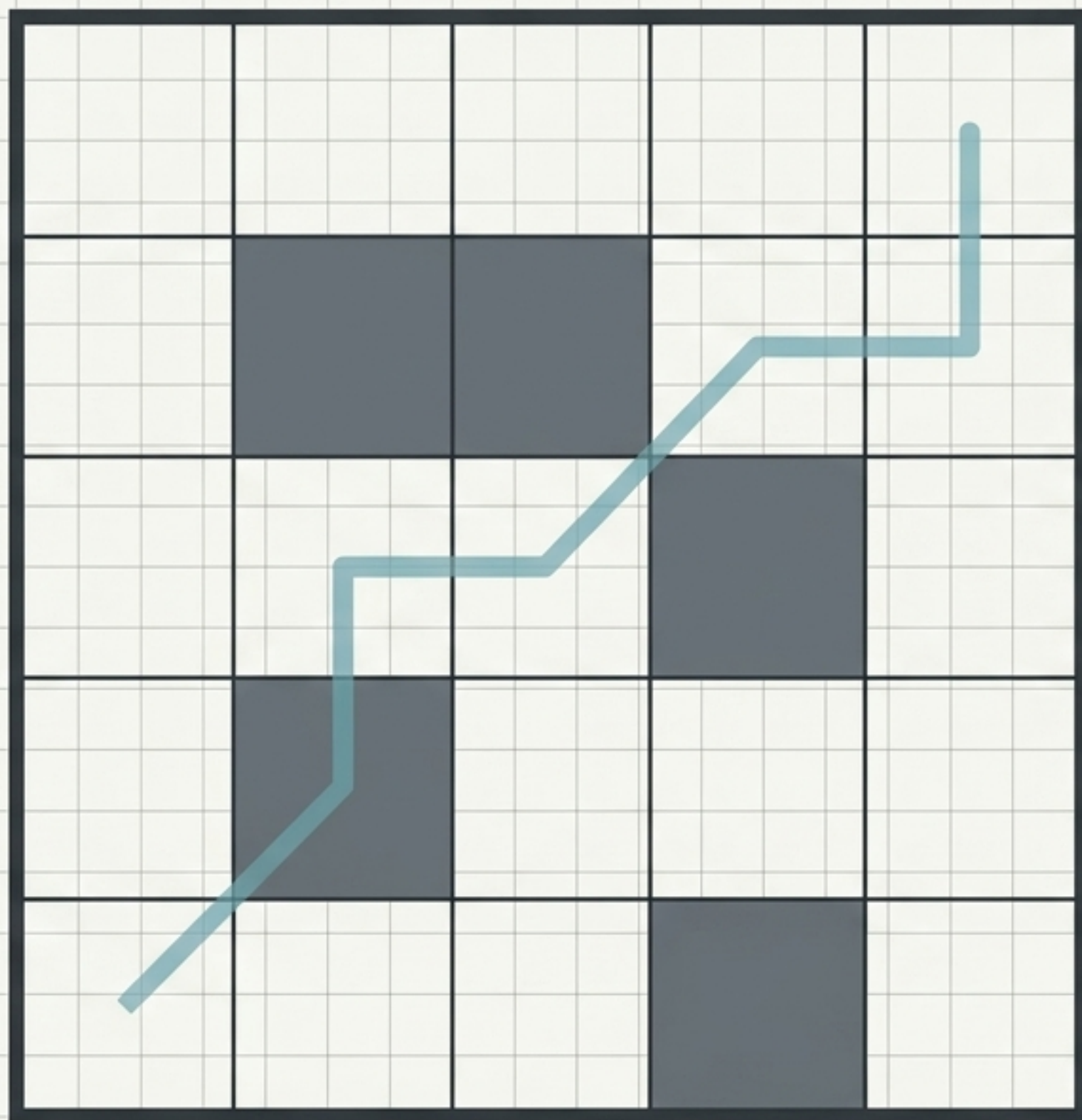


迷路探索アルゴリズムの軌跡： 複数解探索の視覚的アプローチ

応用情報技術者試験 午後問3 設問4(2) — プログラムの動的挙動を追跡する





追跡対象：第2の解に至るまでの「移動」と「訪問」

Target 1

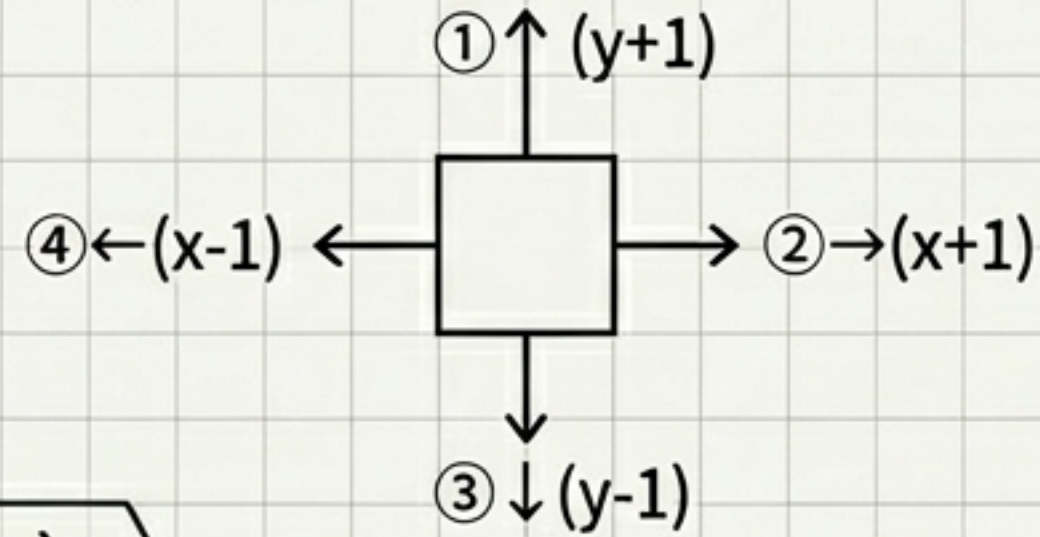
マスの「移動」軌跡の全容

Target 2

座標(4,2)の「訪問」タイミング

1つ目の解を発見した後、プログラムはどのようにして次の解を見つけるのか。本資料では解答の数値（回数）は提示しません。アルゴリズムが歩んだ正確な足跡を視覚化し、その挙動の本質を紐解きます。

