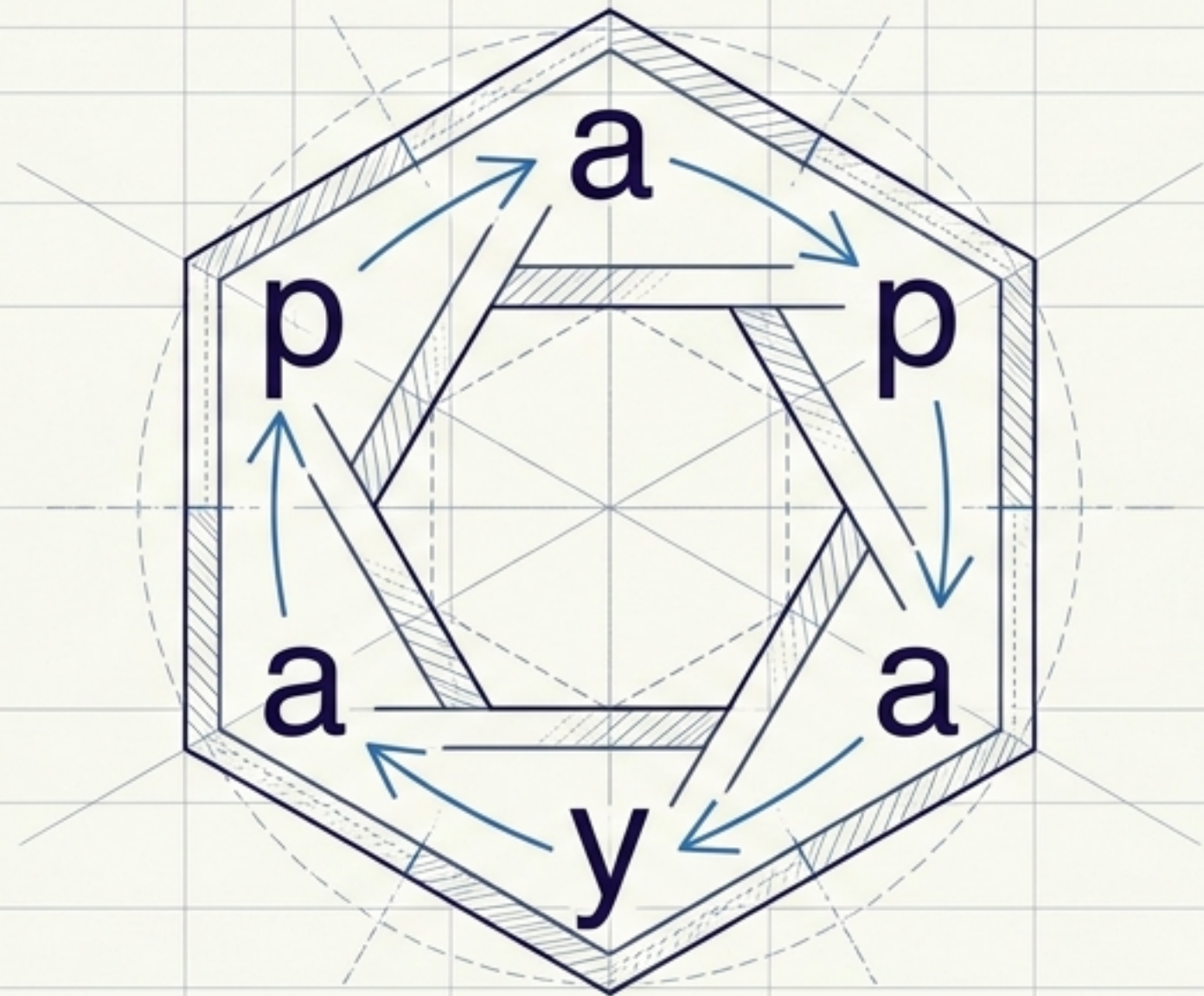




Block-sorting 変換処理の完全解剖

コードから読み解くアルゴリズムの論理
(平成27年 春期 応用情報 問3 設問2 思考プロセス解説)

```
function encode(InputString)
  rString ← InputString
  for( i を ア から イ まで1ずつ増やす )
    EncodeArray[i] ← rString
    rString ← rotation(rString)
  endfor
  sort1(EncodeArray)
  BlockSortString を空文字列に初期化する
  for( k を ア から イ まで1ずつ増やす )
    BlockSortString の末尾に EncodeArray[k]の末尾の1文字を追加する
    if( ウ )
      Line ← k
    endif
  endfor
endfunction
```

