

誤差拡散法の謎を解く

応用情報技術者試験：初心者のためのアルゴリズム読解ガイド

TITLE BLOCK

REFERENCE NO. 2405

DIMENSION A

DIMENSION A

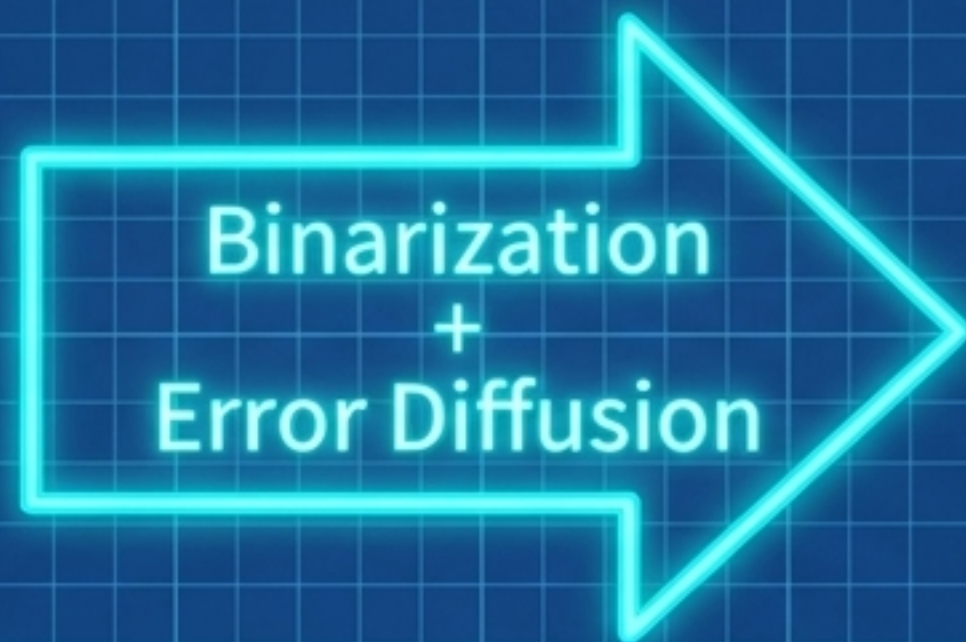
SCALE 1:1

DIMENSION B

2つの色だけで「グレー」を作る魔法



左：階調あり（0～255の濃淡）



右：モノクロ2値（黒と白のみ）

人間の目の錯覚を利用し、黒と白の点の密度でグレーを表現する技術。

コアルール：「判定」して「分配」するだけ

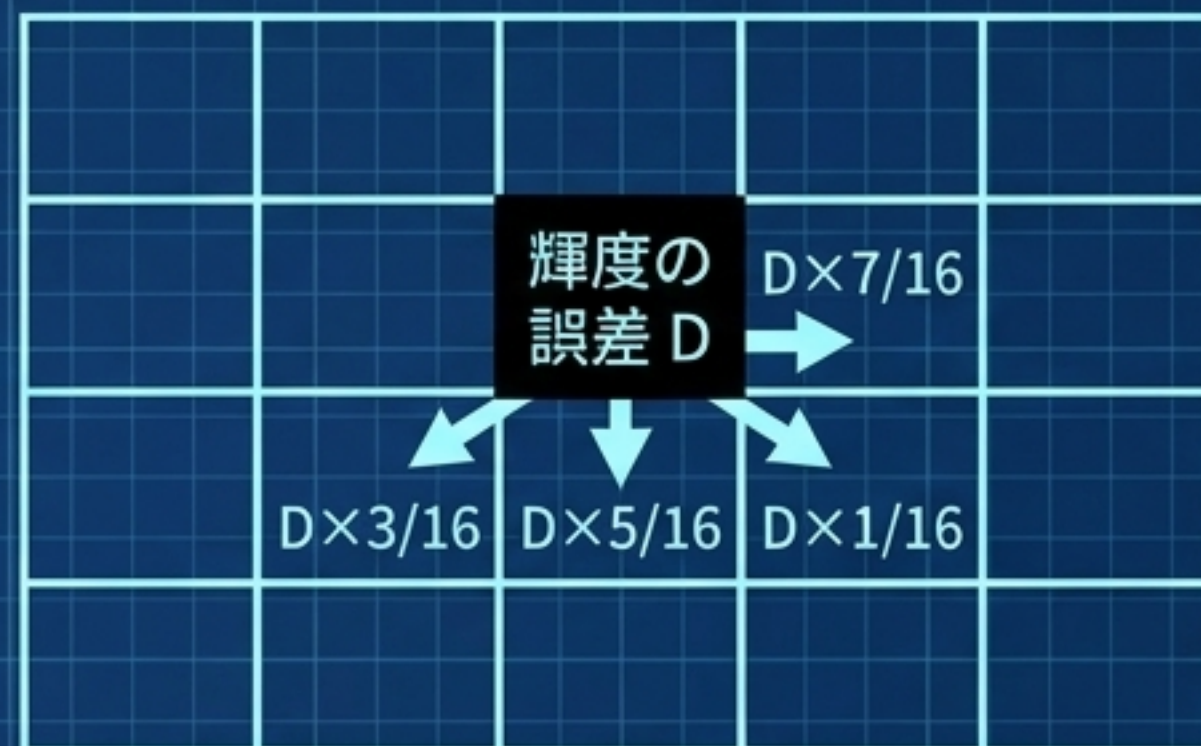
STEP 1: 判定 (Decide)