プログラムの意思決定モデル



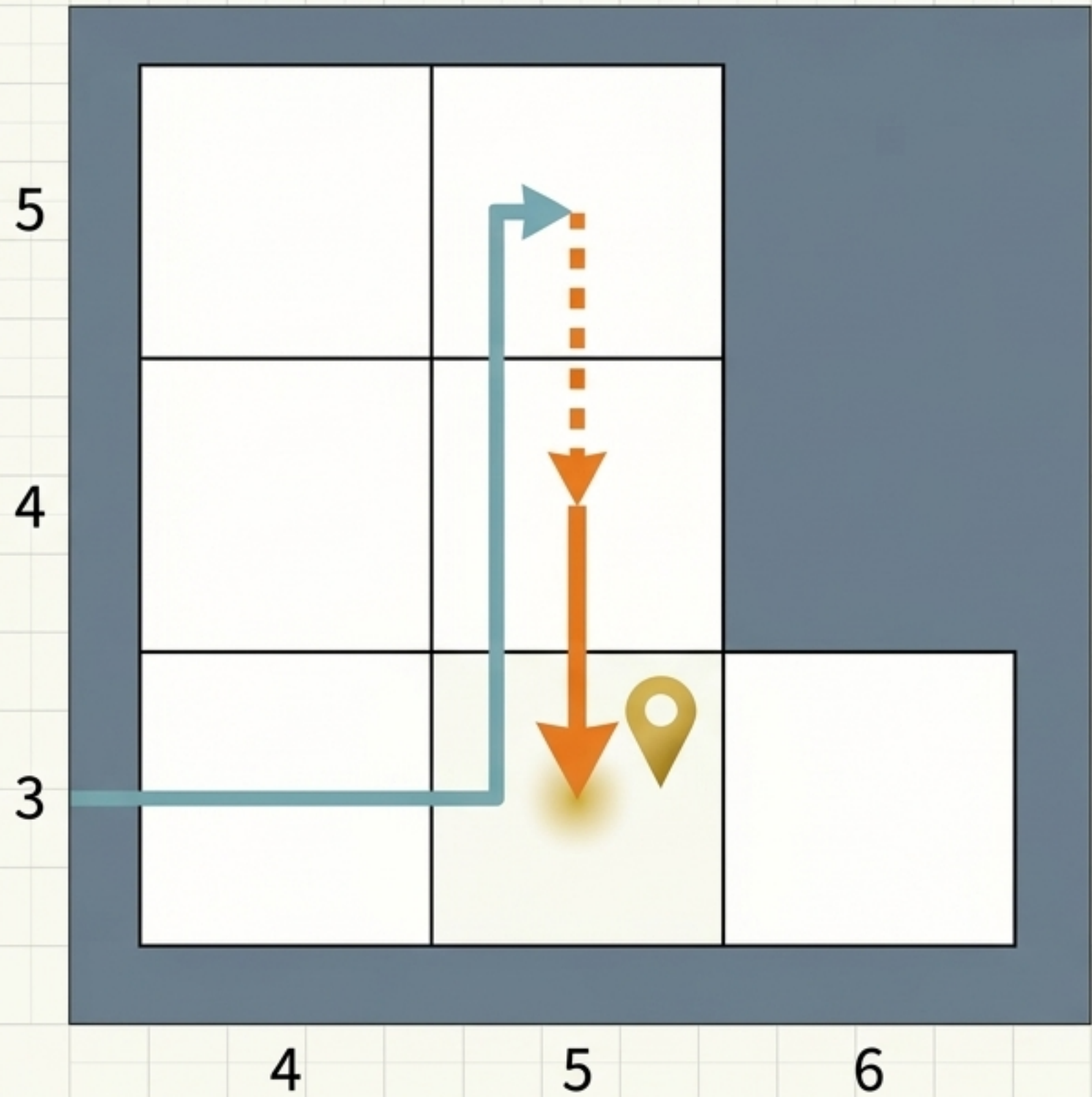
進行条件 (Movement Conditions)

1	【OKフラグ】 	進行可能（白マス）。進入と同時に「VISITED（足跡）」へ変化。
2	【NGフラグ】 	進行不可（外壁・障害物マス）。
3	【VISITED】 	進行不可（既にスタックにある足跡マス）。

【バックトラックの原則】

四方すべてに進めない時、プログラムは現在のマスを『OKフラグ』にリセットし、一つ前のマスへ「戻る」。

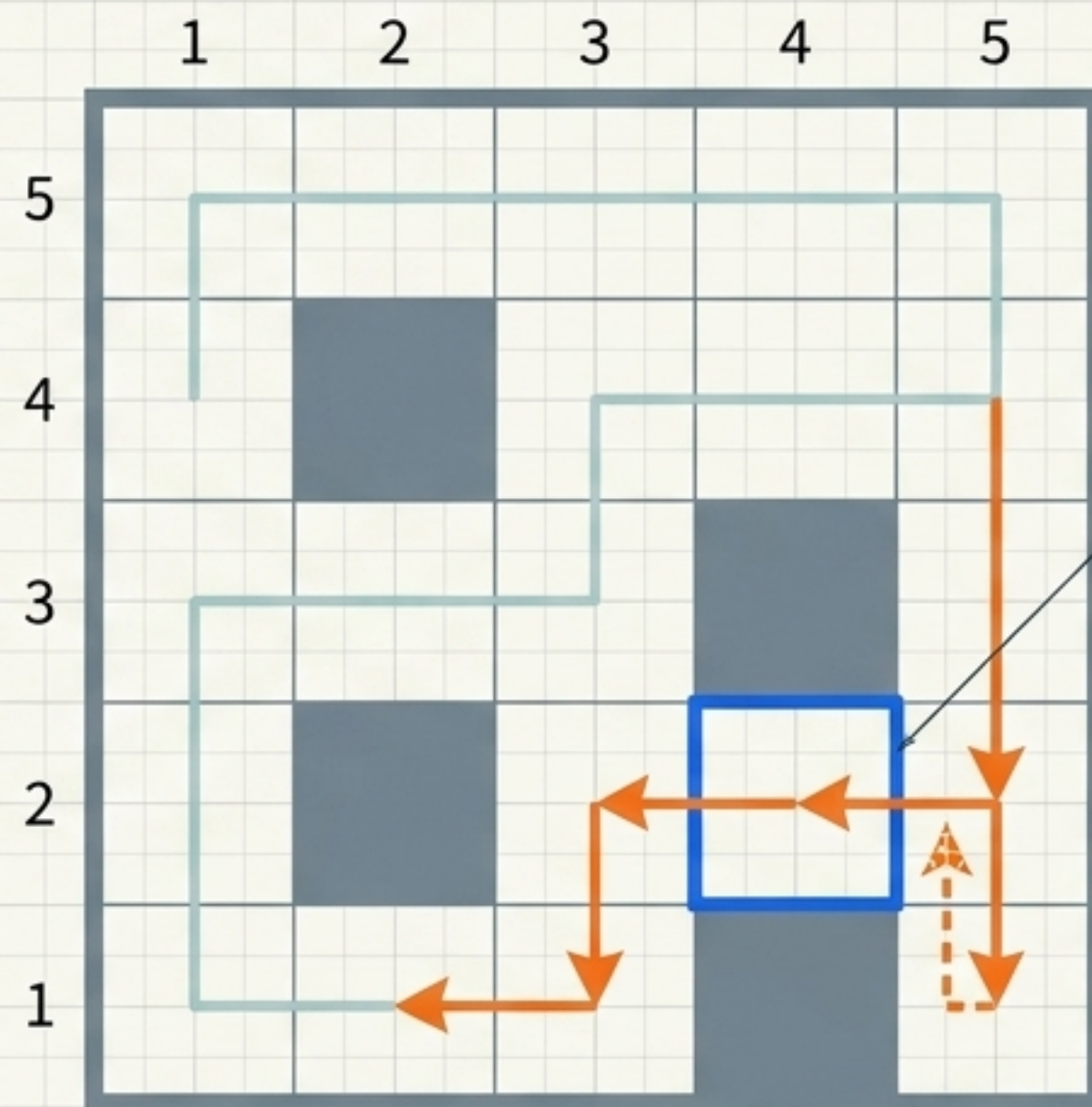
Phase 0: 探索の再開点「ベースポイント」



STATUS PANEL	
1	現在位置: 座標(5,3)
2	状態: 新たな探索関数 <code>visit(5,3)</code> が呼び出された瞬間。

1つ目の解（終点(5,5)）発見後、一つ前の(5,4)へ戻ります。
右(6,4)は外壁で進行不可のため、優先順位③に従い下方向の(5,3)へ進入。
ここが、未知なる第2の解探索の実質的な「起点」となります。

