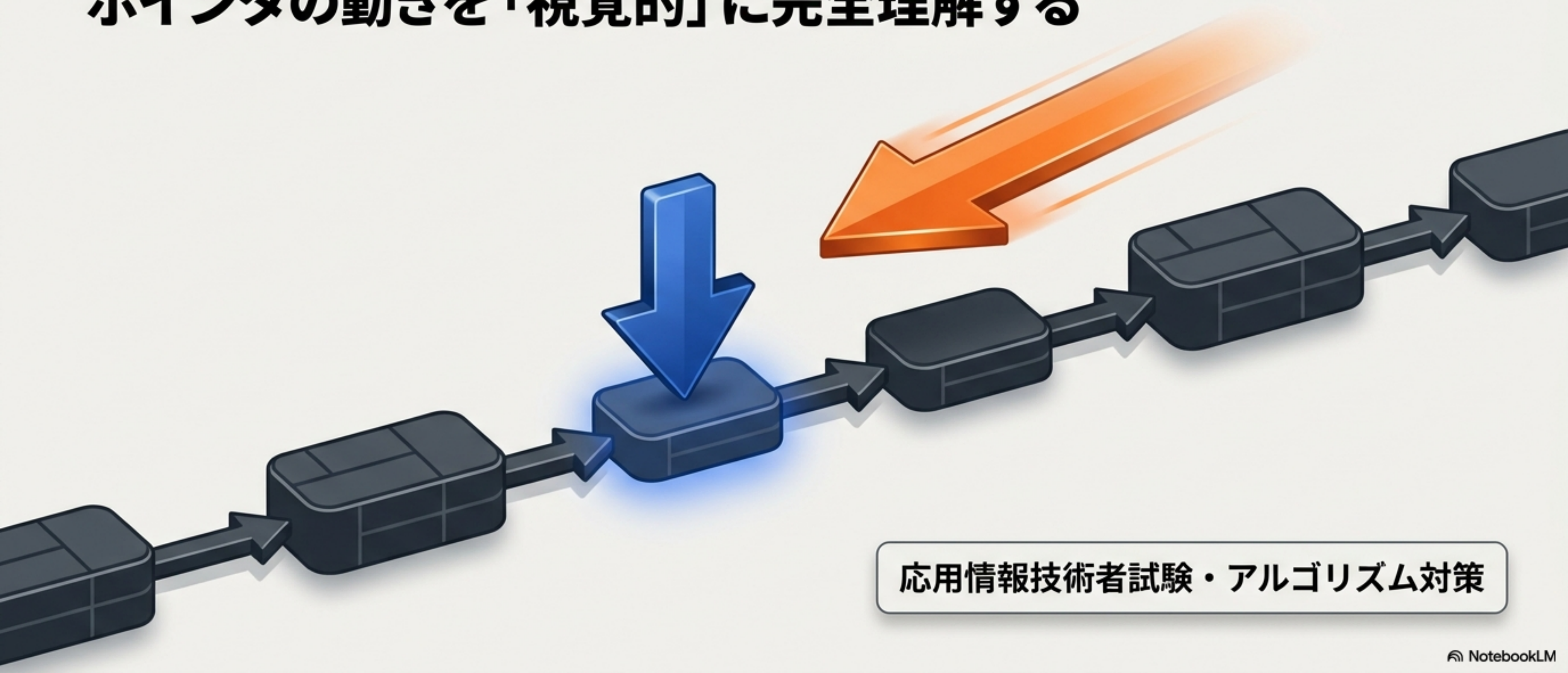


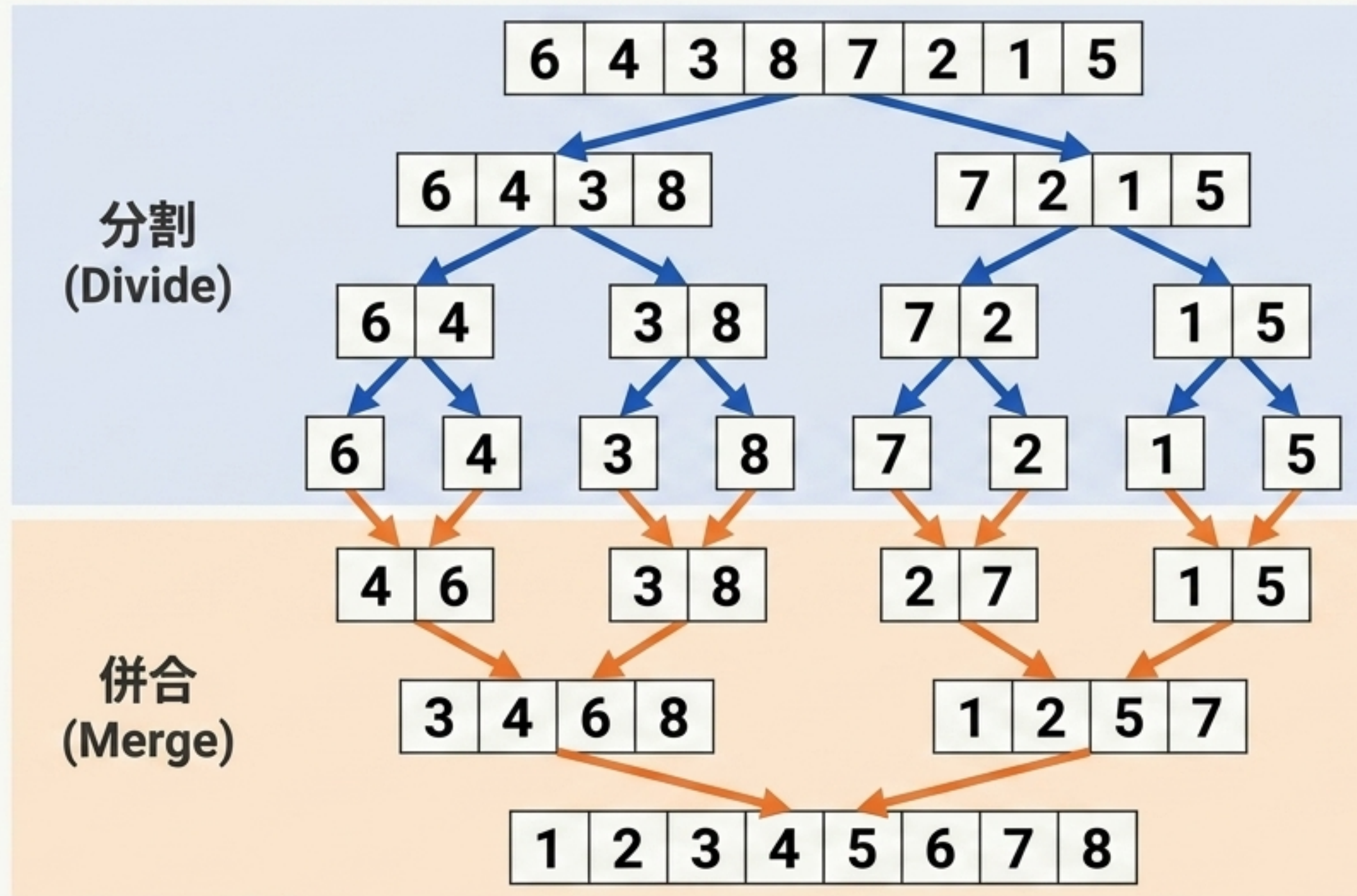
マージソートの完全解剖：連結リスト編

ポインタの動きを「視覚的」に完全理解する



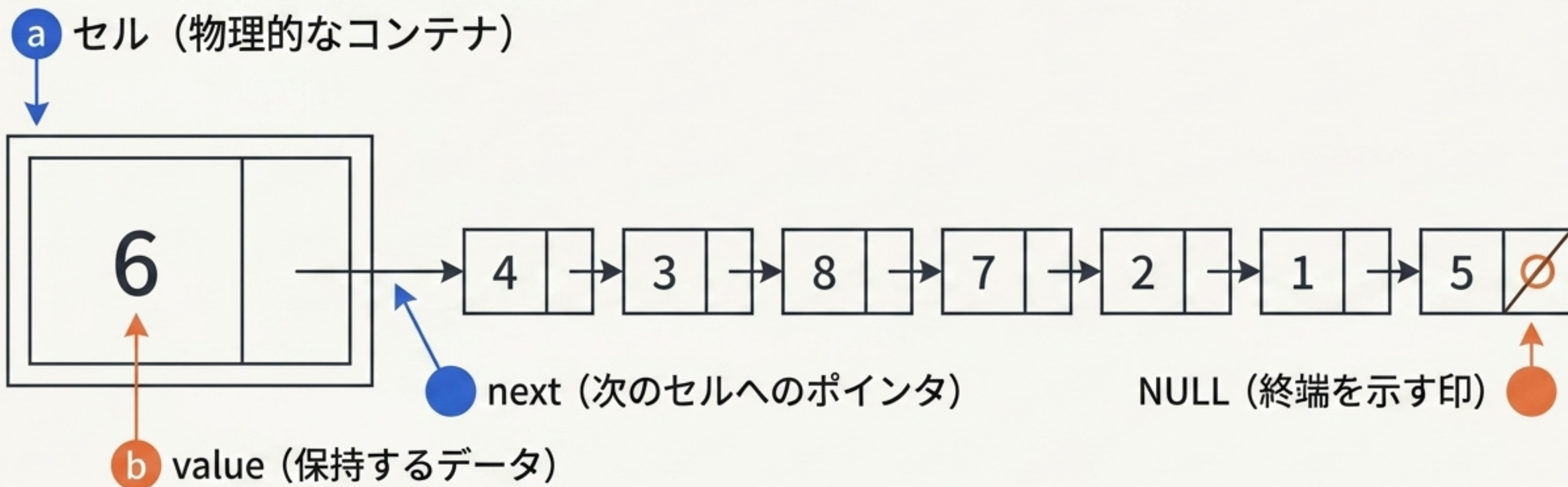