ミッション：3つの欠落したピースを特定せよ

🎯 目的

関数 encode に隠された3つの空欄を論理的に特定する。

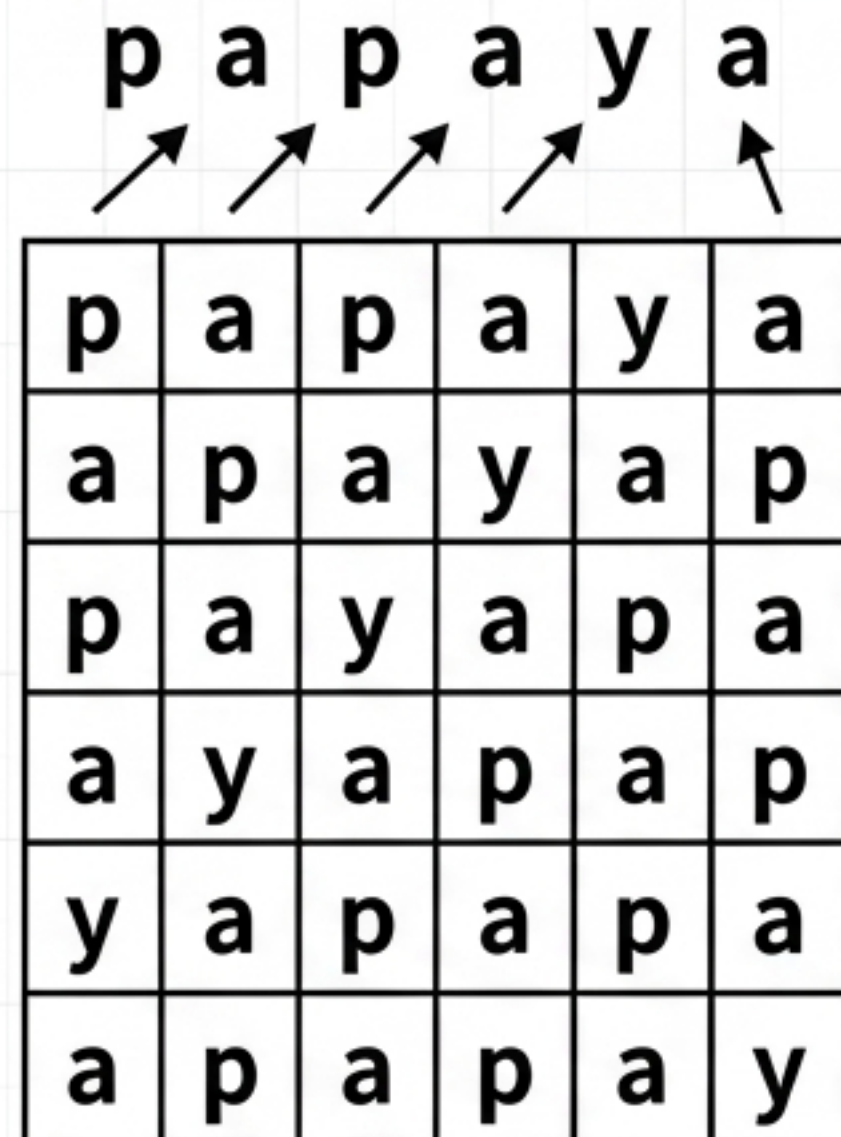
📖 アプローチ

アルゴリズムの「物理的な動き（図解）」と「実装（コード）」をマッピングする。

⚙️ ルール

プログラミング言語の常識による推測は捨てる。すべて「仕様表」と「処理手順」から必然のロジックを導き出す。

1. 生成 (Generate)



2. 整列 (Sort)