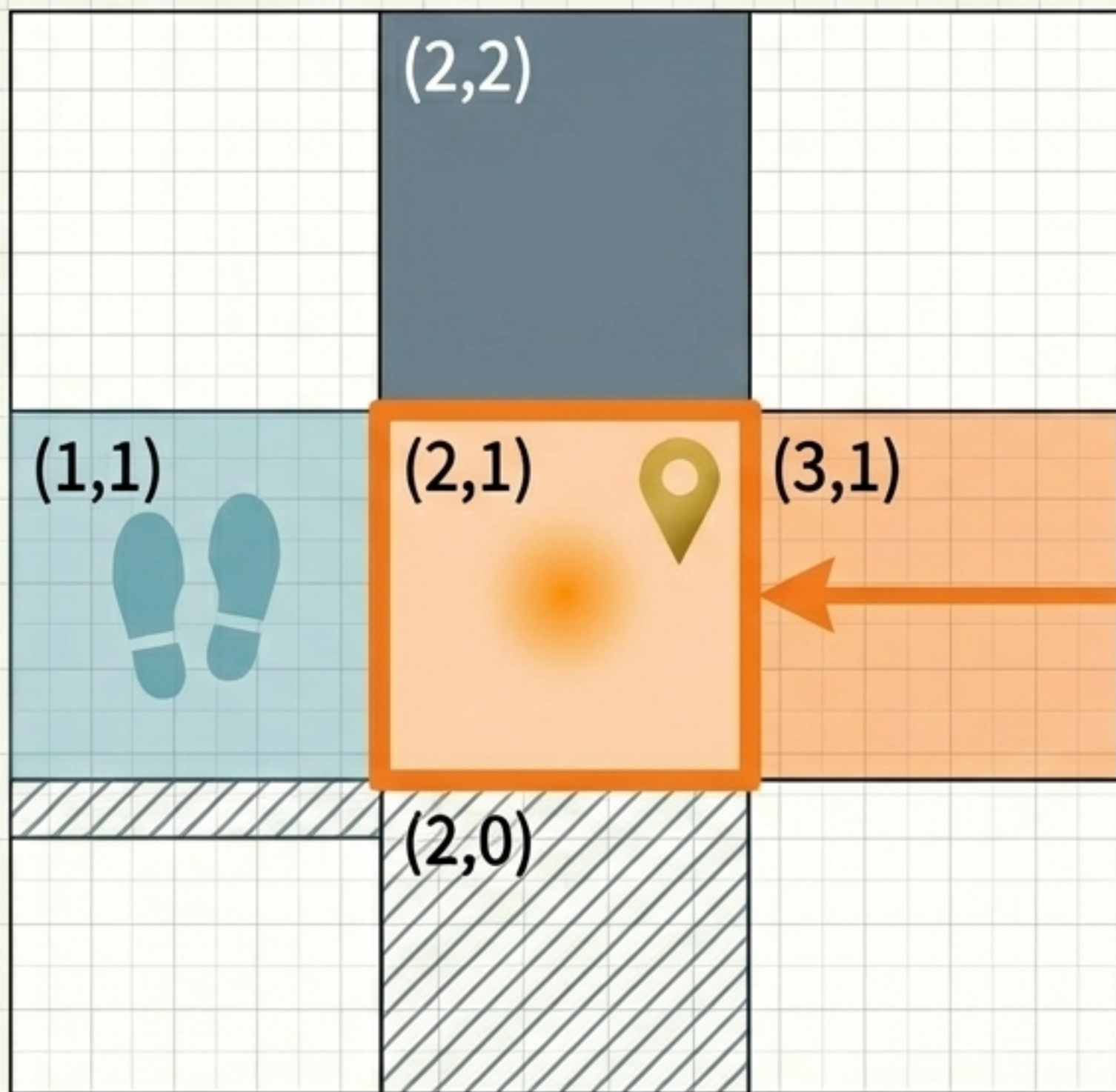
Phase 1: 左下エリアへの深掘り（前進）



Target 2: 訪問録 #1 (進む)