応用情報技術者試験・アルゴリズム対策

マージソートの全体像：「分割」と「併合」



データ列を「1個」になるまで分割し、順番を整えながら「併合」していく。
これがマージソートの基本原則です。

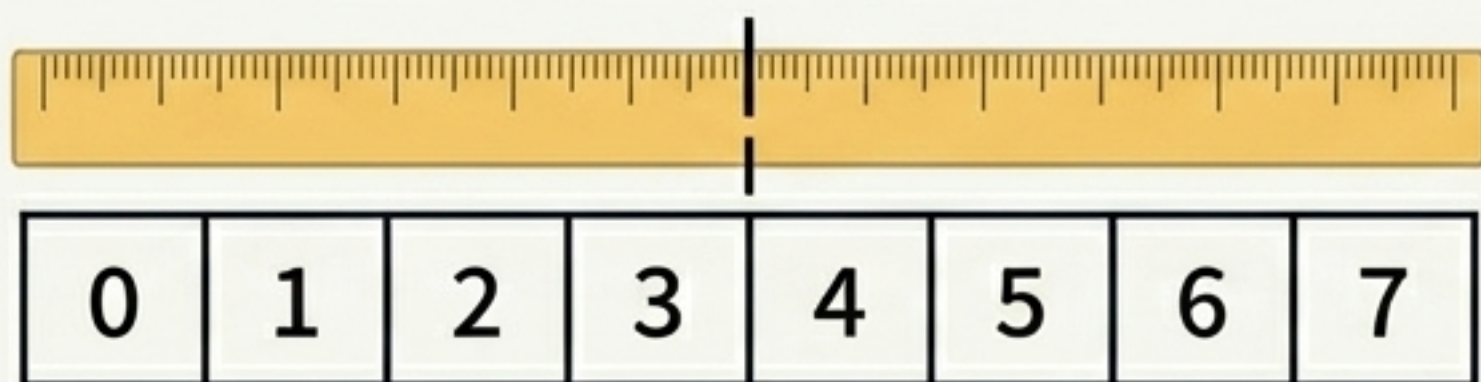
連結リストの構造：「アセンブリ・ライン」



