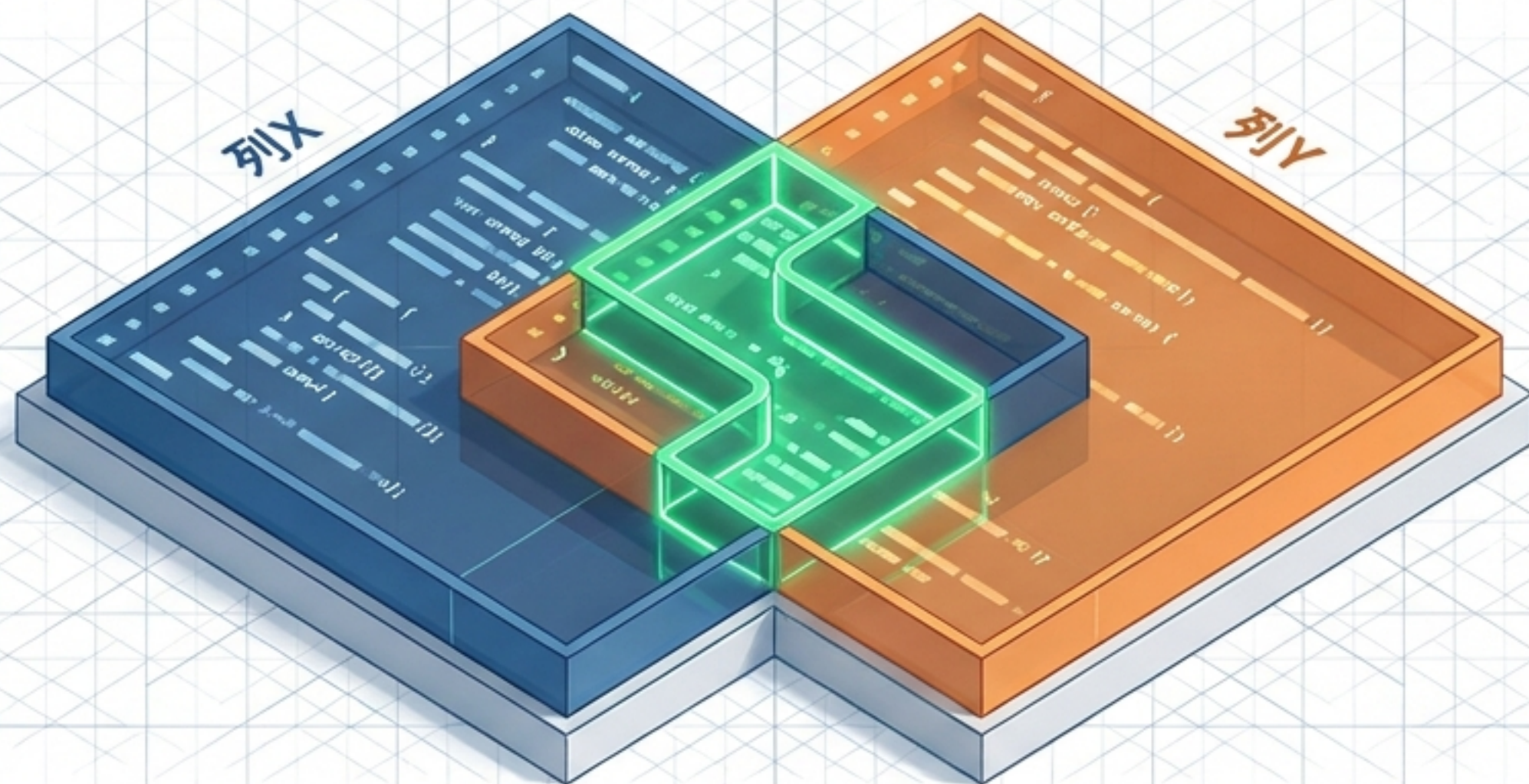


最長共通部分列（LCS）攻略の設計図

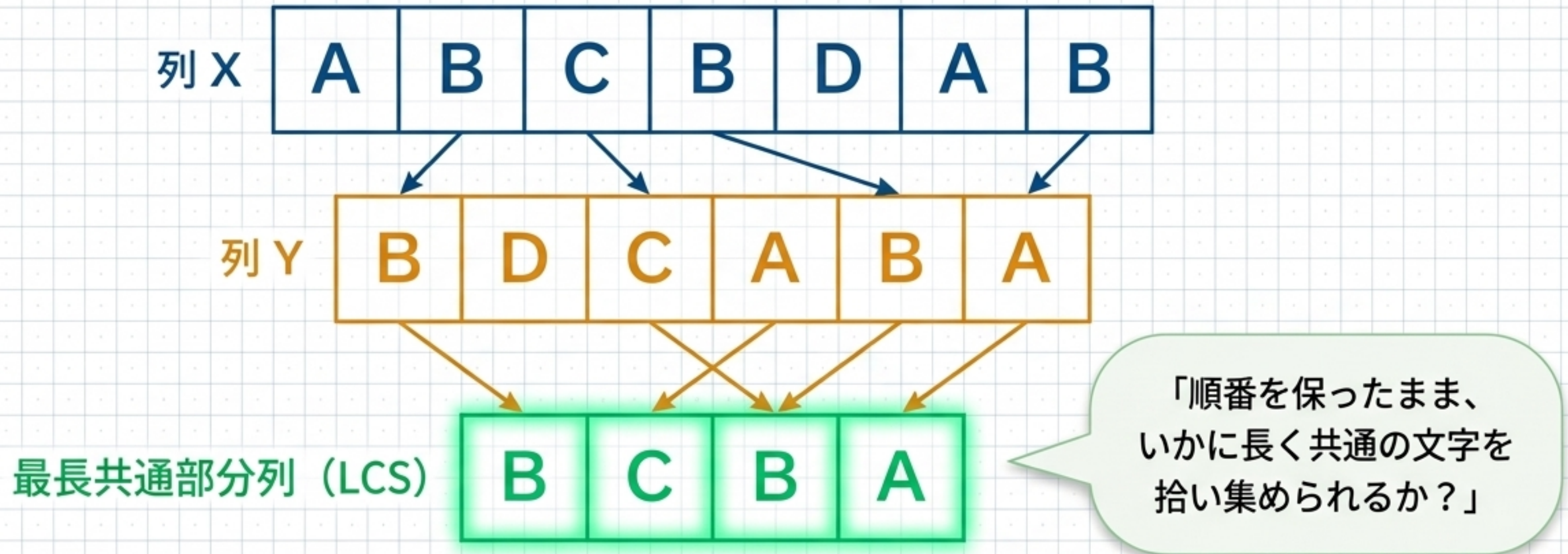
応用情報技術者試験のアルゴリズム問題を「視覚」で解き明かす



台本：

「アルゴリズムの問題、特に最長共通部分列（LCS）と動的計画法と聞くと、身構えてしまう方も多いかもかもしれません。でも大丈夫。このスライドでは直接的な答えは教えません。代わりに、問題をパズルのように自力で解くための『視点』をお渡しします。一緒に設計図を読み解きましょう。」

