

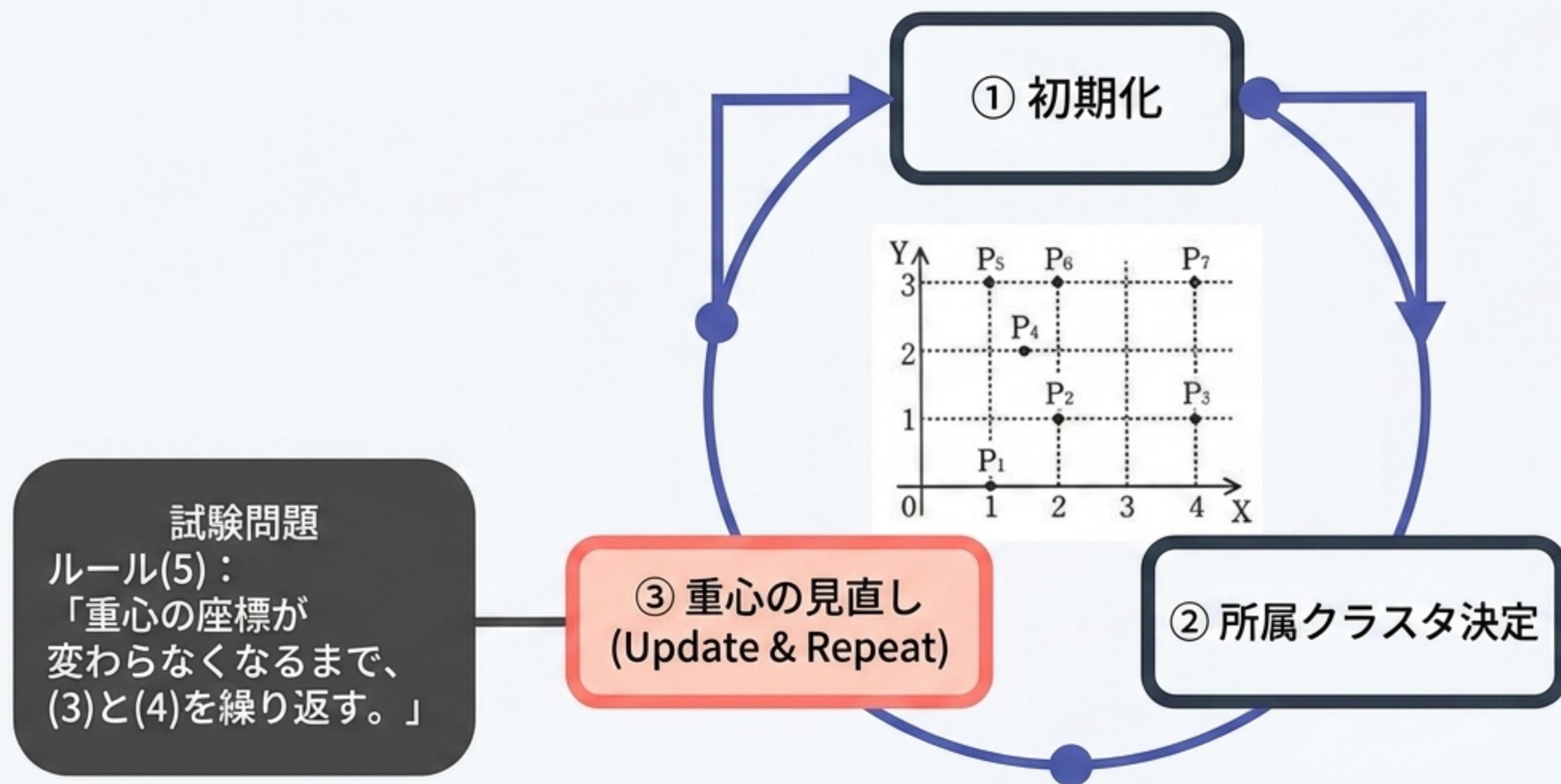
```

function clustering (N, K)
    MaxCount ← 100
    initial(K)
    for( s を 1 から N まで 1 ずつ増やす )
        for( t を 1 から K まで 1 ずつ増やす )
            data_length[t] ← data_dist(s, core[t])
        endfor
        cluster[s] ← min_index(data_length)
    endfor
    for( p を 1 から MaxCount まで 1 ずつ増やす )
        if( p が 1 と等しい )
            for(
                //1 回目の重心の計算
                令和3年 春期 応用情報技術者試験 午後問3
                一 設問2 [イ][ウ][エ] の論理構造を紐解く
                (※解答非公開・完全解説版)
            )
            endfor
        else
            for(
                //2 回目以降の重心の計算
            )
                if( gravity_x(t)と coordinate_x[t]が異なる //重心を修正する