配列とは異なり、全体の長さ（要素数）は事前にはわかりません。
ポインタをたどって初めて次の要素がわかります。

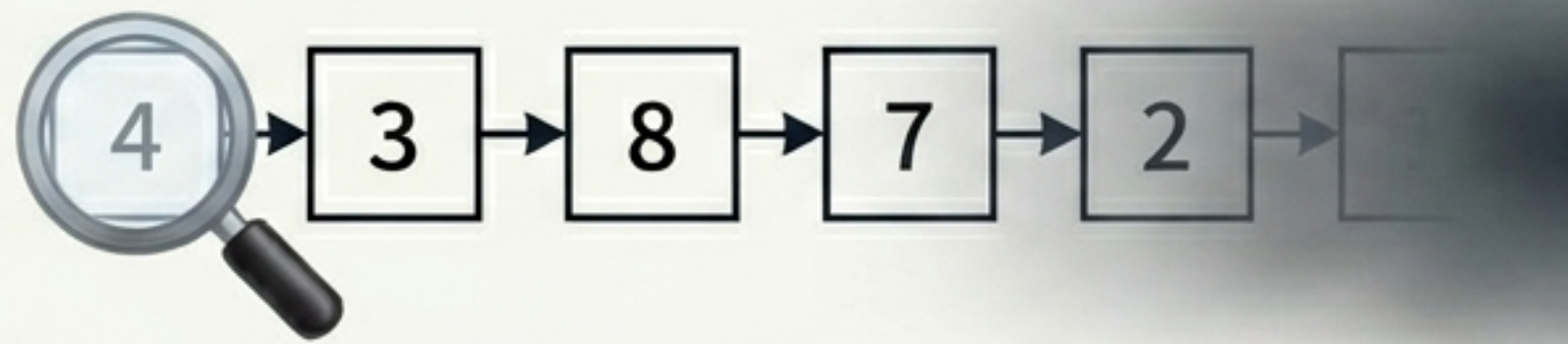
分割の壁：「真ん中」はどこにある？

配列の場合 (The Illusion)



✓ 長さが確定しているため「長さ ÷ 2」で一瞬で中央を特定可能。