目的：2つの文字列から「共通の順番」を抽出せよ

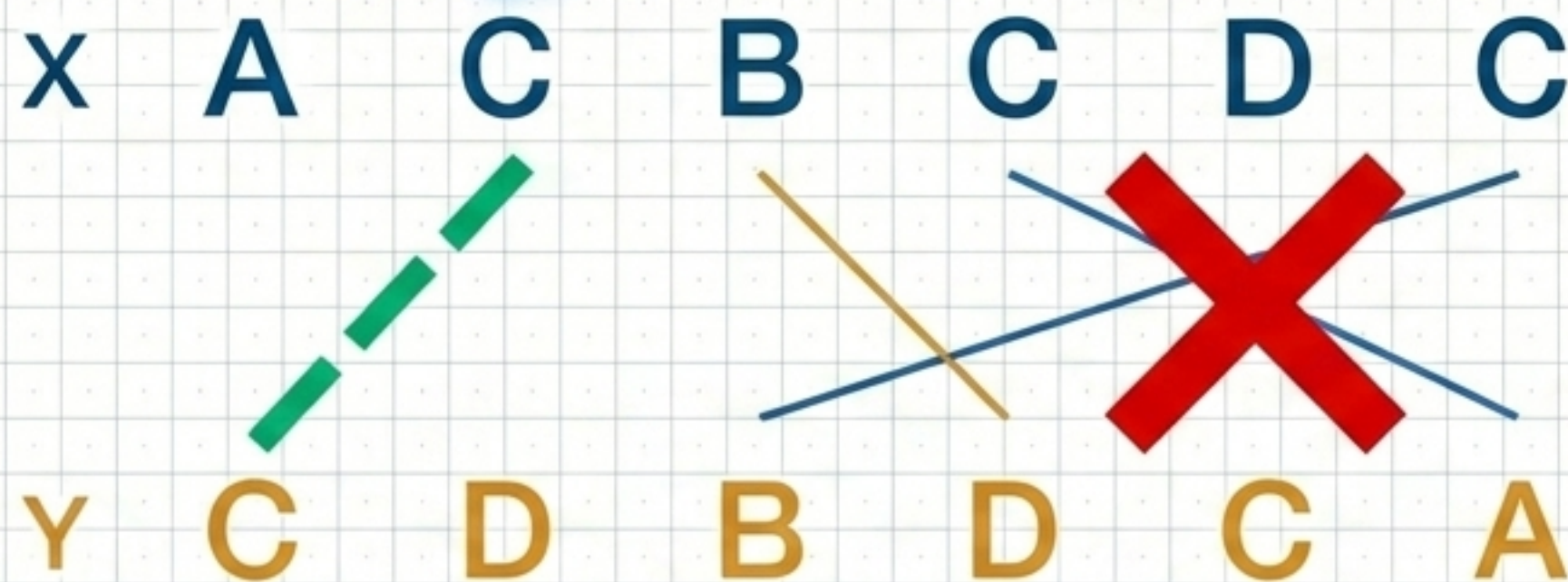


台本

台本：「LCSとは何か。難しく聞こえますが、要するに『2つの文字の並びから、共通している文字を、順番を崩さずにどれだけ長く拾い出せるか』というゲームです。DNAの解析なんかにも使われる考え方なんですよ。」

第一の関門：まずは「目」と「線」で探してみよう（設問1へのアプローチ）

「要素が少ないうちは、自分でアンダーラインを引いてみれば答えが見えてきます。
交差しないように、一番多く線を引けるルートはどれでしょう？」



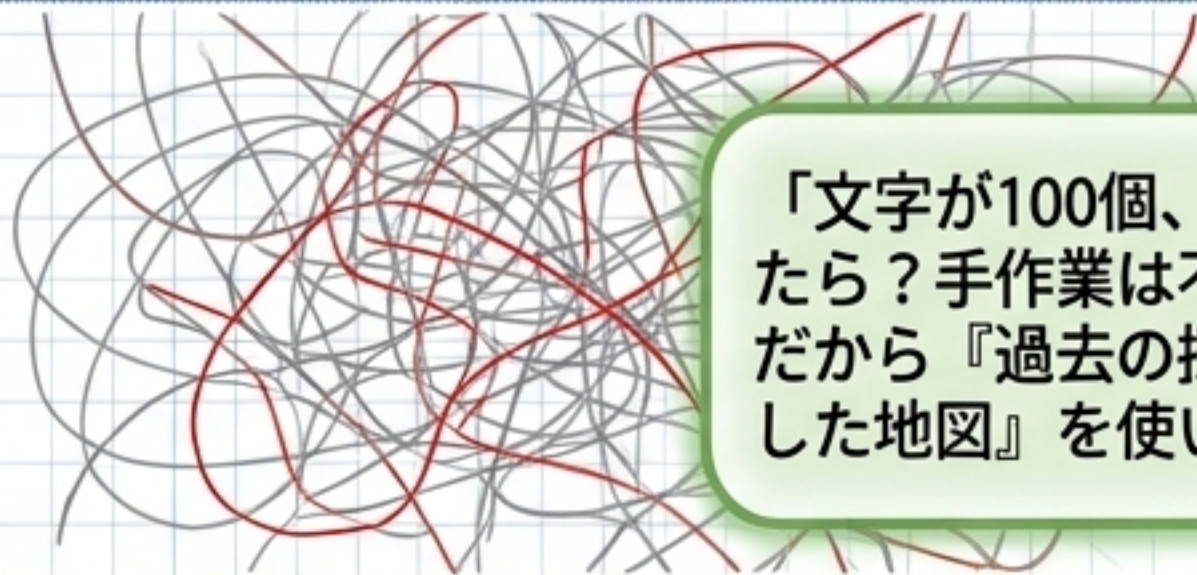
台本

台本：「早速設問1のヒントです。動画解説にもあった通り、文字数が少ないうちは、手を動かして直感的に線を引いてみるのが一番確実です。ルールは一つ、線が交差しないこと（順番を守ること）。一番多く線を引けるルートを探してみてください。」

