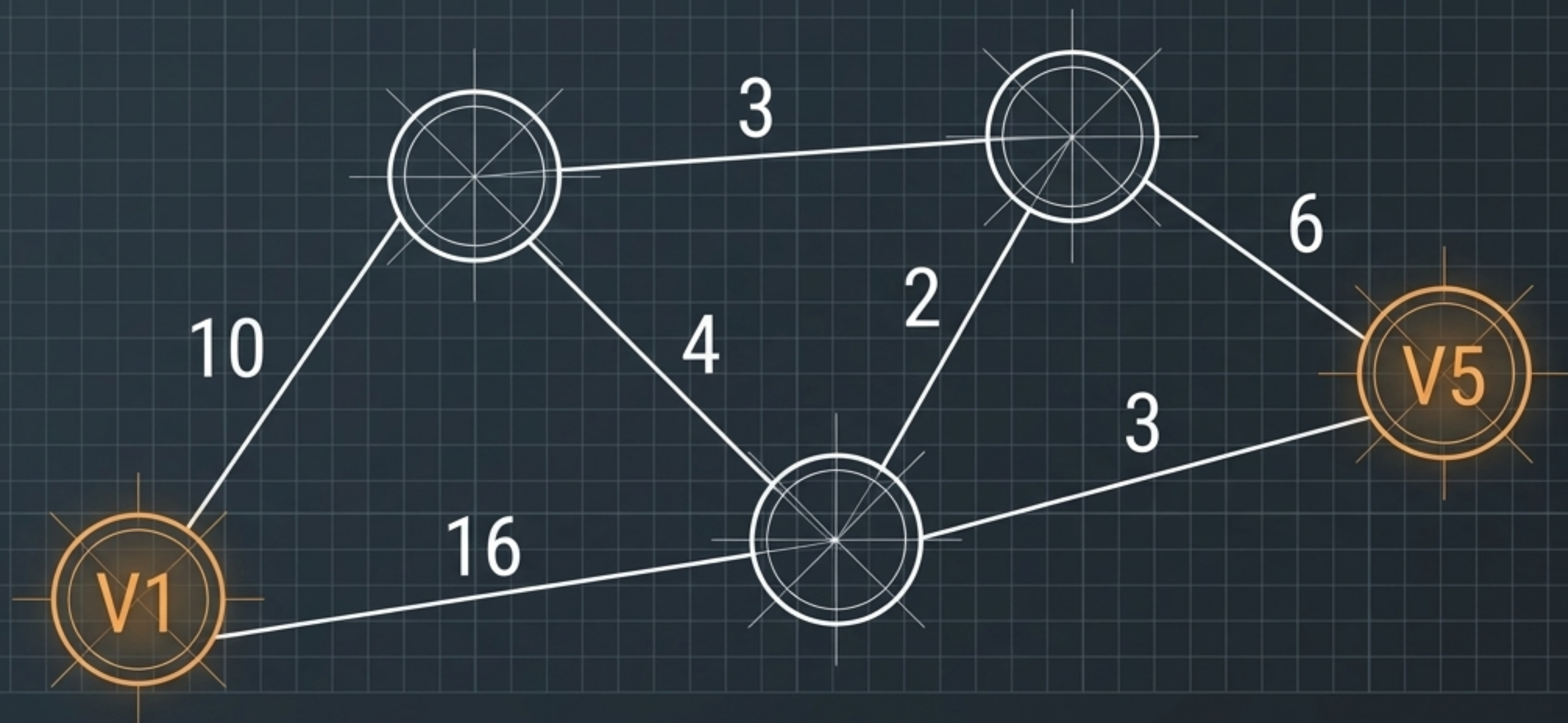




応用情報技術者試験 午後問3 完全攻略ガイド

ダイクストラ法を「暗記」ではなく「理解」で解くための極秘ヒント集

※本資料に解答は一切記載されていません。あなたの論理的思考力を呼び覚ますためのガイドです。

攻略のルール (The Mission)

