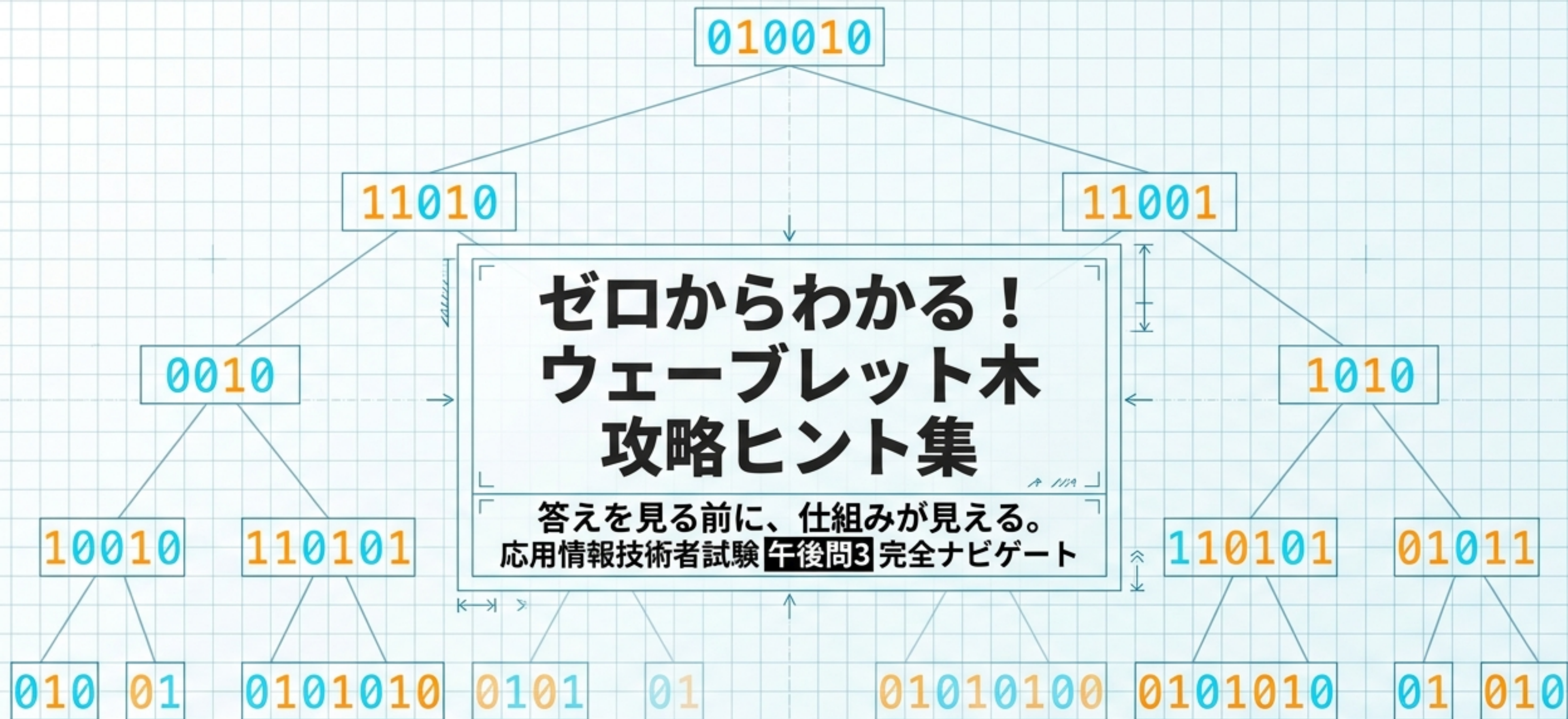
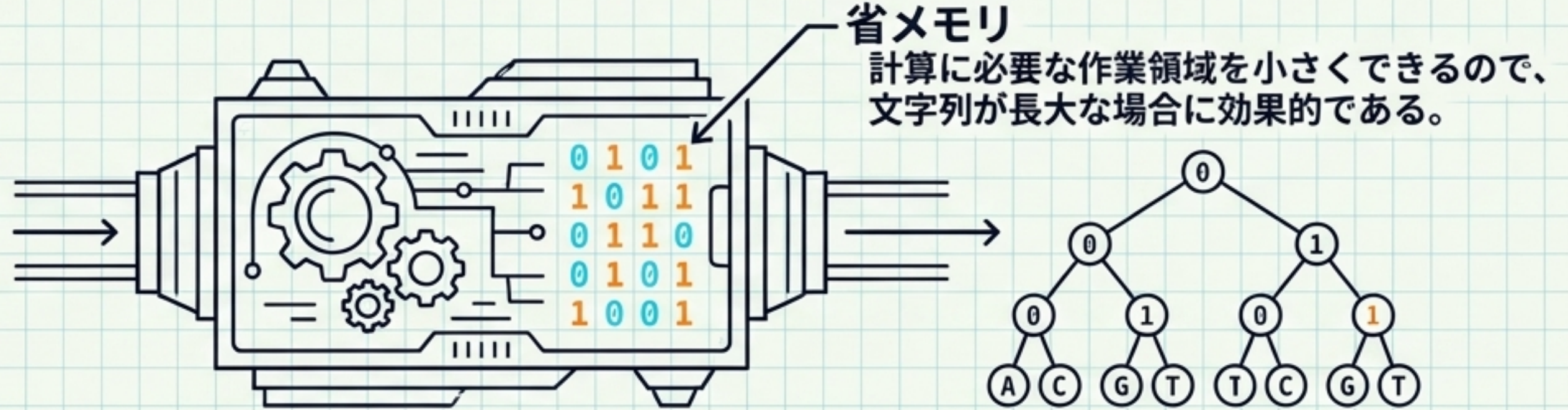