なぜ「動的計画法の表」が必要なのか？

直感の限界

X



「文字が100個、1000個になったら？手作業は不可能です。
だから『過去の探索結果をメモした地図』を使います。」

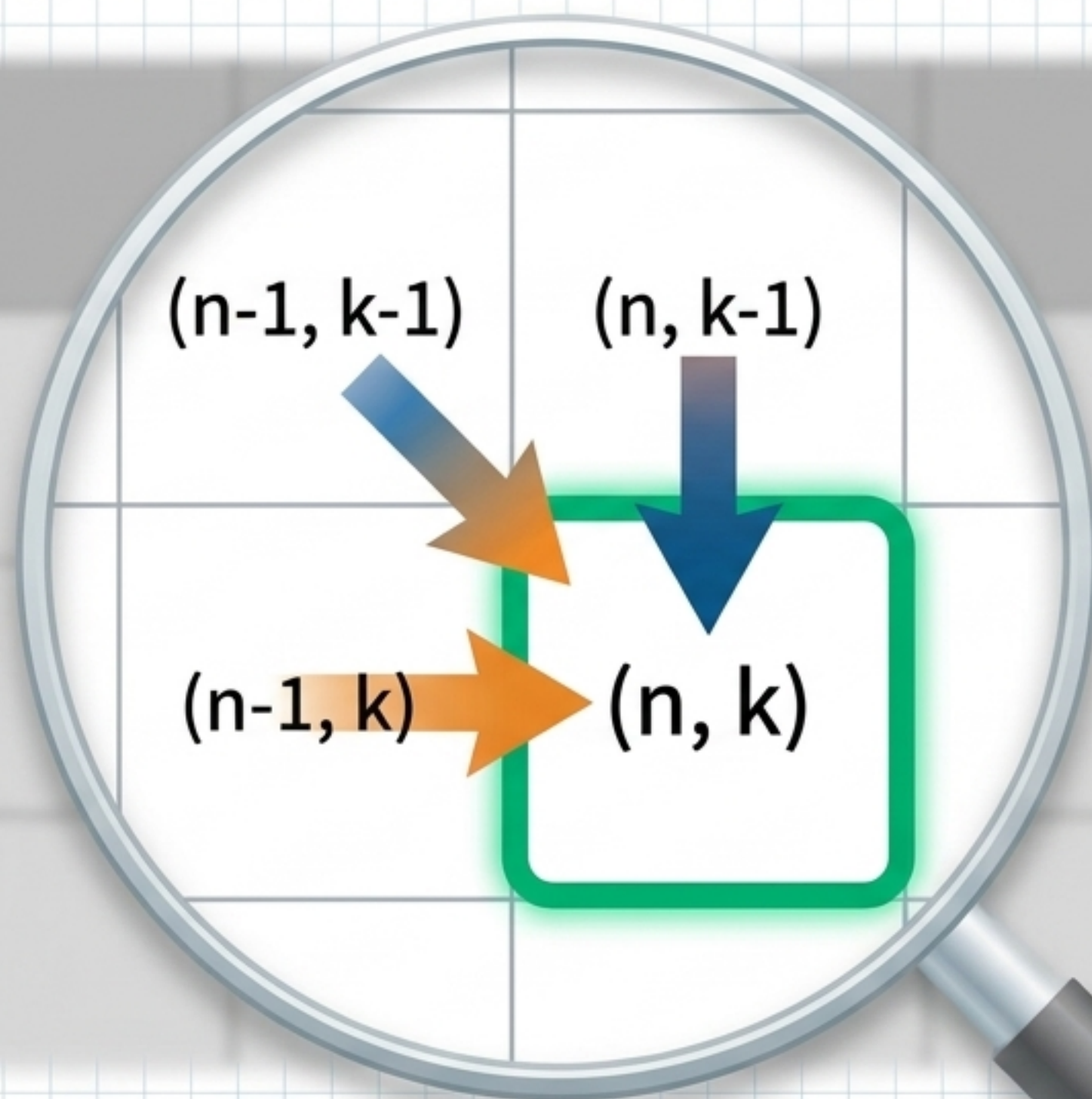
Y

記録と計算

		k	0	1	2	3	4	5	6
n	x_n y_k	—	B	D	C	A	B	A	
0	—	0	0	0	0	0	0	0	0
1	A	0	0	0					
2	B	0	1 (①)	1 (②)					
3	C	0	1	1		ア			
4	B	0	1	1					
5	D	0	1	2					
6	A	0	1	2 (③)					
7	B	0	1					イ	

台本：「設問1は解けましたか？しかし、この文字が100文字、1000文字になったら手作業では無理ですよ。そこで、計算のやり直しを防ぐために結果を記録しながら進める『動的計画法（DP）』の出番です。この無機質な表、実はとってもシンプルなルールの集まりなんです。」