本資料が提供するのは「答え」ではなく「圧倒的なヒント」です。
探偵のように手がかりを集め、自らの頭で論理を組み立ててください。

表1 図1の例における最短距離を求める手順と

適用回数	更新起点ノード	最短距離が確定していない, 更新起点ノードに隣接するノード	最短距離
1回目	V1 <0>	V2 <10>, V3 <16>	V2 <10>
2回目	V2 <10>	V3 <14>, V4 <13>	V3 <14>
3回目	V4 <13>	V5 <19>	V3 <14>
4回目	V3 <14>	V5 <[ア]>	V5 <[ア]>
5回目	V5 <[ア]>	-	-

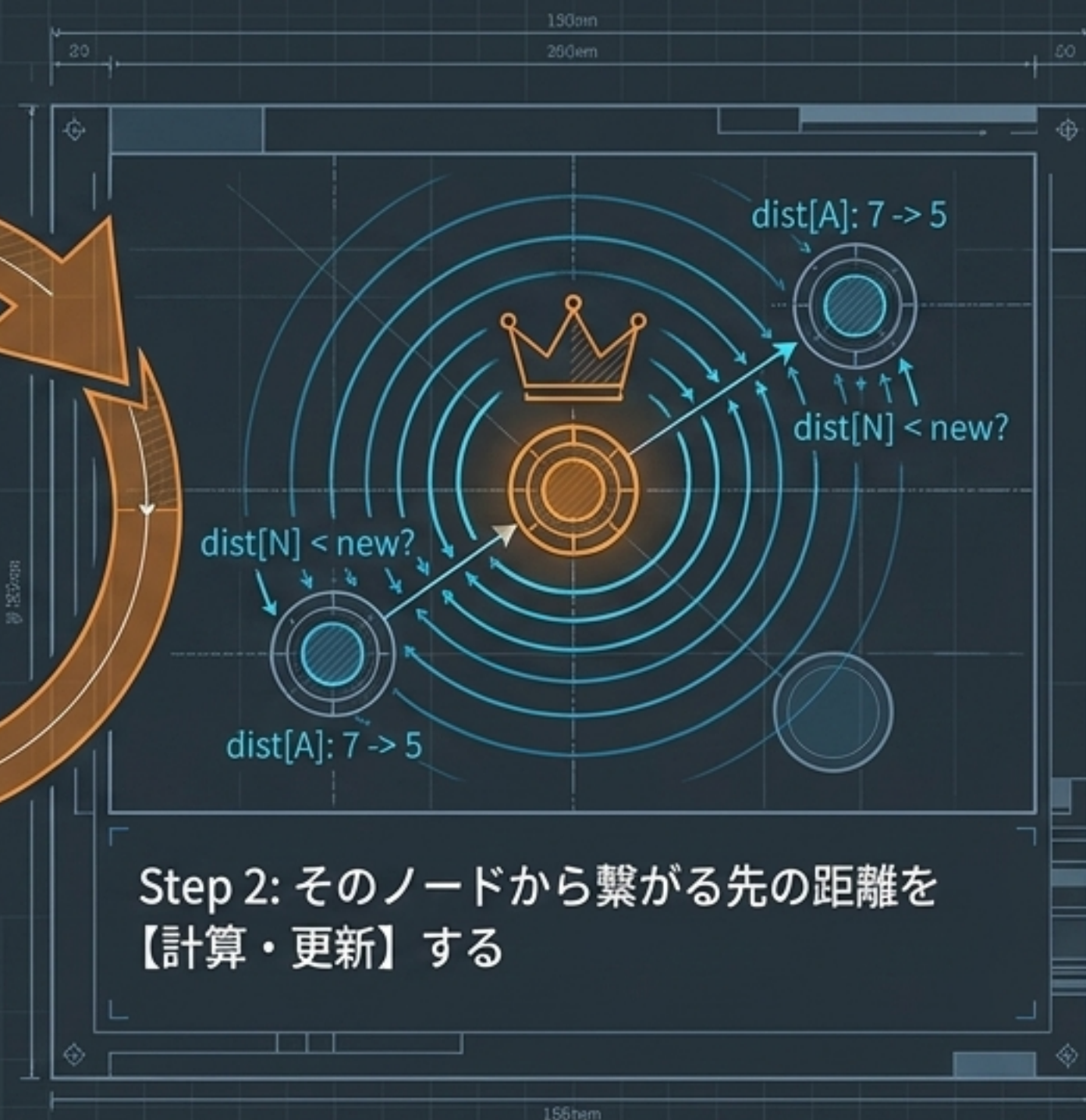
Mission 1: 設問1 (トレース力の試練)
- アルゴリズムの動きを手動で追跡し、
[ア]の数値を導き出せ。

```
2: while (true)
3:   minDist ← INF
4:   for (k を1 から N まで1 ずつ増やす)
5:     if (done[k]が0 かつ [イ] )
6:       minDist ← [ウ]
7:       curNode ← k
8:     endif
9:   endfor
10:  done[curNode] ← 1
11:  if (curNode が GOAL と等しい)
12:    return dist[curNode]
```