※本スライドは解答を直接提示しません。あなた自身で「アハ体験」に辿り着くための設計図です。

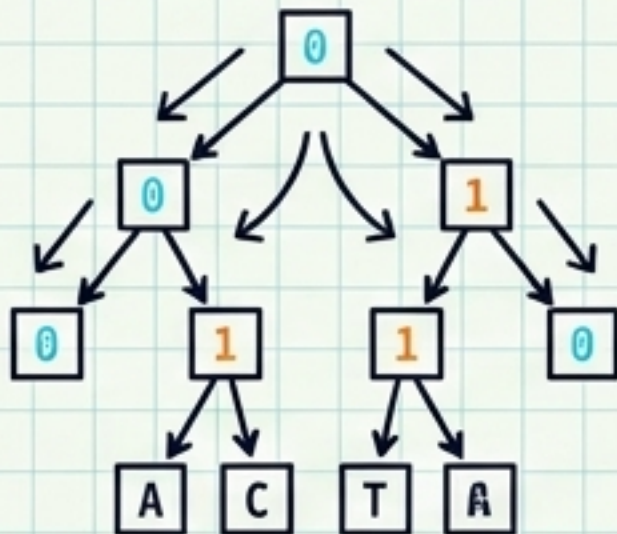
3つのシステムを解説せよ

CTCGAGAGTA



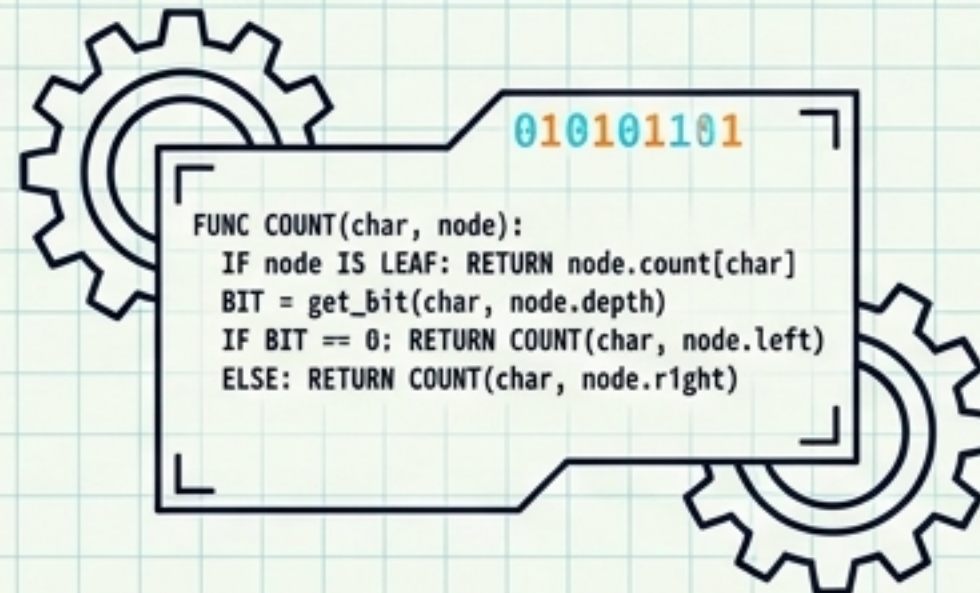
物理構造の構築

設問1・2 [ア][イ][ウ]