魔法のルール：常に「周囲3マス」だけを見る



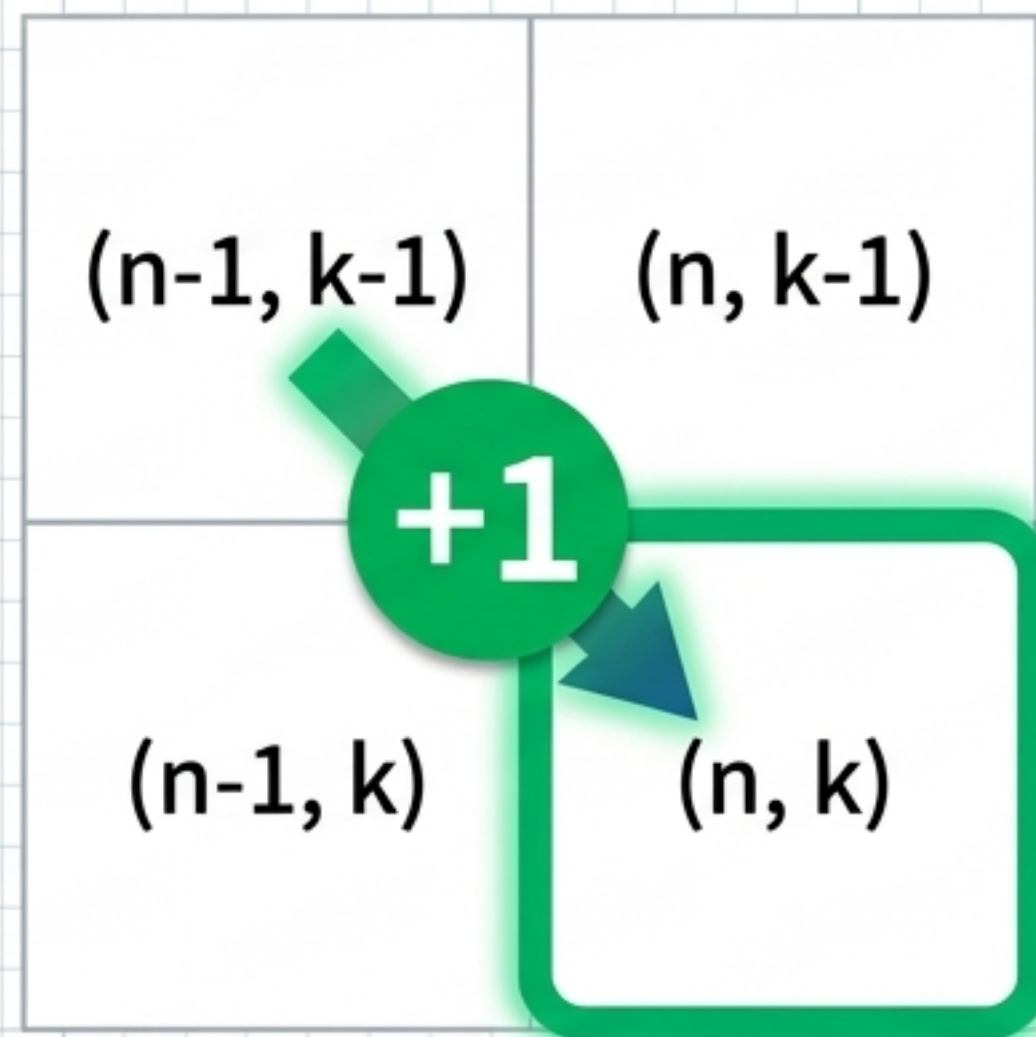
「表全体を見て圧倒される
必要はありません。
どのマスを埋める時も、見
るべき方向はこの3つだけ！」

台本：「表を埋めろと言われると圧倒されますが、安心してください。動的計画法の表を埋める時、あなたが注目すべきは常に『今いるマスの周囲3マス』だけです。左上、上、左。これだけで全てが決まります。」

パターン A：文字が「一致」した時の動き (Match)

$$x_n = y_k$$

「共通文字を1つゲット！
過去の最大値（左上）に1を足そう。」



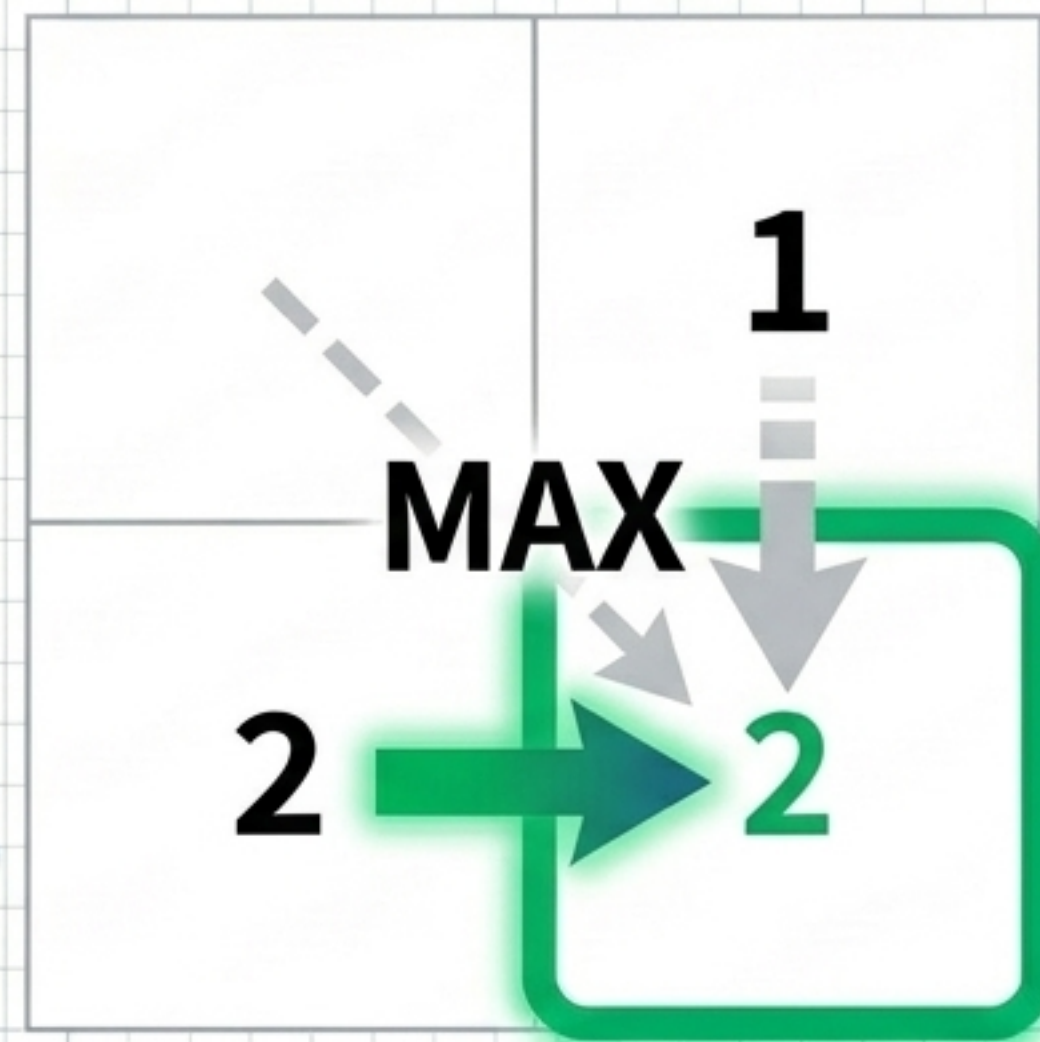
$$LCSL(n, k) = LCSL(n - 1, k - 1) + 1$$

台本：「ルールはたった2つ。1つ目は、列Xと列Yの文字が一致した時。これは共通文字をゲットしたということです。だから、一つ前の状態である『左上のマス』の数字に『+1』して、今のマスに書き込みます。」

パターン B：文字が「不一致」の時の動き (Mismatch)

