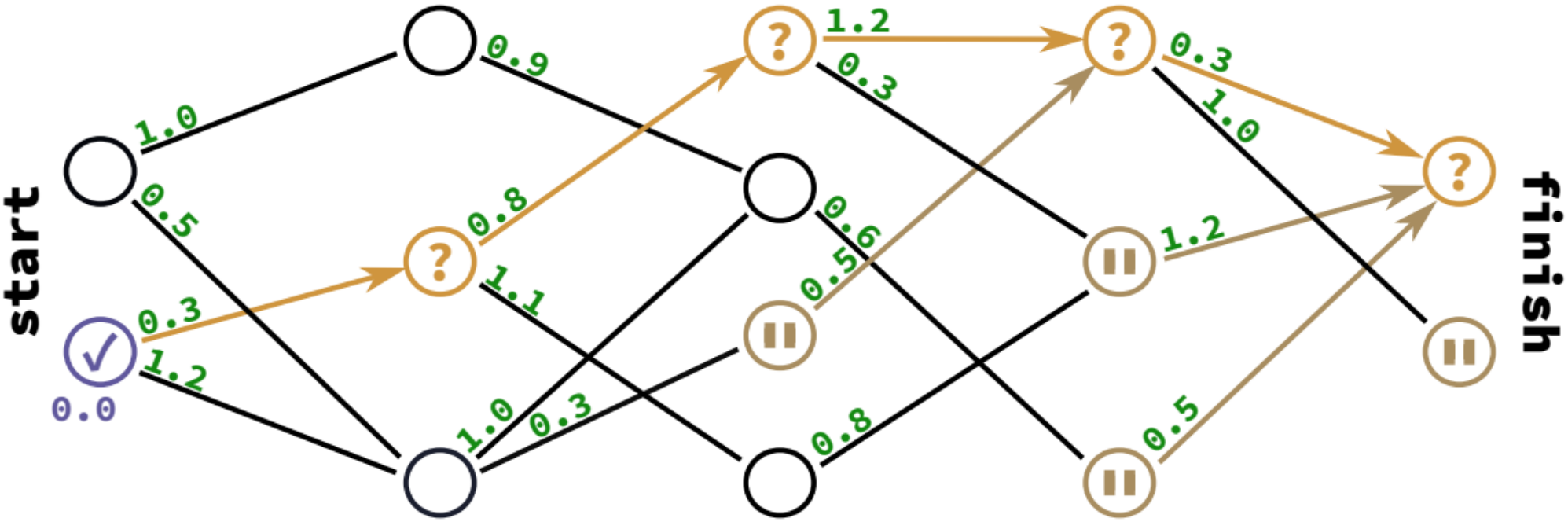
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

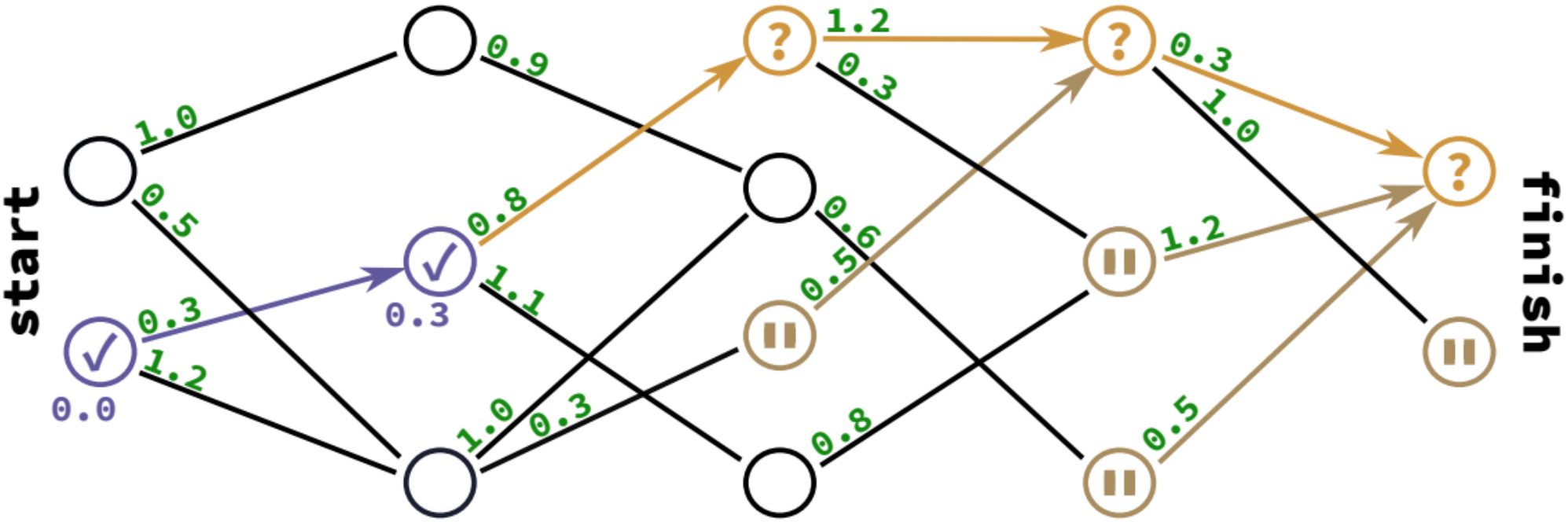
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

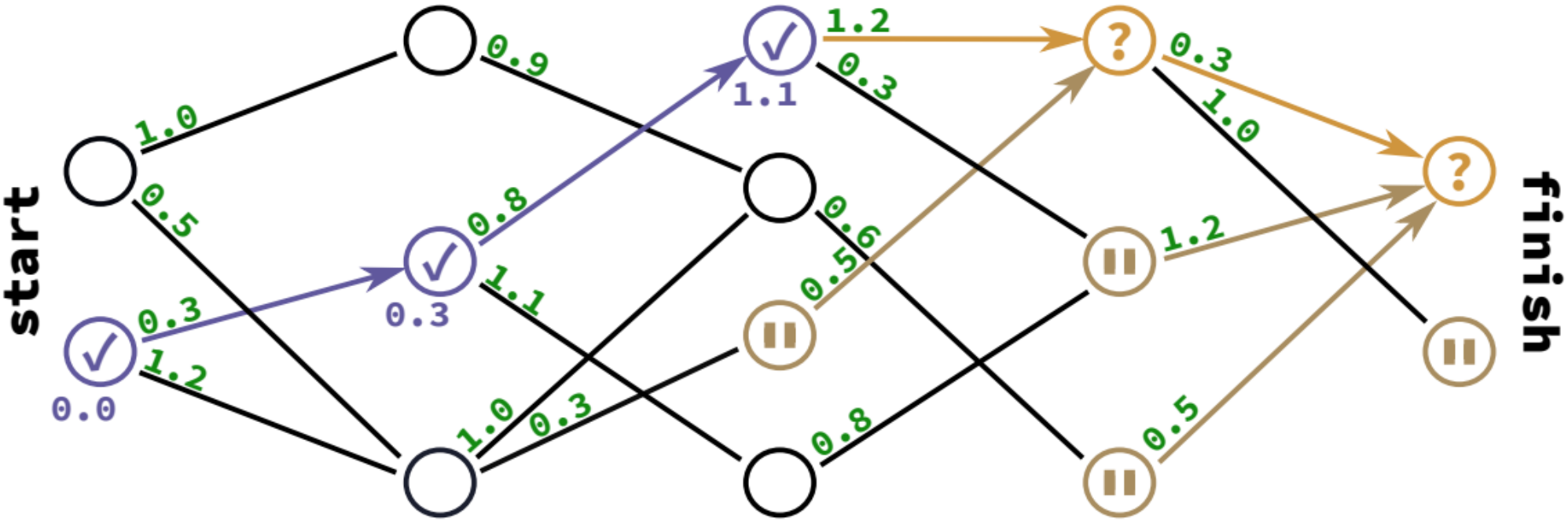
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

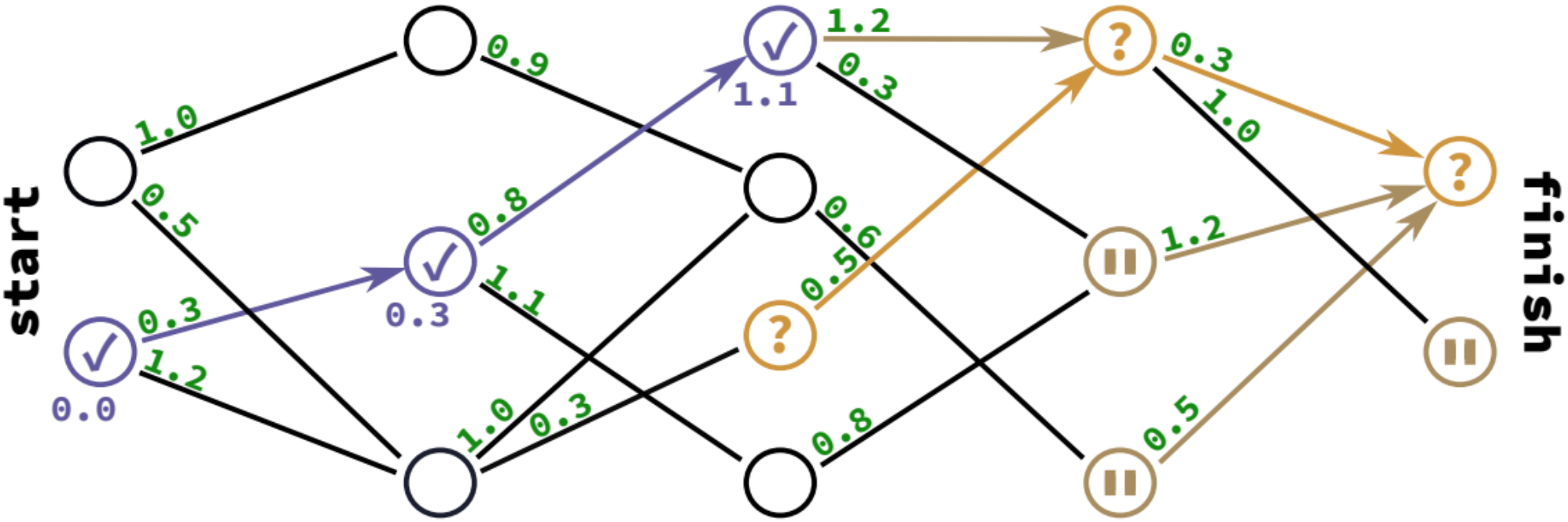
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

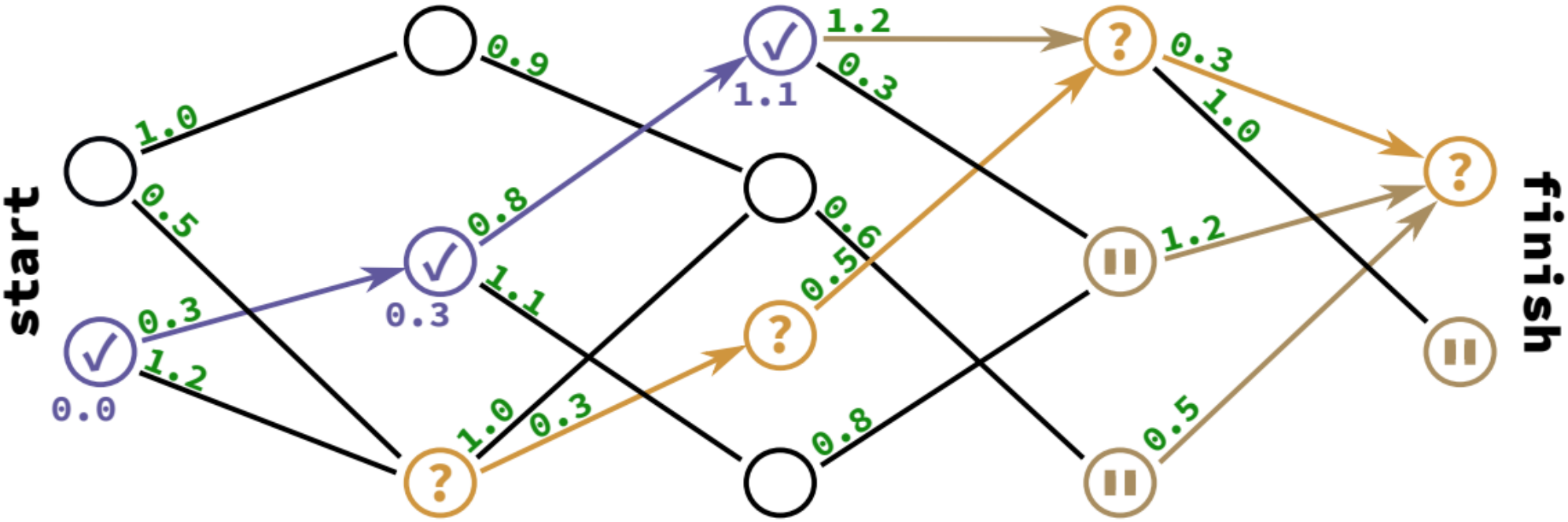
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

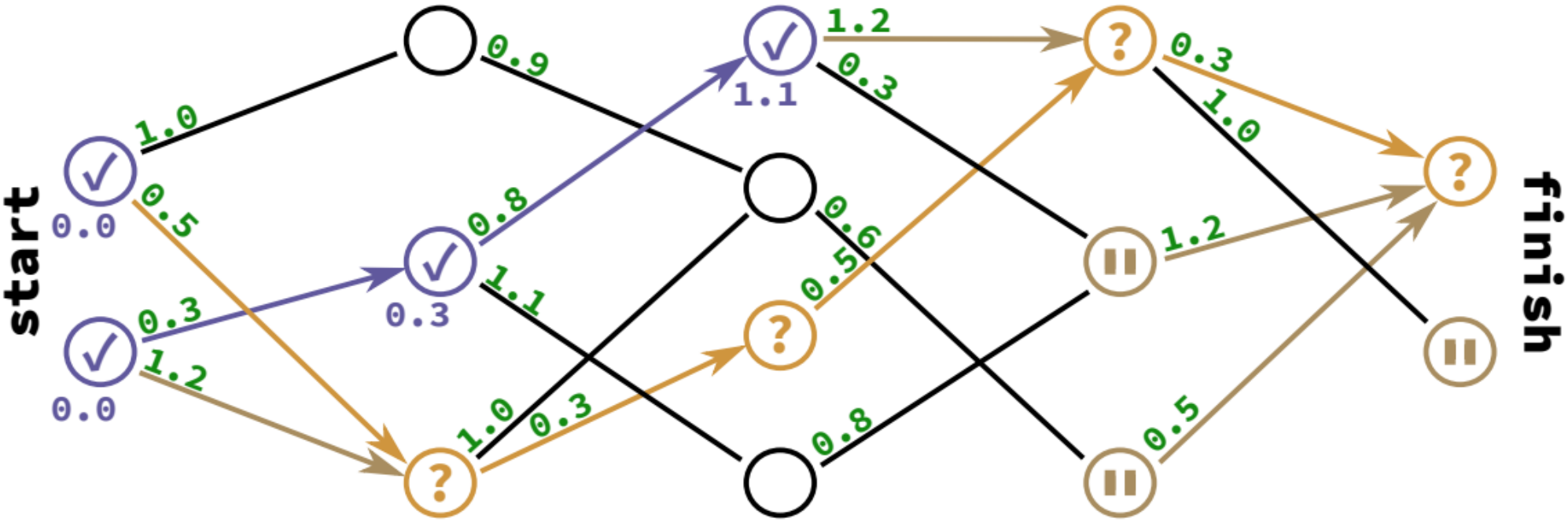
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

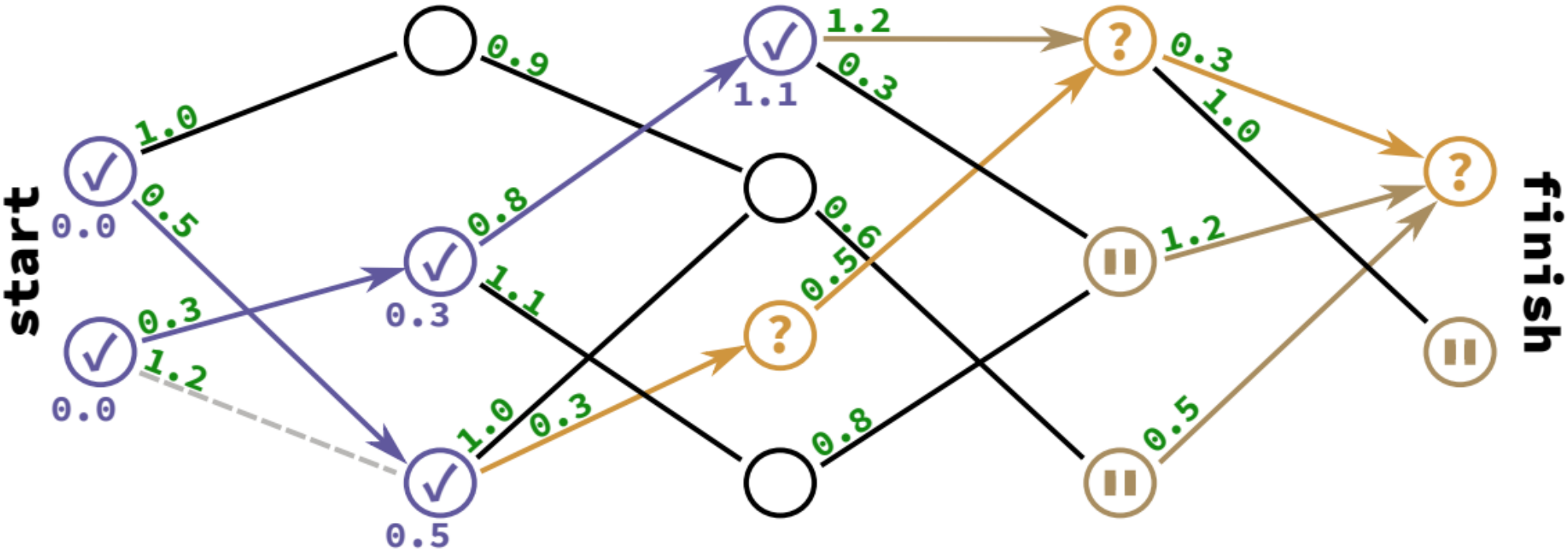
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

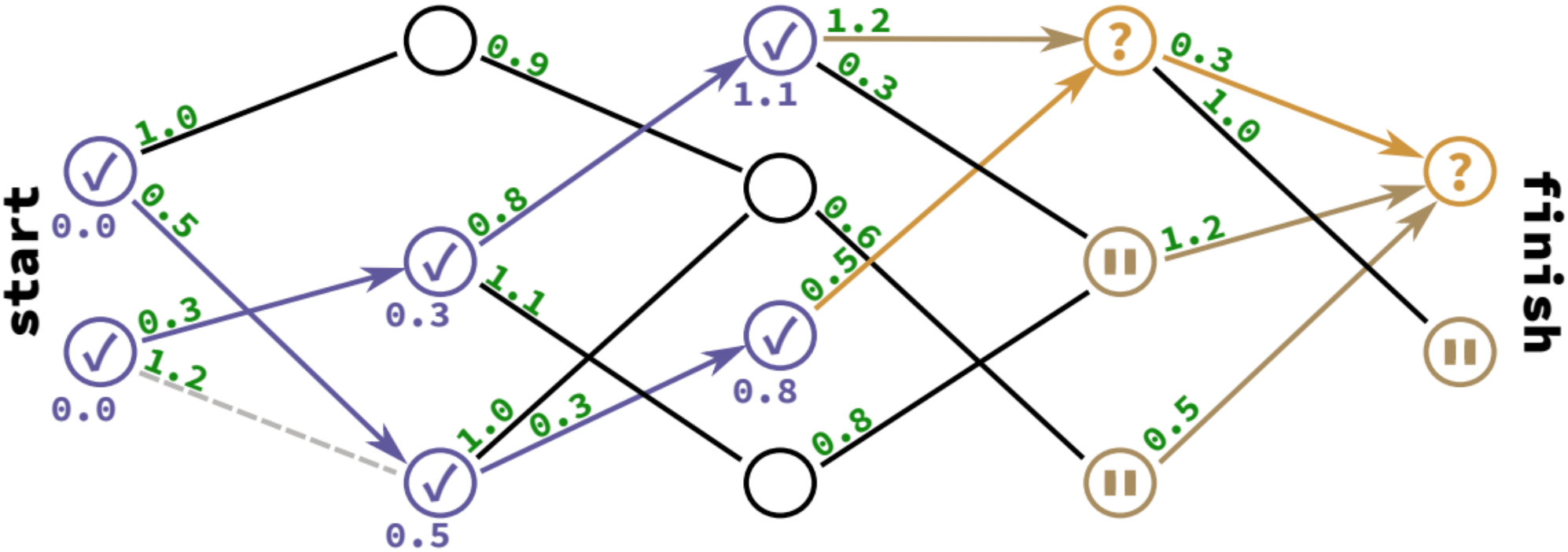
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