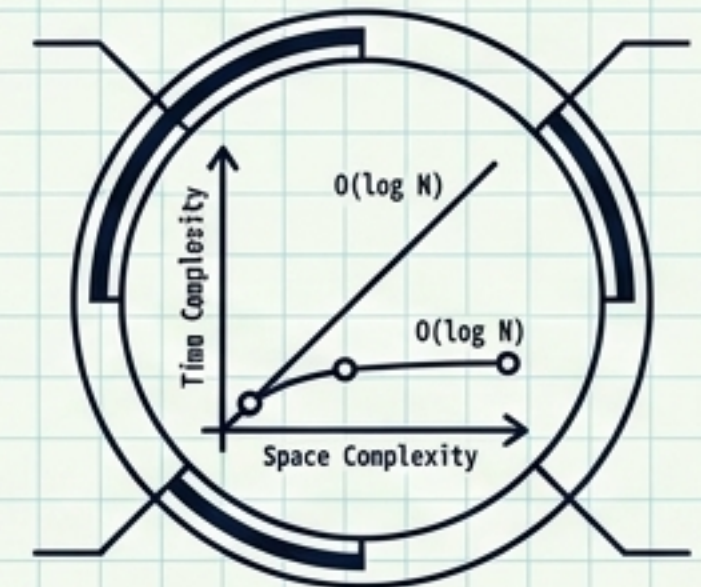
エンジンの解説

設問3 [エ][オ][カ][キ]



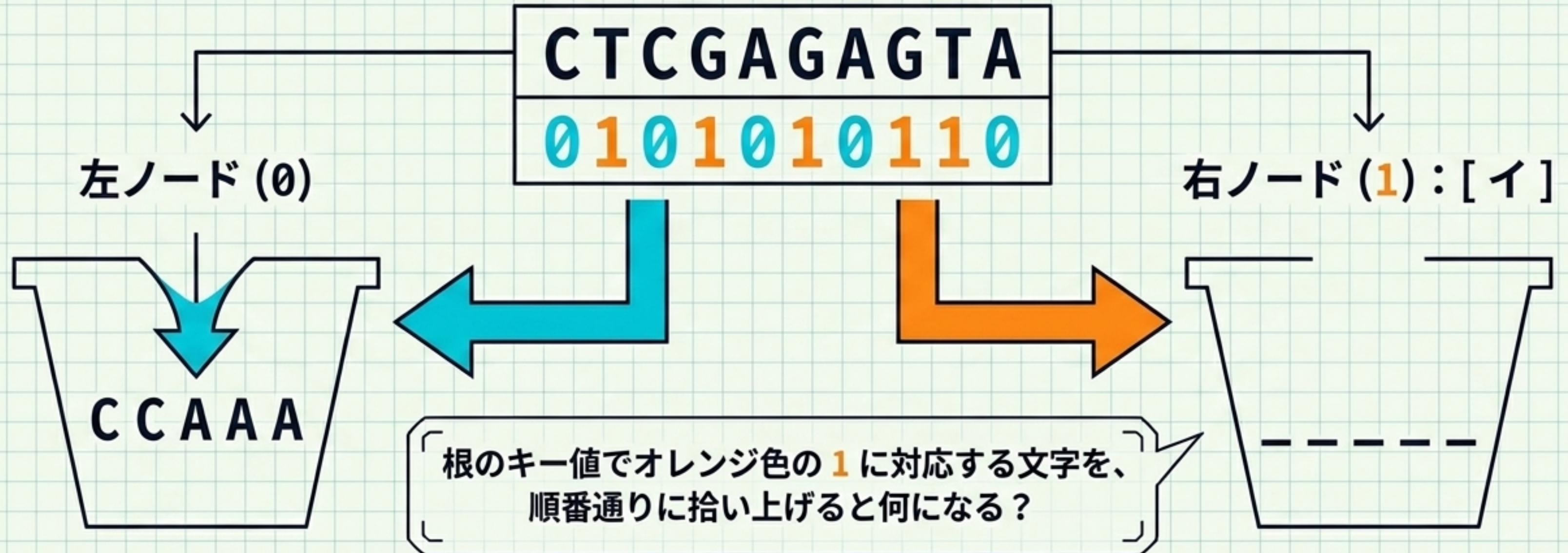
パフォーマンス評価

設問4 [ク][ケ][コ]