優先順位（上・右・下・左）に従い、未訪問のマスへ深く潜ります。
この過程で、ターゲットである座標(4,2)へ最初のアプローチが発生します。

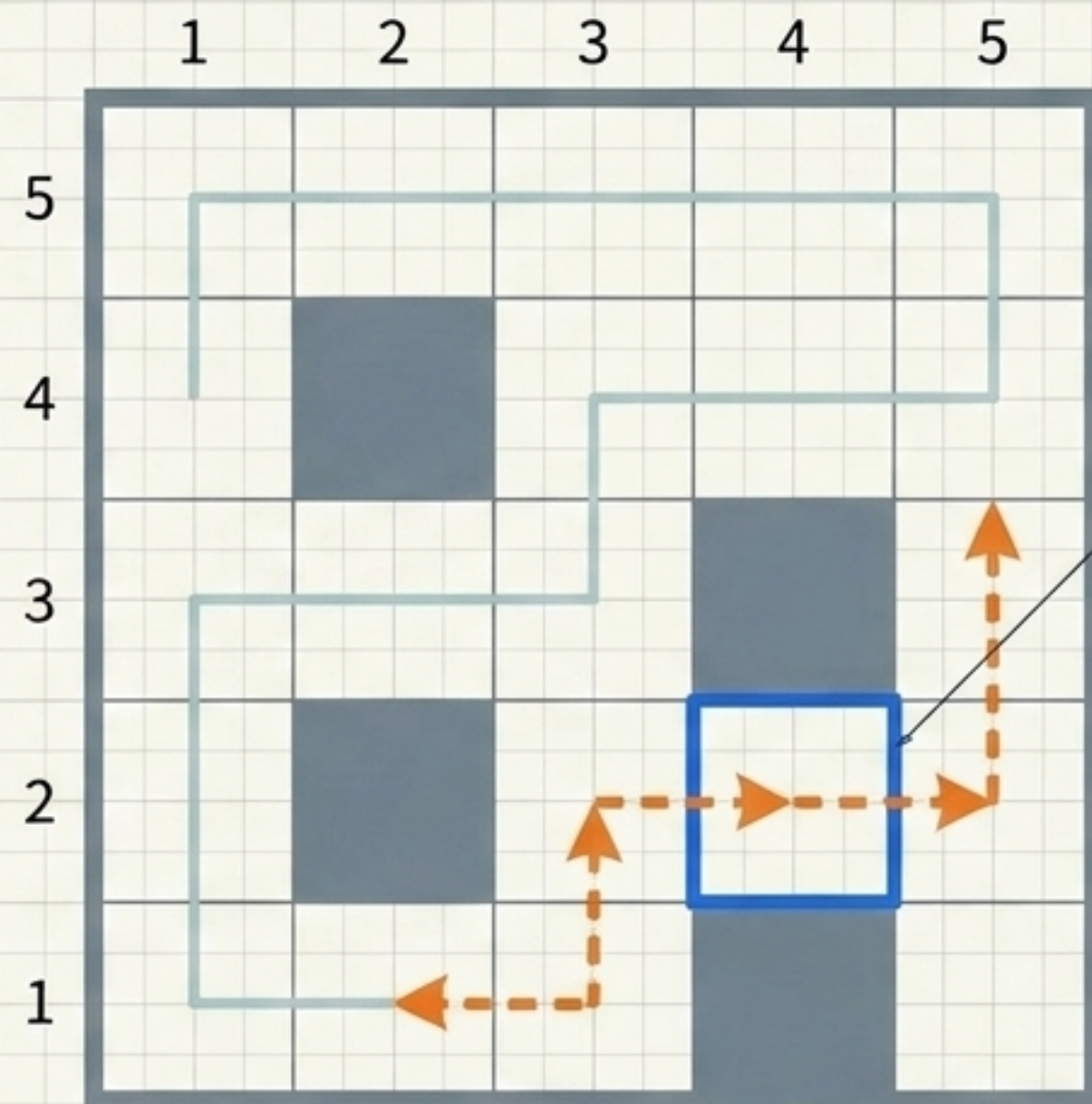
デッドロック：座標(2,1)での状況



4-Way Check Diagram		
①	上 (2,2): [NG] 障害物マス	×
②	右 (3,1): [VISITED] 今来た道	×
③	下 (2,0): [NG] 外壁	×
④	左 (1,1): [VISITED] 1つ目の解の 始点として足跡が残存	×

座標(2,1)に到達したプログラムは、四方すべてが進行不可であることを検知します。ここから、アルゴリズムの真骨頂である「バックトラック(後退)」が発動します。

Phase 1: 行き止まりからの撤退（戻る）

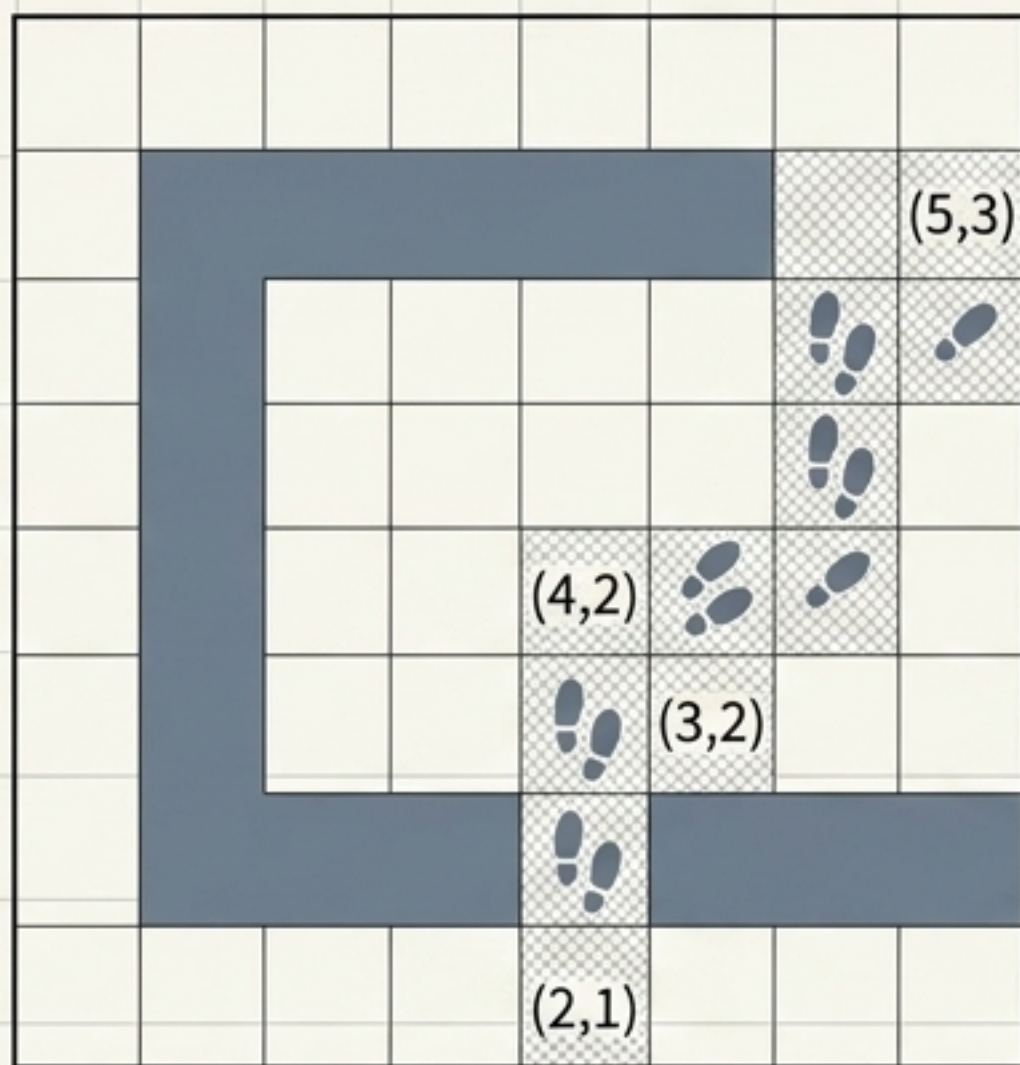


Target 2: 訪問録 #2 (戻る)