連結リストの場合 (The Reality)



⚠ 問題: リストを半分に分割したいが、連結リストは「要素数」が事前にはわからない。

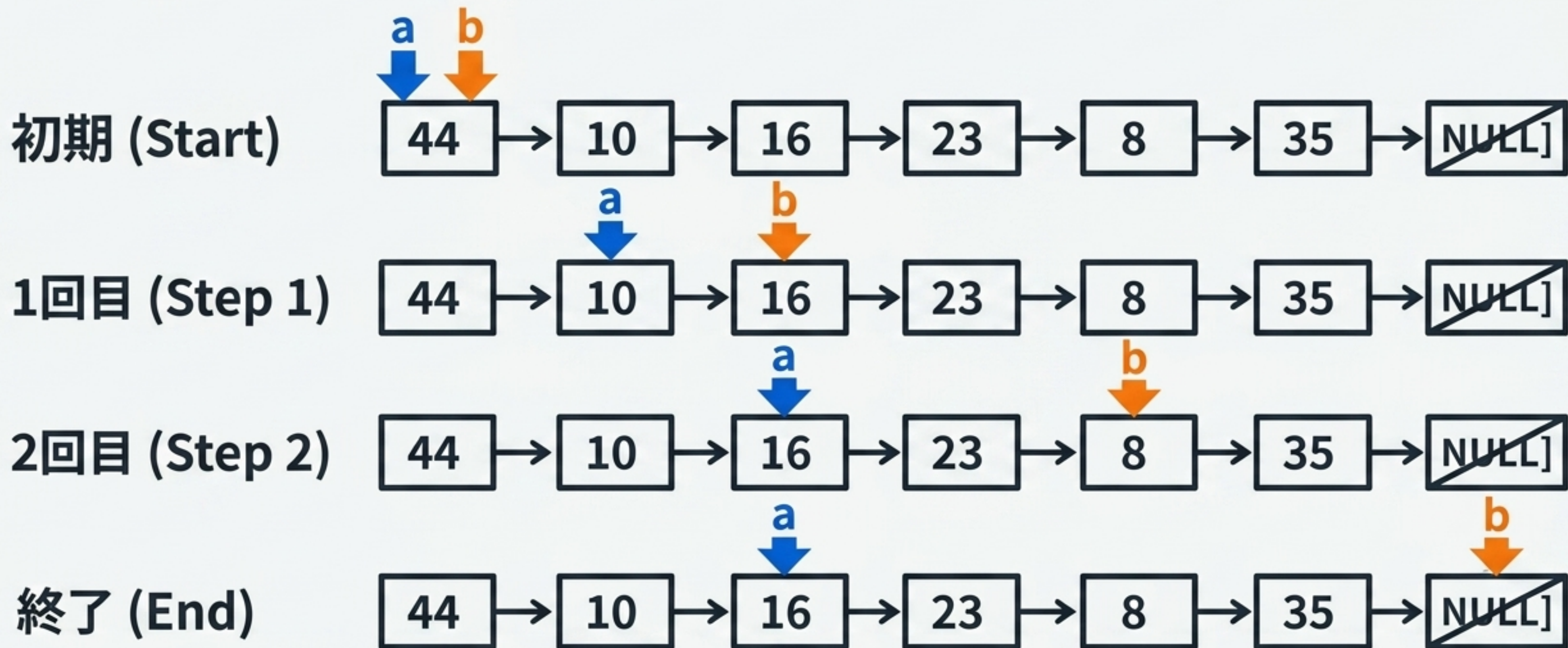
結論: リストの末尾 (NULL) まで探索しないと、全体の長さが確定しないため、単純に「半分」を計算することは不可能です。