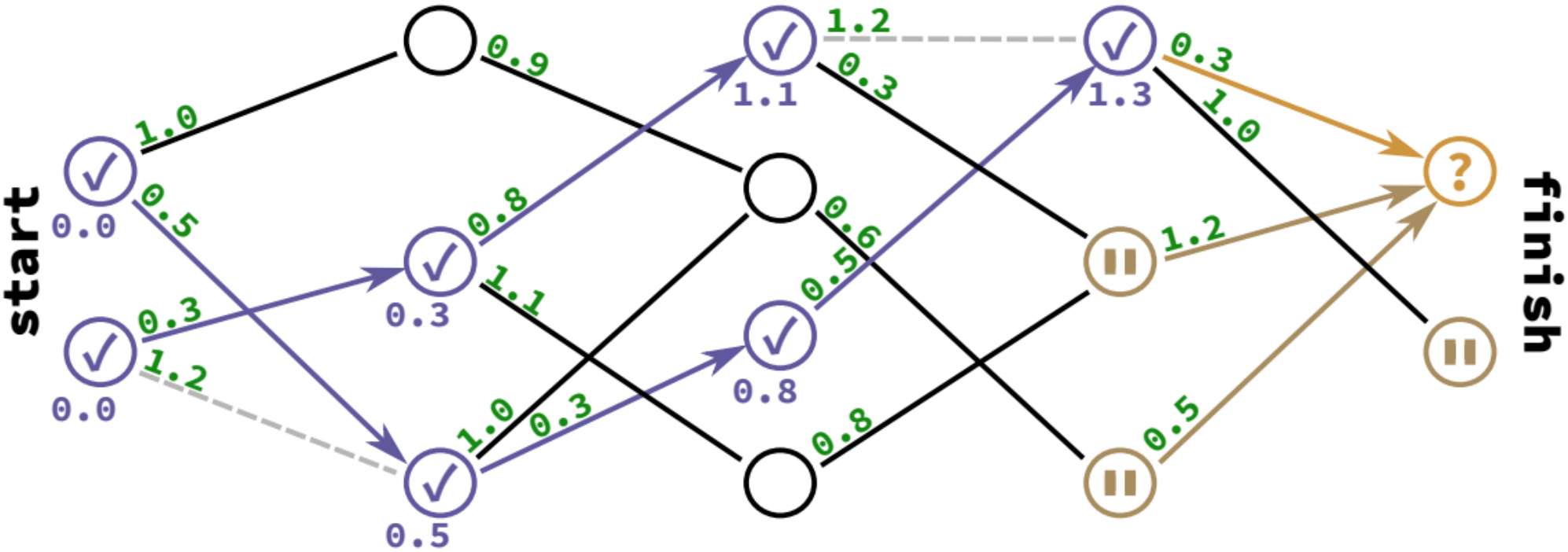
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

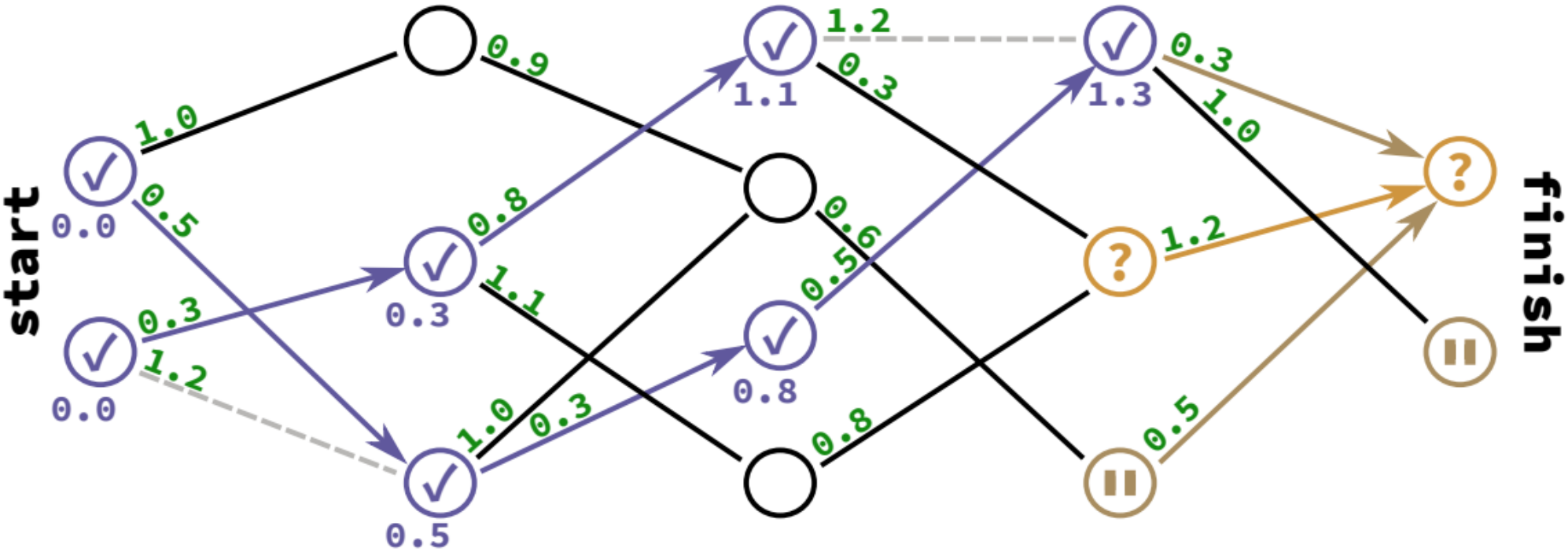
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

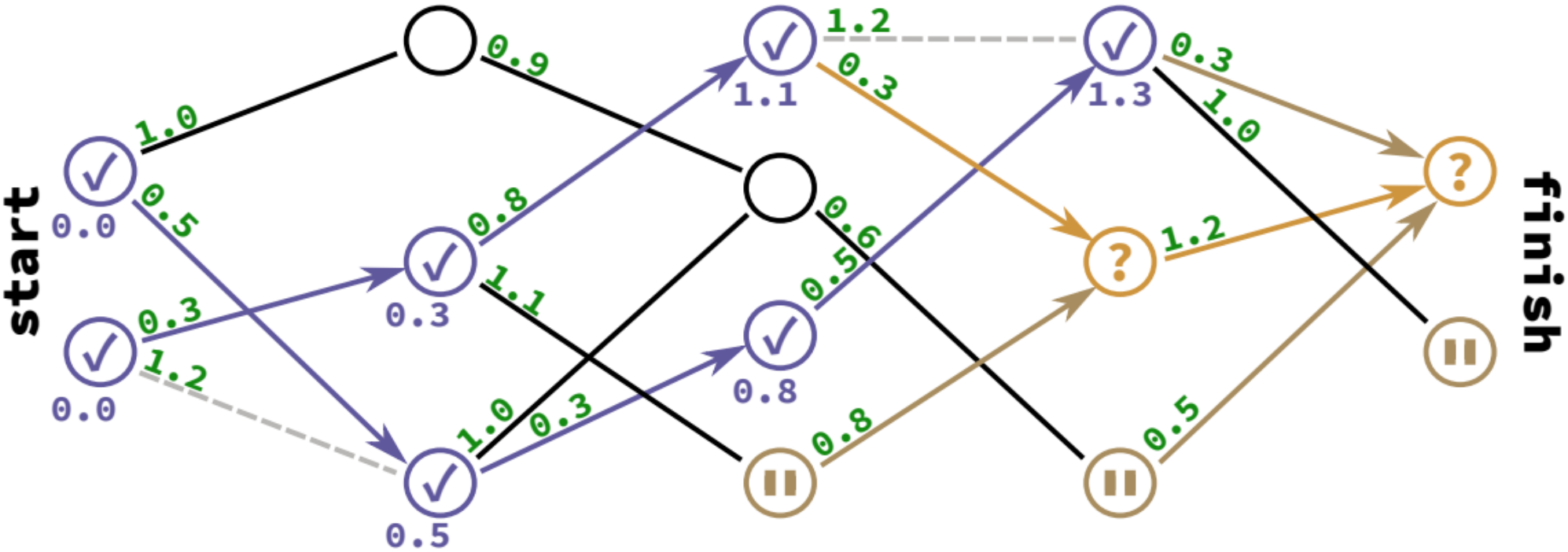
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

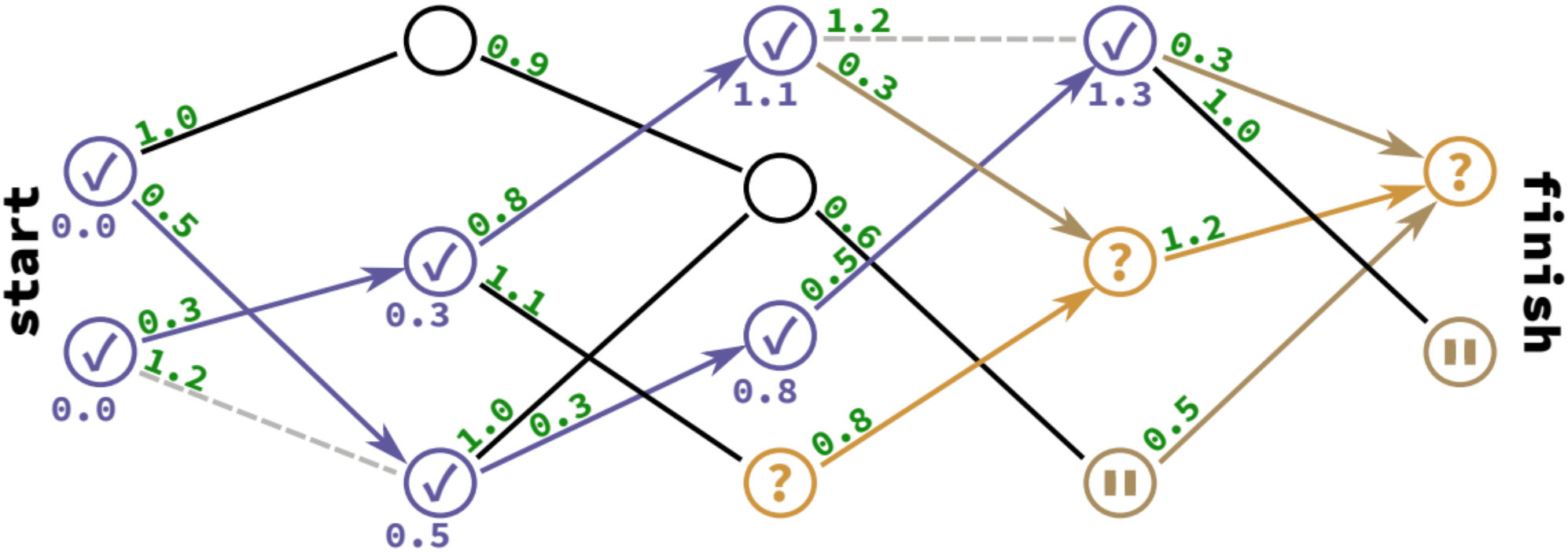
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

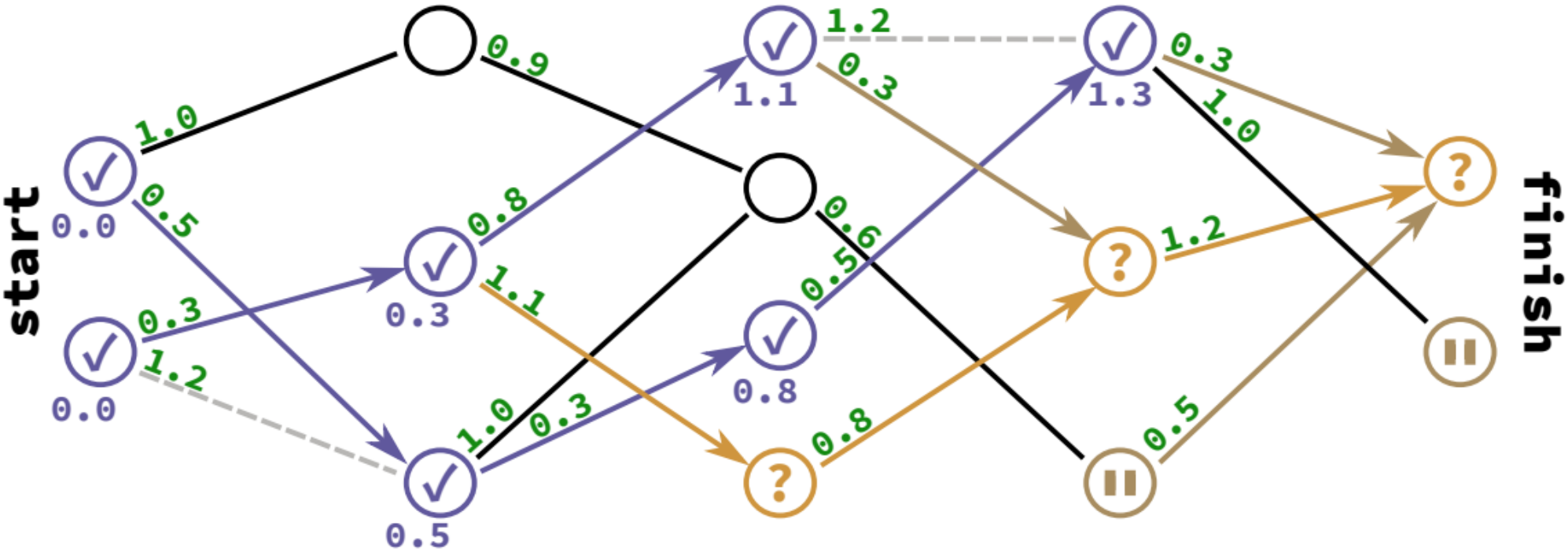
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

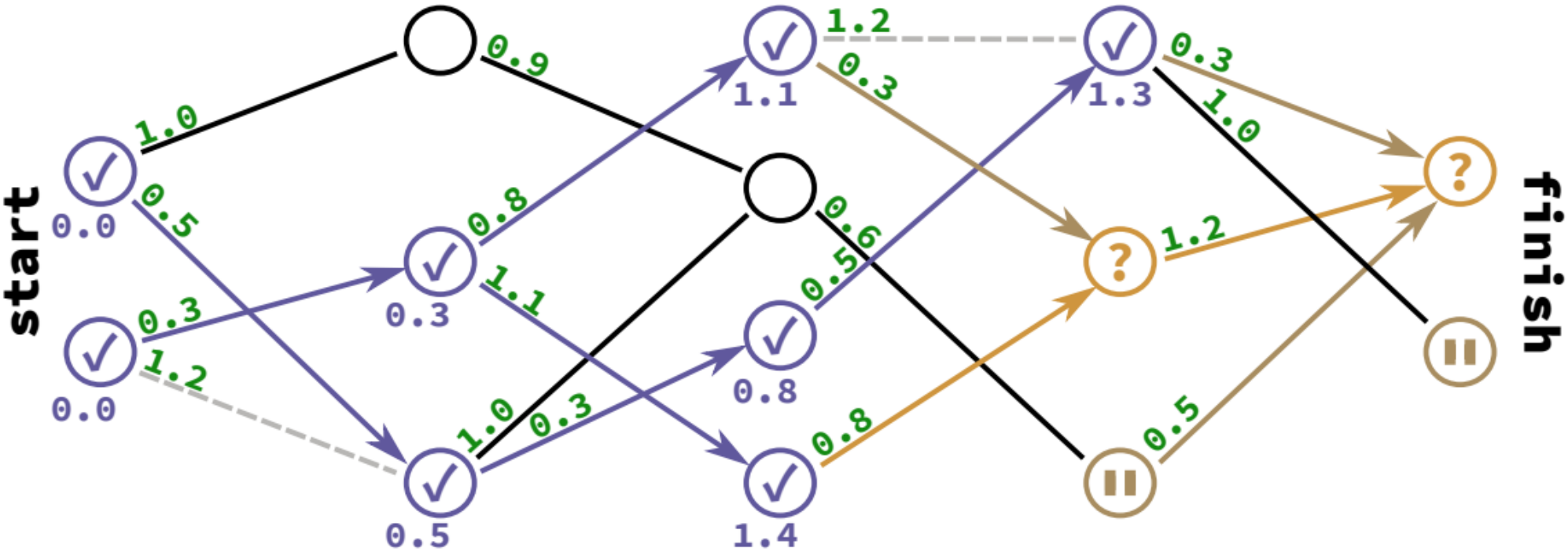
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

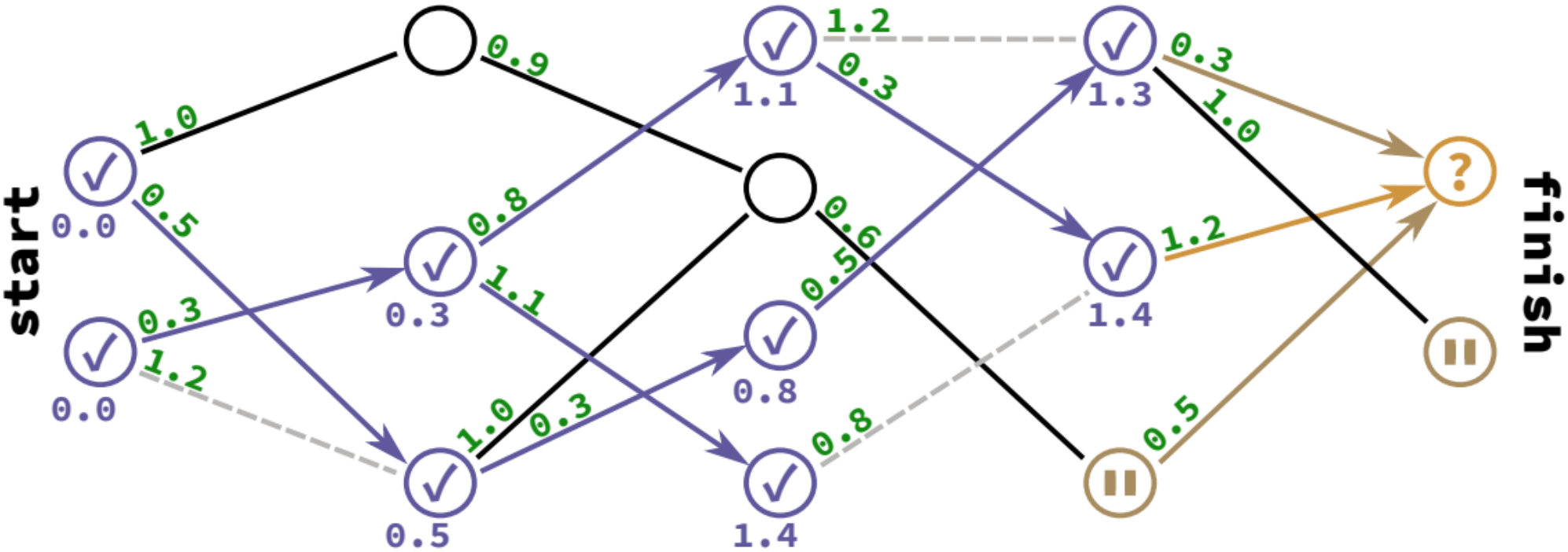
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

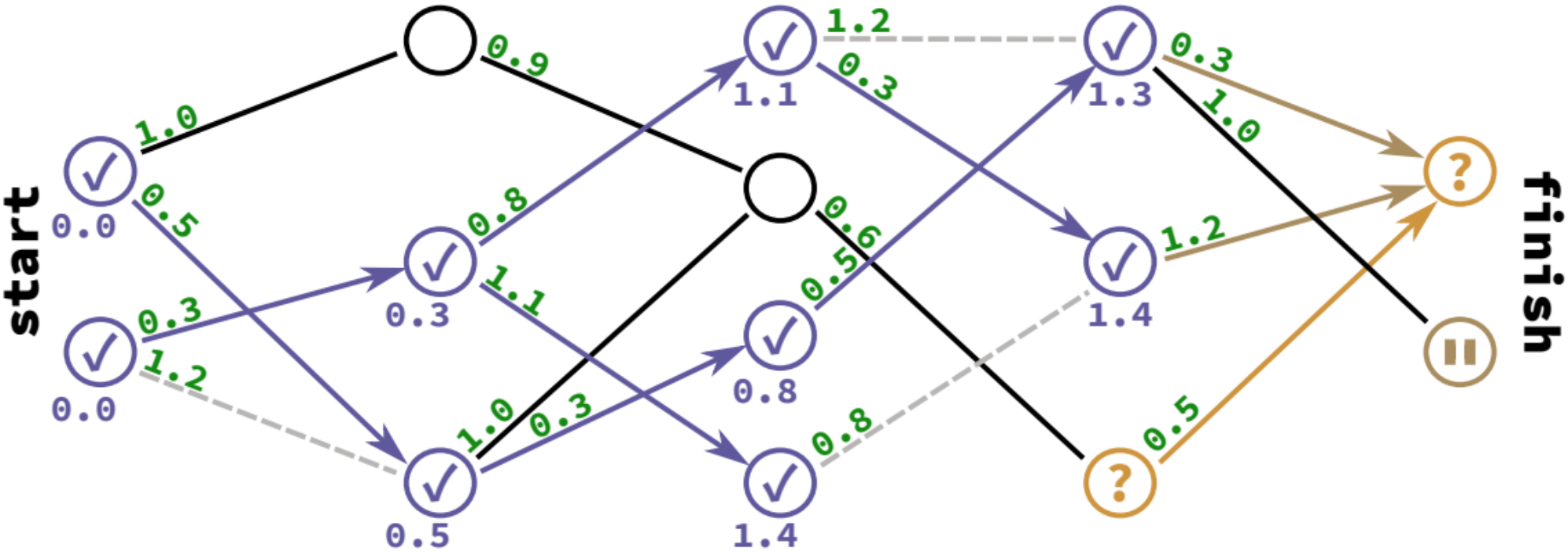
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