```

k-means法 プログラムの解剖学

k-means法の全体像と本設問のスコープ



今回のターゲットである **[イ][ウ][エ]** の空欄はすべて、アルゴリズムの最終段階である「クラスタ重心の見直しと終了判定」のフェーズに集中しています。

プログラムの全体構造とターゲット領域

無限ループを防ぐための
MaxCount ループ内に、目的の
空欄が潜んでいます。
アルゴリズムが「変化の停止」を
どのように検知しているかを解読
することがクリアの条件です。

```
function clustering (N, K) //N はデータ数, K はクラス数
  MaxCount ← 100 //無限ループを防ぐため最大見直し回数を設定
  initial(K) //初期設定
  for( s を 1 から N まで 1 ずつ増やす ) //クラスを決定 (1 回目)
    for( t を 1 から K まで 1 ずつ増やす )
      data_length[t] ← data_dist(s, core[t])
    endfor
    cluster[s] ← min_index(data_length)
  endfor
  for( p を 1 から MaxCount まで 1 ずつ増やす ) //クラスを見直す
    if( p が 1 と等しい )
      for( ⊕[イ]⊕ ) //1 回目の重心の計算
        coordinate_x[t] ← gravity_x(t)
        coordinate_y[t] ← gravity_y(t)
      endfor
    else
      ⊕[ウ]⊕
      for( ⊕[イ]⊕ ) //2 回目以降の重心の計算
        if( gravity_x(t) と coordinate_x[t]が異なる //重心を修正する
          又は gravity_y(t) と coordinate_y[t]が異なる )
          coordinate_x[t] ← gravity_x(t)
          coordinate_y[t] ← gravity_y(t)
          flag ← 1
        endif
      endfor
      if( ⊕[エ]⊕ ) //終了して抜ける
        return
      endif
    endif
  // END OF FULLY VISIBLE SECTION
  for( s を 1 から N まで 1 ずつ増やす ) //近い重心を見つける
    for( t を 1 から K まで 1 ずつ増やす )
      grav_length[t] ← grav_dist(s, t)
    endfor
    ⊕[オ]⊕
  endfor
endfor
endfunction
```

条件分岐の解剖：1回目 vs 2回目以降

1回目の計算

```
if ( p が 1 と等しい )  
  for ( [イ] )
```

2回目以降の計算

```
else  
  [ウ]  
  for ( [イ] )  
    ...  
    if ( [エ] )
```

全く同じロジック

空欄 [イ] は両方の分岐に出現します。つまり、初回でも2回目以降でも「全く同じ目的」で回るループであることを意味しています。

[イ] の解説：何を回すループなのか？

```
for ( [イ] )  
  coordinate_x[t] <- gravity_x(t)  
  coordinate_y[t] <- gravity_y(t)  
endfor
```