Mission 2: 設問2 (コード翻訳力の試練)
- 日本語の仕様をプログラム言語に翻訳し、
[イ][ウ][エ]のロジックを完成させよ。

ダイクストラ法のコアドクトリン (The Core Concept)

迷路の最短ルートを探す方法は、実は驚くほどシンプル。
以下の2ステップを繰り返すだけです。



Mission 1 - 設問1のターゲット確認

探索適用回数「4回目」と「5回目」のV5の始点ノード距離
[ア]を導き出すことが最初のミッションです。

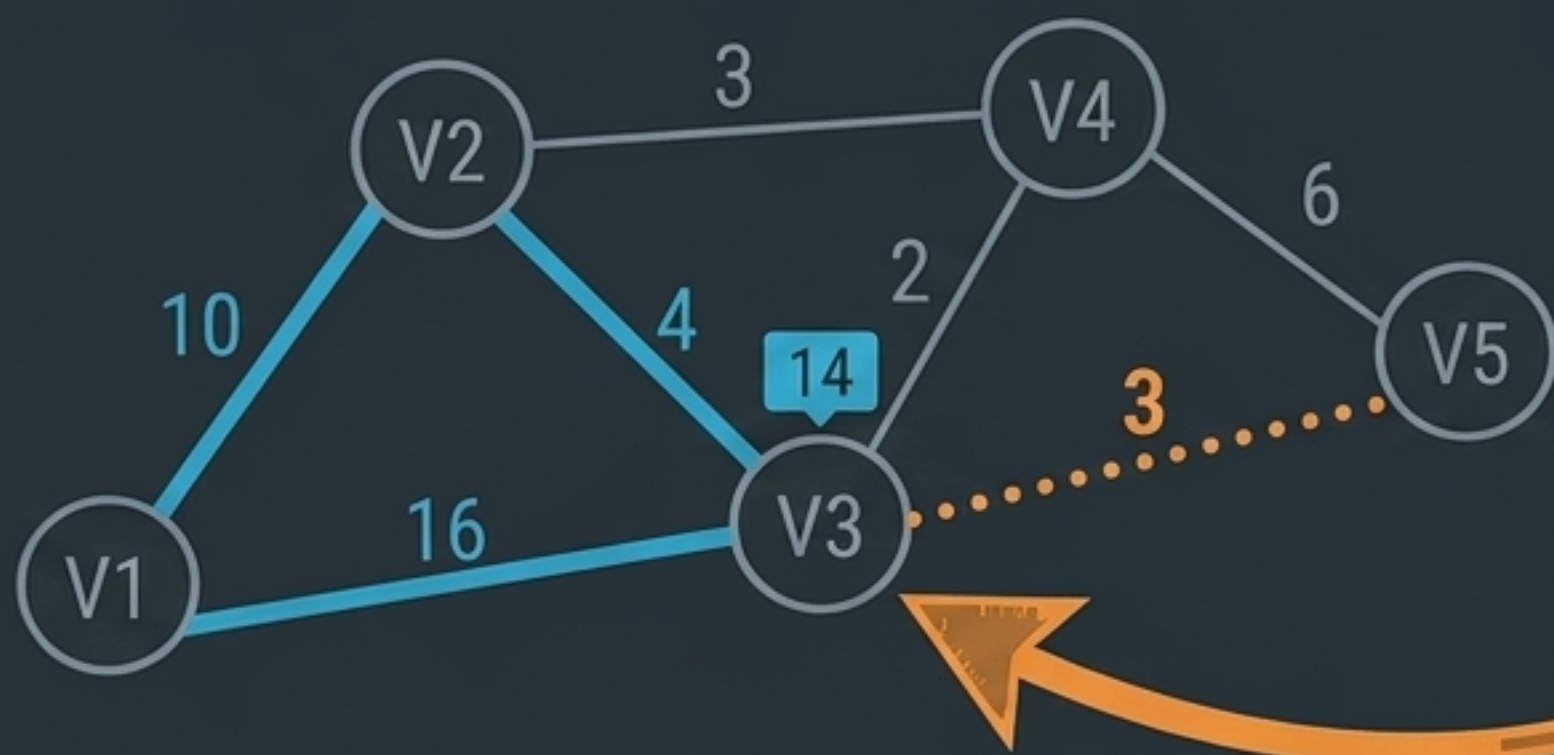
ヒント：3回目の時点では、V5の暫定距離は「19」でした。
これが4回目でどう変化するかを見極めましょう。