1	a	p	a	p	a	y
2	a	p	a	y	a	p
3	a	y	a	p	a	p
4	p	a	p	a	y	a
5	p	a	y	a	p	a
6	y	a	p	a	p	a

A
Z

3. 抽出 (Extract)

1	a	p	a	p	a	y
2	a	p	a	y	a	p
3	a	y	a	p	a	p
4	p	a	p	a	y	a
5	p	a	y	a	p	a
6	y	a	p	a	p	a

プログラムの解読に入る前に、データがどう変形していくか、この「3つの動き」の物理的イメージを頭に焼き付けることが最も重要です。

Phase 1: 生成 (Generate)

シフトと配列への格納

Phase 2: 整列 (Sort)

Phase 3: 抽出 (Extract)

結果の結合と
行番号の特定

```
function encode(InputString)
  rString ← InputString
  for( i を ア から イ まで1ずつ増やす )
    EncodeArray[i] ← rString
    rString ← rotation(rString)
  endfor
  sort1(EncodeArray)
  BlockSortString を空文字列に初期化する
  for( k を ア から イ まで1ずつ増やす )
    BlockSortString の末尾に EncodeArray[k]
    の末尾の1文字を追加する
    if( ウ )
      Line ← k
    endif
  endfor
endfunction
```

実装コードのアーキテクチャを解剖する

- プログラムは先ほどの「3つの動き」を忠実に再現するブロックで構成されている。
- 空欄 ア と イ はループの反復範囲を決定する。
- 空欄 ウ は抽出時の条件を決定する。

Phase 1: ブロック生成ループ

for (i を **ア** から **イ** まで1ずつ増やす)

Phase 3: 文字抽出ループ

for (k を **ア** から **イ** まで1ずつ増やす)

なぜ2つのループは全く同じ範囲を持つのか？

目的が違っても関わらず、反復範囲が同じである理由。
それは、アルゴリズムの過程（ソート）を経ても、
ブロックの「行数」という物理的次元は一切変化しないからである。

ループの起点：仕様書に隠された絶対ルールを読み解く

名称	種類	内容						
EncodeArray[n]	配列	巡回シフト後のブロックを格納する。ブロックの1行を文字列として、配列の一つの要素に格納する、配列の添字は1から始まる。 例 <table><tr><td>“papaya”</td><td>“apayap”</td><td>“payapa”</td><td>“ayapap”</td><td>“yapapa”</td><td>“apapay”</td></tr></table>	“papaya”	“apayap”	“payapa”	“ayapap”	“yapapa”	“apapay”
“papaya”	“apayap”	“payapa”	“ayapap”	“yapapa”	“apapay”			

問い (Q)