tooltip content

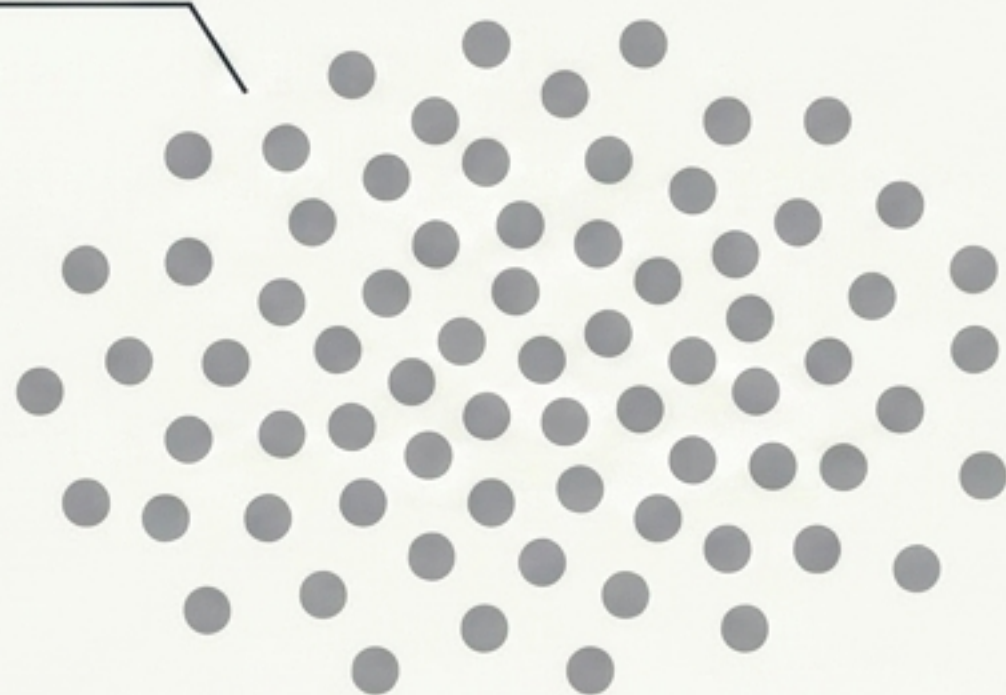
クラスタsの重心Gsを求め、その座標を返す。

ループ内で呼び出されている変数と関数に注目します。これらはすべて「クラスタの重心」を扱うための配列・関数群です。変数 t は、ある特定の対象をカウントアップするためのインデックスとして機能しています。

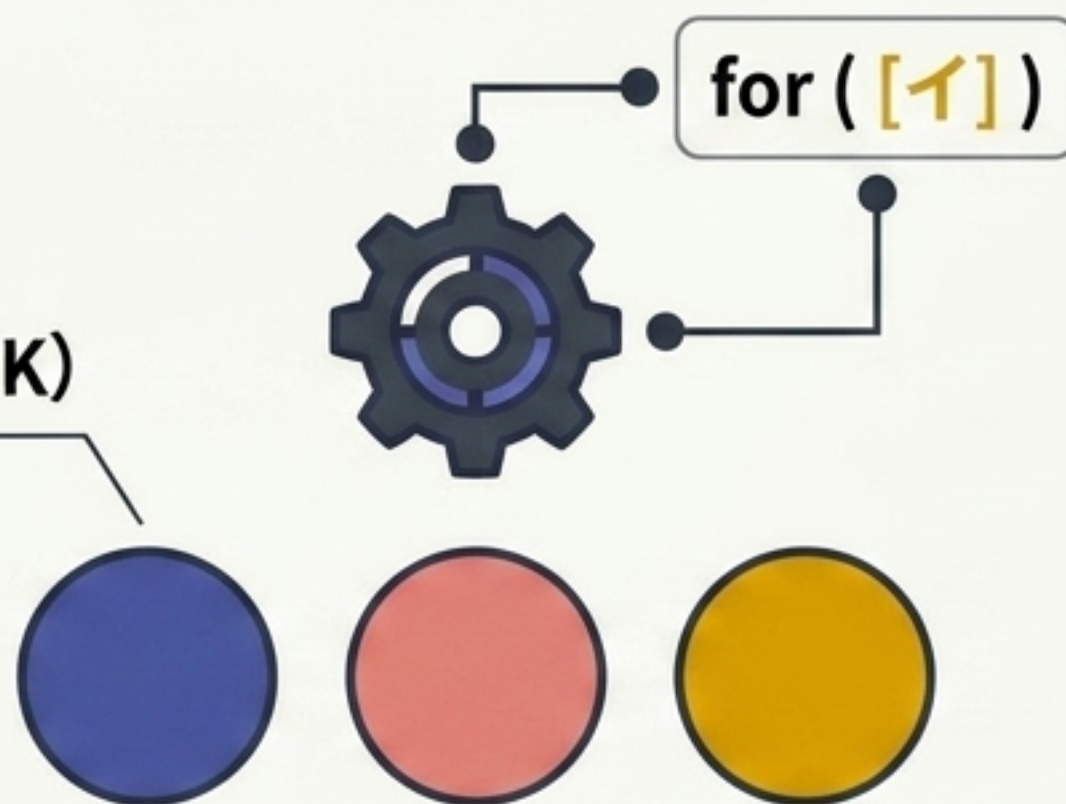
[イ] へのヒント：処理すべき総数は？

```
function clustering (N, K)
```