$x_n \neq y_k$

「ゲットならず。これまでの一番良い成績
(上か左の大きい方) をそのまま引き継ぐ。」



$$\text{MAX}(\text{LCSL}(n, k - 1), \text{LCSL}(n - 1, k))$$

台本：「2つ目は、文字が一致しなかった時。ここでは新しい文字はゲットできません。
なので、これまでの成績のうち、一番良かった数字をそのまま持ってきます。
『上のマス』と『左のマス』を見比べて、大きい方の数字を書き写すだけです。」

第二の関門：ルールに従って表を埋めてみよう（設問2へのアプローチ）

「CとA...一致しない！
じゃあどのマスとどのマスを比べる？」

n \ k		0	A
2		?	?
C		?	≠ [ア]
4			

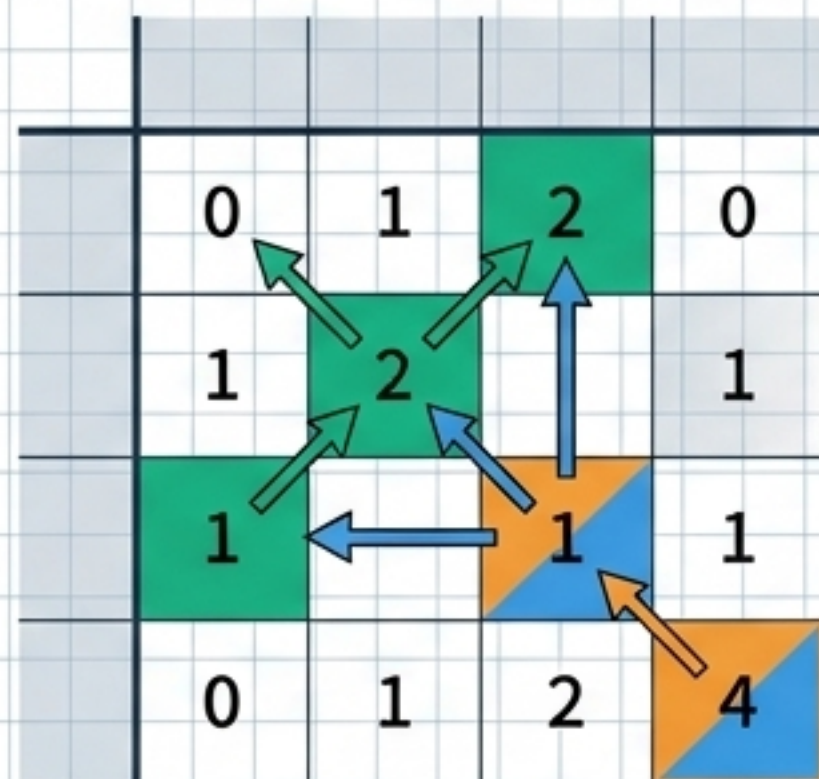
「BとB...一致した！
ナナメ左上に+1だ！」

n \ k		4	B
7		?	
6		?	
B			[イ]

台本：「では、設問2に挑戦です。図3の表の穴埋めですね。先ほどの2つのルールを思い出してください。アの場所は文字が一致していません。イの場所は文字が一致しています。見るべきマスはどこか、もうお分かりですね？」

アルゴリズムをコードに「翻訳」する

表の動き



翻訳



プログラム

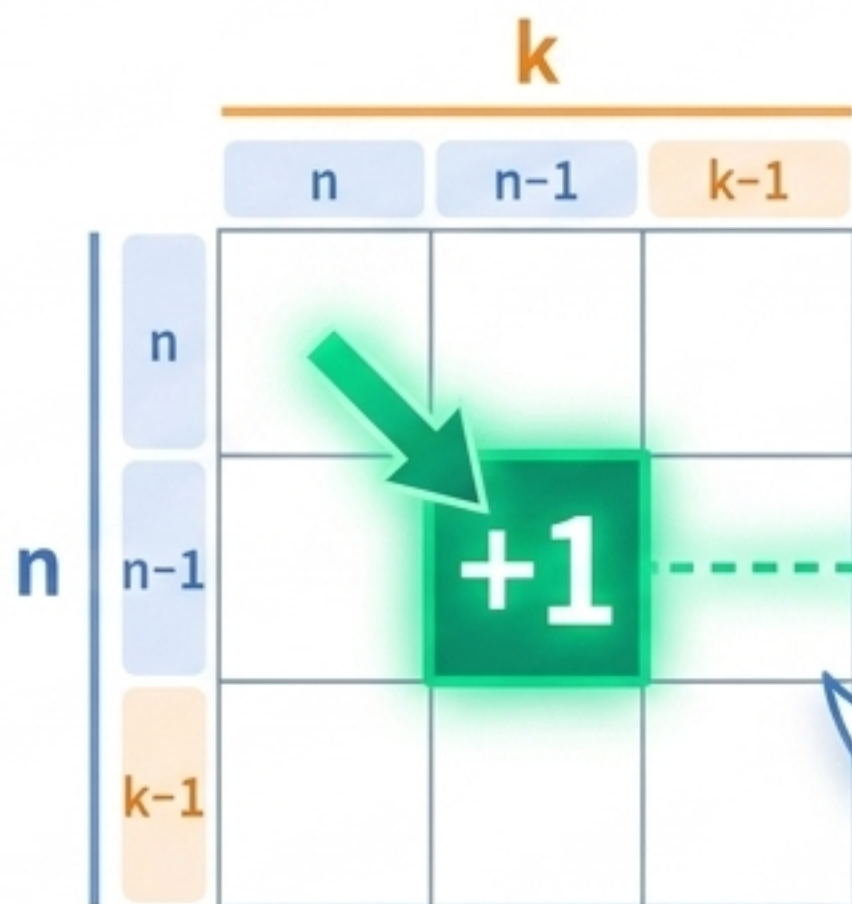
```
dof calculate_lcs1(tamox) {  
  lnt = ウ  
  int = y  
  for (i= 0; i < name; i++) {  
    if (icsl <= )  
      lcst = ウ  
    else:  
      answ = エ  
  } else:  
    return オ  
}  
return カ
```

プログラミングは魔法ではありません。
先ほど確認した『表の動きのルール』を、コンピ
ュータに指示する言葉に翻訳するだけです。

台本：「表が埋められたら、次は設問3です。急にプログラムのコードが出てきて焦るかもしれませんが、やることは同じです。
あなたが表を埋める時に頭の中でやっていたことを、コンピュータ用の言葉に翻訳してあげるだけなのです。」