解決策：「ウサギとカメ」の2倍速ポインタリレー



「b」が2倍の速さでリストの末尾（NULL）に到達したとき、1倍速の「a」は必然的にリストの【中央】に取り残されます。これが長さ不明のリストを半分に割るエレガントな手法です。

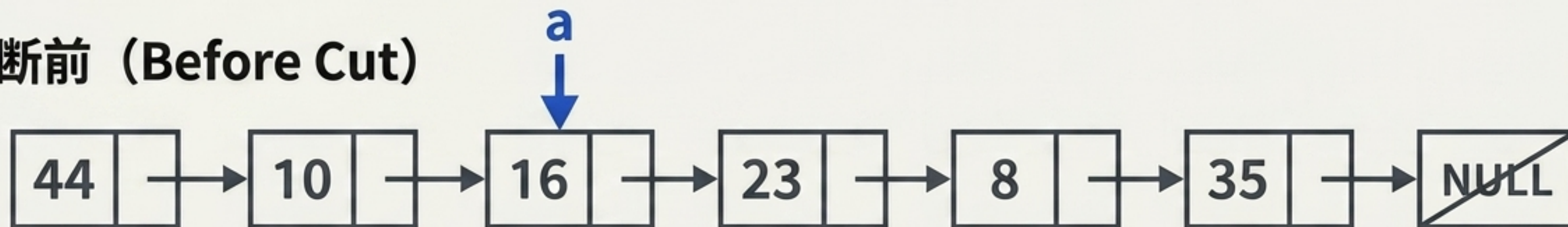
シミュレーション：ポインタの軌跡



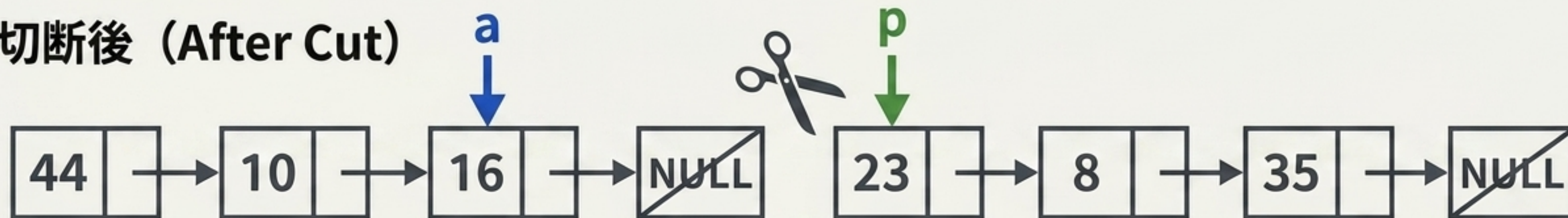
b が末尾に到達した瞬間、a はリストの中央を指している。

リストの切断：「前半」と「後半」の分離

切断前 (Before Cut)