【空欄イ】 右の子ノードへ進む文字列を特定せよ

振り分けの絶対ルール：「1 番目のビットが 0 なら左へ、1 なら右へ」



【空欄ア】 深さ1のノードにおけるキー値を生成せよ

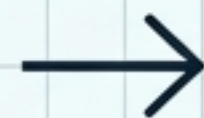
抽出の法則：「深さ + 1 = 抽出するビット位置」



深さ1（右ノード）の対象文字列

TGGGT

深さ 1 + 1 = 2番目のビットを抽出



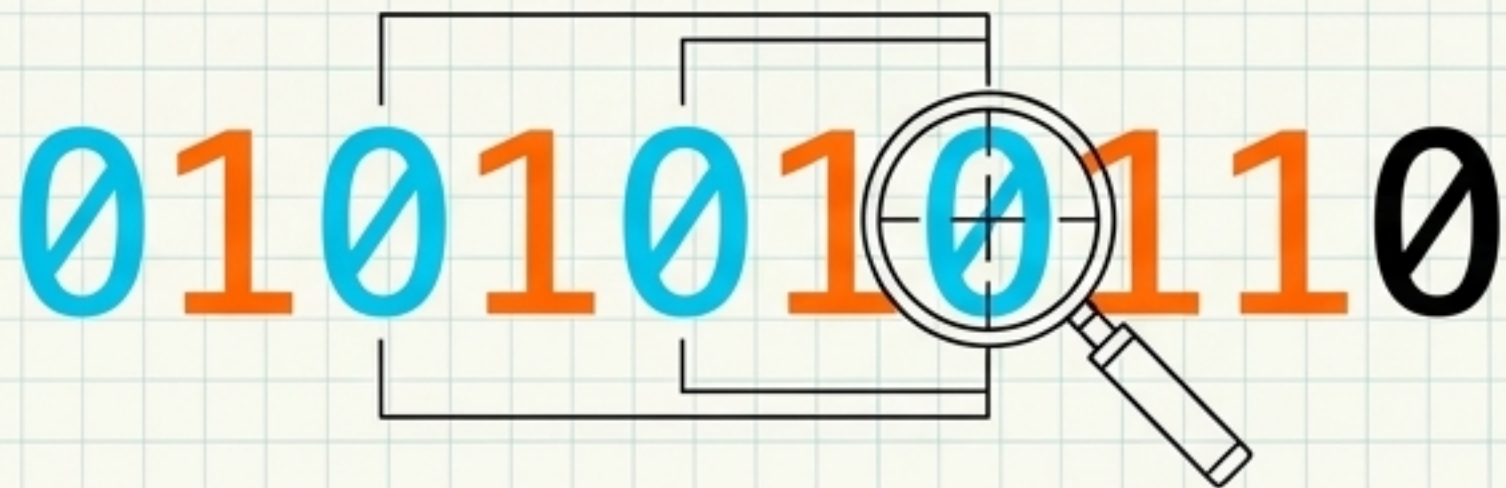
T	→	1	1
G	→	1	0
G	→	1	0
G	→	1	0
T	→	1	1

ハイライトされた2番目のビットを上から順に並べたビット列が、そのまま【空欄ア】の答えになる

【空欄ウ】 左の子ノードに振り分けられる文字の個数を数えよ

カウント法則：「子ノードの要素数 = 親キーの該当ビット数 = 移動した文字列の長さ」

ルートA（ビットを数える）



左に進むための条件は 0。
このキー値の中に 0 はいくつある？

ルートB（文字数を数える）

CCAAA
← 5 文字

移動してきたこの文字列は、
全部で何文字？

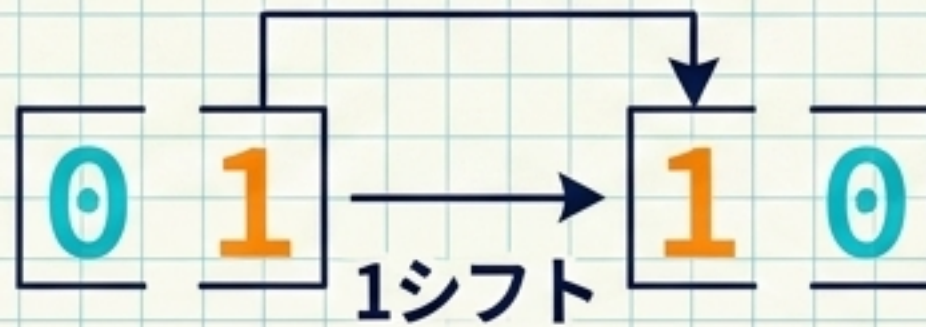
どちらのルートからアプローチしても、導き出される【空欄ウ】の整数は全く同じになる。