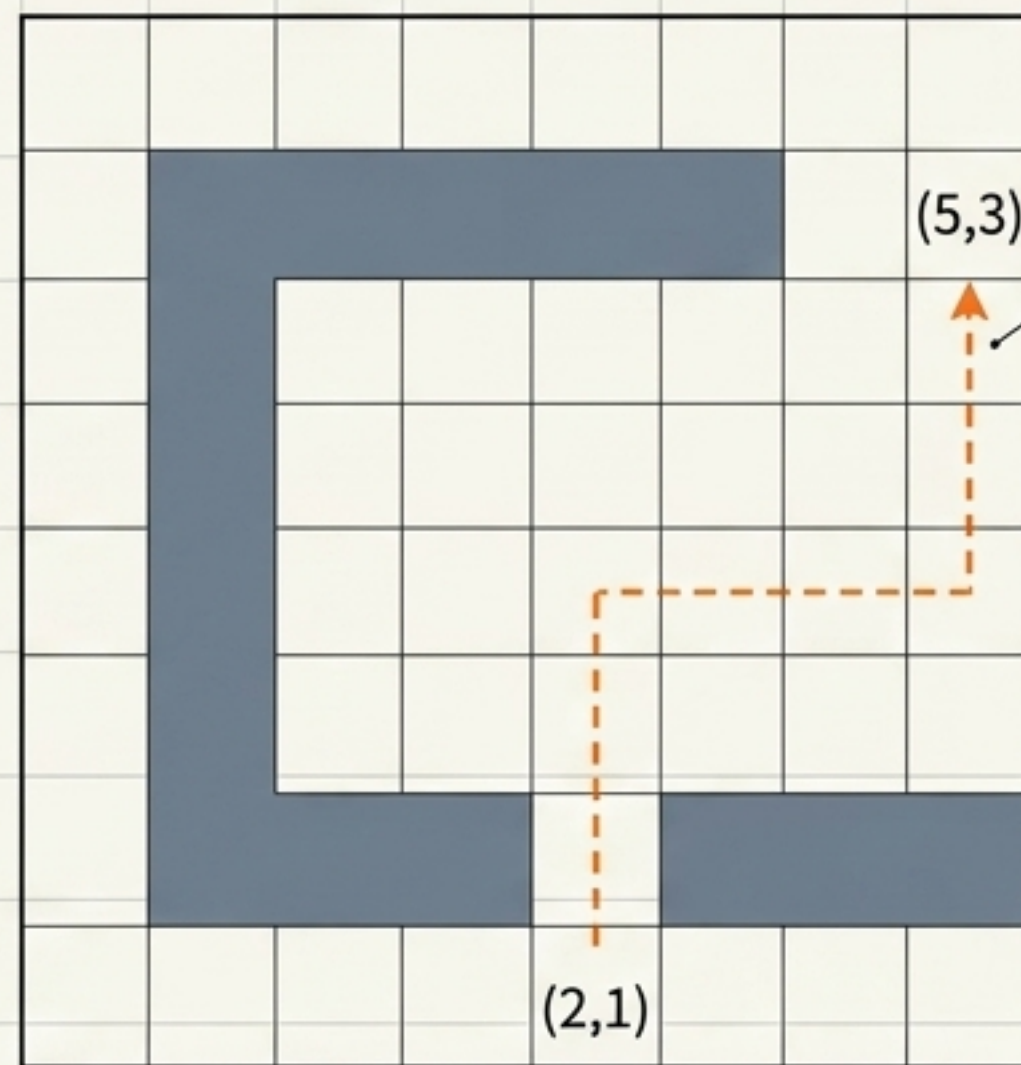
行き場を失ったプログラムは、来た道を忠実に引き返します。「戻る」アクションも1回の移動としてカウントされます。この撤退経路により、座標(4,2)を再び通過します。

バックトラックの副産物：記憶の消去

Before (深層到達時)



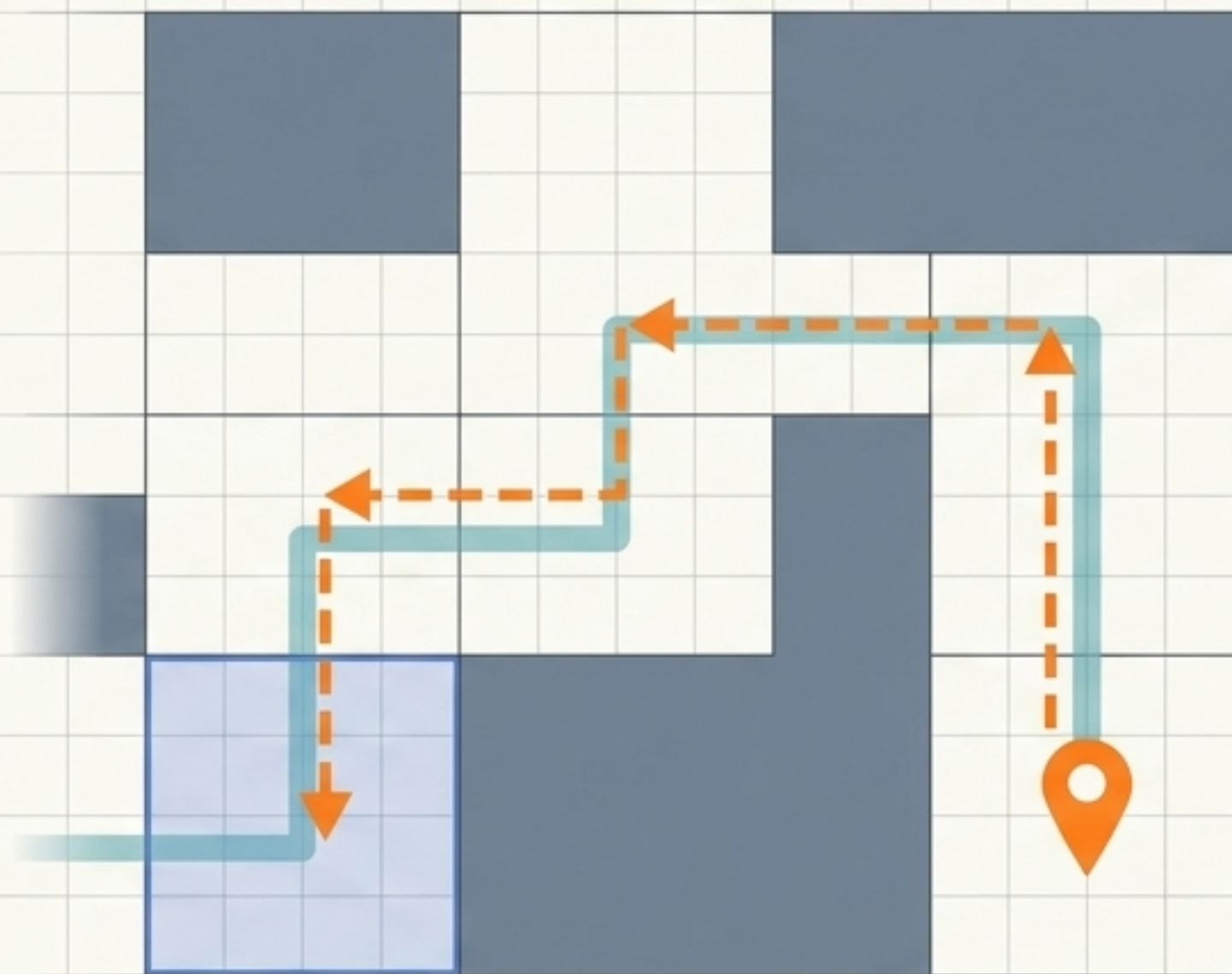
After (撤退完了時)



Target 2:
訪問録 #2
(戻る)