切断後 (After Cut)



$p = a \rightarrow next$
(後半の先頭を確保)

$a \rightarrow next = NULL$
(前半の終端を確定)

設問1 ヒントタイム：空欄 [ア]

```
while ( [ ア ] )
```

目的 a と b を進め続けるループの「停止条件」は何か？

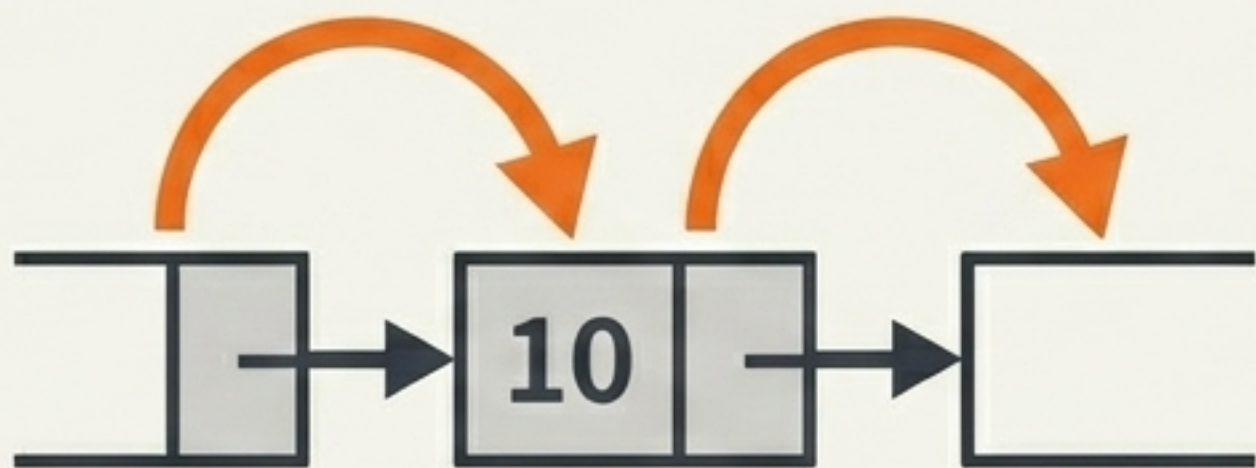
状況 ウサギ役のポインタ b が先頭を切って走り、壁 (NULL) にぶつかったら探索終了。

結論 したがって、ループを「継続」するための条件は、先頭を走る b がまだ壁に到達していないこと。

ヒント：「b がリストの端 (NULL) に到達していない間」という条件が入ります。

設問1 ヒントタイム：空欄 [イ]・[ウ]