このプログラムにおいて、配列には「どこ」からデータが格納される仕様か？

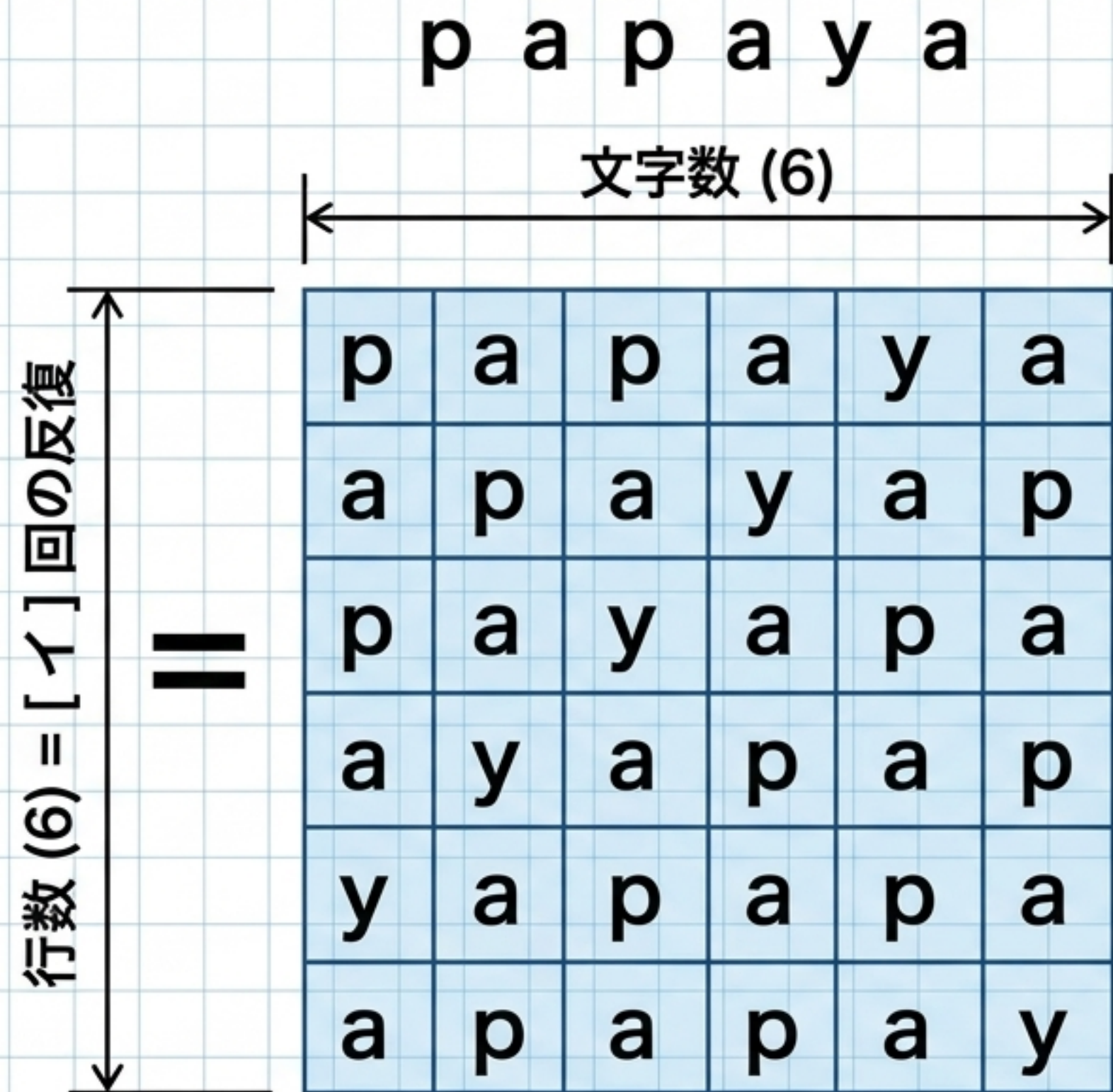
ヒント (Hint)

C言語やJavaの「配列は0から始まる」という一般的なプログラミングの常識を一度捨てること。

結論への導き (Action)

EncodeArray の仕様定義に絶対に従い、ループカウンタの開始値 [**ア**] を決定せよ。

ループの終点：生成される「正方形」の法則



問い (Q)

1文字ずつ巡回シフトしてブロックを作る
ループは、何回繰り返されるべきか？

ヒント (Hint)

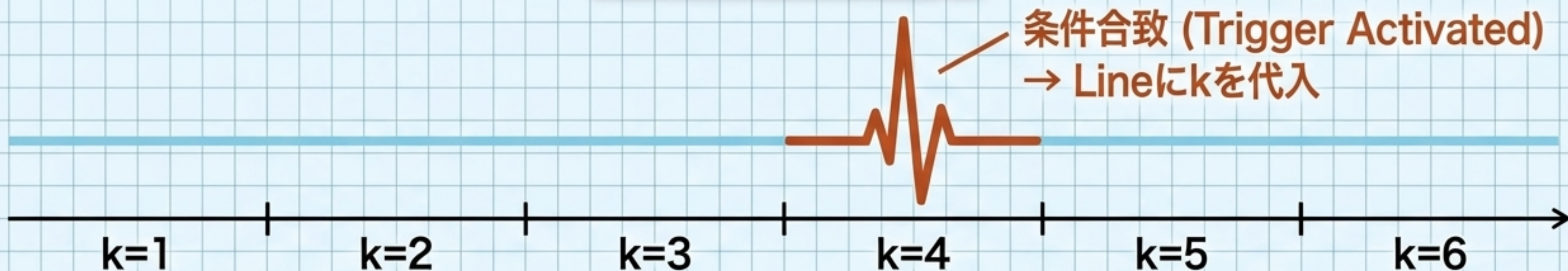
シフトを繰り返して「正方形のブロック」
が完成するということは、生成される「総
行数」は、入力された元の文字列の「どの
属性」に依存しているか？