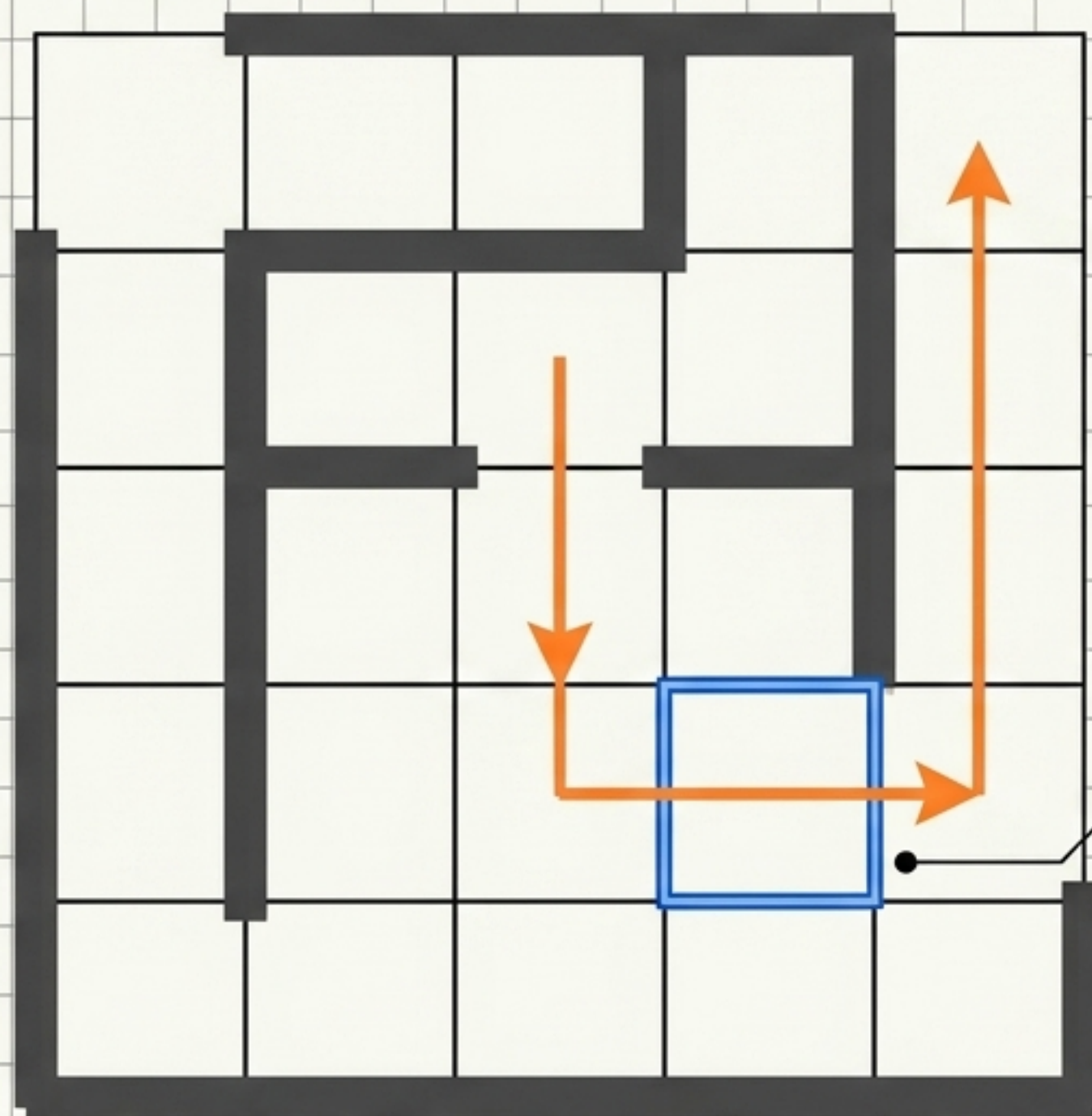
「戻る」動作の極めて重要な役割は、現在のマス情報を『OKフラグ』に戻すことです。一度行き止まりと判定されたルートの周辺マスがリセットされ、後の探索で再利用可能になります。

Phase 2: オリジナル経路の巻き戻し



起点(5,3)でも全方向の探索を終えたため、プログラムは1つ目の解で通ったルート（青線）をさらに遡ります。この深いスタックの後退により、(3,3)周辺のマスの子跡もOKフラグにリセットされていきます。