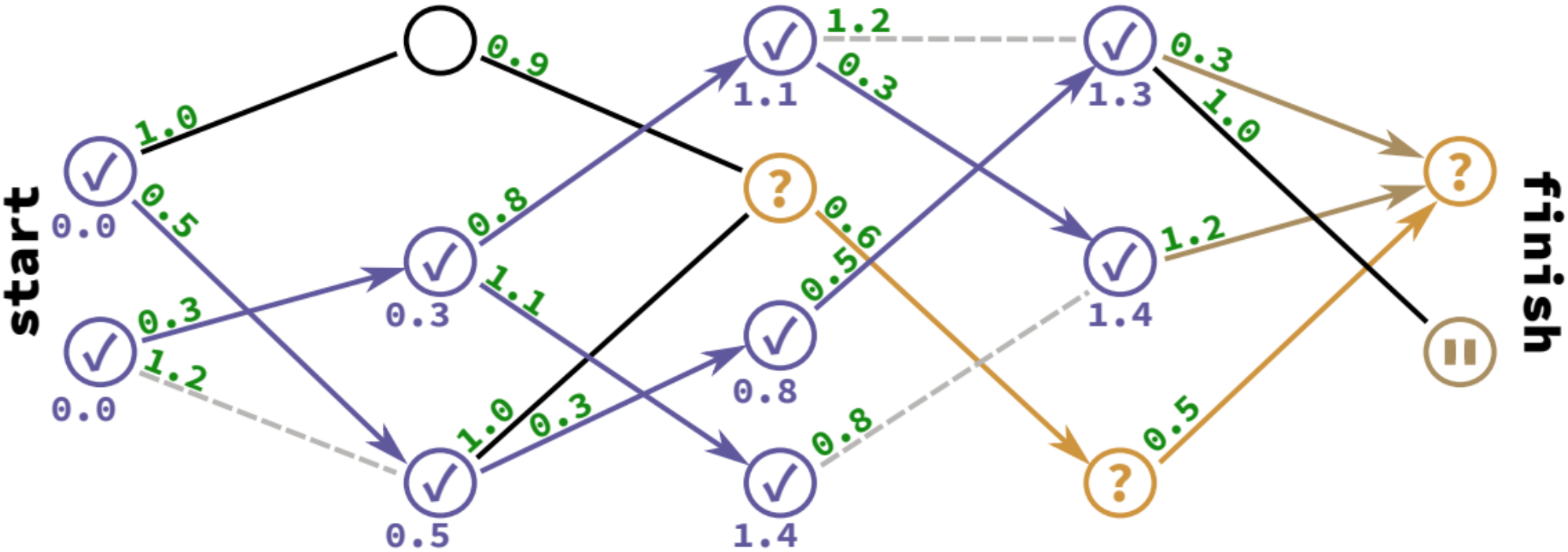
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

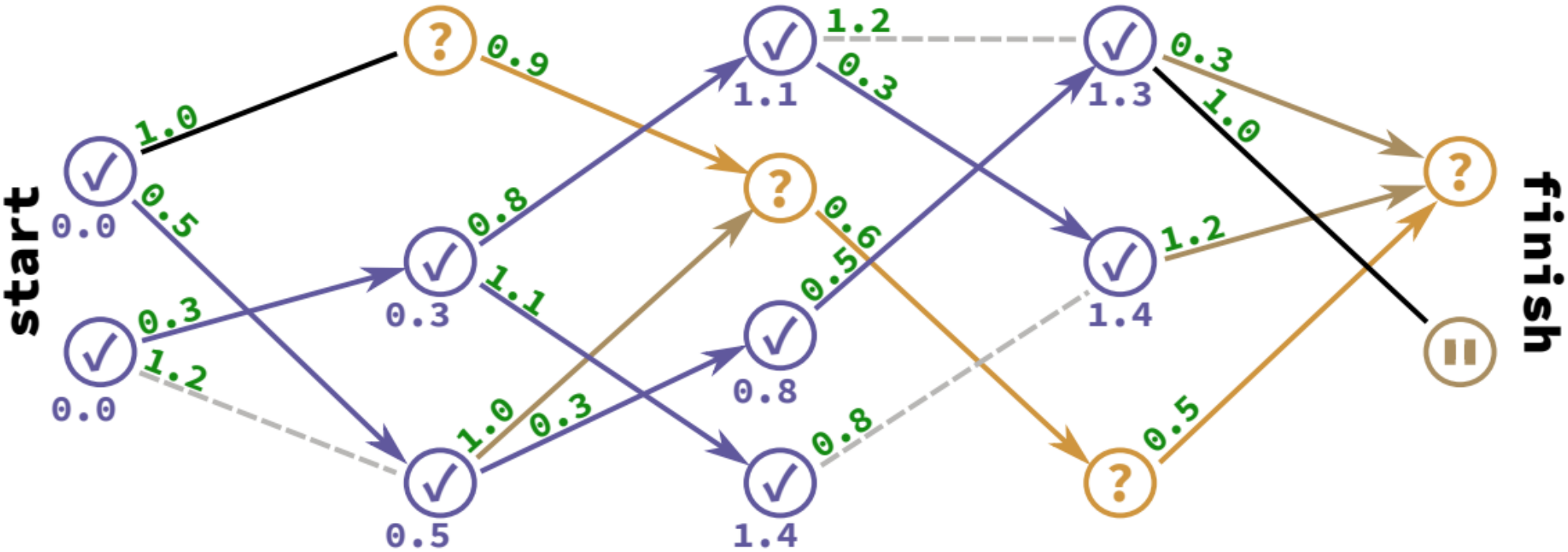
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

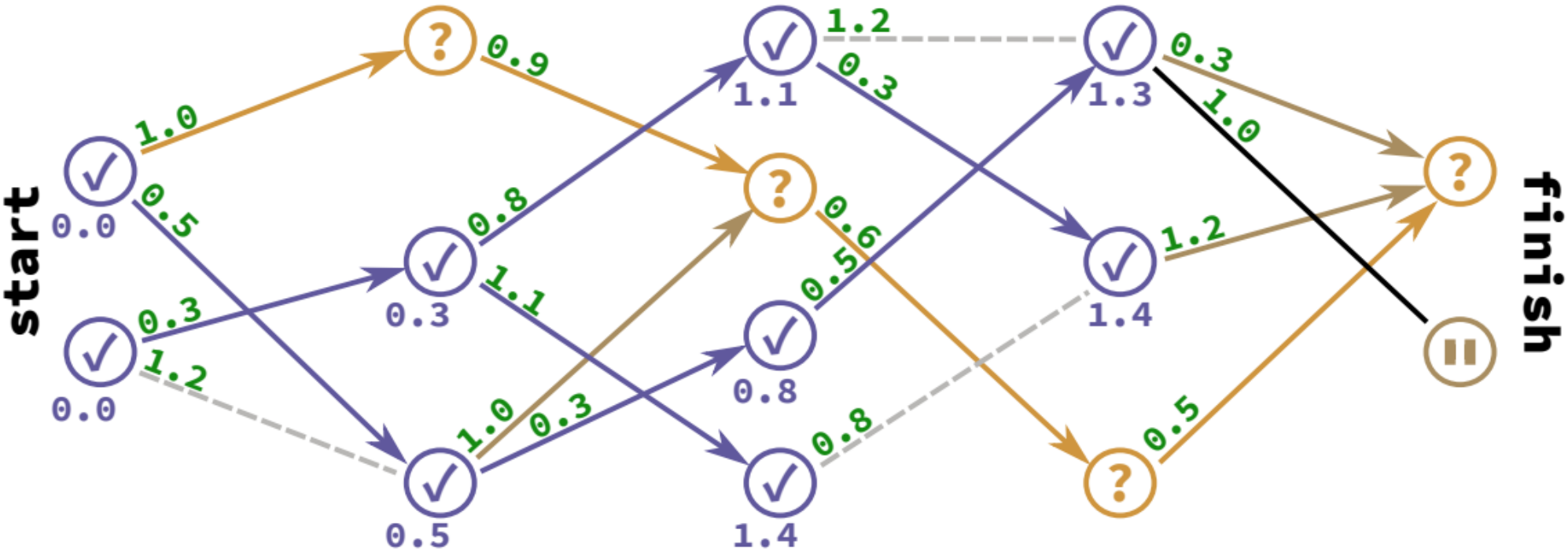
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

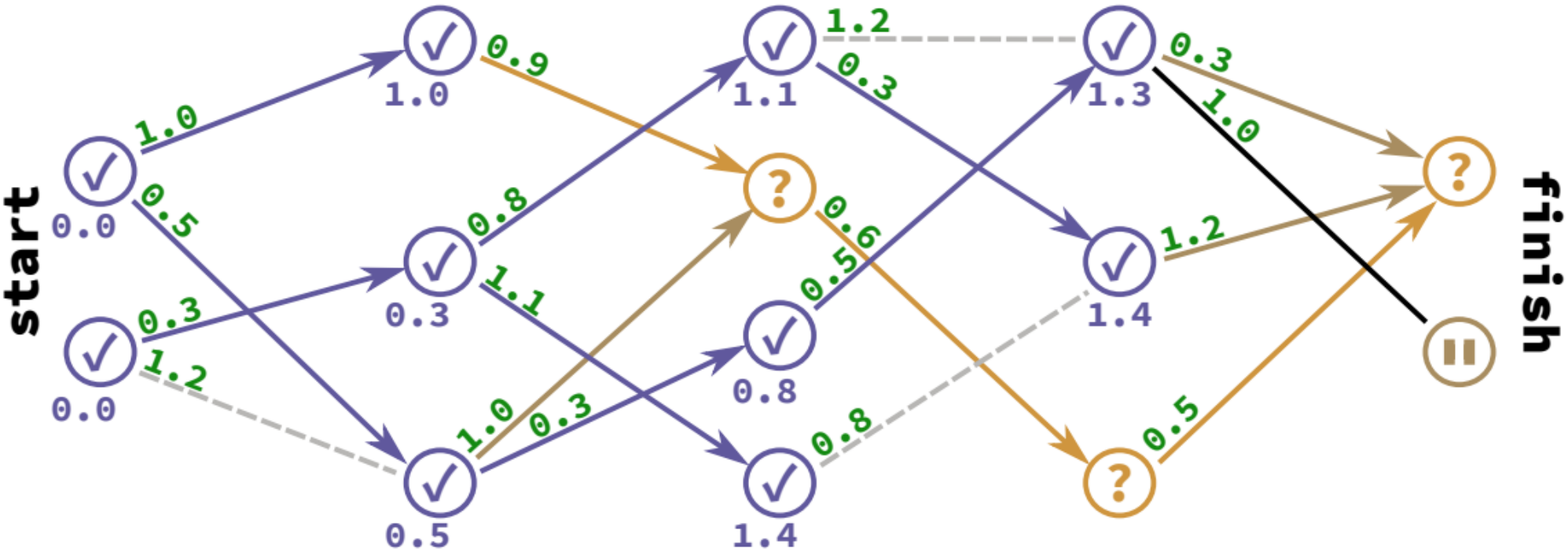
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

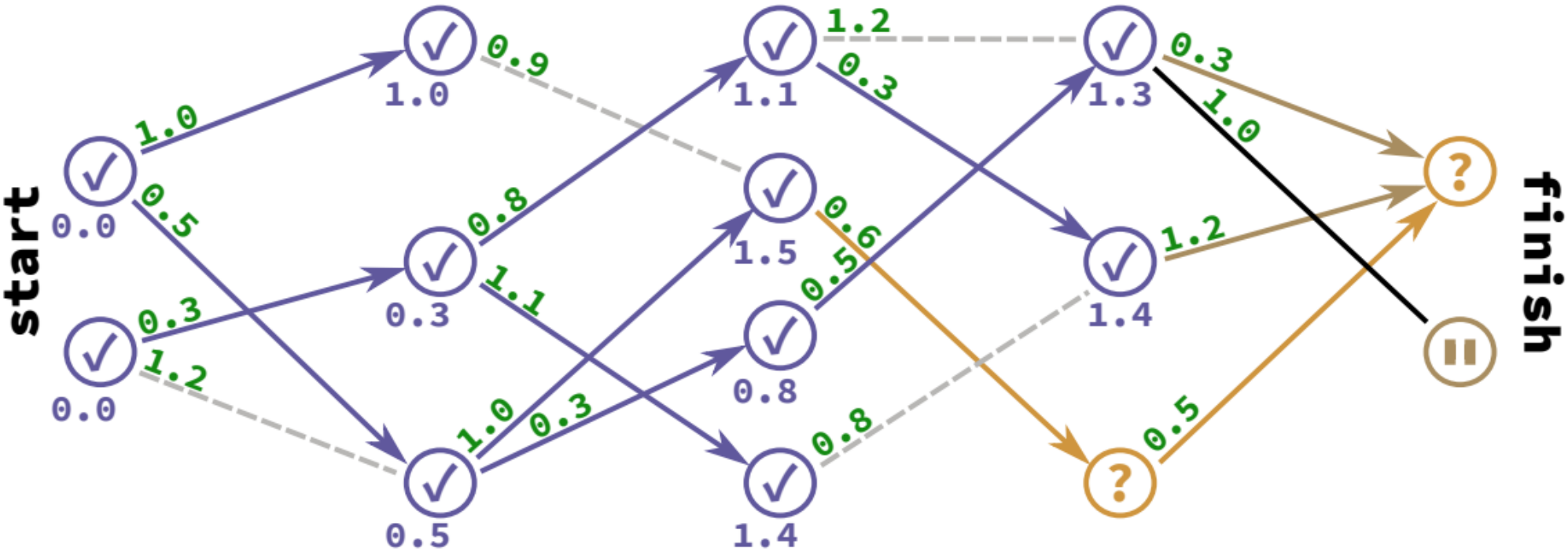
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

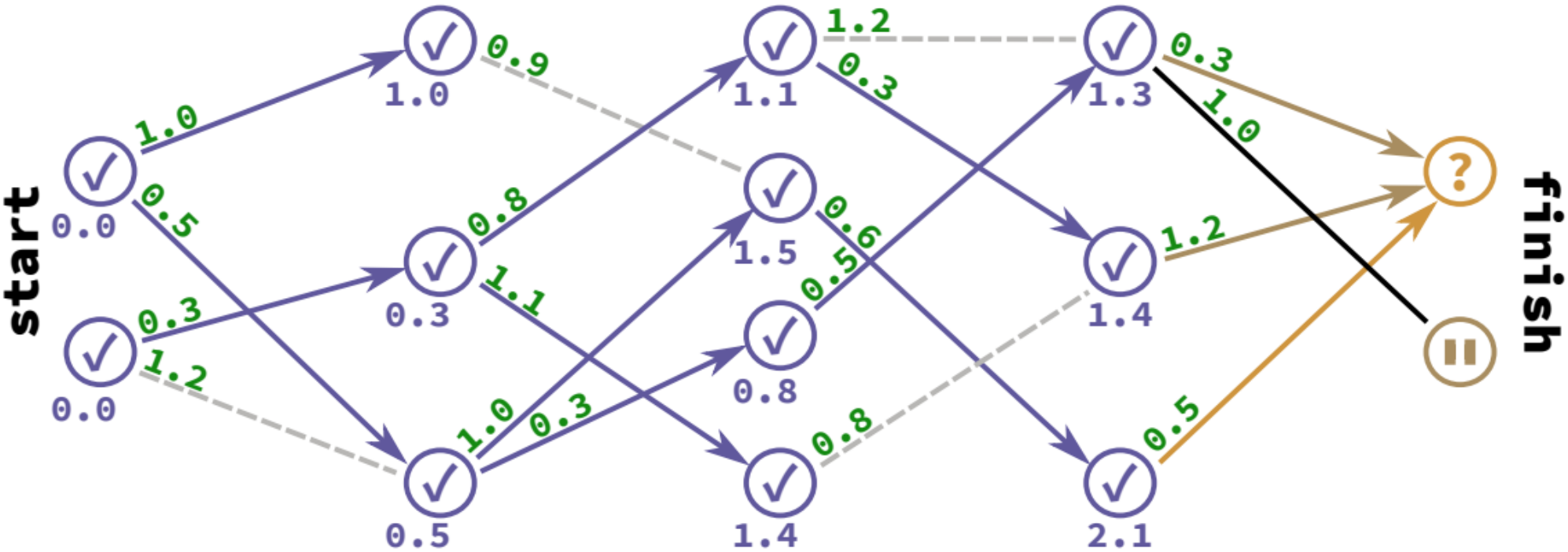
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

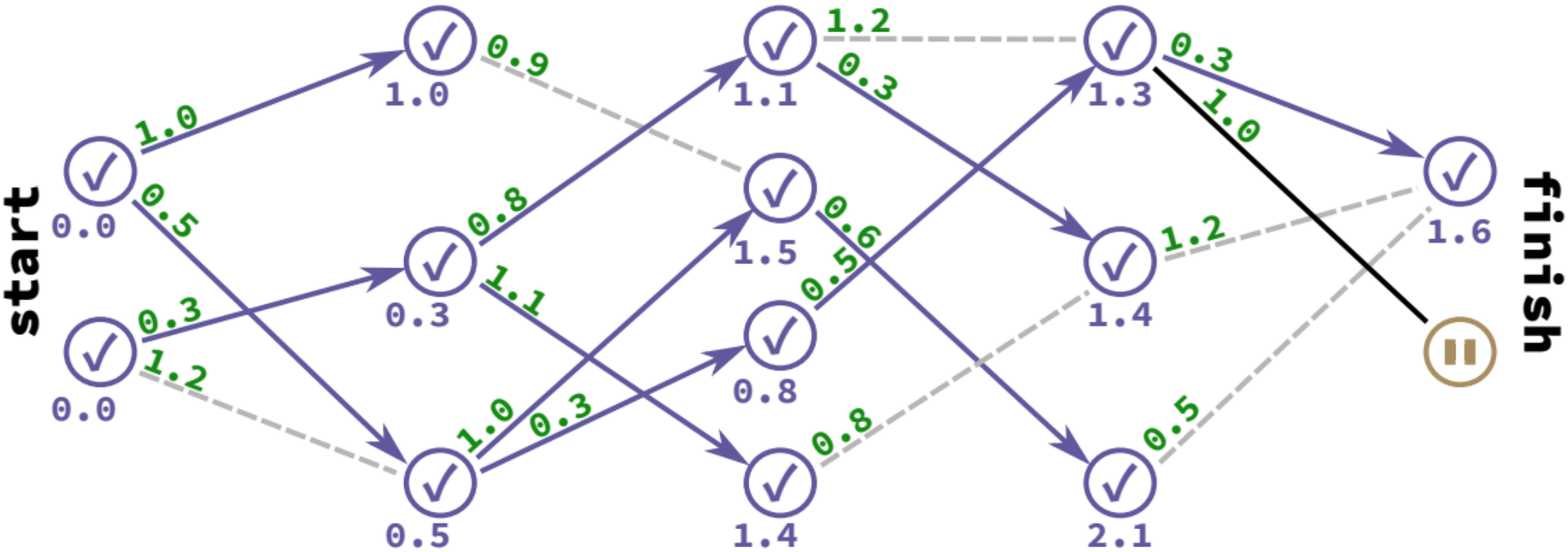
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

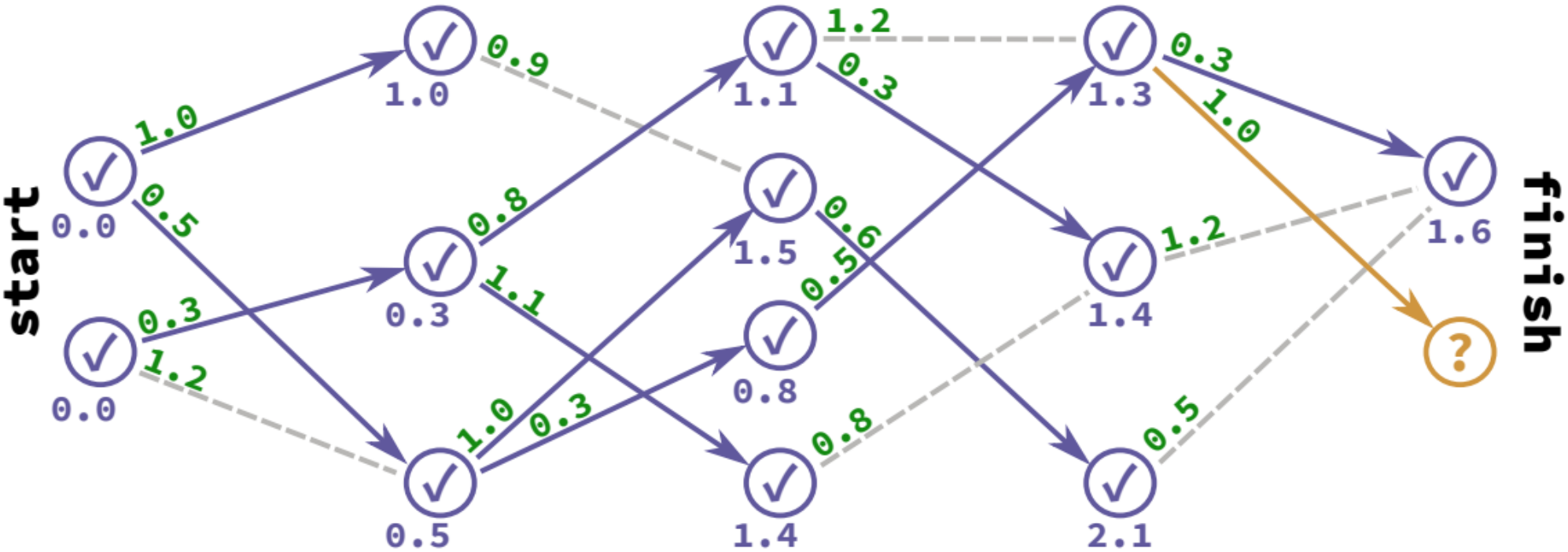
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