【空欄エ】 d番目のビットを抽出するための「左シフト量」を導け

スライディング・ウィンドウ：「調べたい位置（d）に応じて、マスク用の1を動かす」

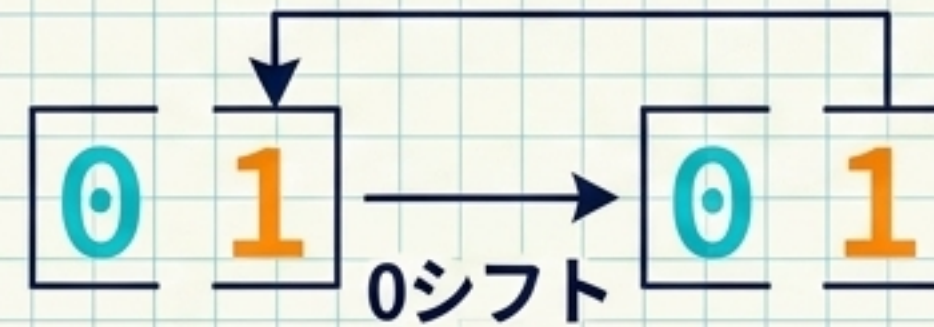
大域変数 DEPTH = 2

左端のビットを狙う（d=1）



$$[?] - 1 = 1$$

右端のビットを狙う（d=2）

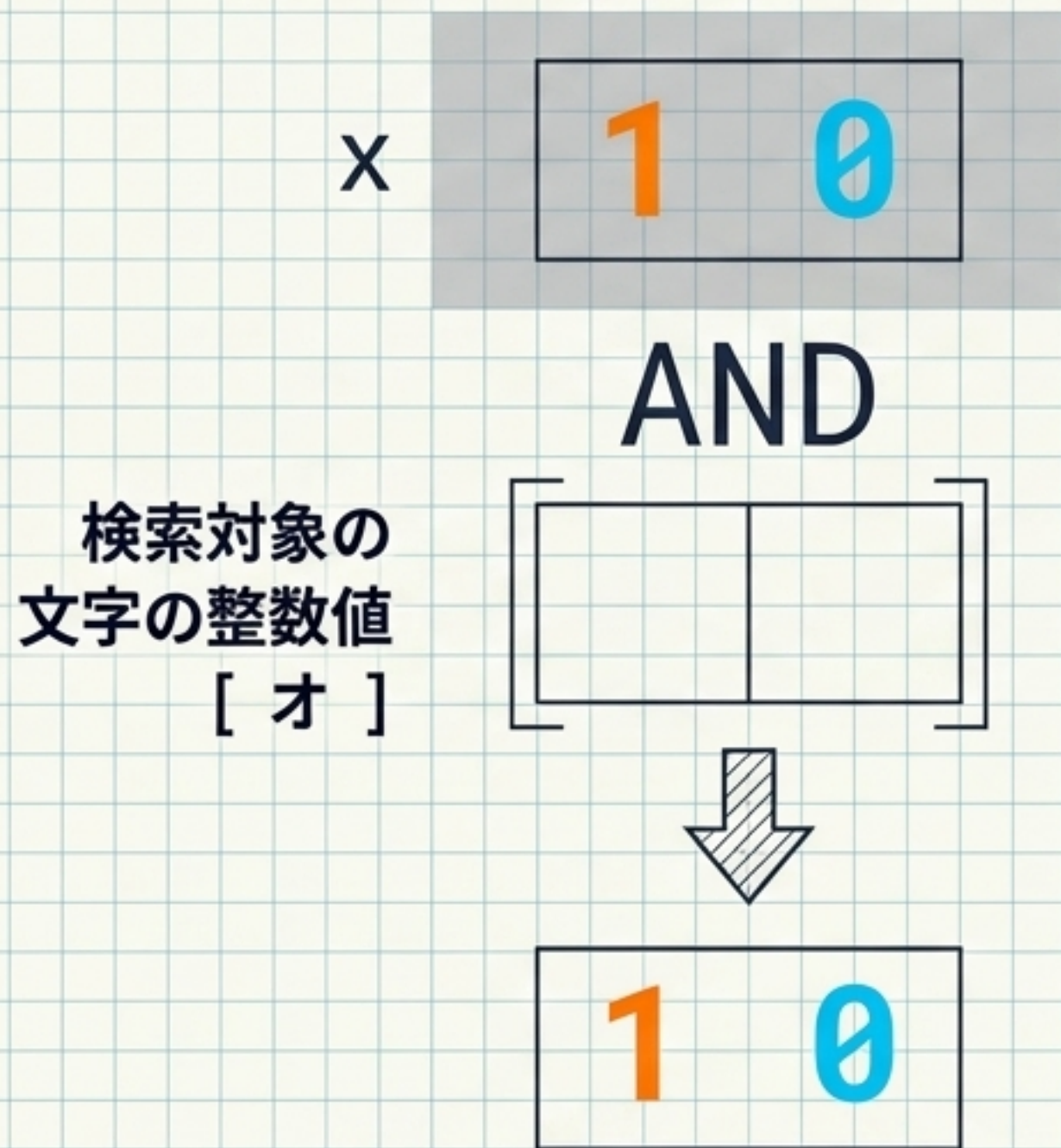


$$[?] - 2 = 0$$

どちらのケースでも共通して使える変数は何？
コード全体のビット数を表す大域変数 DEPTH を使って式を完成させよう。

【空欄オ】 論理積（AND）でマスク処理を行う「対象」を特定せよ

マスク処理の標的：「 $x \text{ and } [\text{ターゲット}]$ 」により、ターゲットの特定ビットだけをくり抜く」



プログラムの目的：私たちが探している文字コード（例：C ならば $r=1$ ）のビット配列配列そのものを透視して調べること。

私たちが今、そのビット配列を調べようとしている『対象文字の整数』が格納された変数はどれだろうか？

【空欄力】 左の子ノードへ遷移するための条件式を完成させよ

ノード遷移の原則：「構築時の物理ルールと、探索時のプログラム論理は完全に一致する」

物理ルール（Slide 3で確認済）

ビット値が 0 の場合：
文字は左の子ノードへ

プログラム論理（図3）

if (b が [力] と等しい)
nodep <- nodep.left

左に進む（nodep.left が実行される）ためには、抽出したビット値 b が何であればよいか？

【空欄キ】 次の階層へ引き継ぐ「検索する文字数（**n**）」を更新せよ

ボタンタッチの法則：「現在のノードで条件に一致した文字の数だけが、次のノードへ引き継がれる」

Step 1

ループで 1 から **n** 文字目までを検査



Step 2

探しているビット **b** と一致するたびに、
変数 count に加算（タリー）



Step 3

木を一段下りる (nodep.left または nodep.right)