結論への導き (Action)

表2の InputString の説明欄に着目し、
その属性を表す表現を [イ] に当てはめよ。

条件分岐のトリガー：変数 Line が動く瞬間

```
if ( [ウ] )  
    Line ← k  
endif
```



Analysis:

- この if 文は、ループが回る中で「探し求めている特定の行を見つけた瞬間」にだけ真 (True) になる。
- その瞬間、変数 Line に現在の行番号 k が記録される。

探索のゴール：我々は一体「何」を探しているのか？

変数名: Line

内容: ソート後のブロックでの
入力文字列の行番号。

ソート後のブロック

a	p	a	p	a	y
a	p	a	y	a	p
a	y	a	p	a	p
p	a	p	a	y	a
p	a	y	a	p	a
y	a	p	a	p	a

Core Concept:

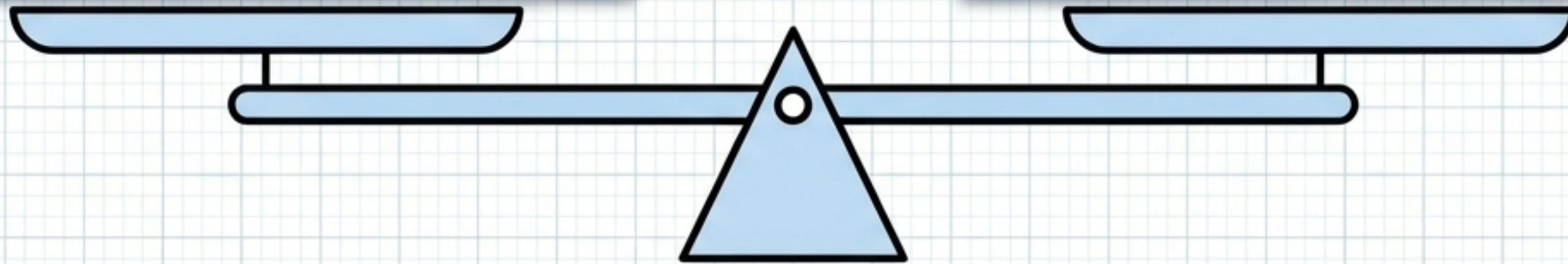
- 配列 EncodeArray には、ソートされた後のすべての行が格納されている。
- プログラムは上から順に1行ずつ調べ、「入力された元の文字列」と全く同じ並びになっている行を探し出そうとしている。

等価判定：天秤に載せるべき2つの変数は何か

探すべき対象 (The Target)
元の入力文字列を保持している変数

=

現在見ている行 (Current State)
現在ループで調べている行
(添字 k を使用)



- | | |
|---------|---|
| Hint 1: | 目的の行を見つけるには、「何」と「何」が完全に一致したかを判定すればよいか？ |
| Hint 2: | 表2から「元の入力文字列」を保持している変数名を見つける。 |
| Hint 3: | プログラム中から「現在調べているソート後の行」を指し示す配列の要素を見つける。 |

Action (結論への導き):