基準値（128）で白か黒かを決める。



STEP 2: 分配 (Share)



理想との「ズレ（誤差）」を隣のピクセルに押し付ける。

設問1 ヒント：配列のトレース術

元の輝度:35									
0	223	128	35	220	0	255	0	49	0
30	22	18	55	197	-6	11	33	7	0
35	122	250	105	15	0	0	0	0	0
38	153	251	120	18	0	0	0	0	0
変換前画像					変換後輝度配列				

太枠の右隣のピクセル（元の輝度:35）を処理します。

Mental Checklist

1

【Hint 1】

Fを計算：元の輝度(35) + 蓄積された誤差(49) = ?

2

【Hint 2】

判定：Fは128以上？
未満？（白か黒か決定）

3

【Hint 3】

誤差Dを計算：Fの値を
そのまま誤差Dとするか？

4

【Hint 4】

分配：真下のピクセルへの
拡散割合は？（図2を確認）

ルール（日本語）とコードの翻訳マップ

日本語の仕様 (Rules)

3. 変換前画像の輝度と、変換後輝度配列の
同じ要素の値を加算し、これをFとする。

4. Fの値が128以上なら...
白(255)とする。

5. Dの値について...
定義された割合に従って配分し、
拡散先の要素に加算する。

プログラム (Code)

```
for ( y を 1 から height まで繰り返す )  
  for ( x を 1 から width まで繰り返す )  
    f ← [ ア ]  
    if ( [ イ ] )  
      d ← f - 255  
      bmpTo[x, y] ← 255  
    else  
      d ← f  
      bmpTo[x, y] ← 0  
    endif  
    for ( c を 1 から ratioCount まで繰り返す )  
      px ← x + tdx[c]  
      py ← y + tdy[c]  
      if ( (px が 1 以上) かつ (px が width 以下)  
          かつ (py が 1 以上) かつ (py が height 以下) )  
        bmpTo[px , py] ← [ ウ ]  
      endif  
    endfor  
  endfor  
endfor
```

設問2 [ア][イ] ヒント：輝度の合算と境界線

```
for ( y を 1 から height まで繰り返す )
  for ( x を 1 から width まで繰り返す )
    f ← [ア]
    if ( [イ] )
      d ← f - 255
      bmpTo[x, y] ← 255
    endif
  endfor
endfor
```