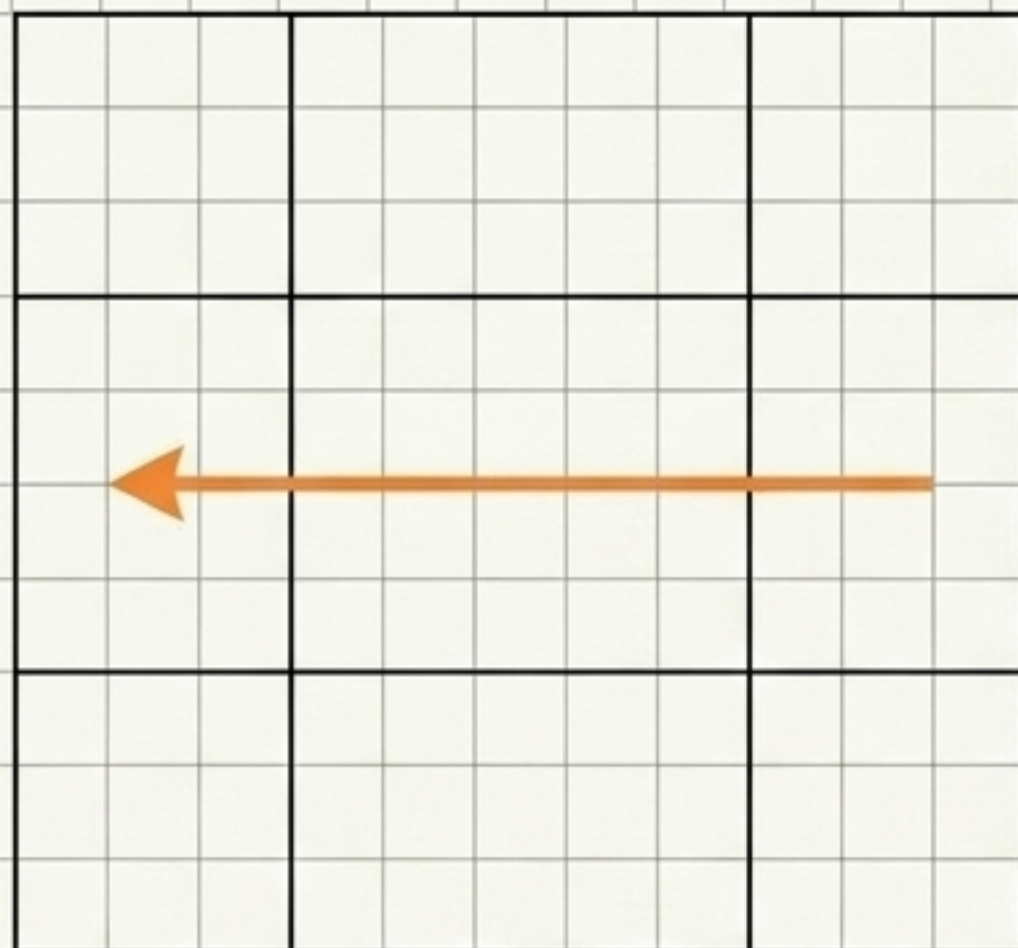
Phase 3: リセットされた盤面からの再出発



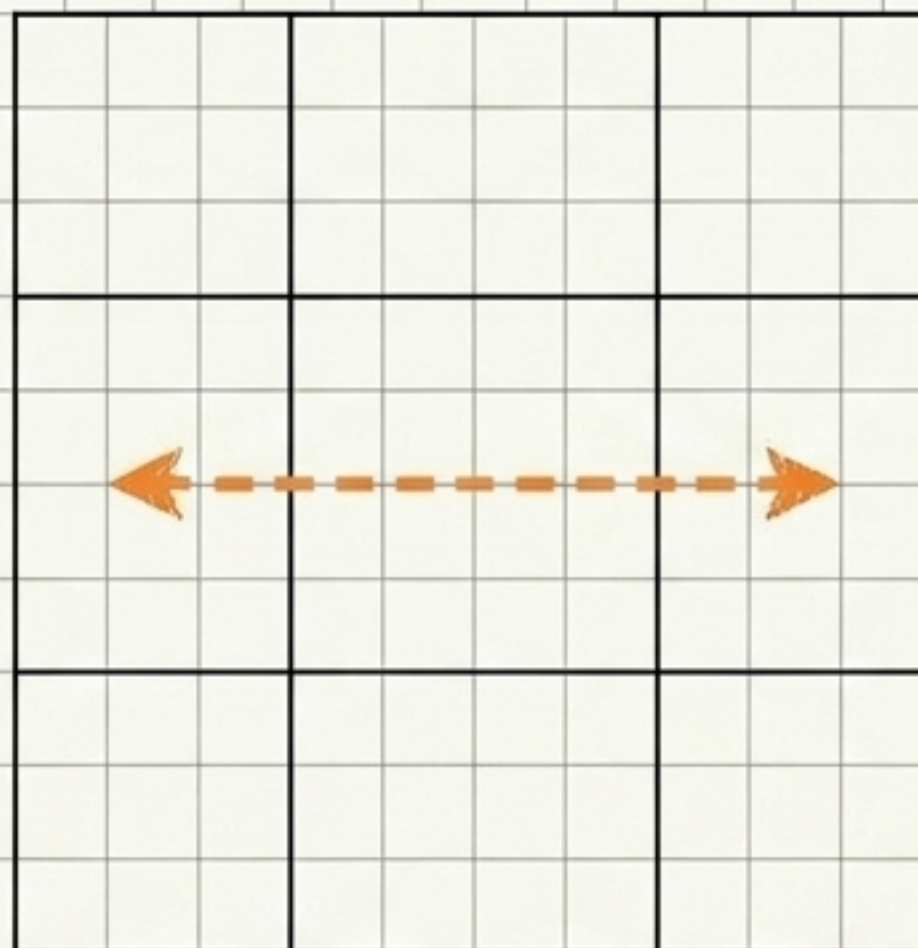
Target 2:
訪問録 #3 (進む)

以前は行き止まりだった(3,2)ですが、Phase 1でのフラグリセットにより進入可能となりました。ここからは迷うことなく、一直線に第2の解（終点）へと到達します。