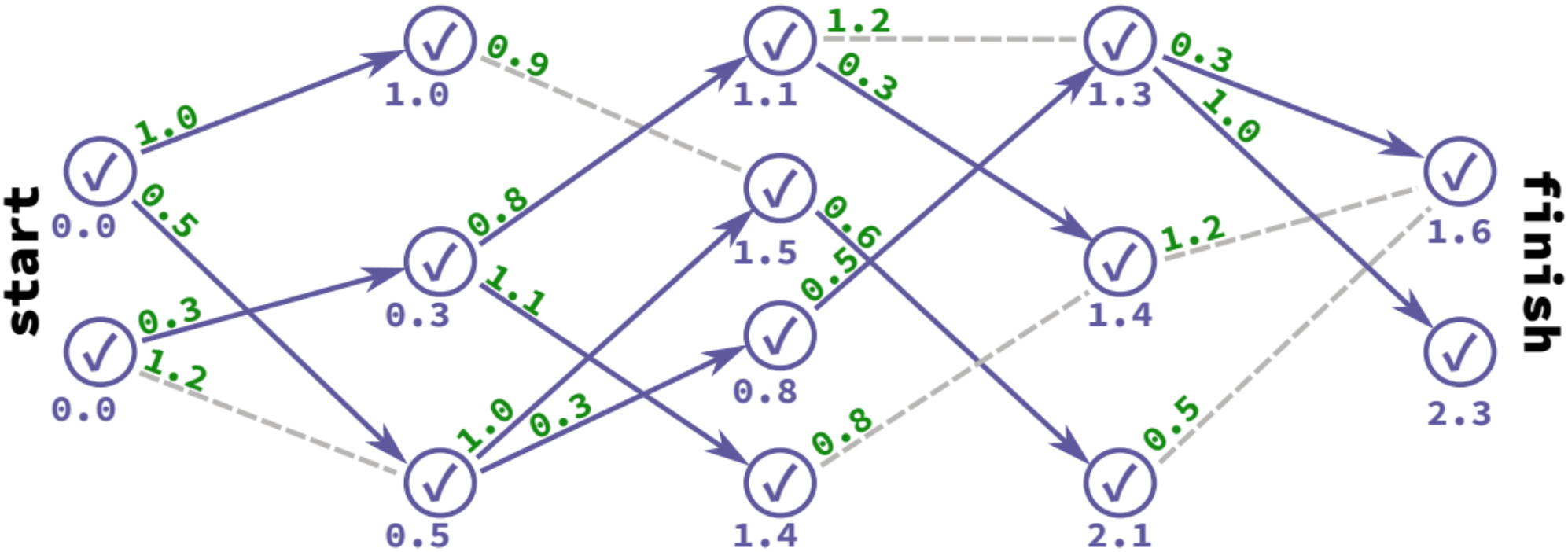
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

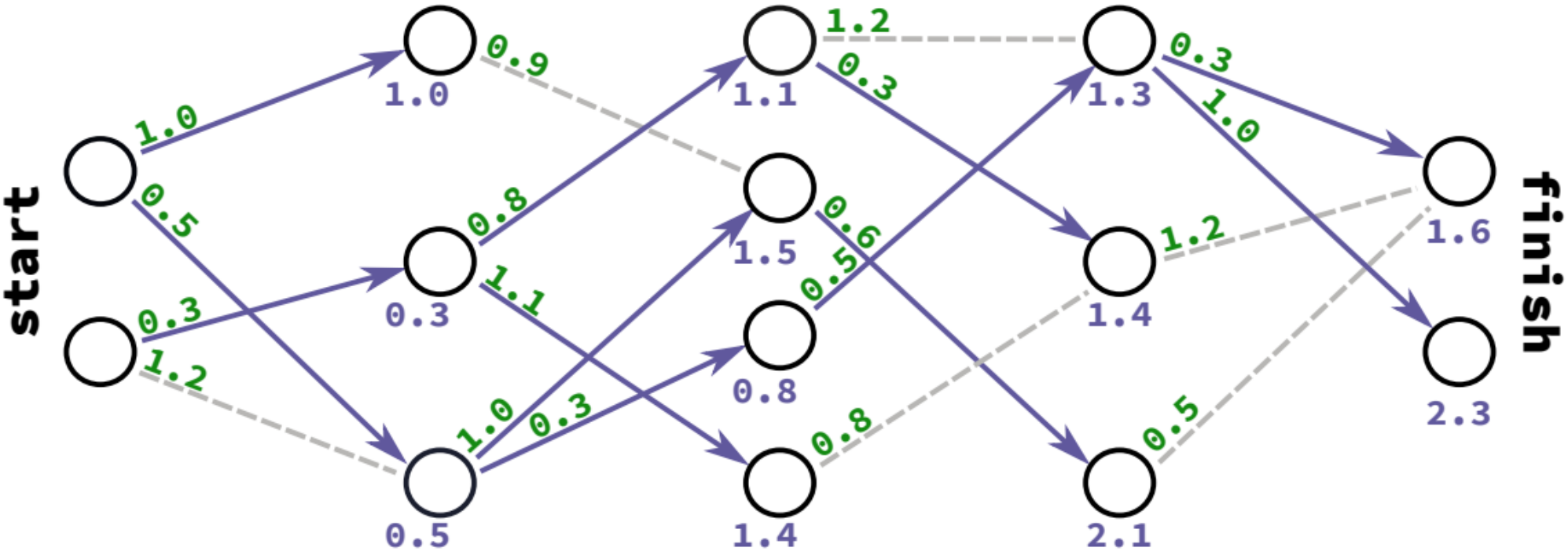
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

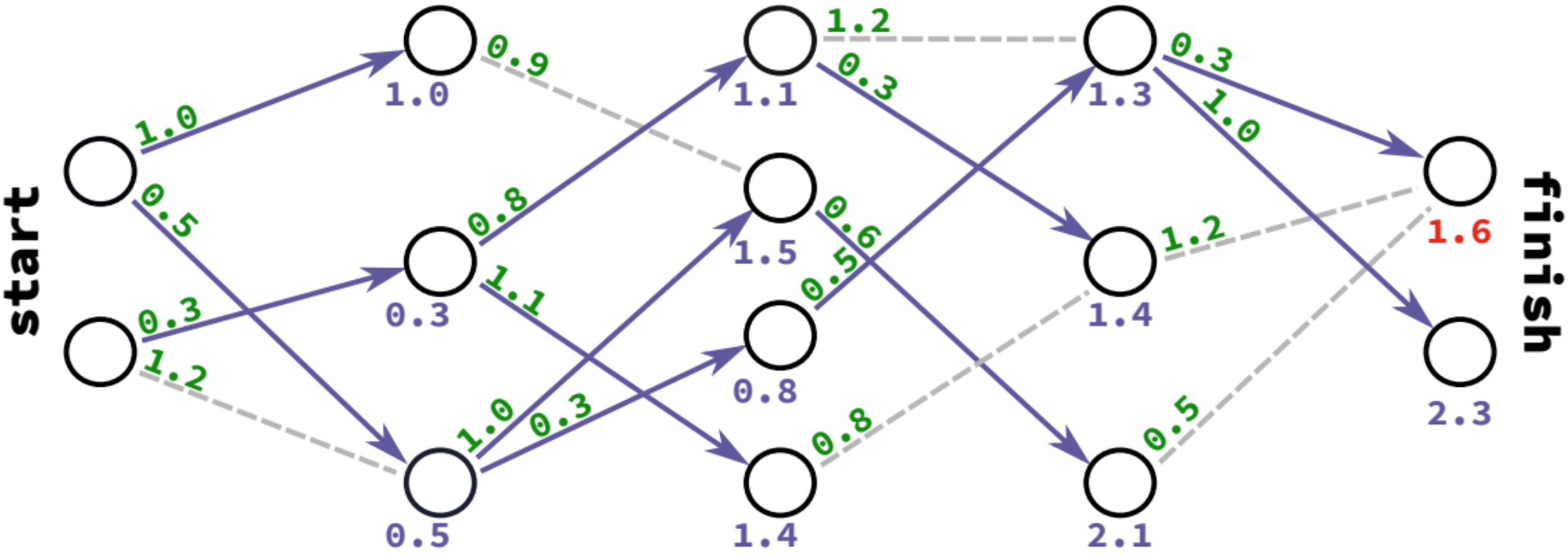
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Forward pass completed