変数テーブル（抜粋）

	変数名	説明
1	bmpFrom[x, y]	変換前画像の輝度
2	bmpTo[x, y]	変換後輝度（蓄積された誤差）
3	f	判定に使う輝度の合計

【アのヒント】

手順3を翻訳。現在の「元の輝度」と「蓄積された誤差」を足し合わせる式は？
（表の配列名を確認）

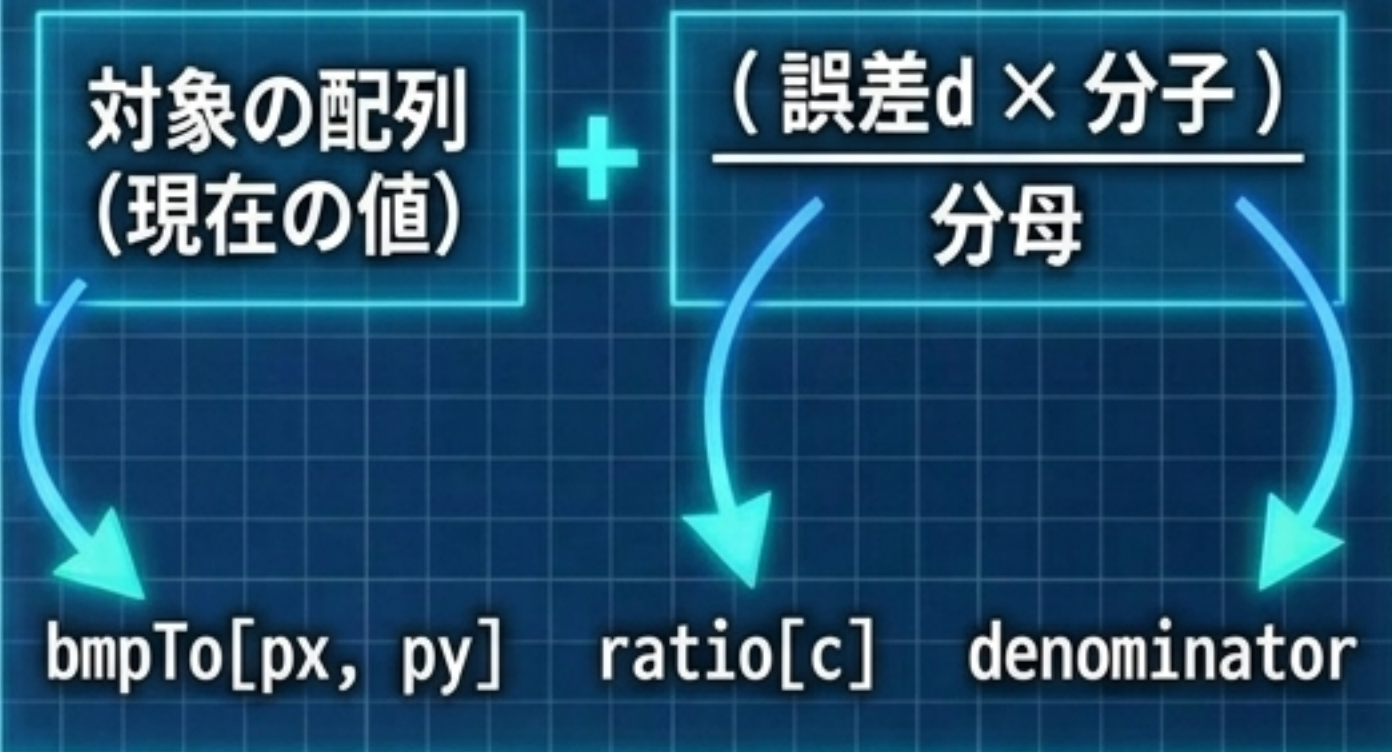
【イのヒント】

手順4を翻訳。Fの値がいくつ以上なら白（255）になる？

設問2 [ウ] ヒント：誤差の配分と加算

```
for ( c を 1 から ratioCount まで繰り返す )  
for ( c を 1 から ratioCount まで繰り返す )  
    px ← x + tdx[c]  
    py ← y + tdy[c]  
    if ( クリッピング判定 )  
        bmpTo[px, py] ← [ウ]  
    endif  
endfor  
endfor
```

拡散先のピクセルを
更新する式を作ります。



【ウのヒント】

「現在の値」に「新しく配分される誤差」を加算する。表から分子と分母の変数名を探そう。

画質向上の工夫：「ジグザグ」スキャン

Y座標 = 1 (奇数: Odd)



piled up
errors

Y座標 = 2 (偶数: Even)