これら2つが等しいことを示す条件式を【ウ】に構築せよ。

アルゴリズムと論理の完全な一致

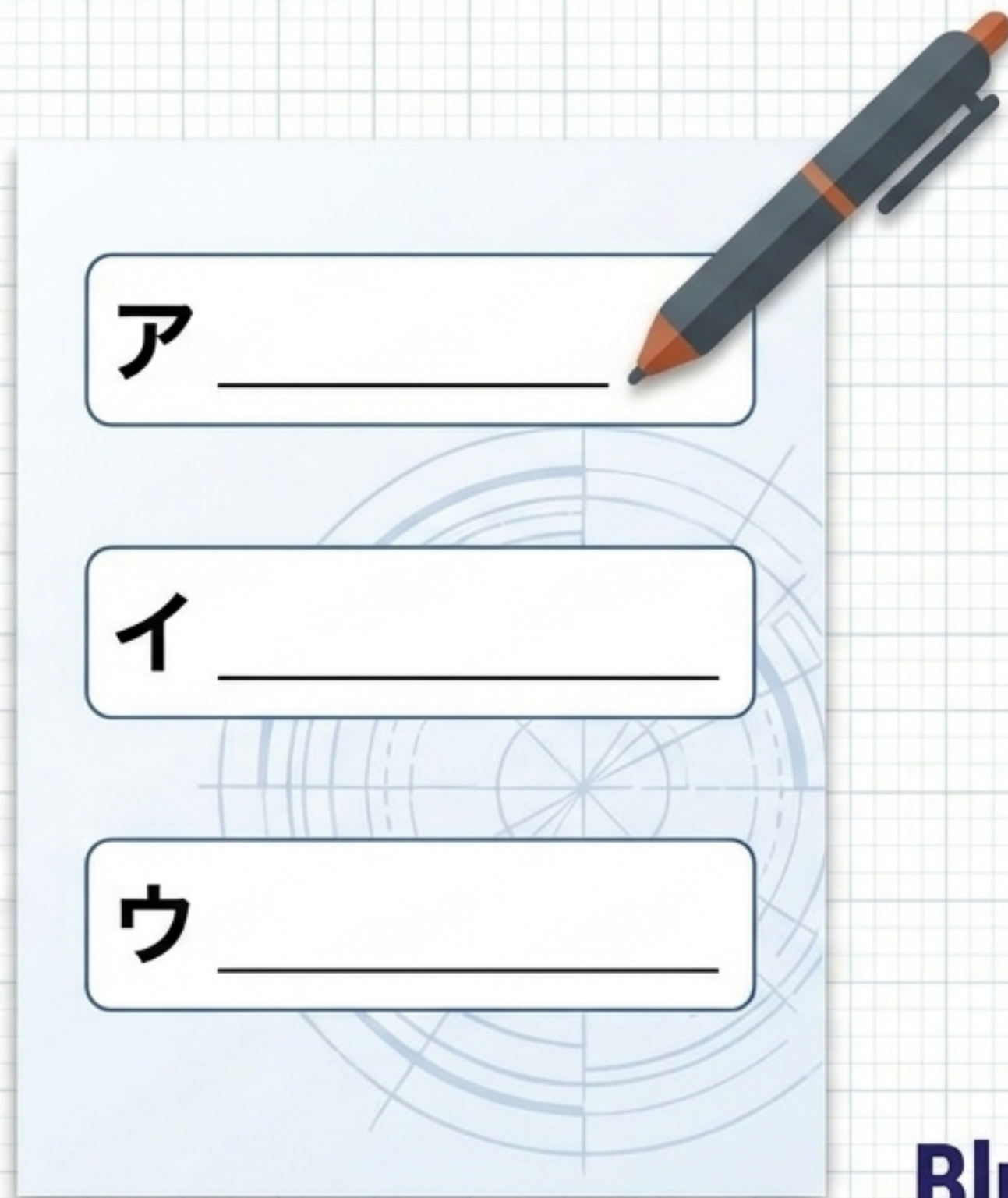
Diagnostic Matrix		
空欄 (Blank)	役割 (Role in Code)	根拠となる仕様/図 (Basis in Spec)
[ア]	配列の開始インデックス	表2の「配列の添字は1から始まる」という定義を厳守。
[イ]	処理すべき総行数	入力データの属性（文字列の長さ）が正方形の行数となる。
[ウ]	元の文字列を発見する条件	目的の変数と、現在の配列要素同士の等価判定。

Summary:

コード上の空欄は単なる文法の穴埋めではない。図2で示されたアルゴリズムの物理的な動きと、表2の厳密な仕様定義を橋渡しする論理の結節点である。

論理から解答へのボタンタッチ

- 全ての論理構造は出揃いました。
- もう一度「表2」の変数名と、「図3」のコードの記法を見直してください。
- あなたの言葉で、最終的な解答を組み立ててください。



ア _____

イ _____

ウ _____

Blueprint Complete.
(設計図の完成)