探索適用回数	更新起点ノード	最短距離が確定していない，更新起点ノードに隣接するノード	最短距離が確定していないノード
1 回目	V1 <0>	V2 <10>, V3 <16>	V2 <10>, V3 <16>, V4 <INF>, V5 <INF>
2 回目	V2 <10>	V3 <14>, V4 <13>	V3 <14>, V4 <13>, V5 <INF>
3 回目	V4 <13>	V3 <14>, V5 <19>	V3 <14>, V5 <19>
4 回目	V3 <14>	V5 < ア >	V5 < ア >
5 回目	V5 < ア >	—	—

注記 1 INF は，定数で十分大きい数を表す。

注記 2 <>内の数値は，当該ノードの始点ノード距離を表す。

ヒント1 - V5への道のりはどこから来る？ (Visualizing the Route)



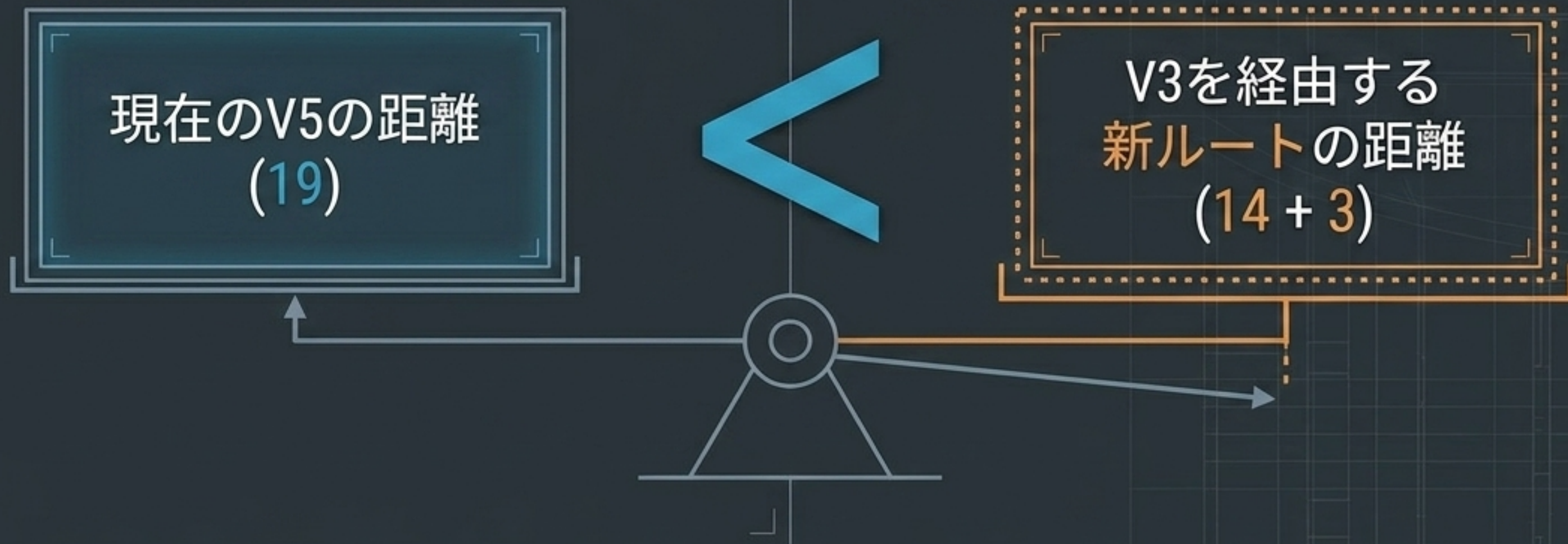
探索適用回数	更新起点ノード	最短距離が確定していない, 更新起点ノードに隣接するノード	最短距離が確定していないノード
4 回目	V3 <14>	V5 < ア >	V5 < ア >