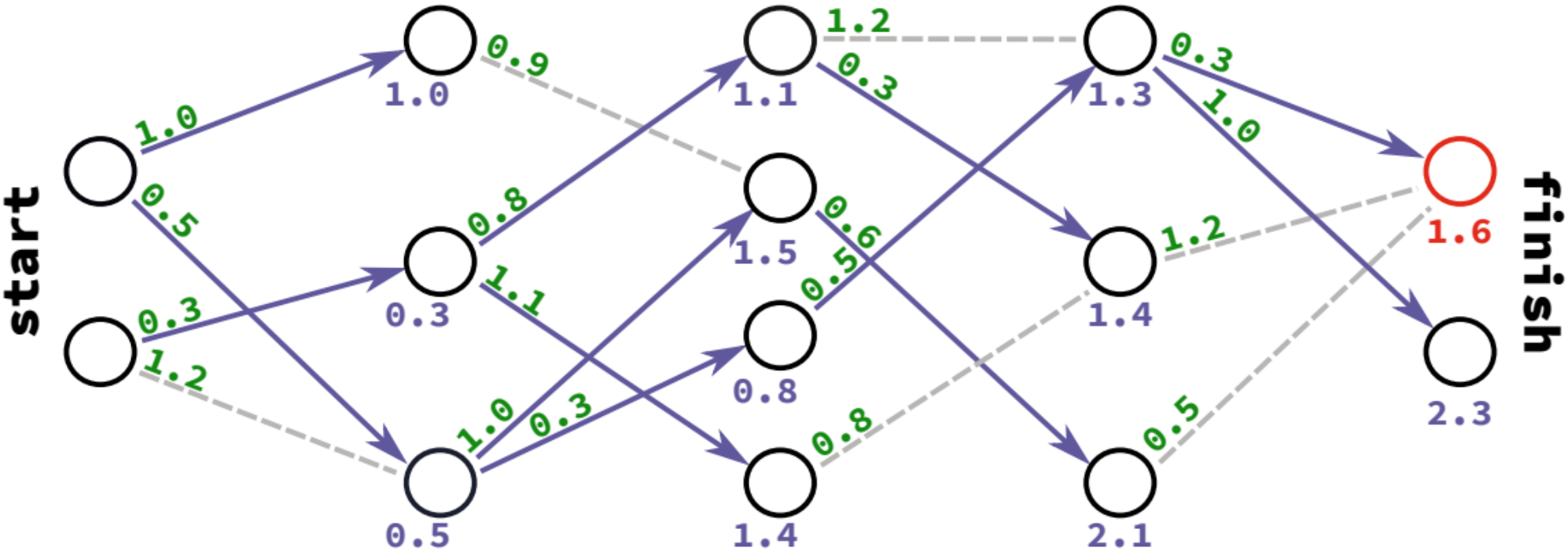
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
```

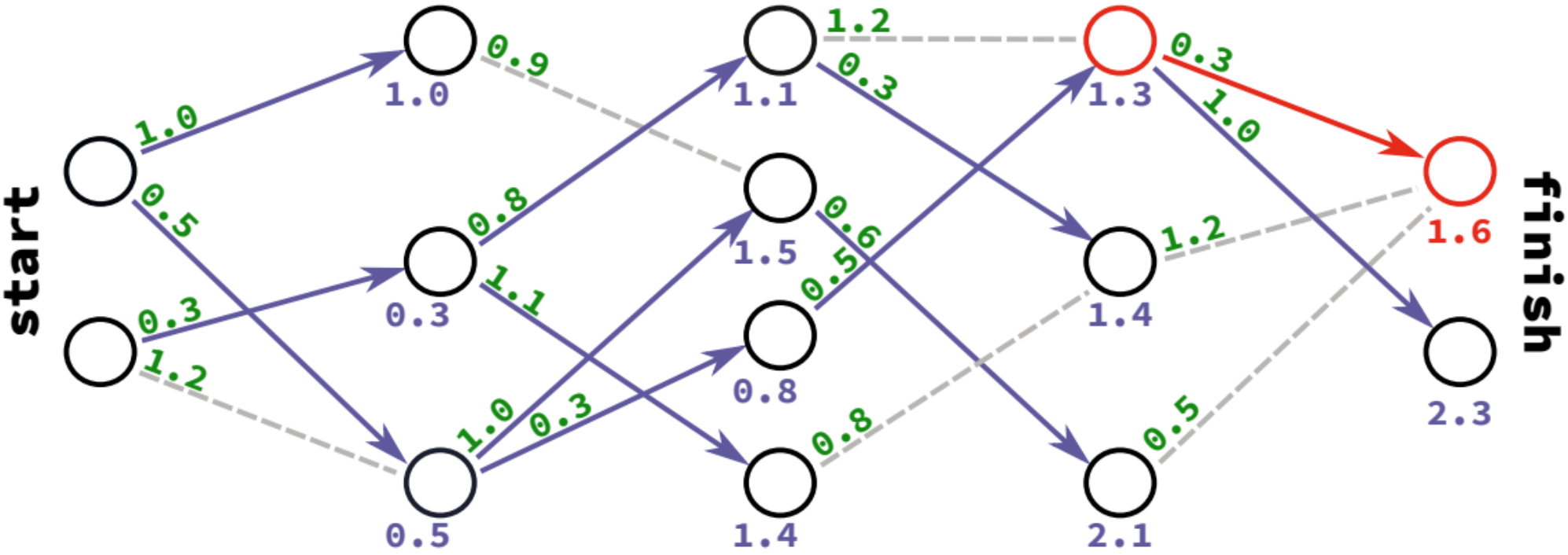
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

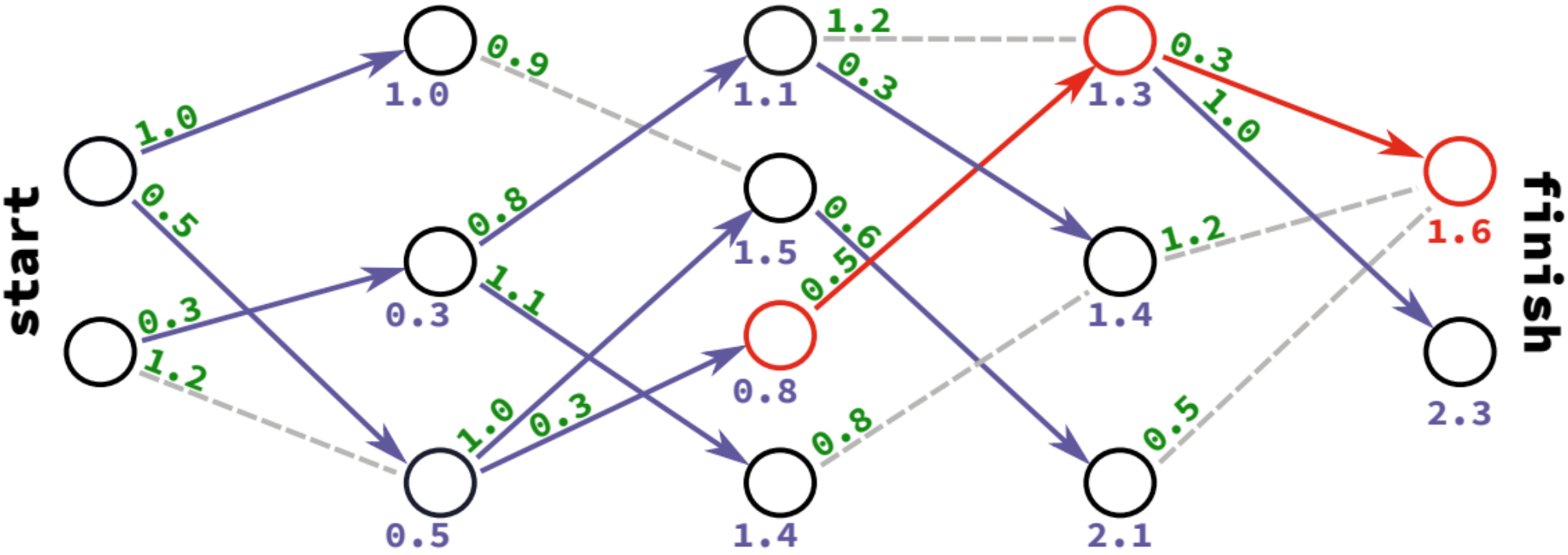
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

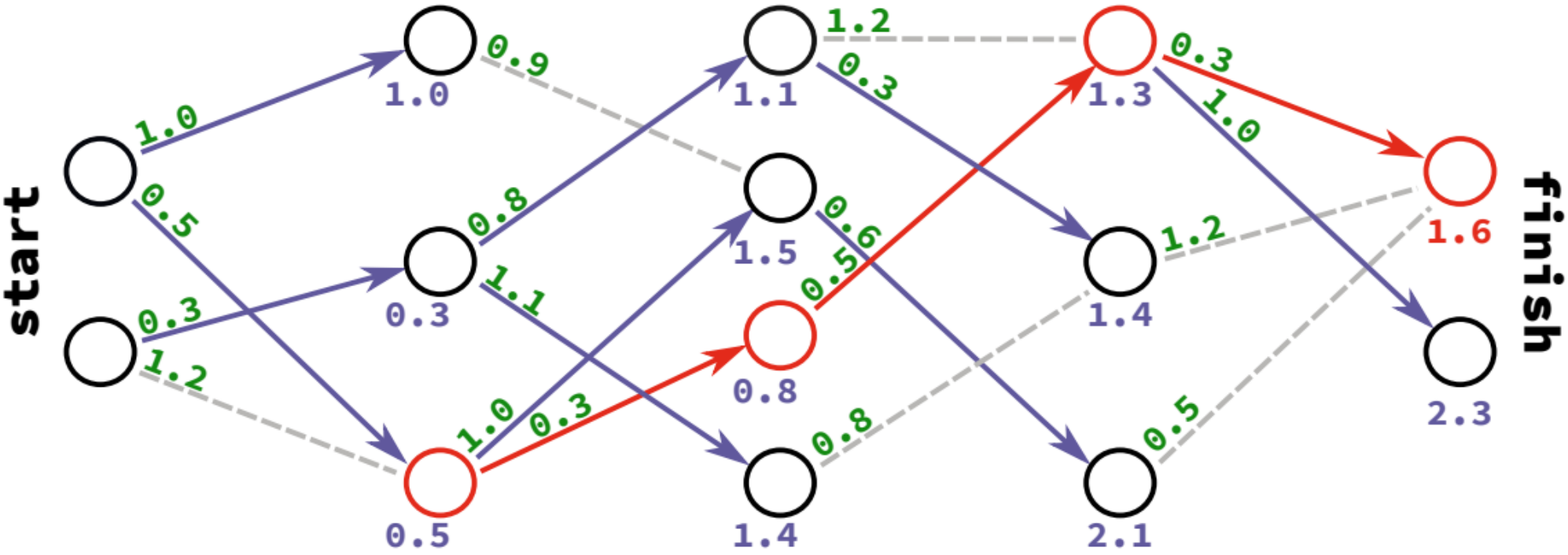
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

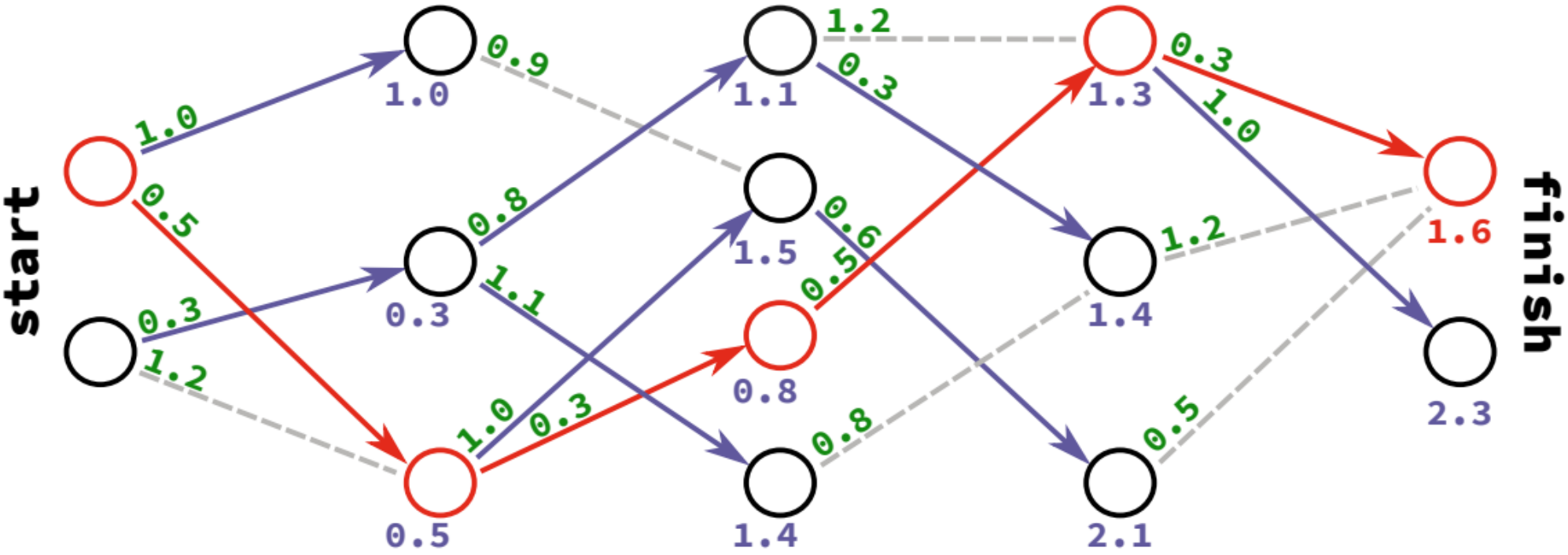
Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

Example: optimal left-to-right path (recursive version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

What made this work?

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost

What made this work?

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost
- Recursive problem structure:
  - Given a node, the best cost to reach it is the minimum among the best costs at the previous layer plus the cost of the final step to the node [recursive step]
  - Exception: nodes at the start layer: the cost to reach those is just zero [base case]

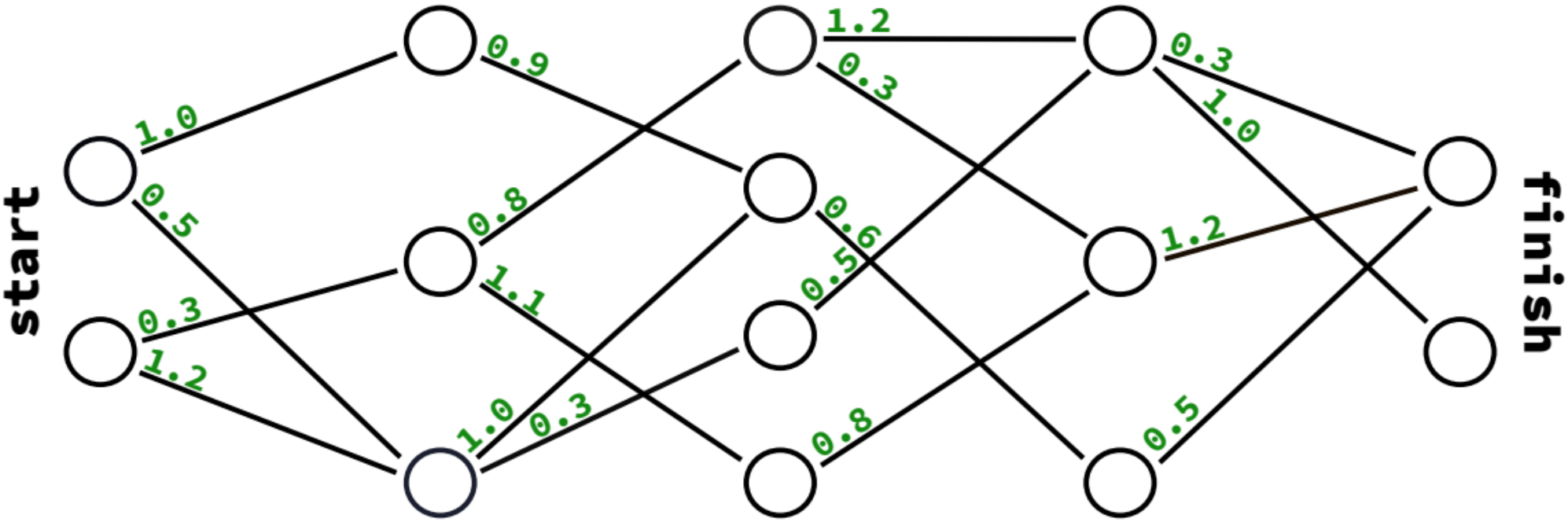
What made this work?

- A configuration is the result of a sequence of decisions (at each layer, pick a node)
- Partial configurations (incomplete decisions) have a well-defined cost
- Recursive problem structure:
  - Given a node, the best cost to reach it is the minimum among the best costs at the previous layer plus the cost of the final step to the node [recursive step]
  - Exception: nodes at the start layer: the cost to reach those is just zero [base case]
- The recursive structure (an optimal path is made up from optimal partial paths) allows the re-use of previous computations, which makes the problem polynomial
  - Crucially, the amount of optimal partial configurations is polynomial (one per node)