Step 4

次の階層での新しい検索上限数 **n** を [**キ**] で上書きする

Mini Example

具体例：根ノード（全体 $n=10$ ）で
条件に合う文字が 5 つ 見つかった場合、次のノードではその 5 文字
だけを対象に検索を行う。

いま数え上げたばかりの『条件に一致した総数』が格納されている変数はどれか？

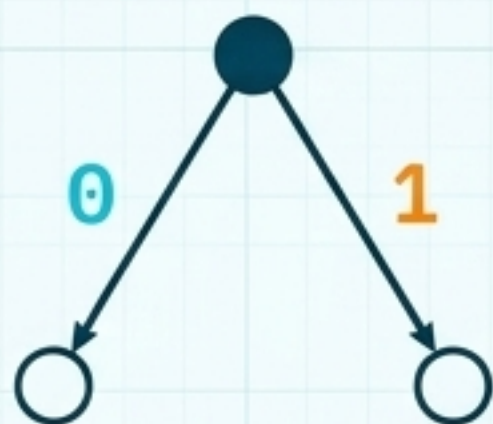
Mission

【空欄ク】 1つの文字が葉に辿り着くまでのノード数（木の深さ）を数式化せよ

Rule

2分木の深さ：「 σ 種類の文字を完全に1つずつに仕分けるために必要な分岐の回数」

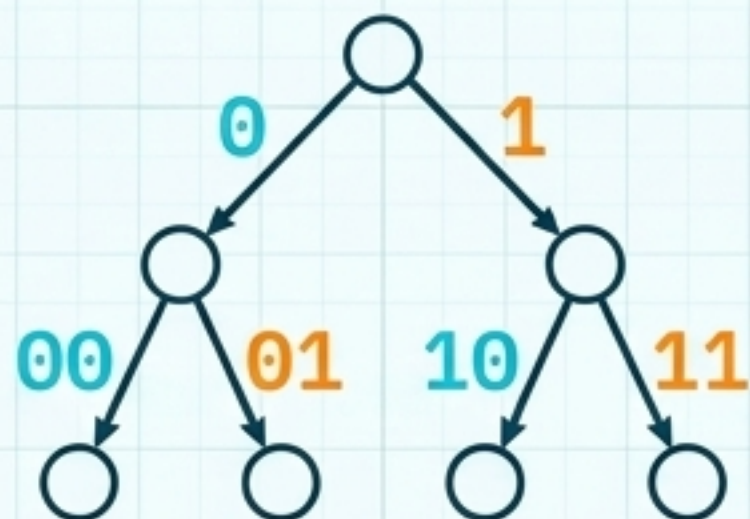
$\sigma = 2$ 種類



1回の分岐（深さ1）

$$\log_2(2) = 1$$

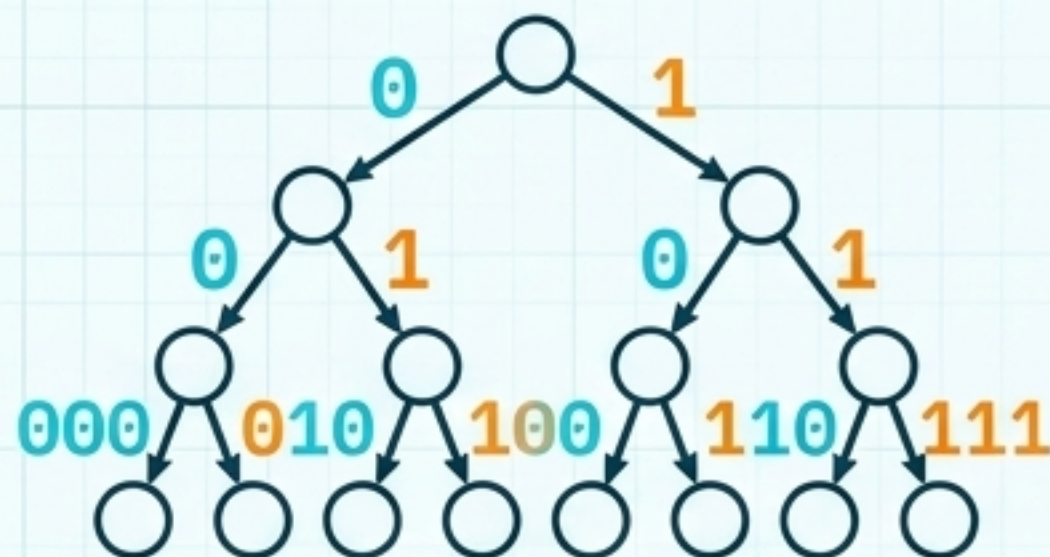
$\sigma = 4$ 種類（DNA等）



2回の分岐（深さ2）

$$\log_2(4) = 2$$

$\sigma = 8$ 種類



3回の分岐（深さ3）

$$\log_2(8) = 3$$

対数 $\log_2([?])$ の括弧の中に入るべき、文字の種類数を表す変数は何だろうか？

Mission

【空欄ケ】 【空欄コ】 木全体の「総計算時間」と「データ長の総和」を導け

Rule

面積のメタファー：「全体量 = 横（1階層の幅） × 縦（木の深さ）」

Visual Clue

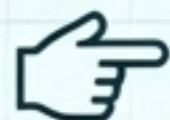
横幅：元の文字列の長さ（[ケ]）

どの階層でも、ビットを横に足すと必ず全体の長さと同じになる。

縦幅：木の深さ
($\log_2 \sigma$)