第三の関門：翻訳 その1 「一致した時」

表の動き（視覚的記憶）



コード翻訳

```
if (S[n - 1] が T[k - 1] と等しい)  
    lcs[n, k] ← [ウ]
```

「配列のインデックスで『左上』はどう表現する？
そして忘れてはいけない『+1』！」

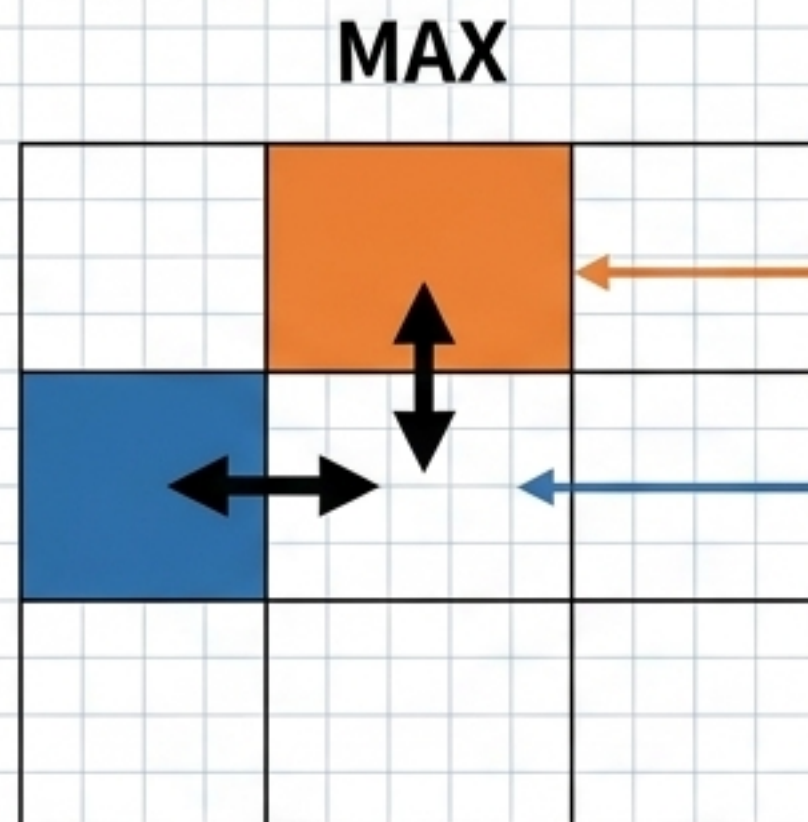
台本

台本：「設問3の[ウ]です。ここはif文の条件を見ると、文字が一致した時の処理ですね。表でいうと『斜め左上のマスに+1』するルールでした。では、配列lcsにおいて『斜め左上』はどう書くのか？そして何を足すのか？考えてみましょう。」

第三の関門：翻訳 その2 「不一致の時」

表の動き（視覚的記憶）

コード翻訳



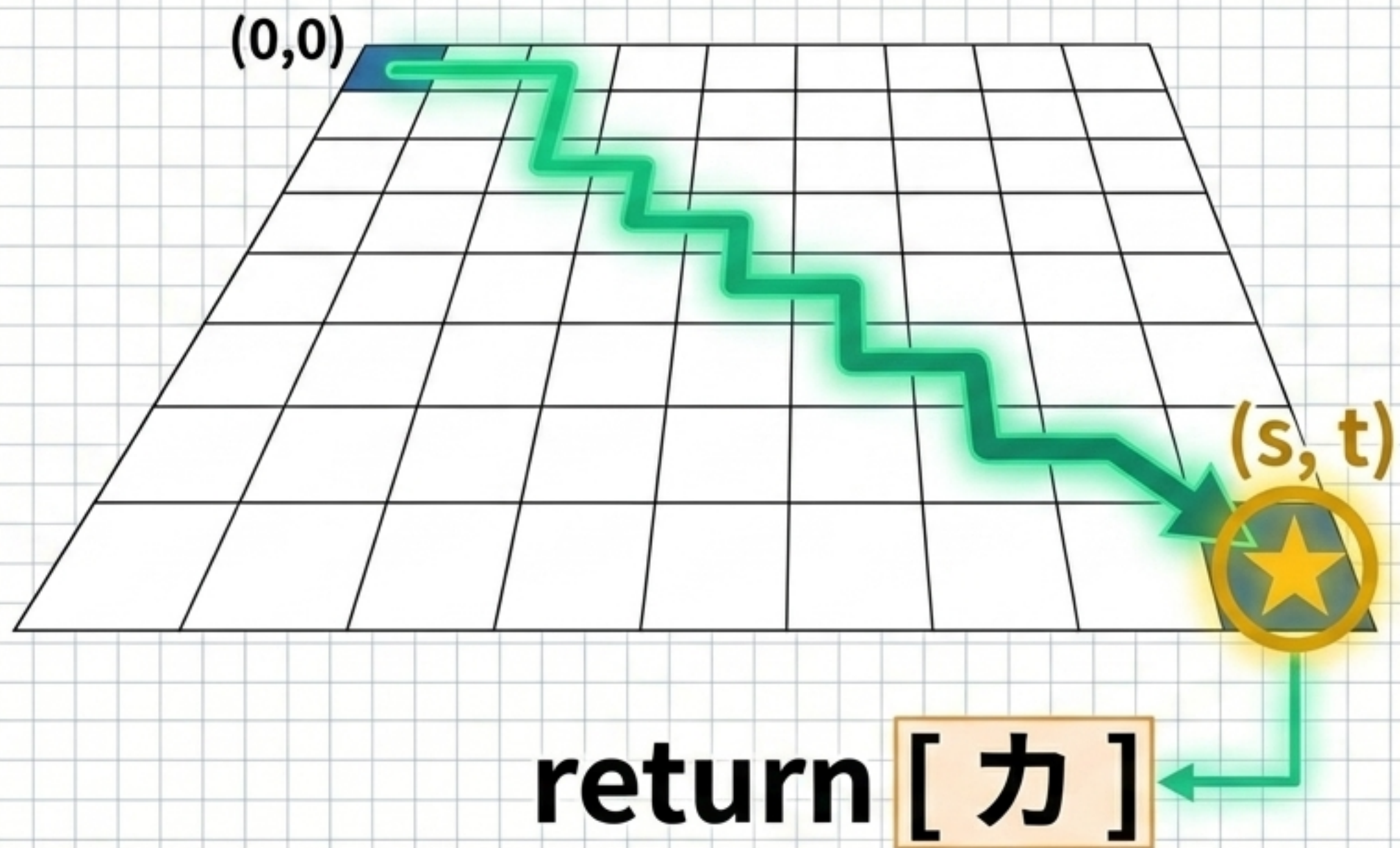
```
elseif (lcs[n, k - 1] が lcs[n - 1, k] より大きい)  
    lcs[n, k] ← [エ]  
else  
    lcs[n, k] ← [オ]
```