  - This means that every time you apply the recursive relation you take a minimum over a polynomial number of terms
  - Overall, we perform a number of computations proportional to the number of links

Top-down vs Bottom-up

- What we've just seen is recursion+memoization ↔ Top-down approach
  - Easy to code (relatively…)
  - Performance drawback: many function calls, may be expensive (in Python…)
- We can perform the computations from the start instead ↔ Bottom-up approach
  - Pre-allocate the space for the data you need (e.g. in a matrix)
  - No (explicit) recursion
  - Might cost some unnecessary computations (depends on the problem, normally not an issue)

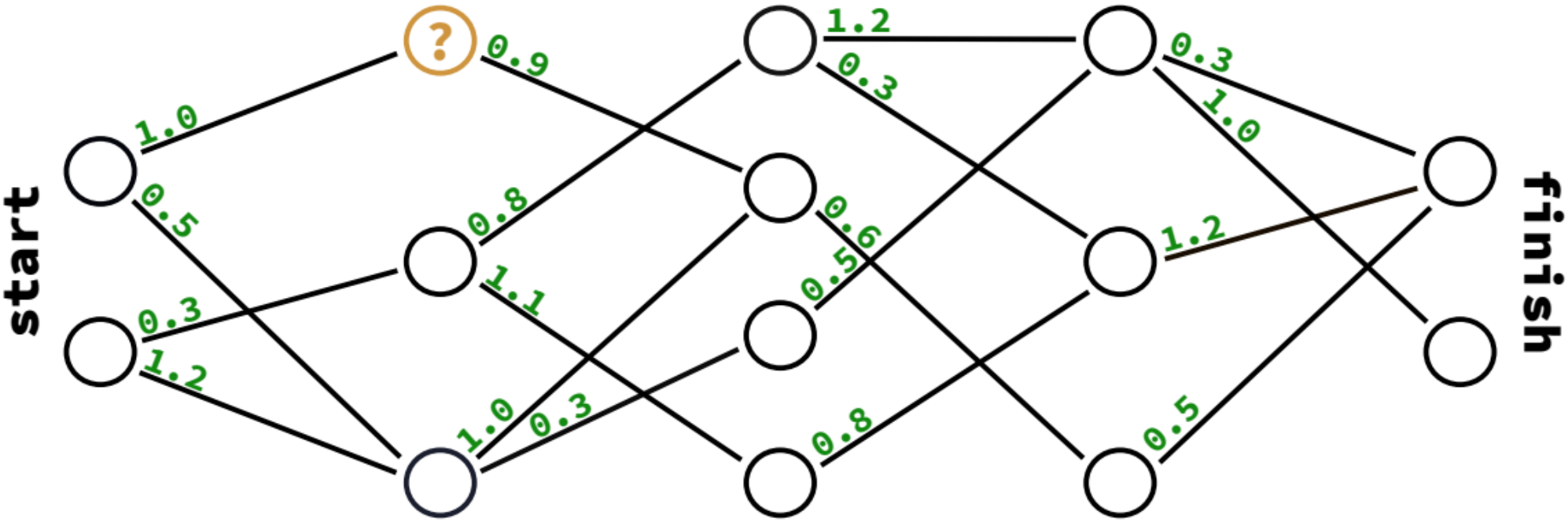
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

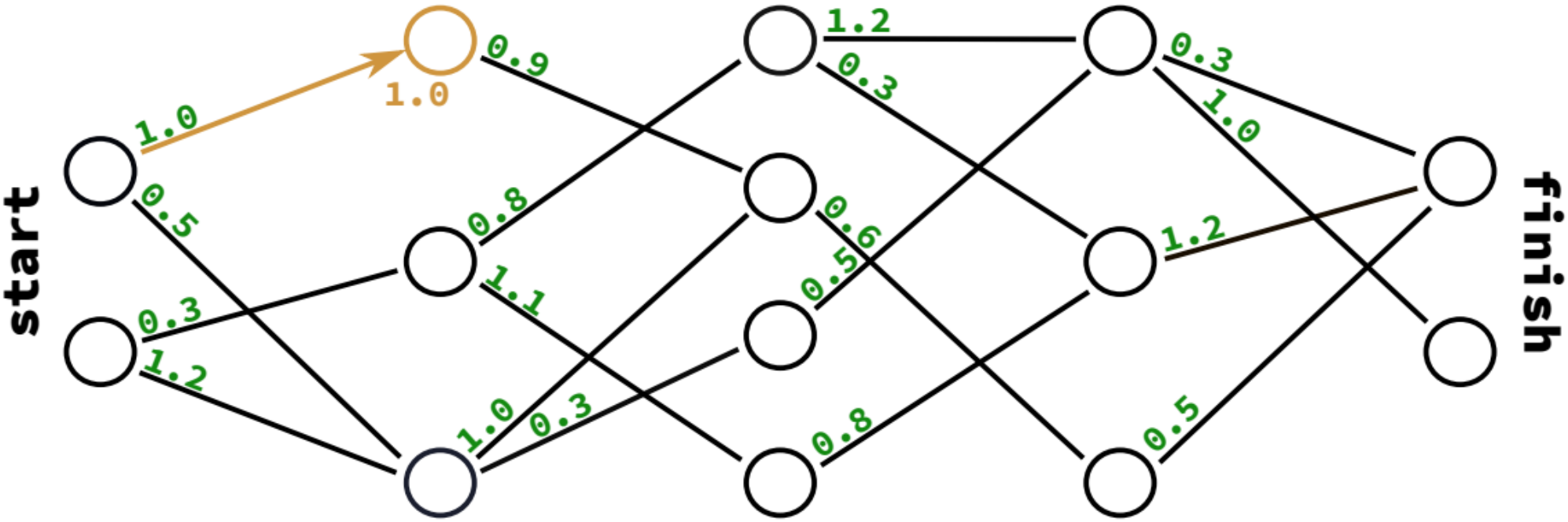
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

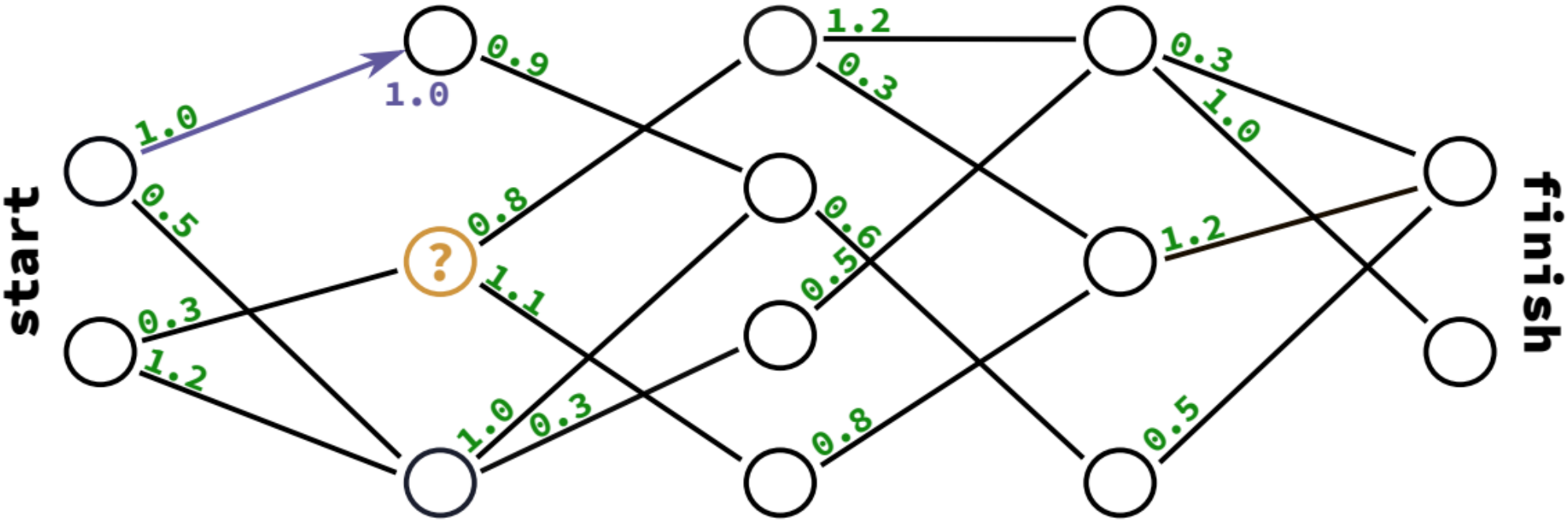
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

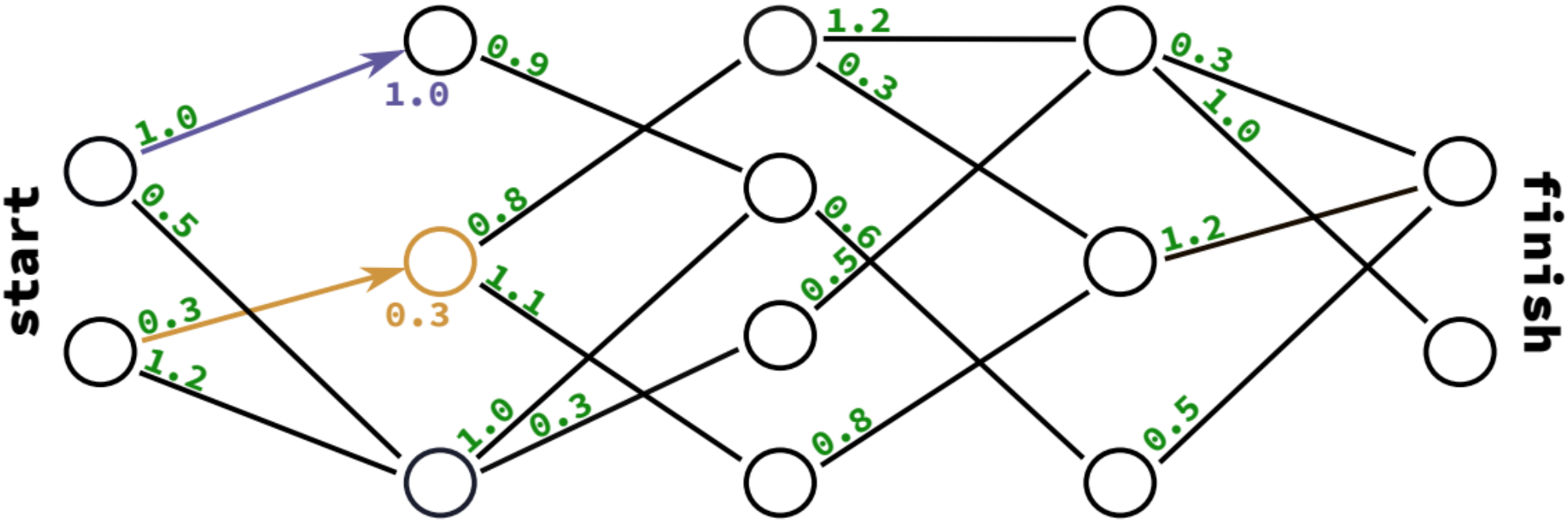
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

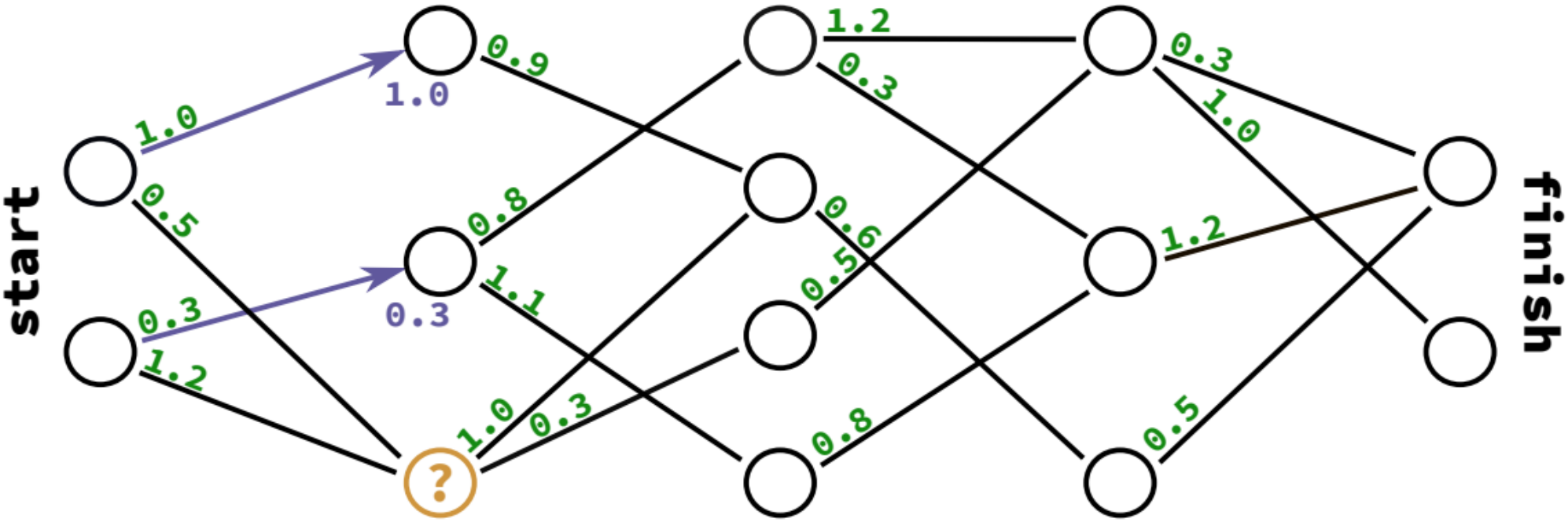
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

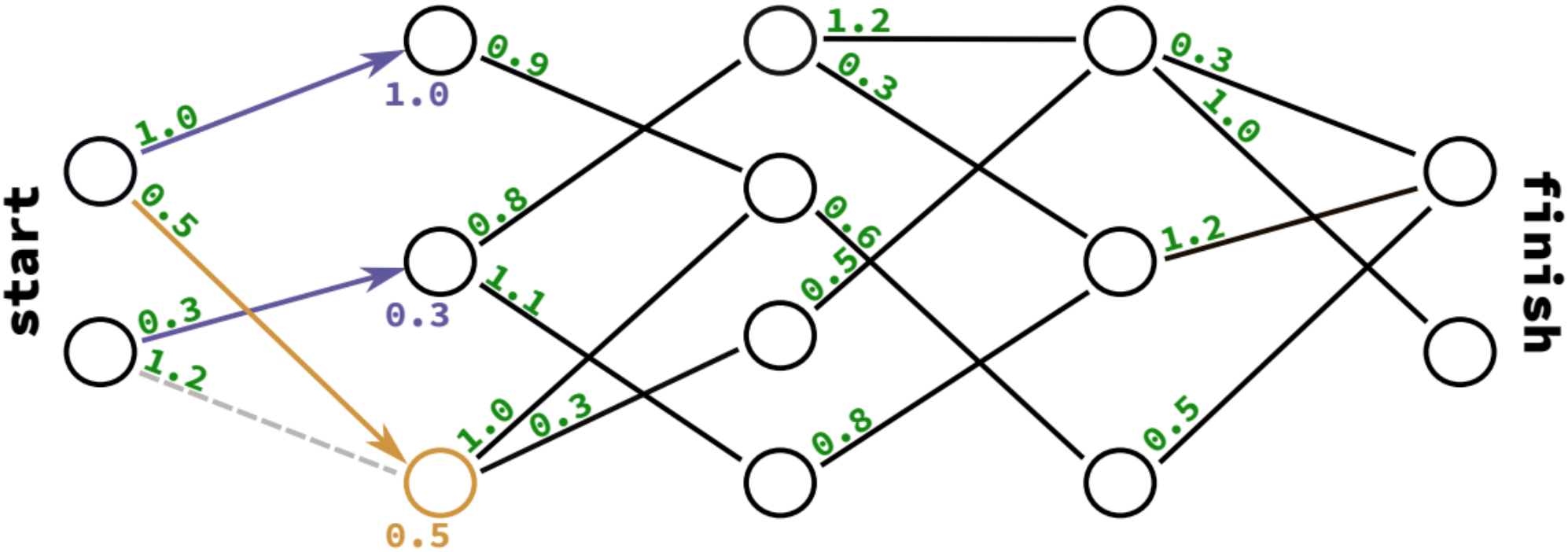
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

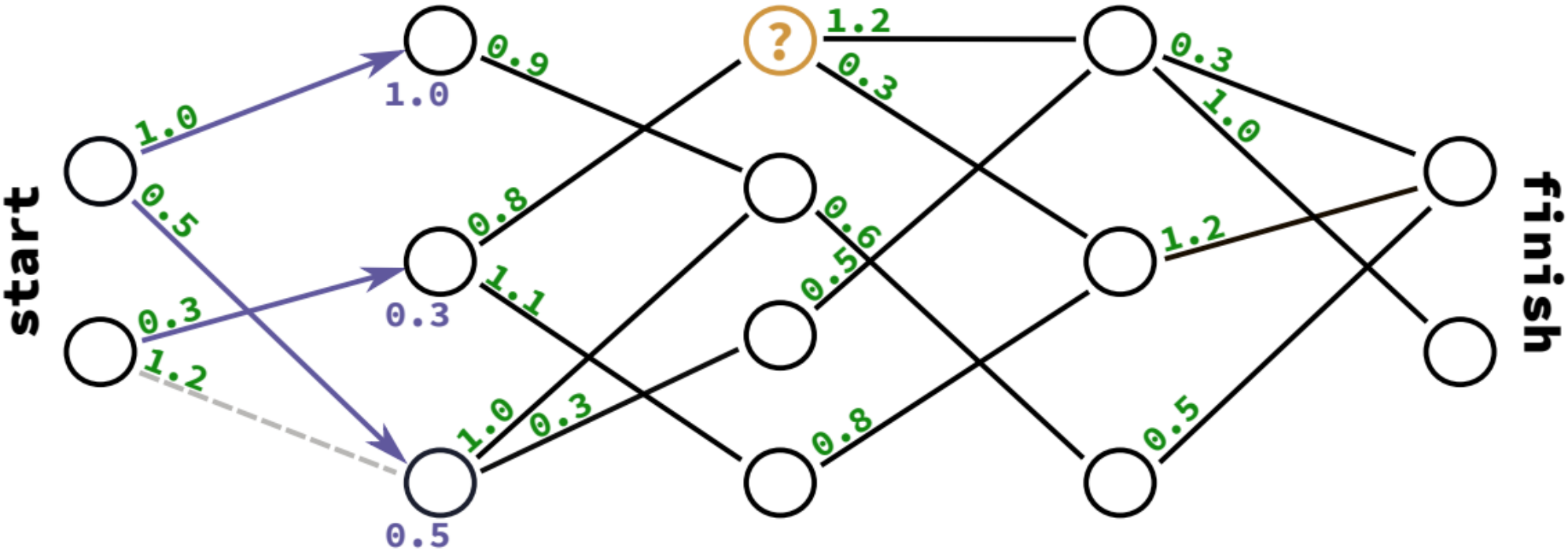
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

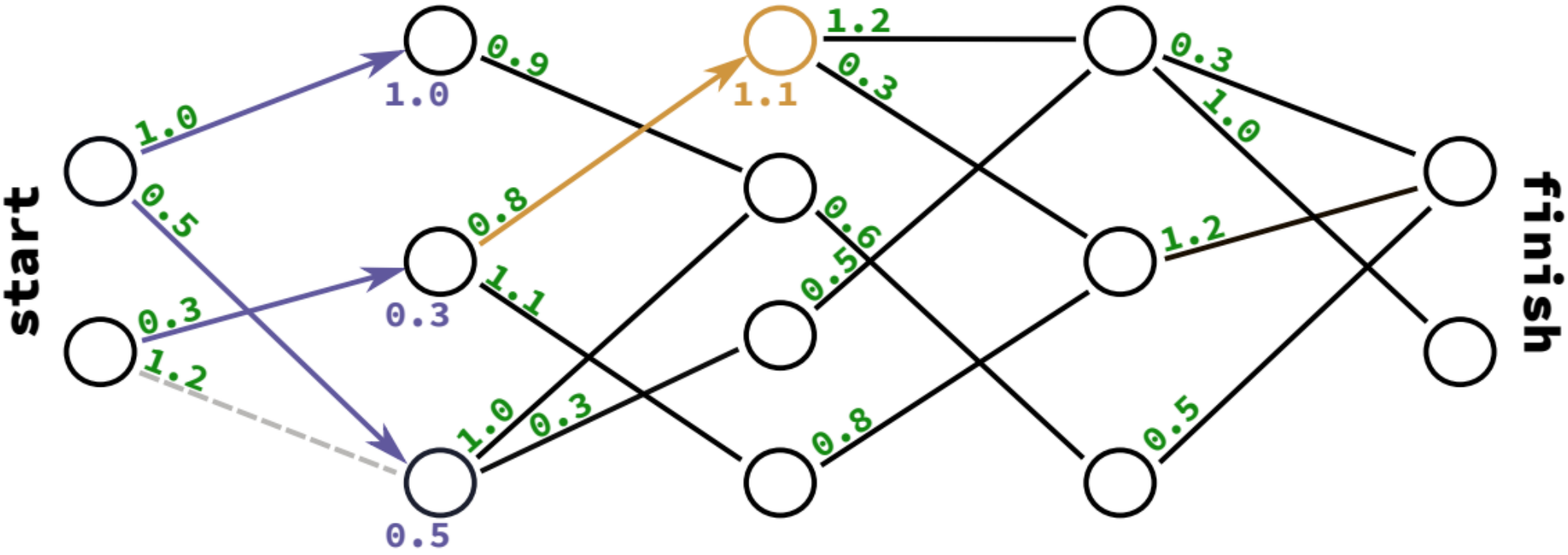
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

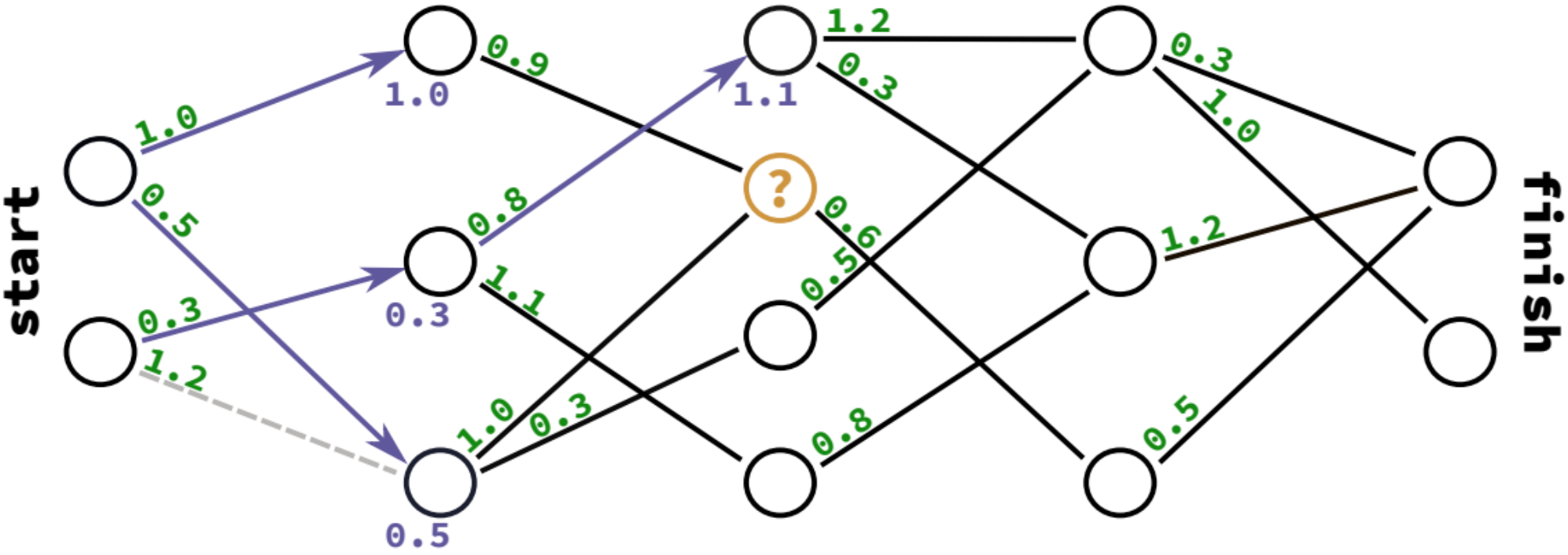
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

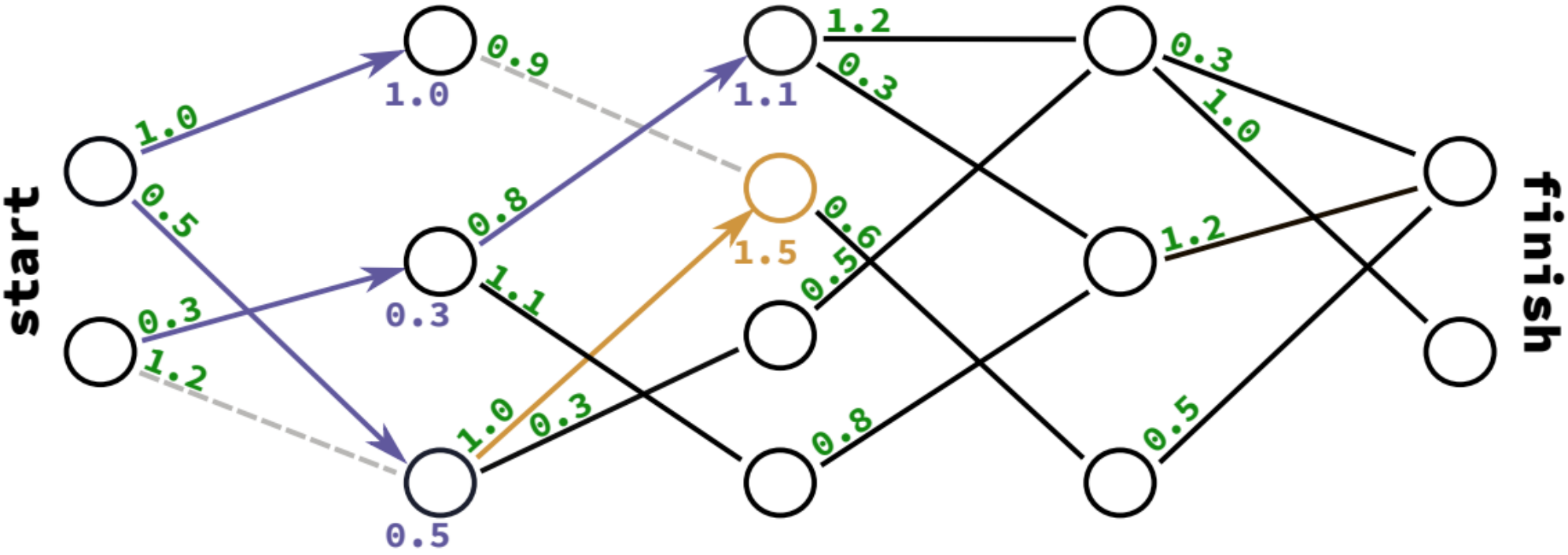
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

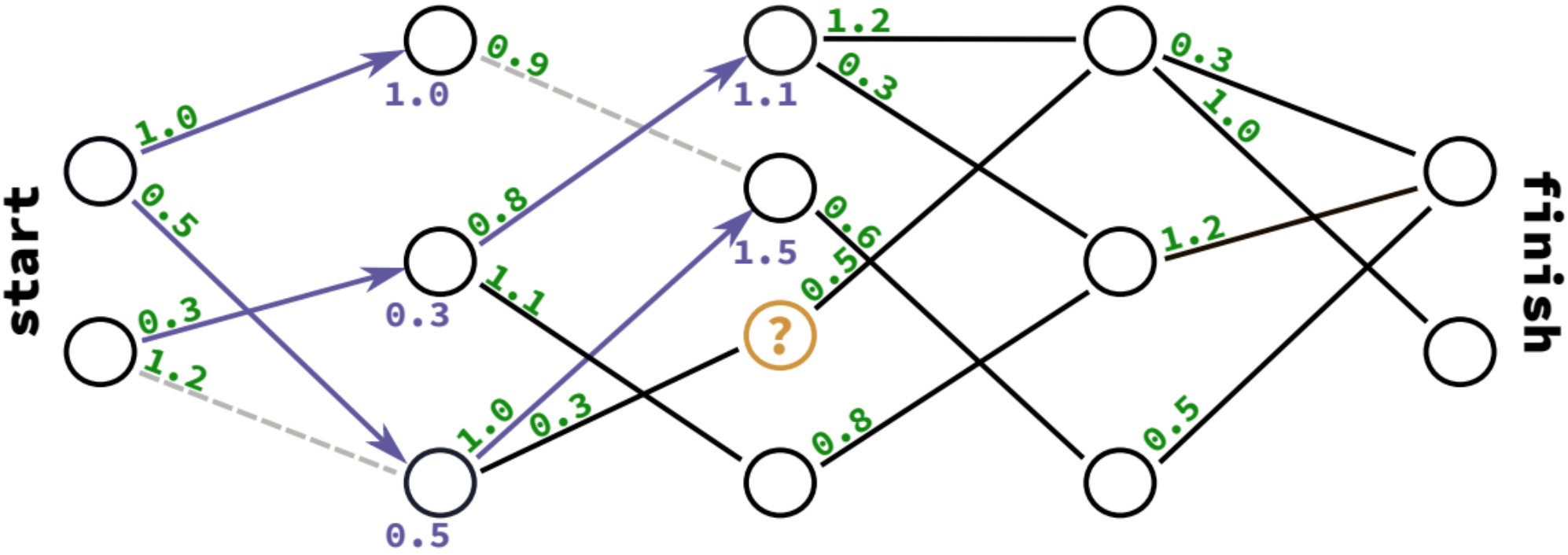
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

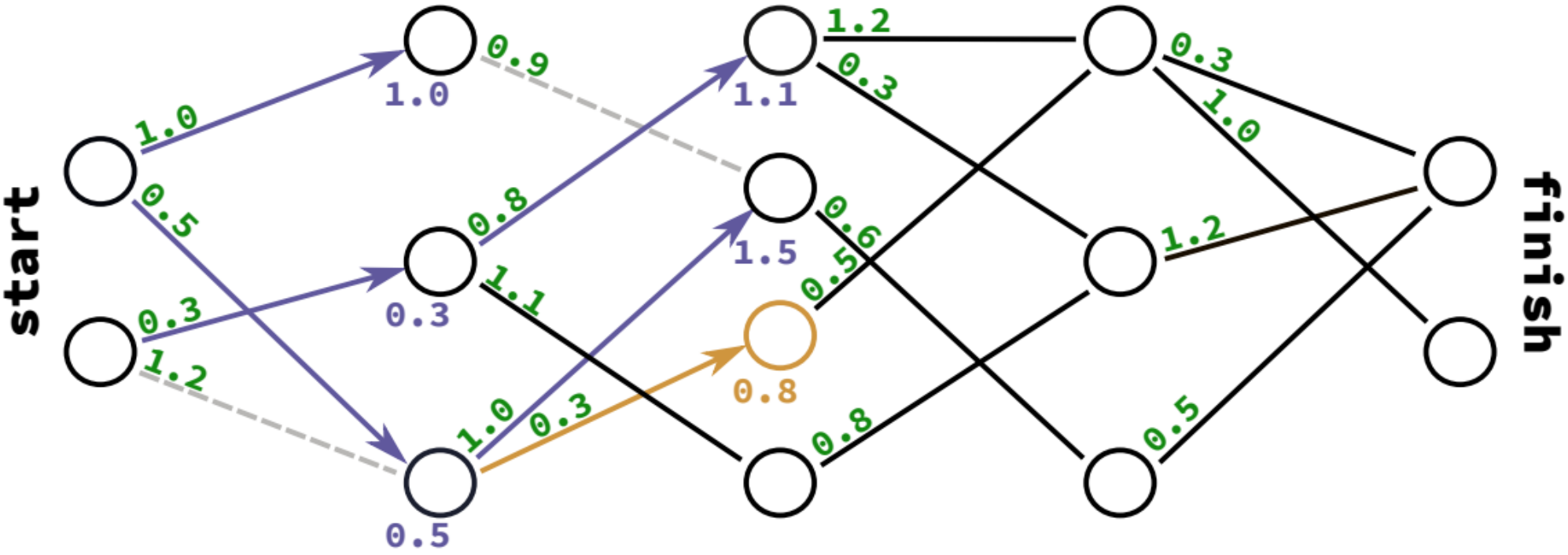