データ数 (N)



クラスタ数 (K)



全てのクラスタの重心を計算（または修正）する必要があります。データ総数は N 、クラスタ総数は K 。では、クラスタを対象とするインデックス変数 t は、どこからどこまでカウントアップすべきでしょうか？

[ウ]と[エ]を解く鍵：「フラグ（状態変数）」の設計パターン



プログラムが「変化があったかどうか」を記憶するためには、フラグ変数を用いたこの3ステップの設計パターンが不可欠です。本アルゴリズムも、この普遍的なパターンに完全に従っています。

[ウ] の解説：変化を検知する前の「準備」

```
else
```

```
[ウ]
```

```
for ( ... )
```

```
  if (gravity_x(t) と  
      coordinate_x[t] が異なる... )
```

```
    ...
```

```
    flag <- 1
```

```
  endif
```

変化あり

flag <- 1 は、重心の座標が「異なる (= 変化があった)」場合にのみ実行されます。

だとすれば、クラスタを一つも確認する前（ループに入る直前）である [ウ] の位置では、flag は「まだ変化がない」ことを示す状態にリセットされている必要があります。

[エ] の解説：アルゴリズムの終了条件

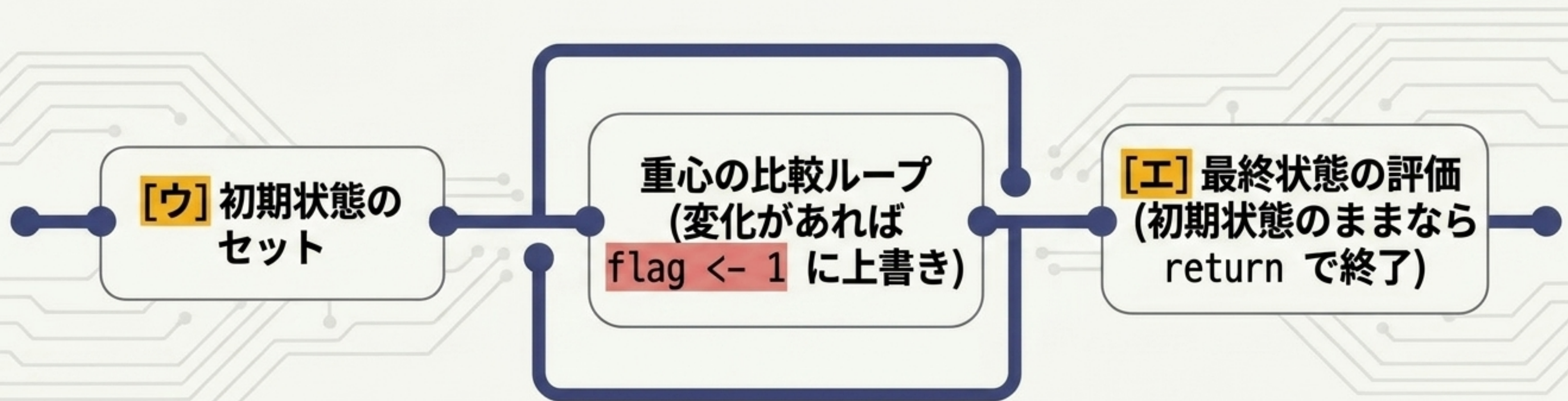
ルール(5)：重心の座標が**変わらなくなる**まで繰り返す



```
if ( [エ] )  
    return
```

return は関数を直ちに終了させる命令です。全てのクラスタをチェックしたループの後、フラグが「どのような状態」を保っていれば、変化がなかった（＝分類が完了した）と断定して処理を抜け出せるのでしょうか？

フラグ変数のライフサイクル (総括)



[ウ]で初期化し、ループ内で状態を監視し、[エ]で評価する。この一連の役割と「変化が止まる」ことの数学的意味を紐付けることで、ソースコードが語る論理構造が完全に可視化されます。解答のコードは、このロジックを正確に記述したものになります。