問題点：

常に左から右へ処理すると、数学的な誤差が右側に偏って蓄積してしまう。

解決策：

Y座標が「奇数」なら左から右、「偶数」なら右から左へ処理の進行方向を反転させる。

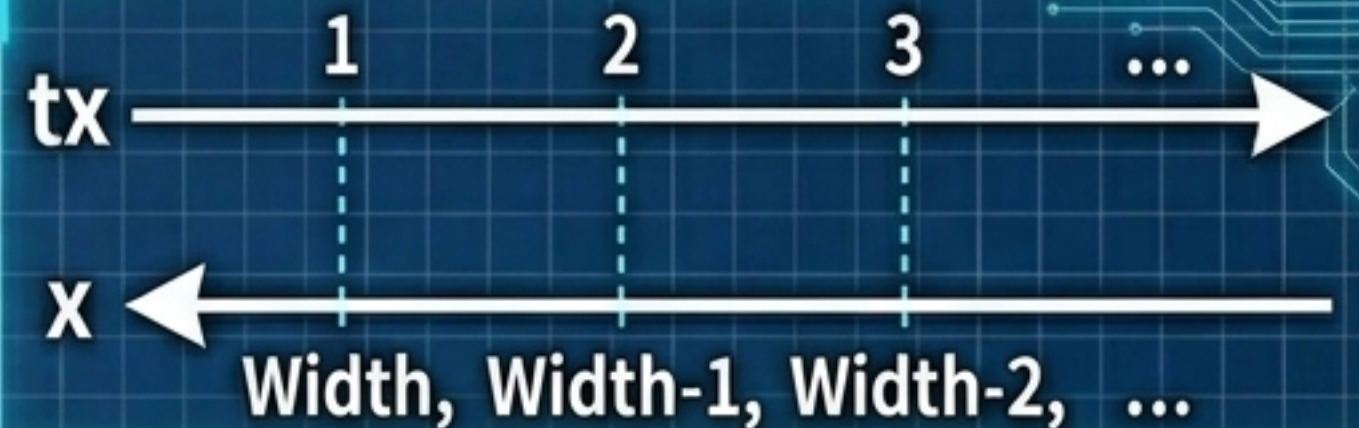
設問3 [エ][オ][カ] ヒント:偶数の判定と逆順ループ

```
for ( tx を 1 から width まで繰り返す )  
  x ← tx  
  if ( ( [エ] mod [オ] ) が 0 に等しい )  
    x ← [カ]
```