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

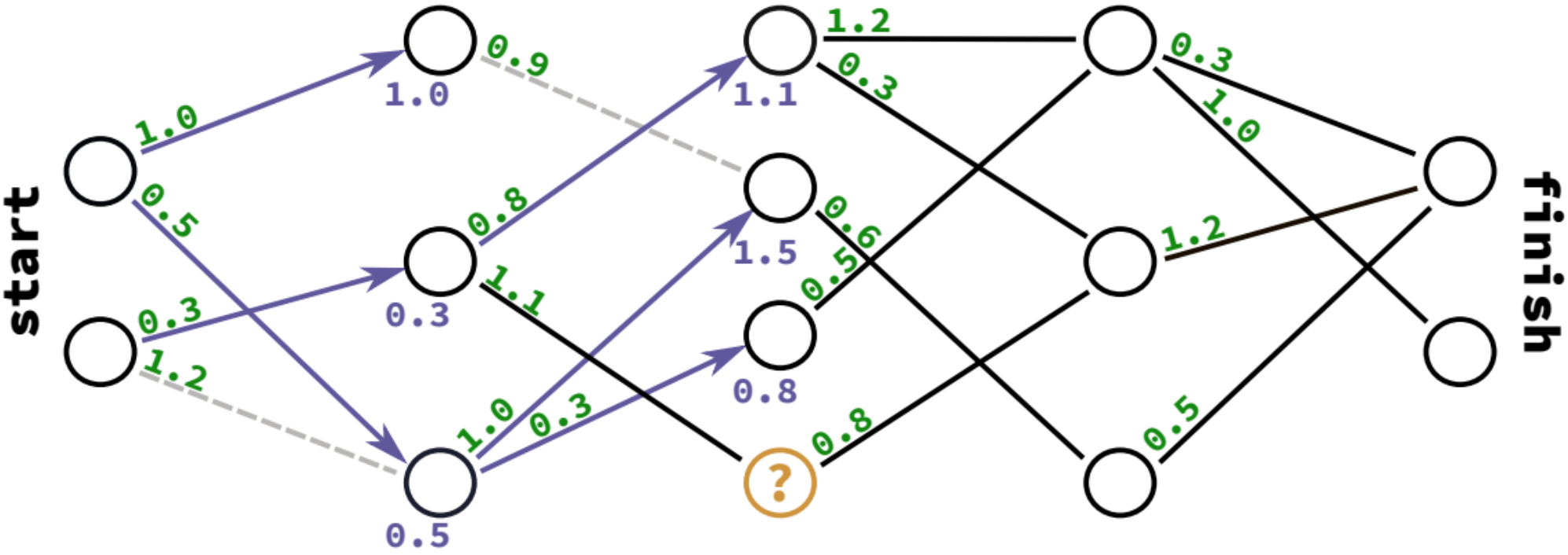
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

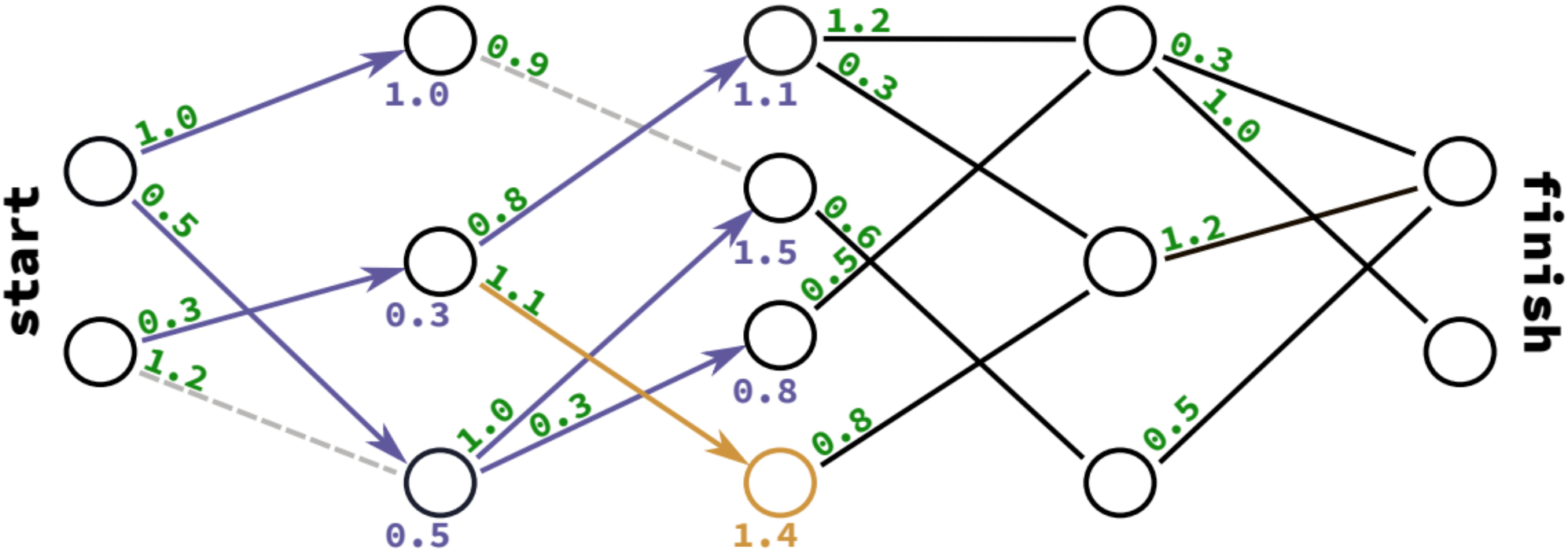
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

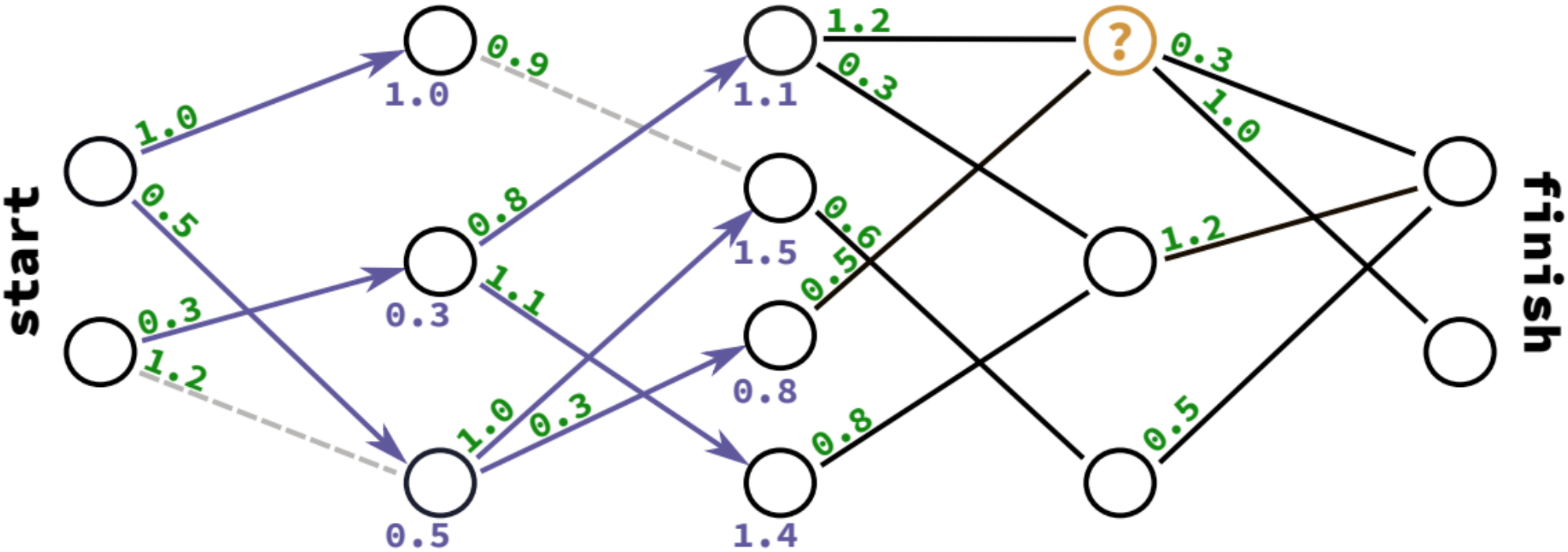
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

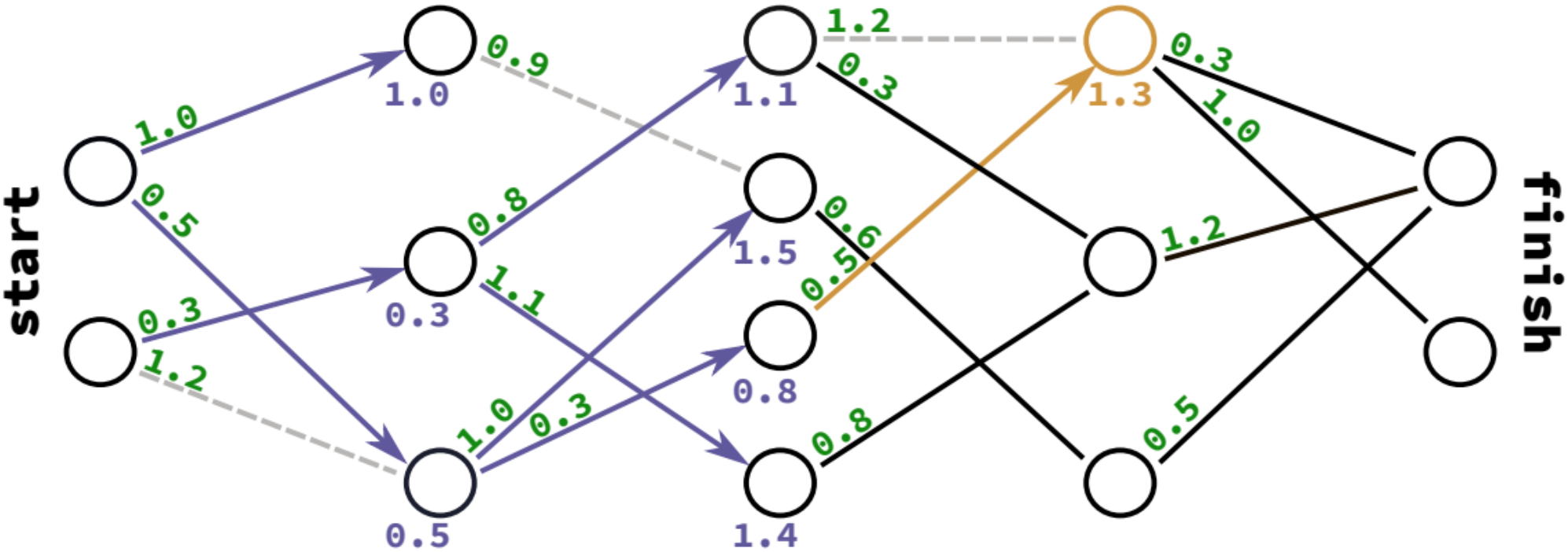
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

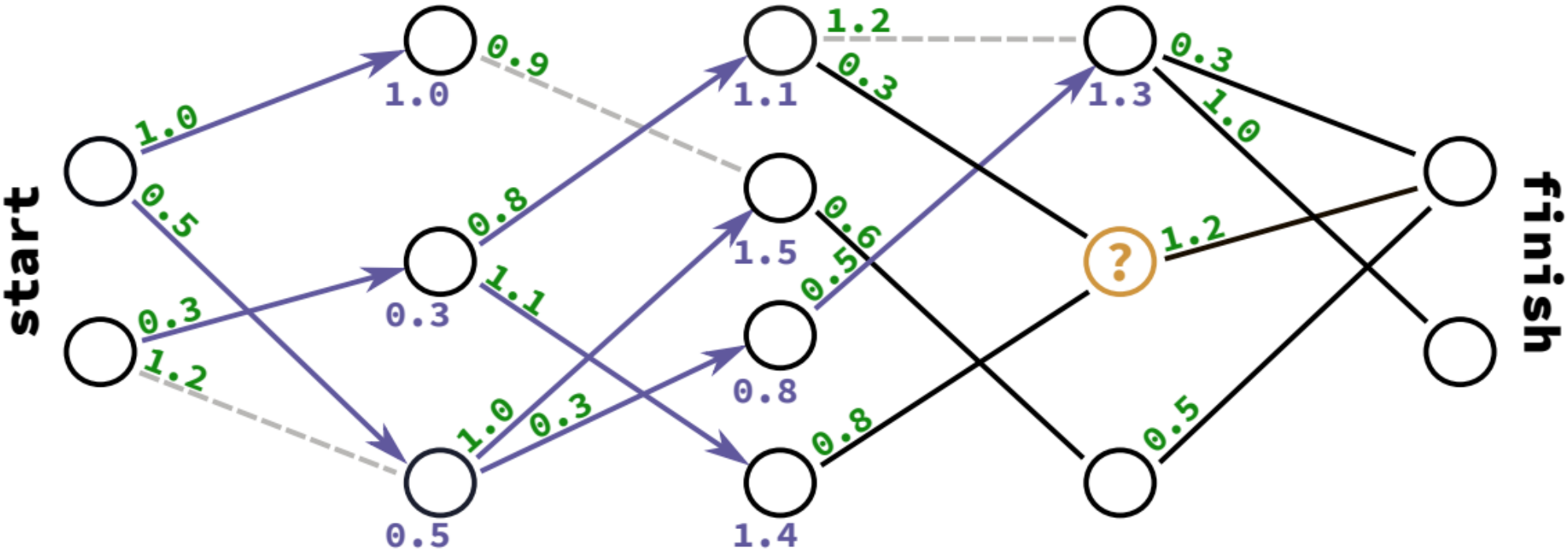
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

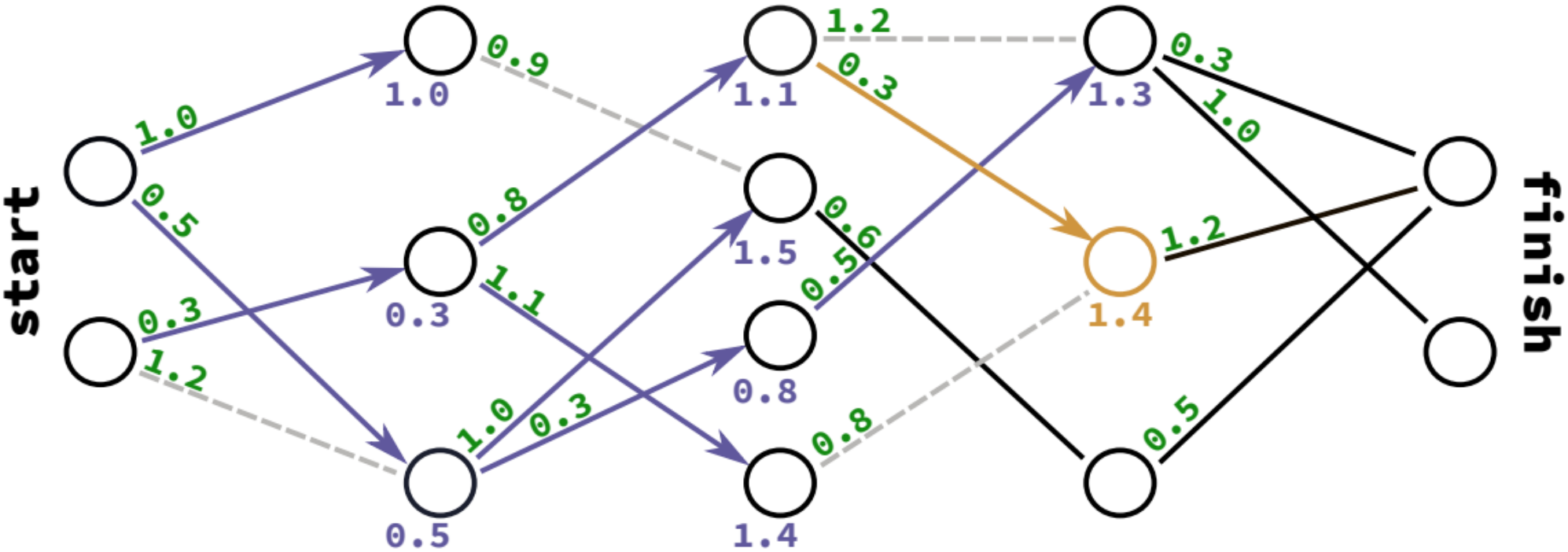
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

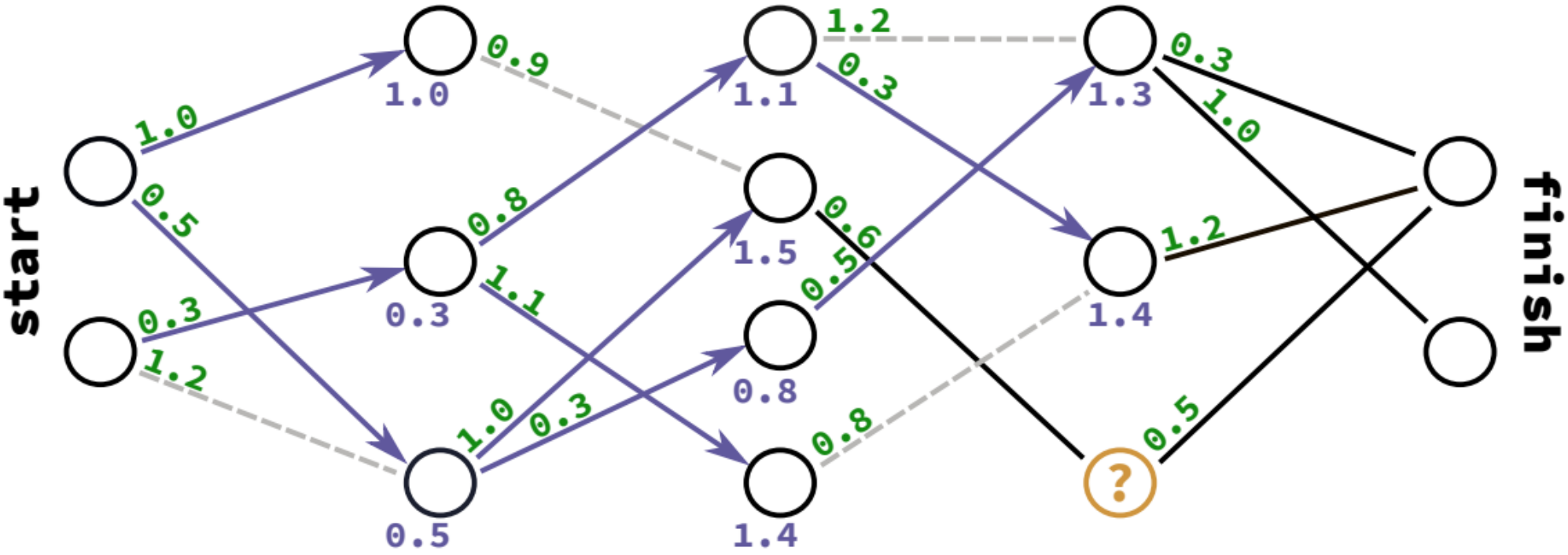
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

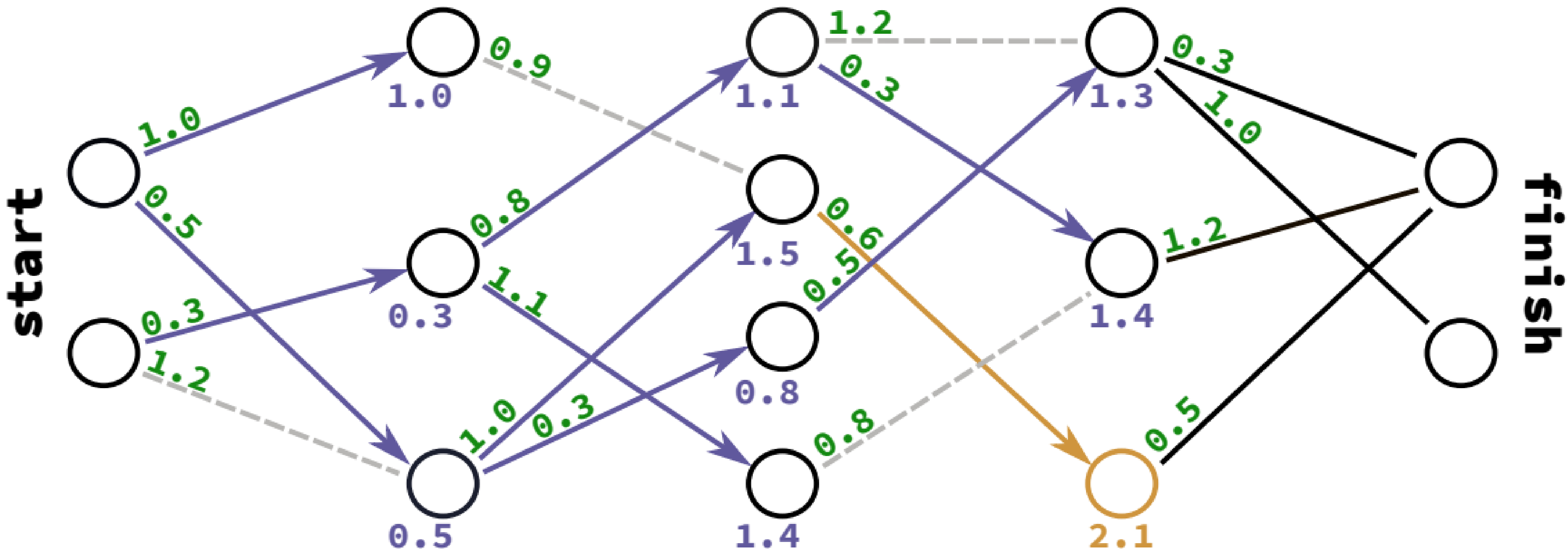
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

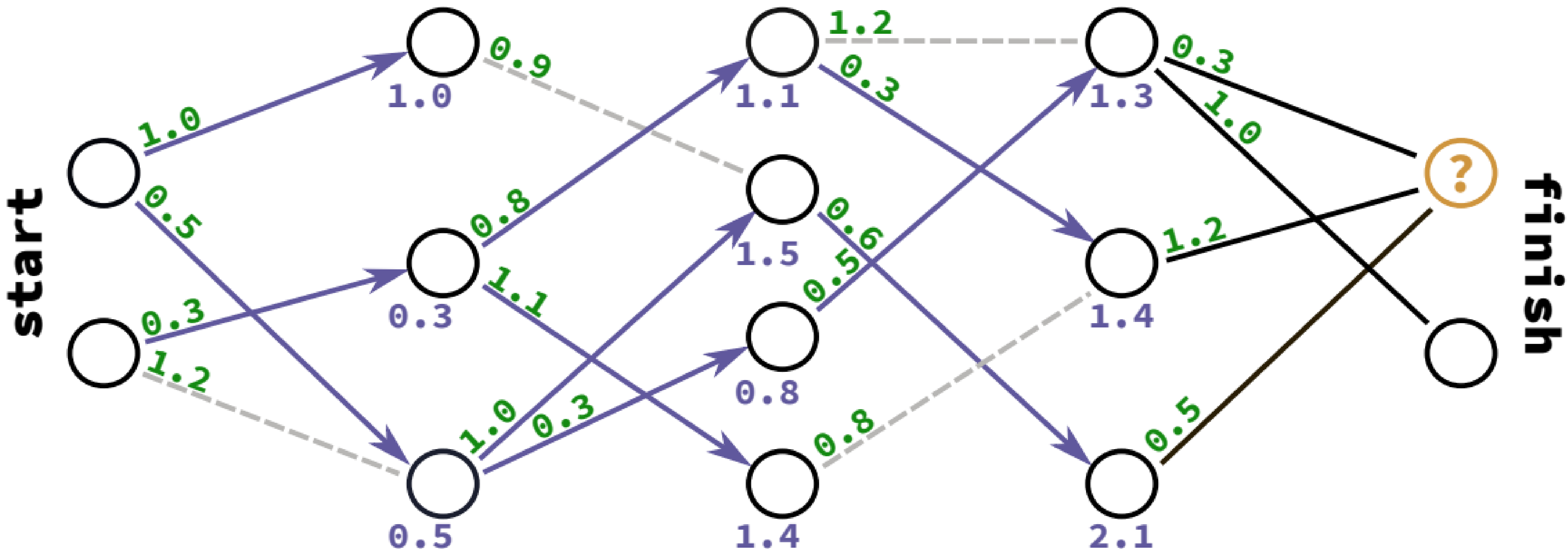
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

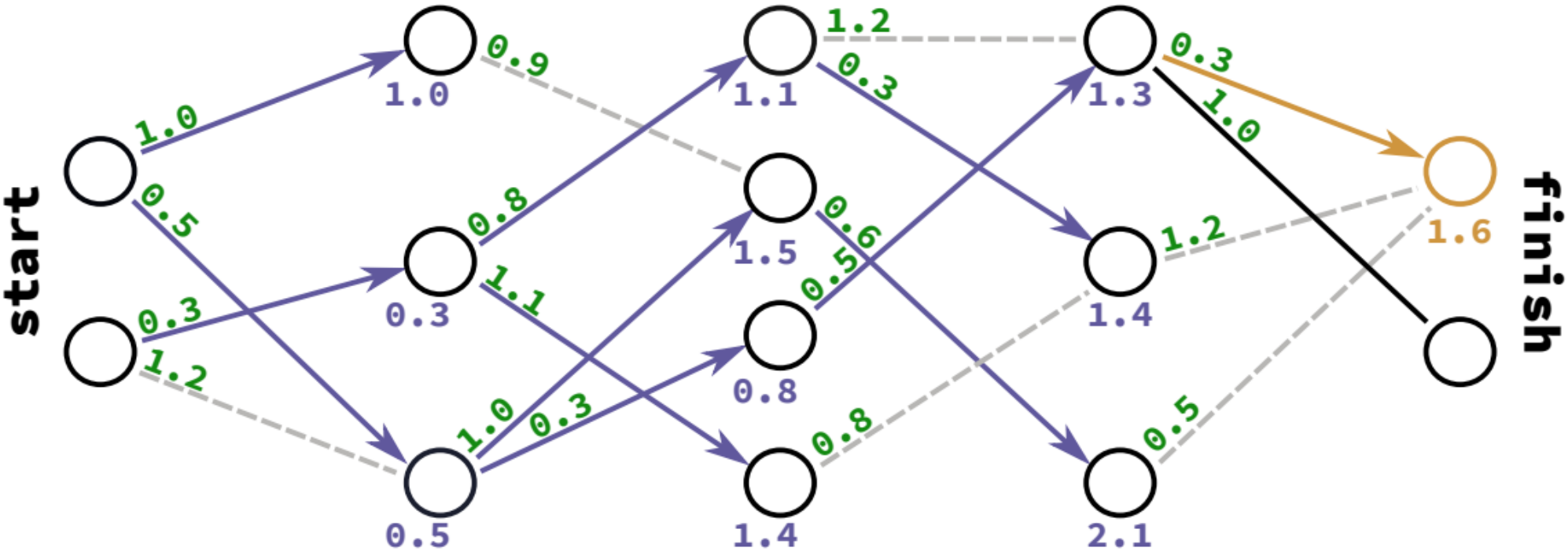
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

<u>Key question</u>: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

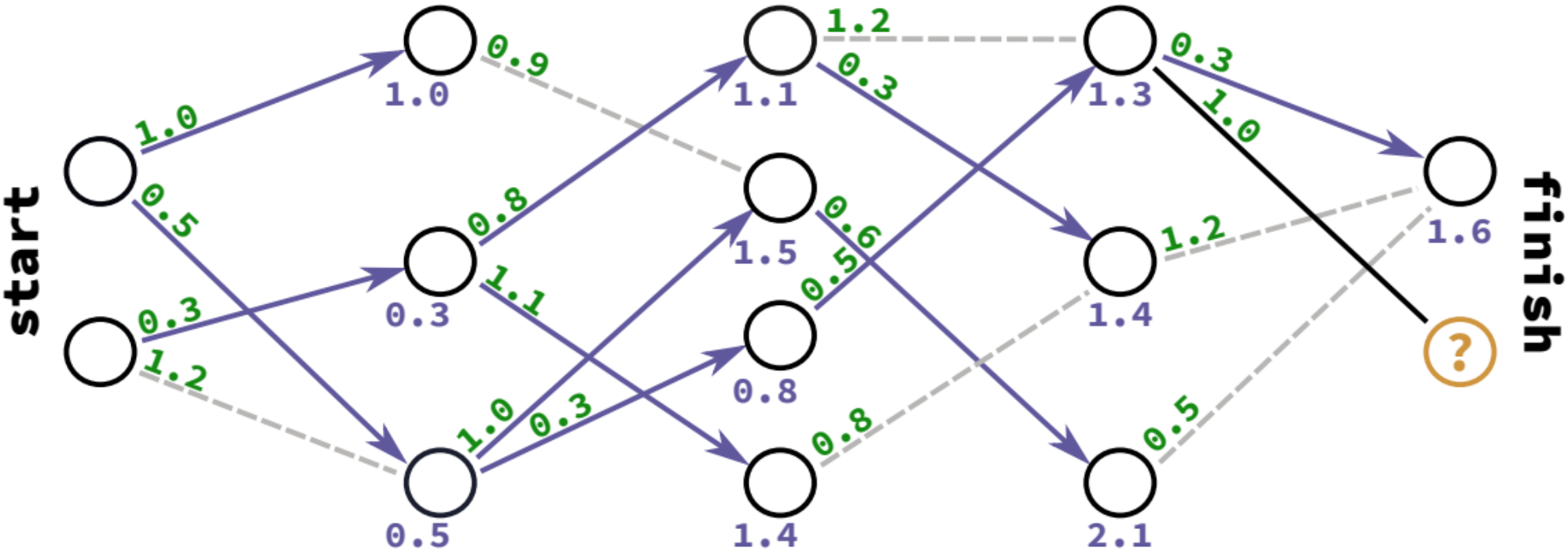
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

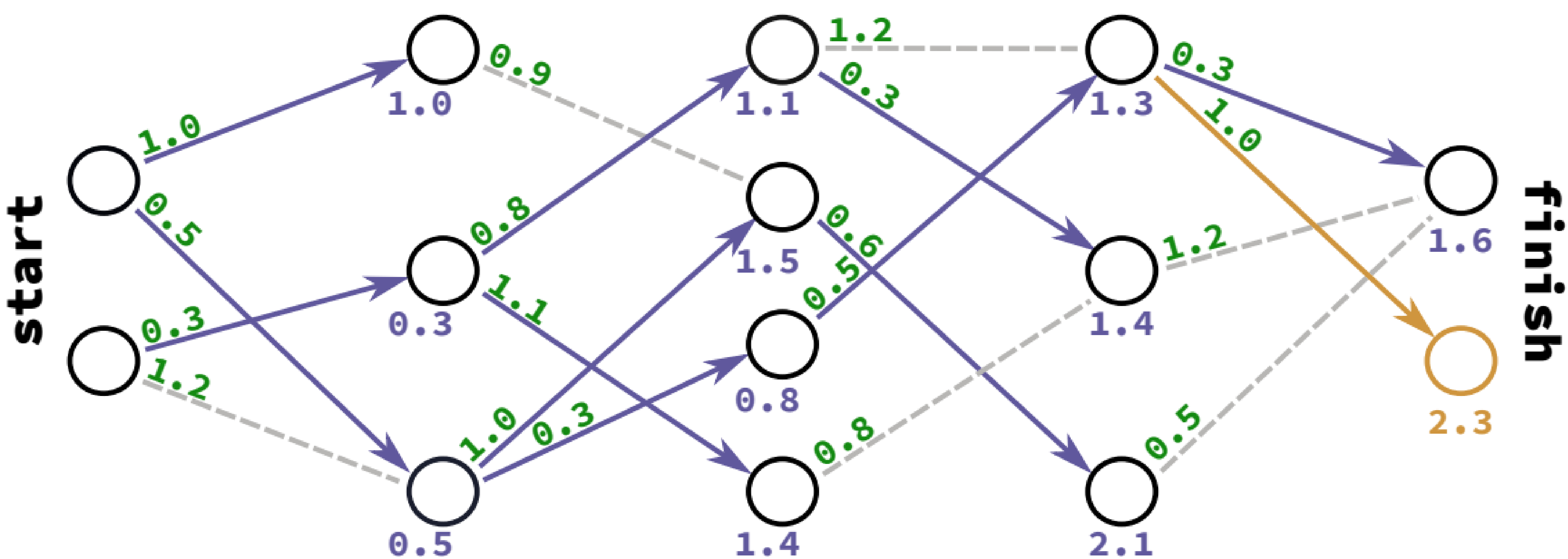
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Key question: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

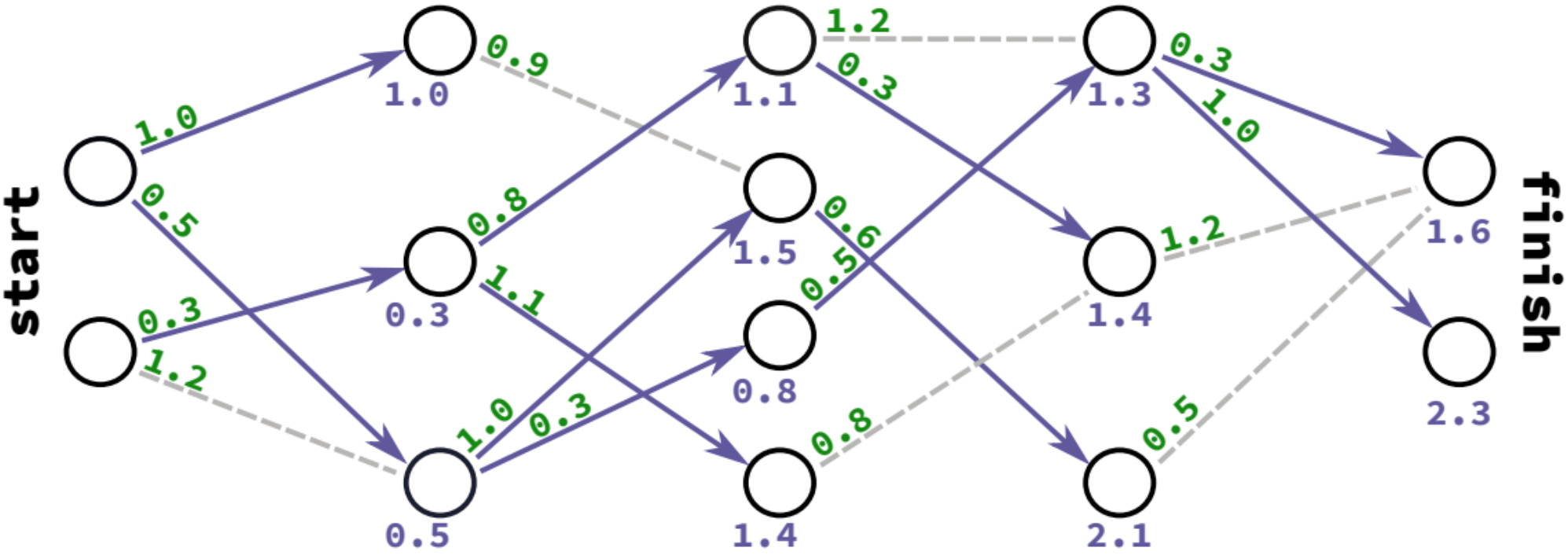
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

<u>Key question</u>: given a node, what's the min cost to get there? (We ask this question for the last layer and then recur backwards)

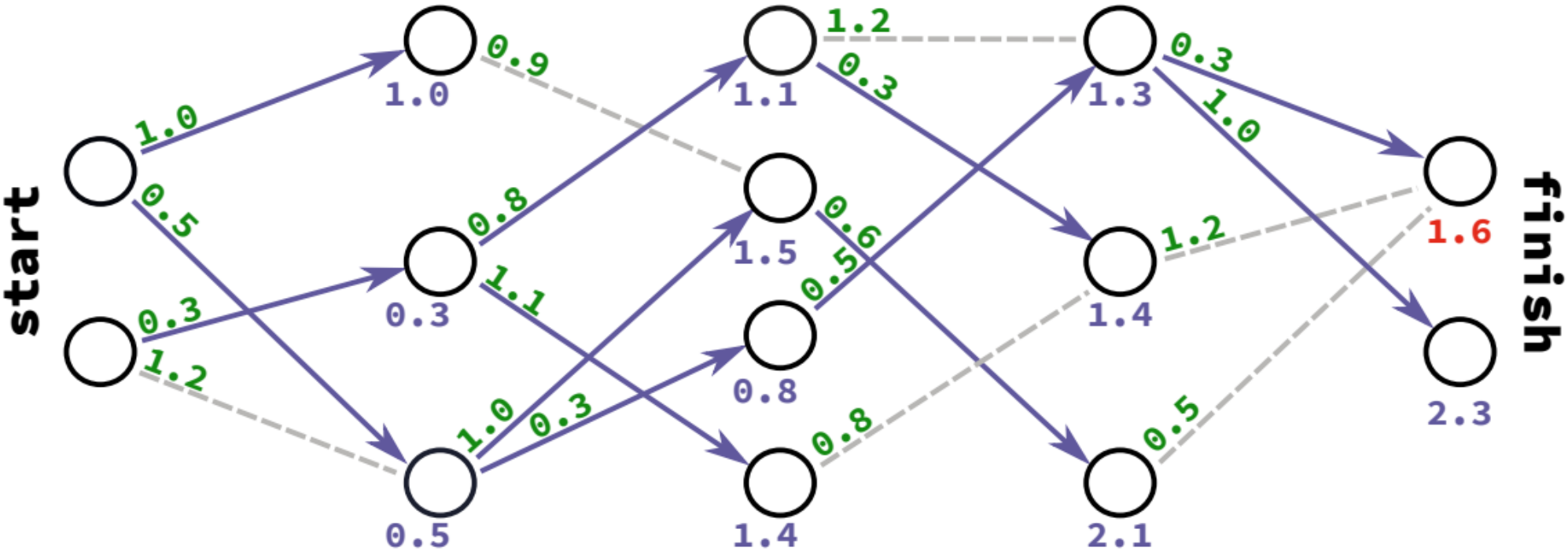
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

Forward pass completed

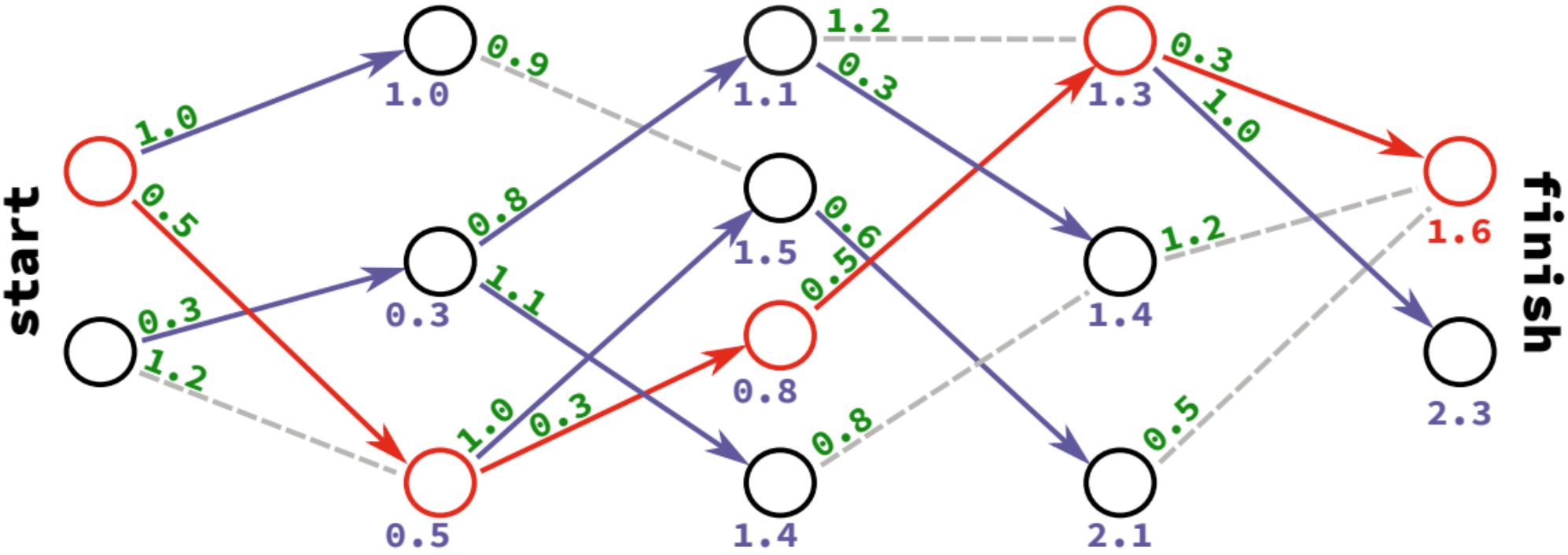
Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
```