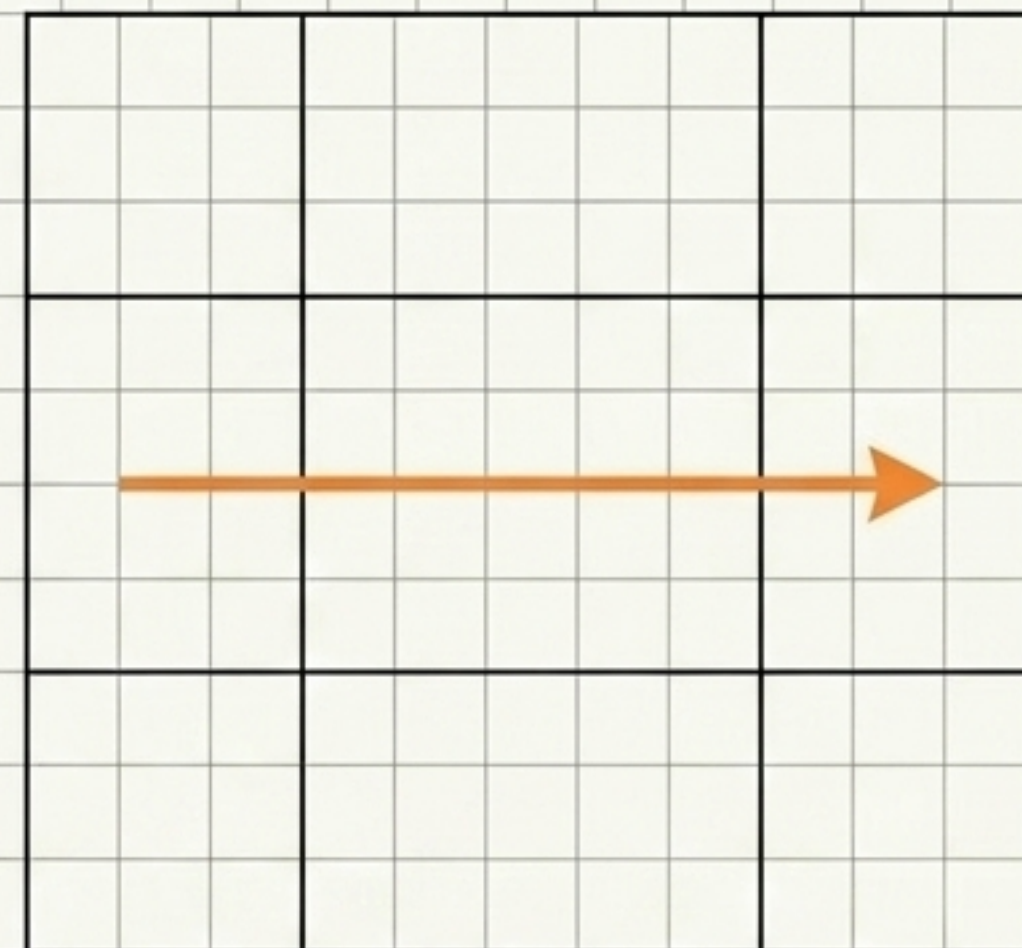
Target 2: 座標(4,2)のステータス履歴



1回目の訪問（探索）



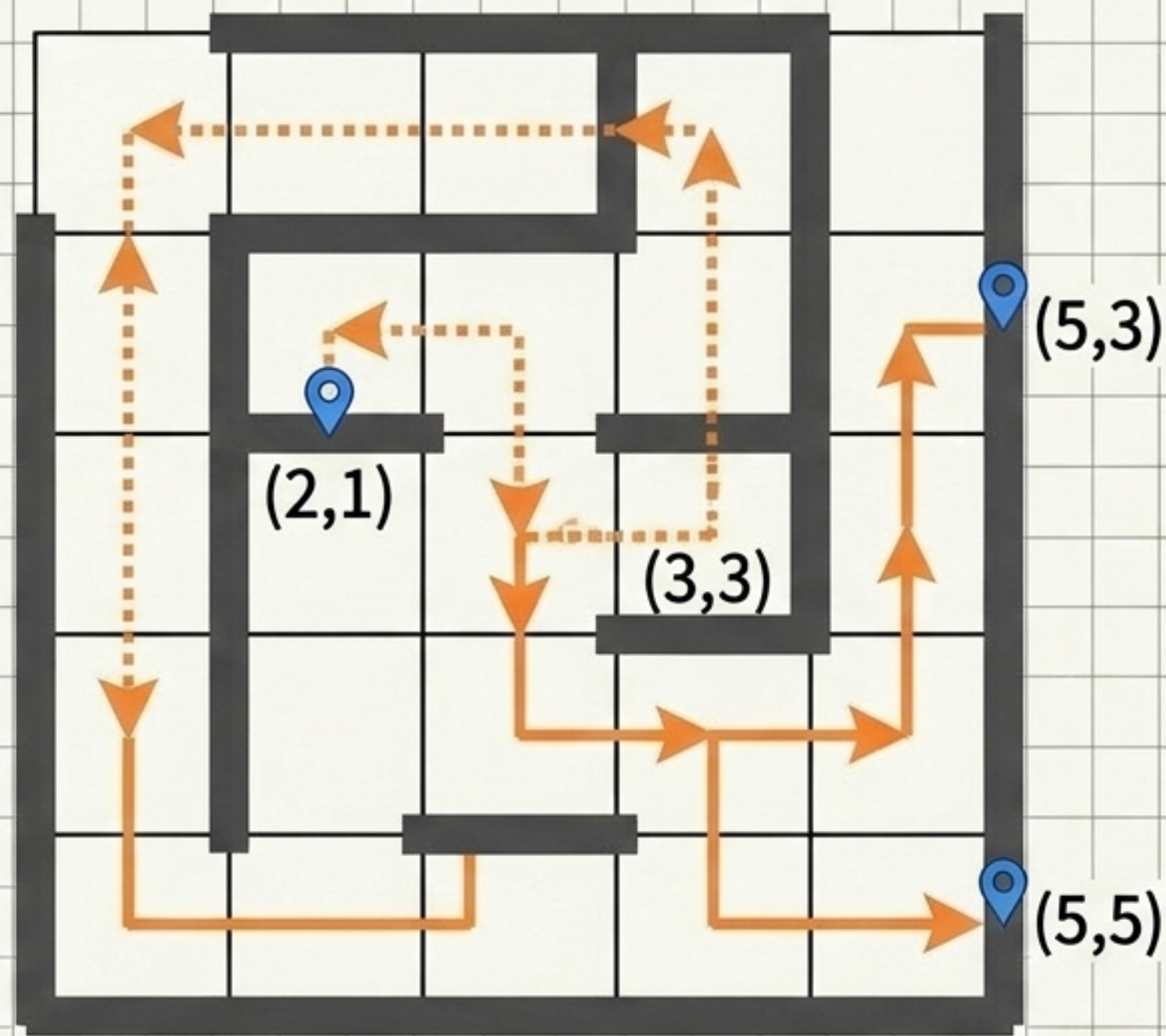
2回目の訪問（後退）



3回目の訪問（新ルート）

特定のマスに注目すると、探索アルゴリズムが同じ座標を「行き止まりへの進行」「撤退」「新ルートの開拓」という全く異なる文脈で複数回利用していることが分かります。

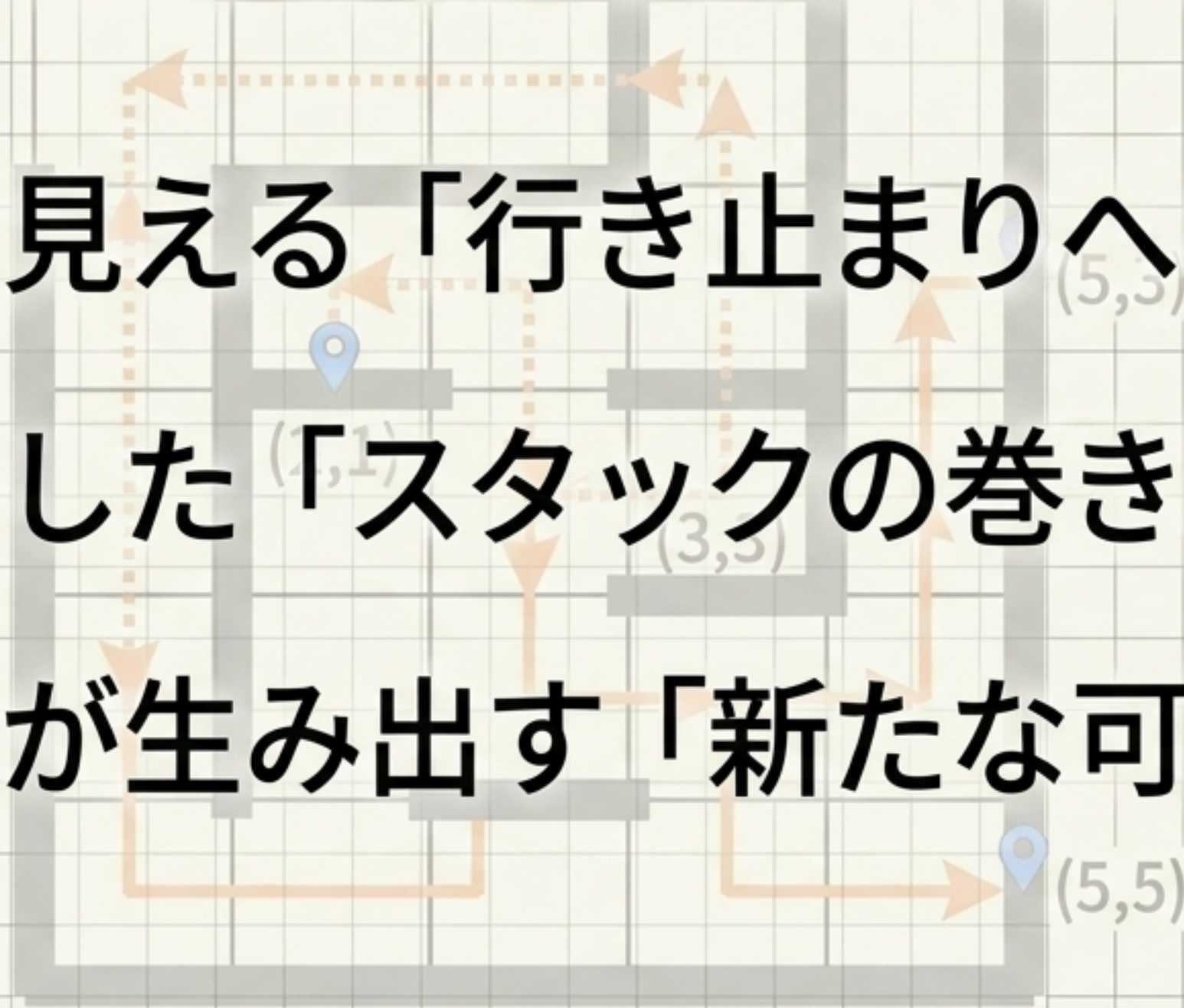
Target 1: アルゴリズムの全足跡



起点(5,3)から2つ目の解を発見するまでに発生したすべての「進む」と「戻る」の軌跡です。画面上に描かれた矢印の総数が、設問が求める総移動回数となります。

Conclusion: 深さ優先探索の美しさ

- 一見無駄に見える「行き止まりへの迷走」
- 地道で徹底した「スタックの巻き戻し」
- 記憶の消去が生み出す「新たな可能性」



人間の目には遠回りに見える軌跡も、コンピュータにとっては「漏れなく全ての解を見つけ出す」ための最も確実なプロセスです。バックトラックの愚直な反復こそが、アルゴリズムの美しさの源泉です。