$$Y \div 2 = \text{商} \text{ 余り } 0$$

【エ・オのヒント】

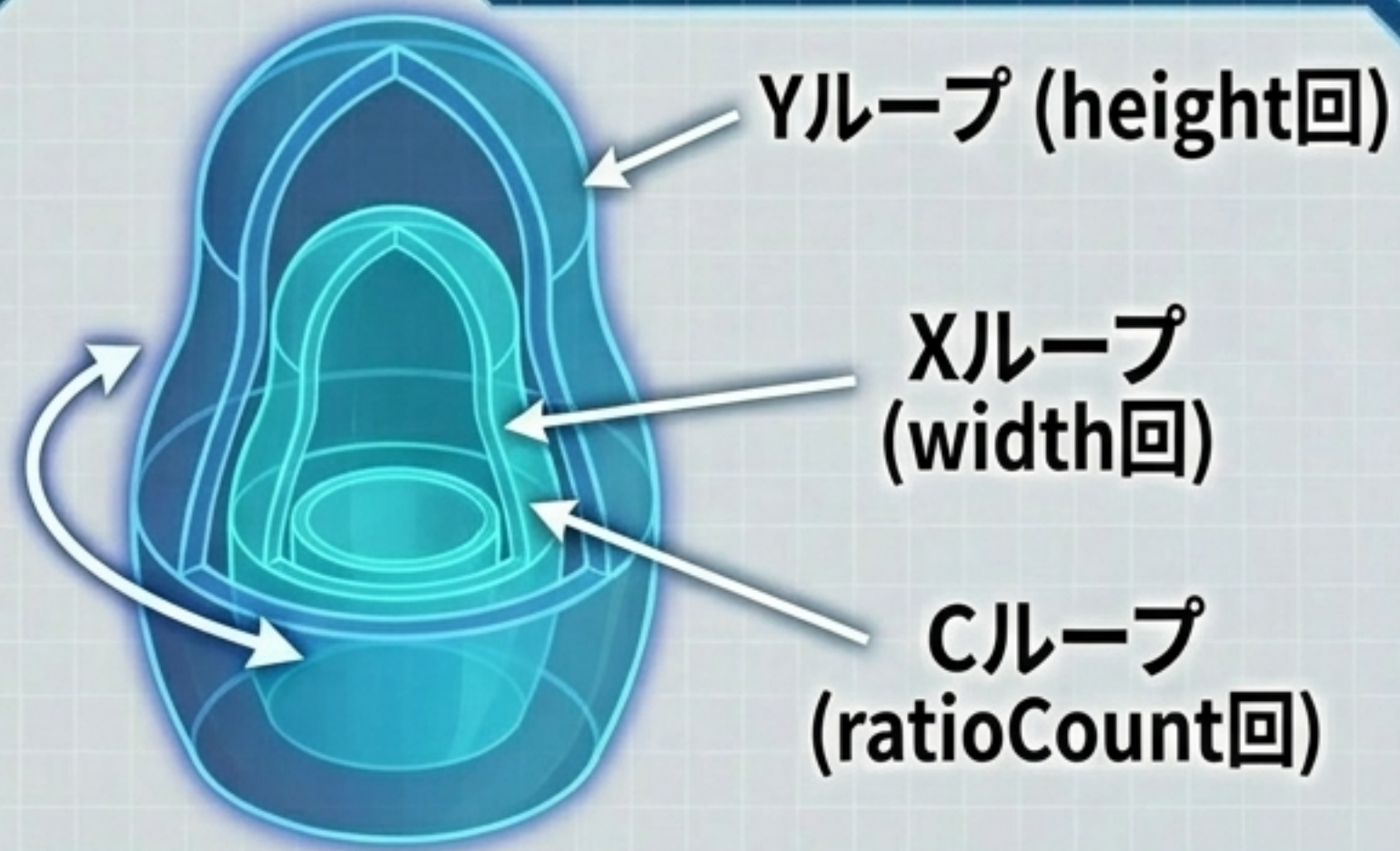
ある数が「偶数」とするとプログラムで判定するには?
「[対象] を [数字] で割った余り(mod) が 0」



【カのヒント】

変数 tx が 1, 2, 3... と増える時、
x を width, width-1... と減らす計算式は?
(例: width=10, tx=1の時に10になる式を検証)

設問4 [キ][ク] ヒント：実行回数の見積もり



【キのヒント】

3重ループの総実行回数。一番外側 × 真ん中 × 一番内側の最大回数を掛け合わせる。

```
if ( (px が 1以上) かつ  
      (px が width 以下) かつ  
      (py が 1以上) かつ  
      (py が height 以下) )
```

比較演算
($\geq$, $\leq$)

論理演算
(AND)

比較演算
($\geq$, $\leq$)

【クのヒント】

if文の中身を数えるだけ！
「～以上/以下」(比較演算)はいくつ？
「かつ」(論理演算)はいくつ？両方の合計を出そう。

アルゴリズム読解の3つの鉄則



1.

日本語とコードを繋ぐ

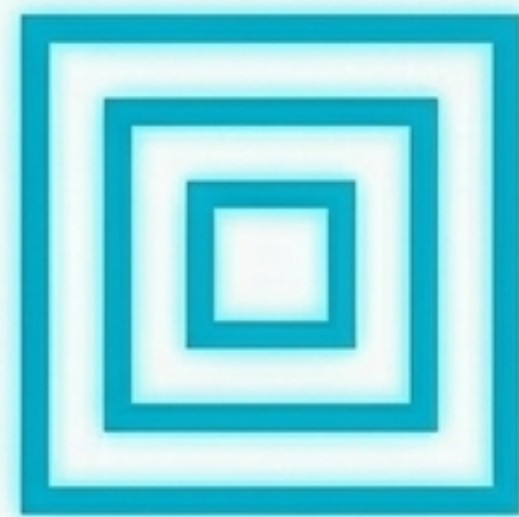
仕様書の文章がそのまま
コードの構造になる。
穴埋め前に、テキストと
if/for文を線で結びつける。



2.

変数を紙で追う

頭の中だけで計算せず、
配列の更新状態を必ずメモする。トレースこそが
確実な得点源。



3.

ループの階層を図解する

何が何度繰り返されるか、
箱の中に箱があるイメージ
で俯瞰して捉える。

Let the text guide your coding.