「『大きい方を選ぶ』という判断がif文になっている。条件が真（上が大きい）の時は、どちらを引き継ぐ？」

台本 (Script)

台本：「続いて[エ]と[オ]。文字が不一致の時です。ルールは『上か左の大きい方を引き継ぐ』でした。コードを見ると、elseifで『上が左より大きいか』を判定していますね。ならば、[エ]には何が入り、そうでない場合の[オ]には何が入るのでしょうか。」

第三の関門：翻訳 その3 「ゴール地点」



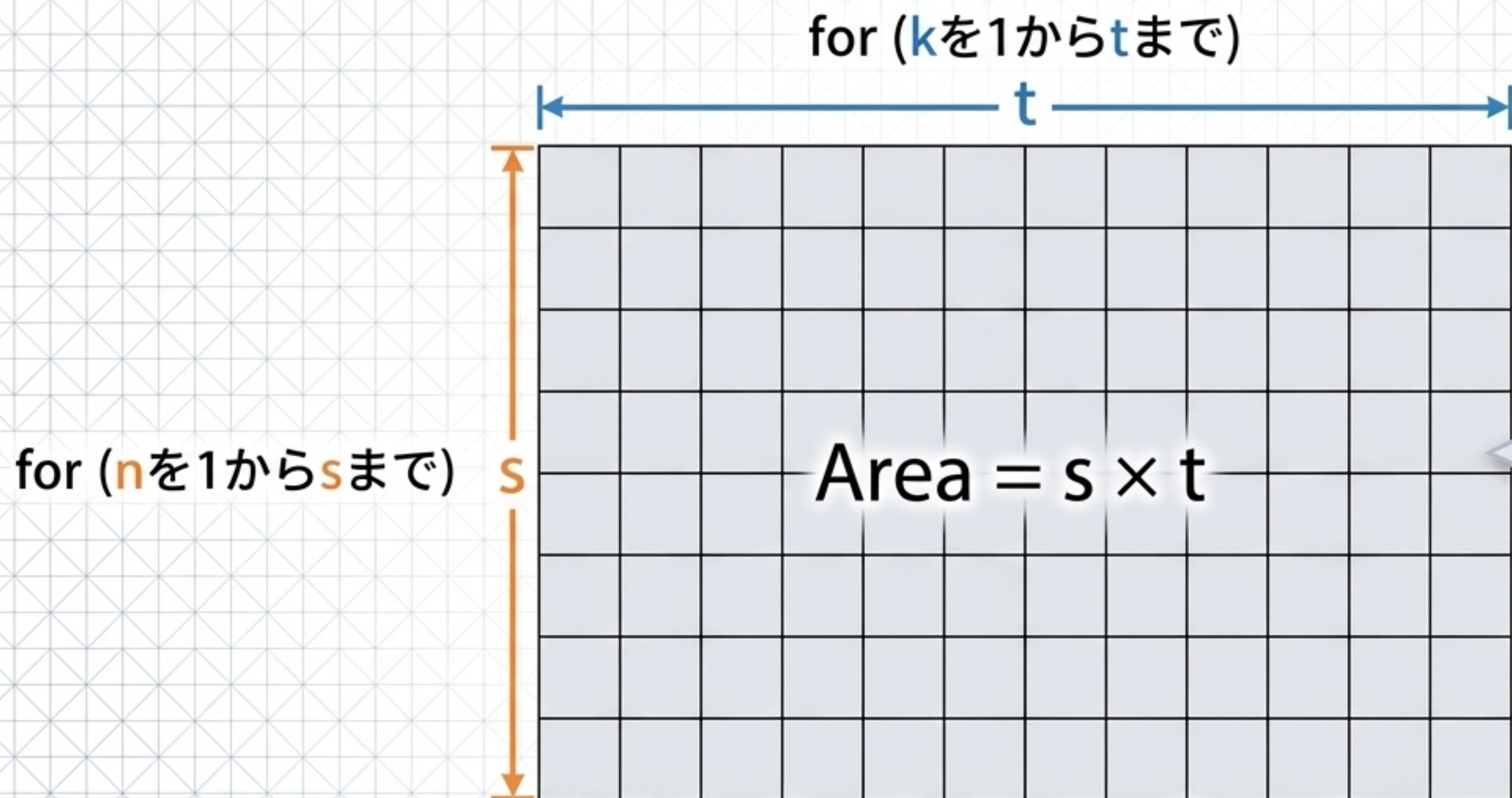
「表を全て埋め終わった時、最終的な『最長共通部分列の長さ』がるのはどのマス？」

「表を全て埋め終わった時、最終的な『最長共通部分列の長さ』が記録されているのはどのマス？」

台本 (Script)

台本：「表の穴埋めループが全て終わりました。最後に、この関数が返す値[カ]は何でしょうか？表を最初から最後まで順番に埋めていった時、最終的な答えが導き出されるのは、表のどこだったかを思い出してください。」

最終関門：計算量（オーダー）とは「マスの総数」

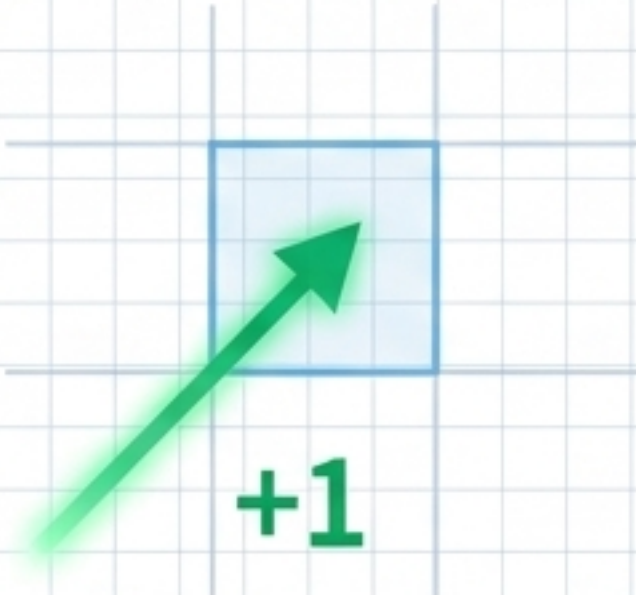
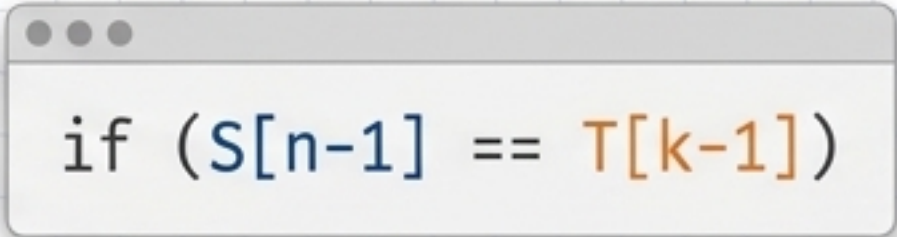