4回目の探索時、「今立っている場所（更新起点ノード）」は V3 です。

始点V1からV3までの最短距離は 14 で【確定】しています。

あなたは今、V3にいます。ここからV5へ向かうルートが見えますか？

ヒント2 - 「更新」の条件を思い出す (The Update Math)



「新しい距離」が「現在の距離」より小さければ、値は上書き（更新）されます。さあ、計算して[ア]を導き出しましょう。

Mission 2 – 設問2のターゲット確認

プログラム全体を漫然と読んでは
いけません。

このコードは、大きく2つの

「役割（ブロック）」に分かれています。**[イ][ウ]**は「最小値探し」、**[エ]**は「隣接ノードの距離更新」のブロックに存在します。

```

1 #include <start.h>
2 #include <stars.h>
3 #include <start.h>
4
5 int main(j) {
6
7     int *ource_read = 0;
8     int int dezmot = [5];
9     for (vresult = 1)
10         set.printf("abtaathk{+});
11
12     enlf8. = anbrasth;
13
14     for (nnz = 0; is:4+) {
15         int ss = 0;
16         for (int j= 0; lengthk,j
17             shors ar:aisra{[j, 2
18             return nhal+ = 00;
19         }
20     }
21
22     return reuat;
23
24     print(dtoc;
25     for (int k = 0; set:i++) {
26         if (length = 0{f{ i+ < 1
27             return ariasson[i],
28     }
29 }

```

Block A: チャンピオン決定戦 ([イ][ウ])

Block B: 周辺ルートの開拓 ([エ])

ヒント3 – 日本語仕様からコードへのマッピング (Mapping the Specs)

情報処理技術者試験のアルゴリズム問題の鉄則。それは「問題文の日本語手順」と「プログラム」が必ず1対1で対応していることです。

The Specs

手順①「始点ノード距離が最小のノードを選ぶ」

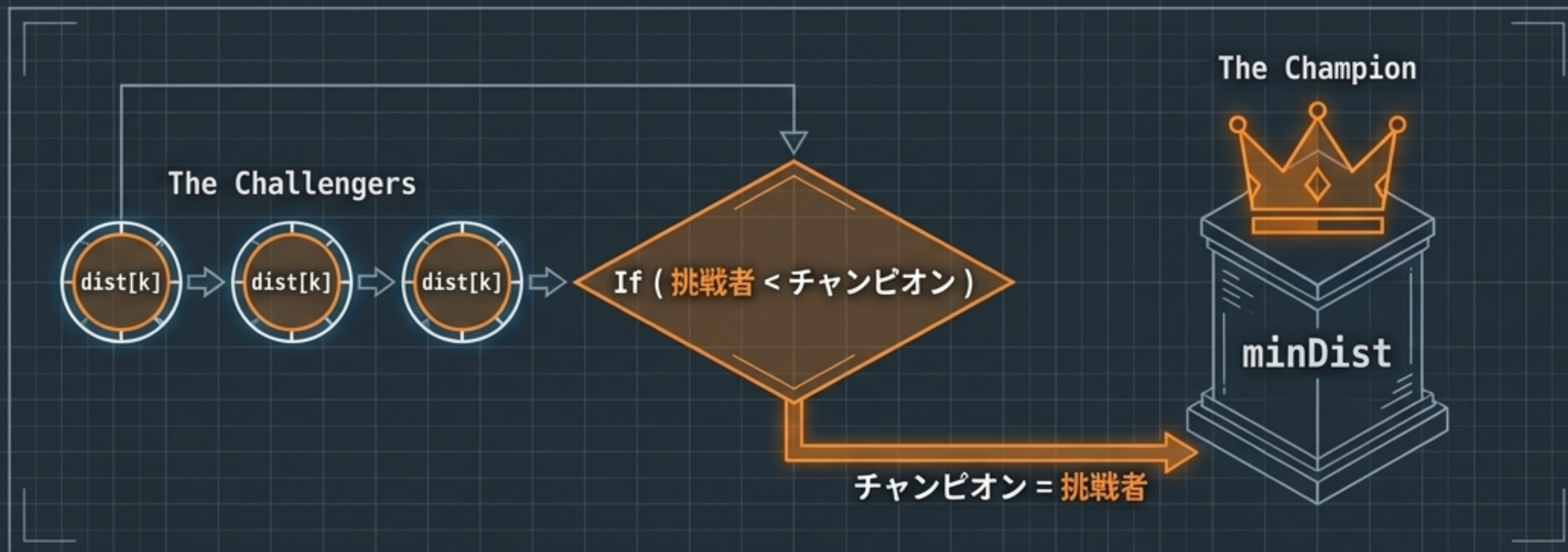
手順③「更新起点ノードを経由した
手順③「更新起点ノードを経由した場合の距離を計算し…更新する」