Example: optimal left-to-right path (bottom-up version)



goal = find the best path that goes from the left to the right
circles = path nodes
lines = available connections
green numbers = costs (partial)
one whole path = configuration x

```
Forward pass completed
Found the optimal cost
Backtrack the optimal path
```

Some important general facts

- It's always possible to use the bottom-up approach
  - The memory structures you need depend on the problem
  - Usually slightly better performance (also easier to debug)
- The recursive relation is always about the cost
- But usually you want the configuration too (e.g. the best path)
- Recursive (Top-down) approach
  - Memoize the partial costs and the partial best solutions
  - Compose the partial solutions in the recursive step
- Bottom-up approach:
  - Forward pass: compute partial costs + the decision made to achieve it
  - Backward pass: backtrack the best configuration

Part II
Seam Carving
(now with code)

Another example: Seam Carving

- Image manipulation technique to resize images along one direction trying to preserve the contents
- DEMO

Another example: Seam Carving

- Image manipulation technique to resize images along one direction trying to preserve the contents
- DEMO
- Based on these ideas/assumptions (here's a more thorough explanation):
  - We seek the lowest-content "seam" – a wiggling path that connects two sides of the image
  - An edge-detection filter is applied to the image to determine where the content is
  - So "lowest-content" means "less overall edges": the optimal seam passes through smooth parts of the image
    - Remove optimal seam: preserves the "content", is not too noticeable
  - Computing the optimal seam can be done efficiently with dynamic programming!

Another example: Seam Carving

- Image manipulation technique to resize images along one direction trying to preserve the contents
- DEMO
- Based on these ideas/assumptions (here's a more thorough explanation):
  - We seek the lowest-content "seam" – a wiggling path that connects two sides of the image
  - An edge-detection filter is applied to the image to determine where the content is
  - So "lowest-content" means "less overall edges": the optimal seam passes through smooth parts of the image
    - Remove optimal seam: preserves the "content", is not too noticeable
  - Computing the optimal seam can be done efficiently with dynamic programming!
- Basically the same optimization problem as before! (But easier to implement)

Seam Carving in brief (I)

- Example of a seam: g is the filtered image (a 2-d array with the cost of each pixel)
  - Goes from top to bottom
  - Horizontal displacement from one row to the next is `-1`, `0` or `1`

`g`

	0	1	2	3
0	1.0	1.0	**0.5**	0.0
1	2.0	**1.0**	1.5	3.0
2	**0.0**	0.5	1.5	2.0
3	**0.0**	0.75	1.0	0.5
4	1.0	**0.5**	0.5	1.0

`seam`

2
1
0
0
1

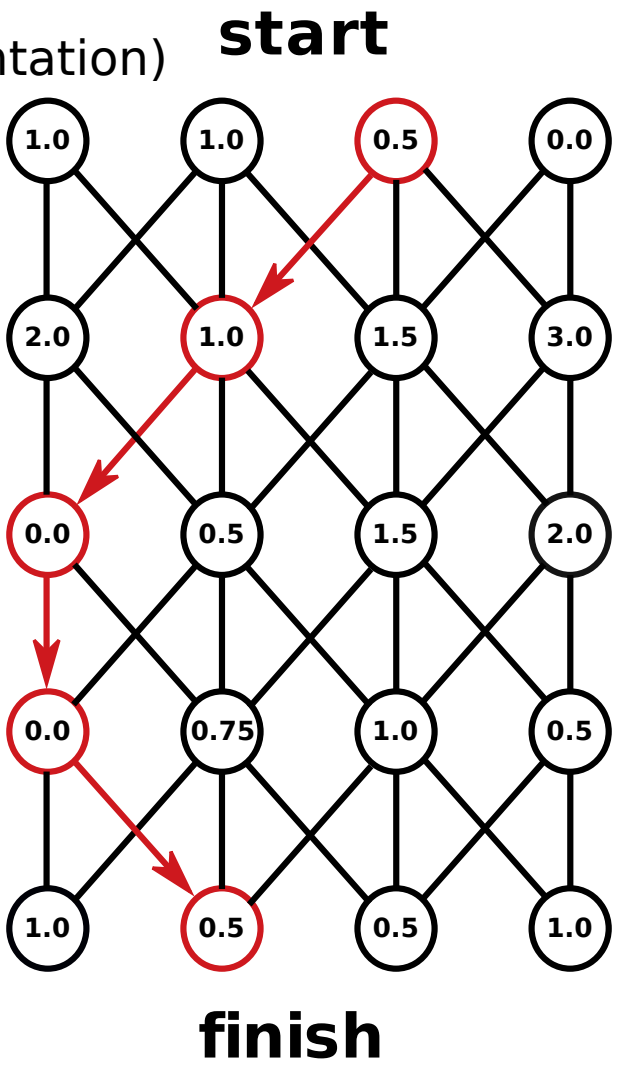
`Total cost = 0.5 + 1.0 + 0.0 + 0.0 + 0.5 = 2.0`

Seam Carving in brief (II)

- Basically the same problem as before (difference: cost on nodes, no cost on edges; also uniform graph structure).
- Exact same solving strategy (different implementation)

g

	0	1	2	3
0	1.0	1.0	0.5	0.0
1	2.0	1.0	1.5	3.0
2	0.0	0.5	1.5	2.0
3	0.0	0.75	1.0	0.5
4	1.0	0.5	0.5	1.0



Seam Carving in brief (III)

- Dynamic programming scheme: optimal costs computation

base case: $c_{0j} = g_{0j}$

recursive relation: $c_{ij} = g_{ij} + \min\left\{c_{(i-1)(j-1)},\ c_{(i-1)j},\ c_{(i-1)(j+1)}\right\} \quad \text{if } i > 0$

g

	0	1	2	3
0	1.0	1.0	0.5	0.0
1	2.0	1.0	1.5	3.0
2	0.0	0.5	1.5	2.0
3	0.0	0.75	1.0	0.5
4	1.0	0.5	0.5	1.0

row 0 copied

c

	0	1	2	3
0	1.0	1.0	0.5	0.0
1	3.0	1.5	1.5	3.0
2	1.5	2.0	3.0	3.5
3	1.5	2.25	3.0	3.5
4	2.5	2.0	2.75	4.0

1.0 + min(2.0, 3.0, 3.5)

Seam Carving in brief (IV)

g

	0	1	2	3
0	1.0	1.0	0.5	0.0
1	2.0	1.0	1.5	3.0
2	0.0	0.5	1.5	2.0
3	0.0	0.75	1.0	0.5
4	1.0	0.5	0.5	1.0

c

	0	1	2	3
0	1.0	1.0	0.5	0.0
1	3.0	1.5	1.5	3.0
2	1.5	2.0	3.0	3.5
3	1.5	2.25	3.0	3.5
4	2.5	2.0	2.75	4.0

best

Dyn. Progr.

whence

	0	1	2	3
0				
1				
2				
3				
4				