面積（全体量） = [横] × [縦]
オーダー表記： $O([ケ] \times \log_2 \sigma)$

00 01 10 11 0 01 011 111 11 010 10 110111

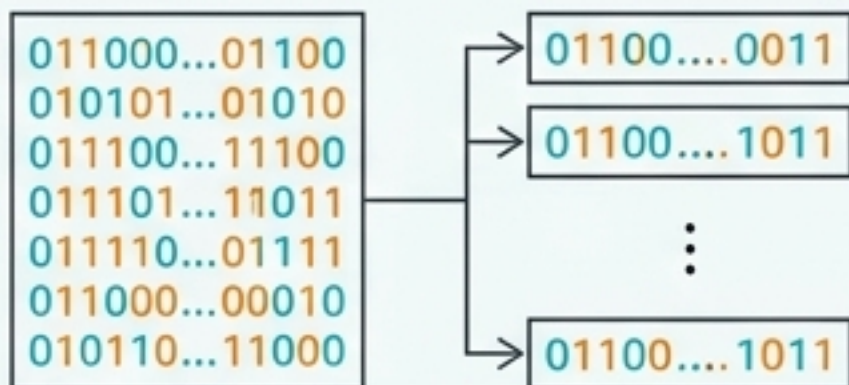


元の文字列の長さを表す変数『N』と、深さを表す式を掛け合わせて【空欄コ】の式を作ろう。

全体像の統合：なぜウェーブレット木は「省メモリ」なのか？

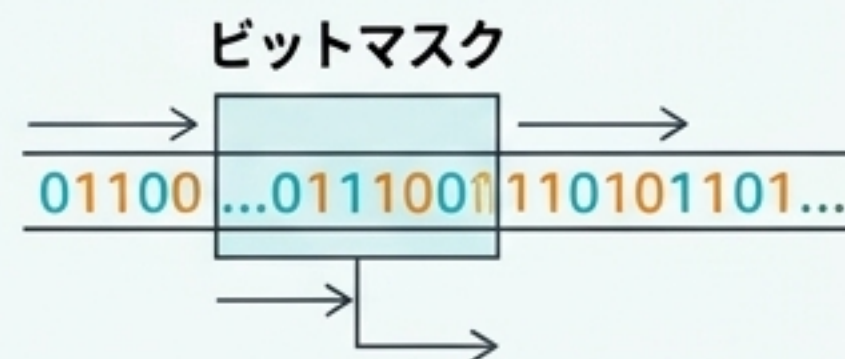
1. 物理（構造）

巨大なテキスト配列を、0と1のコンパクトな振り分けルールで再構築。



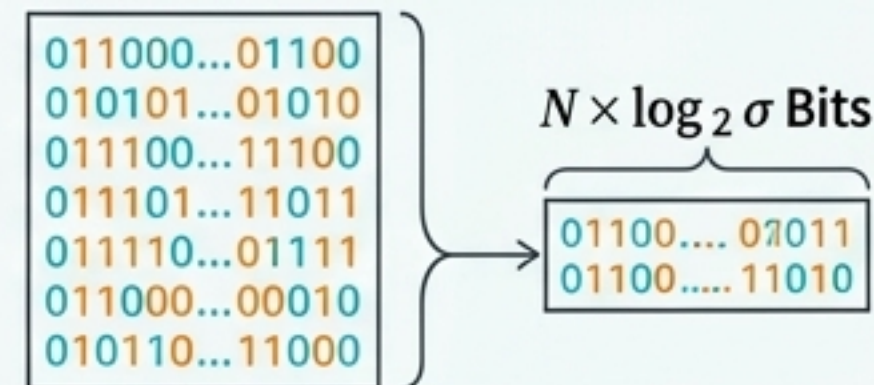
2. エンジン（探索）

ビットマスクによる高速なスライディング・ウィンドウで、目的データピンポイント検索。



3. 評価（結論）

巨大なDNA文字列も、最終的には $N \times \log_2 \sigma$ ビットの極小スペースに圧縮完了。



物理構造のルールを理解し、プログラムがそれをどうなぞるかを見極め、最後に幾何学的な図形として計算量を捉える。この設計図があれば、午後問題のアルゴリズムは確実に解読できる。