The Code

```
14:  for (kを1からNまで1ずつ増やす)
15:    if (done[k]が0 かつ イ)
16:      minDist ← ウ
17:      curNode ← k
18:    endif
19:  endfor
... (lines 20-23 omitted) ...
24:  for (kを1からNまで1ずつ増やす)
25:    if (エ が dist[k]より小さい かつ done[k]が0)
26:      dist[k] ← エ
27:    endif
28:  endfor
```

ヒント4 - 空欄[イ][ウ]の攻略法（最小値探しロジック）

プログラミングにおける「最小値探索」は、チャンピオン勝ち抜き戦です。



[条件 イ]:

挑戦者（現在のノード）が、現在のチャンピオンより強い（値が小さい）かを判定する条件式が入ります。

[処理 ウ]:

挑戦者が勝った場合、チャンピオンの座（最小値）が挑戦者の値に上書き（交代）されます。

ヒント5 – 変数の役割を整理せよ (Variable Diagnostics)

先ほどの「勝ち抜き戦」のロジックを、この変数たちを使って翻訳してください。

`dist[k]`

各ノードkまでの暫定距離
(挑戦者)

`minDist`

探索中の最小値
(チャンピオン)

`curNode`

更新起点ノード

空欄[イ]への問い: 「挑戦者」が「チャンピオン」より小さい ($<$) ことを数式で書くと?
空欄[ウ]への問い: 新しいチャンピオンとして「挑戦者」の値を代入するには?

ヒント6 – 空欄[エ]の攻略法（距離の更新ロジック）

```
25: if ( [エ] が dist[k] より小さい かつ done[k]が0 )
```

現在のルートの距離

新しいルートの距離

25行目の if 文は、まさに先ほどあなたが設問1（手動トレース）で行った計算をプログラムにやらせている部分です。

「今までのルート」よりも「新しいルート」の方が近いですか？ と聞いています。

つまり、[エ]には「更新起点ノードを経由した場合の、新しい距離を求める計算式」が入ります。

ヒント7 - 距離計算のパーツを組み立てる (Assembling the Math)

設問1で「 $14 + 3$ 」を計算したあのロジックを、下の変数パーツを組み合わせて汎用的な数式に組み立ててください。

[始点から今いるノードまでの確定距離] + [今いるノードから隣のノードまでのエッジ距離]

部品辞書 (Parts Dictionary)

dist[x]

始点からノードxまでの距離

edge[x, y]

ノードxとノードyの間の距離

curNode

今いるノード

k

隣のノード

Hint: 「今いるノード」を表す変数はどれですか? 「隣のノード」を表す変数はどれですか?
これらを **dist** と **edge** の **[]** の中に当てはめて足し合わせましょう。

最後のチェックポイント (Final Checklist)

- ☒ グラフ上でV1からV3を経由してV5へ向かう距離を自力で計算できた（設問1 [ア]）。
- ☒ dist配列とminDist変数を使って、最小値を更新するロジックを作れた（設問2 [イ][ウ]）。
- ☒ 「今いるノードまでの距離」と「ノード間のエッジ距離」を足し合わせる式を作れた（設問2 [エ]）。

すべてのチェックが埋まったなら、あなたの導き出した答えはIPAの模範解答と完全に一致しているはずです。
アルゴリズムの「設計図」は、もうあなたの頭の中にあります。