「このプログラムは、
何回の作業（マスの穴埋め）
を行っているか？
2つのループがどう掛け合わ
さっているかを見よう。」

台本
(Script)

台本：「最後の設問4です。時間計算量 $O()$ 。難しそうな言葉ですが、要するに『このプログラムは表のマスを何個埋める作業をしましたか？』という質問です。縦に s 行、横に t 列。二重のforループがどのように動くかを考えれば、自ずと答えは出ます。」

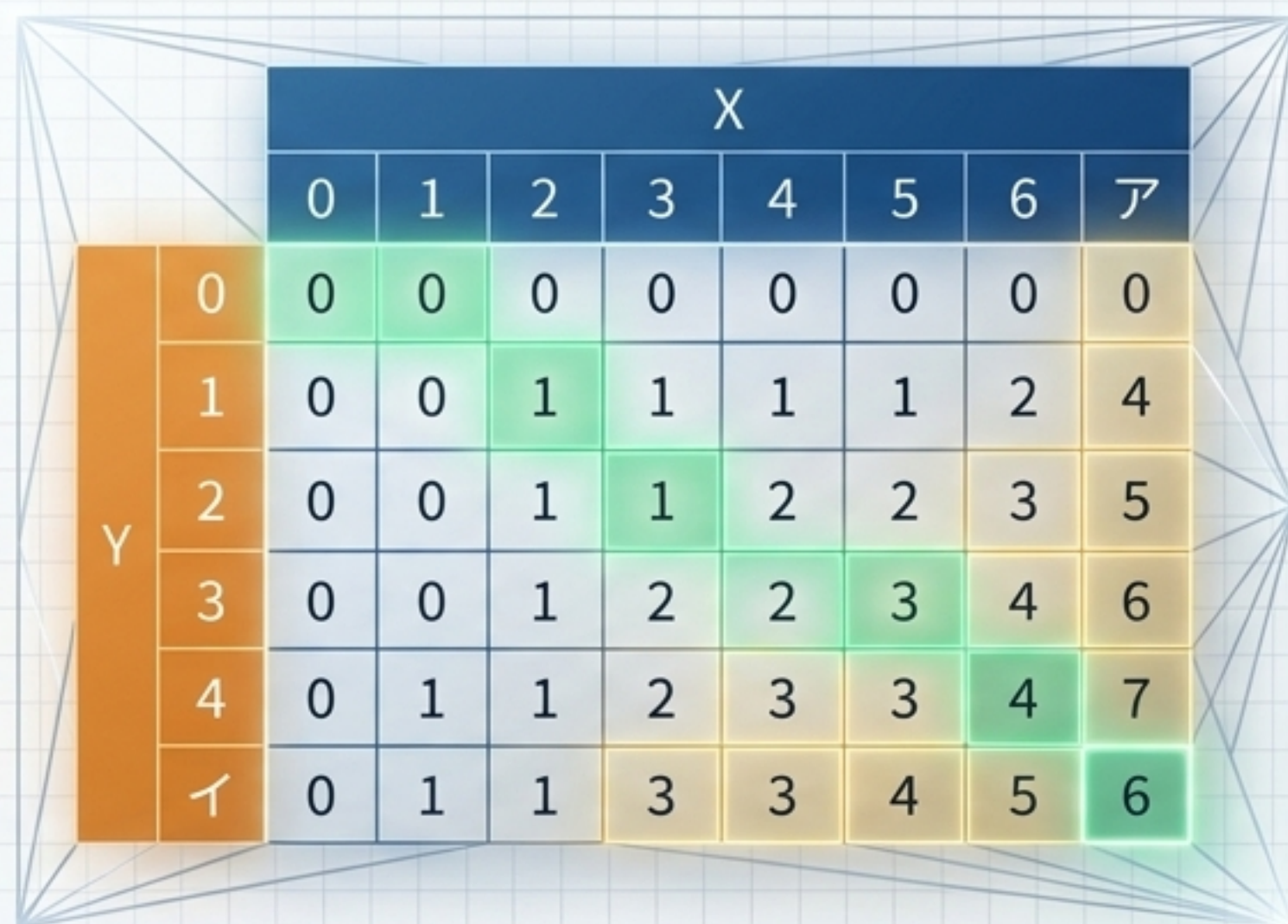
3つのアプローチは、すべて同じことを語っている

文字の一致を見つける (Finding a Character Match)		
1 直感 (Intuition)	2 視覚 (Visual)	3 抽象 (Abstraction)
		

台本 (Script)

台本：「ここまで解いてきたあなたなら、もう気づいているはずです。手作業で線を引いたのも、表で斜め上に+1したのも、コードでif文を書いたのも、実はすべて『共通の文字を見つける』という全く同じ現象を、別の表現で追っただけなのです。」

アルゴリズムの「設計図」を手に入れたあなたへ



		X							
		0	1	2	3	4	5	6	ア
Y	0	0	0	0	0	0	0	0	0
	1	0	0	1	1	1	1	2	4
	2	0	0	1	1	2	2	3	5
	3	0	0	1	2	2	3	4	6
	4	0	1	1	2	3	3	4	7
	イ	0	1	1	3	3	4	5	6

視点が分かれば、アルゴリズムは暗記ではなく『パズル』になる。
さあ、実際の解答欄を自分の力で埋めてみよう。

台本 (Script)

台本：「いかがでしたか？アルゴリズムは暗記するものではなく、仕組みを読み解くパズルです。今日手に入れた視点を持って、ぜひもう一度、問題文に向き合ってみてください。きっと、答えが浮かび上がってくるはずです。応援しています！」