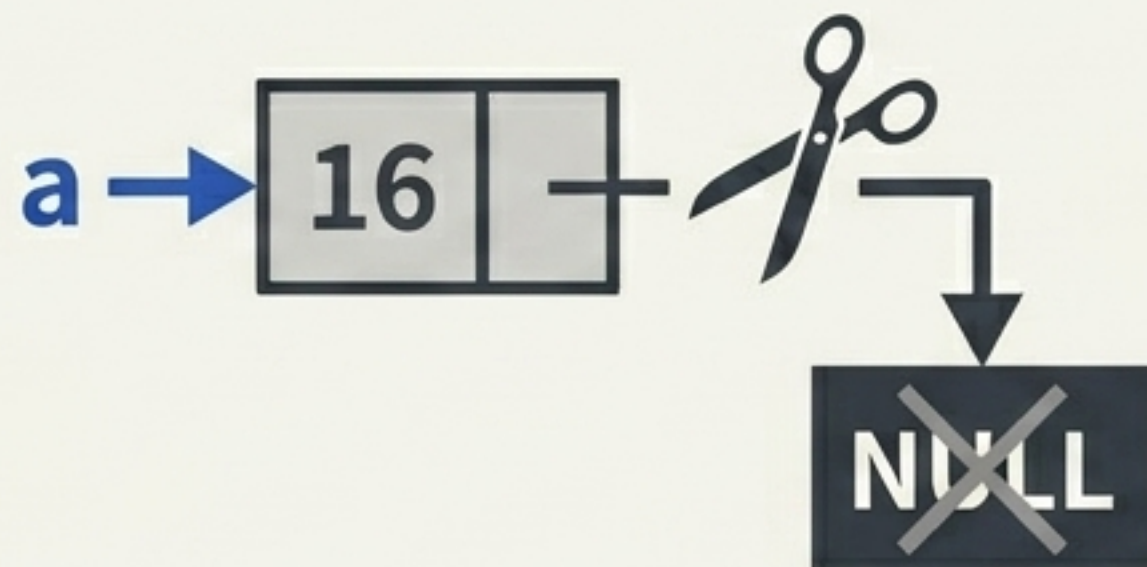
空欄 [イ] - 2倍速の実現



コード内には既に `b = b->next` が1つ存在します。

ヒント：2倍速で進むには、もう1段階進める必要があります。代入文はどうなるでしょうか？

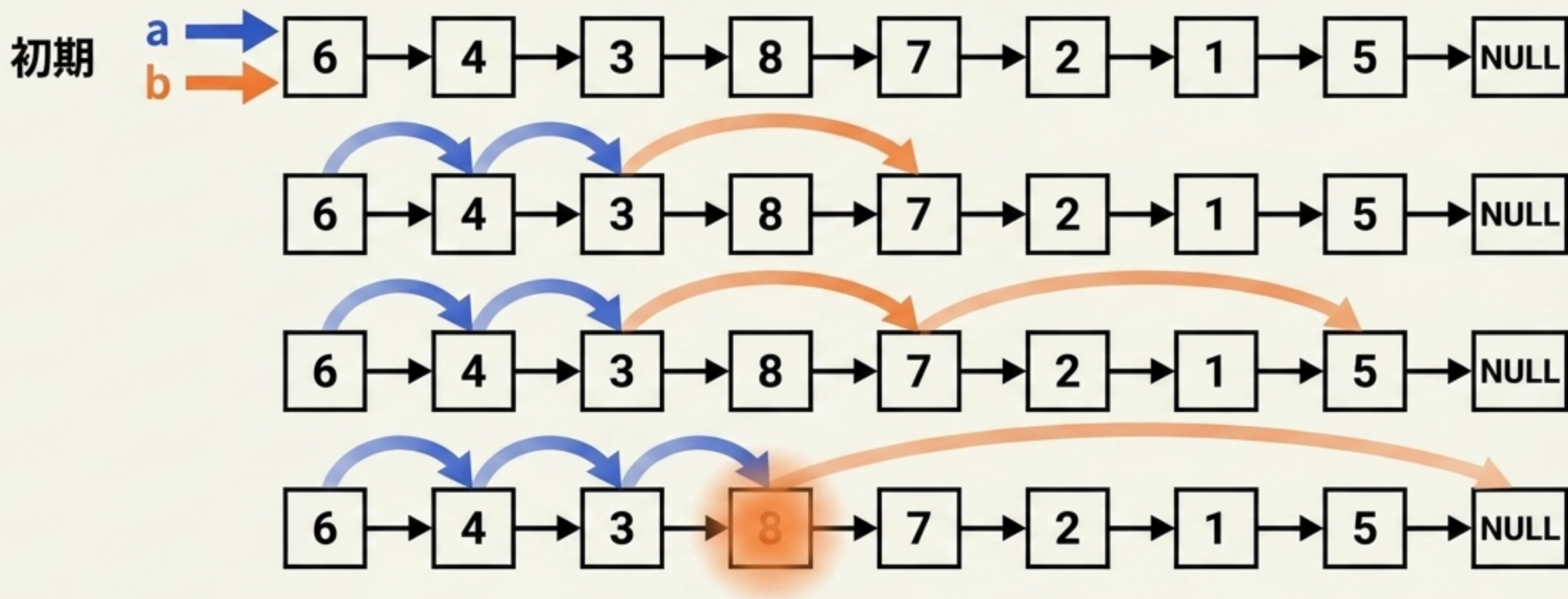
空欄 [ウ] - リストの分断



a は前半リストの最後の要素を指しています。

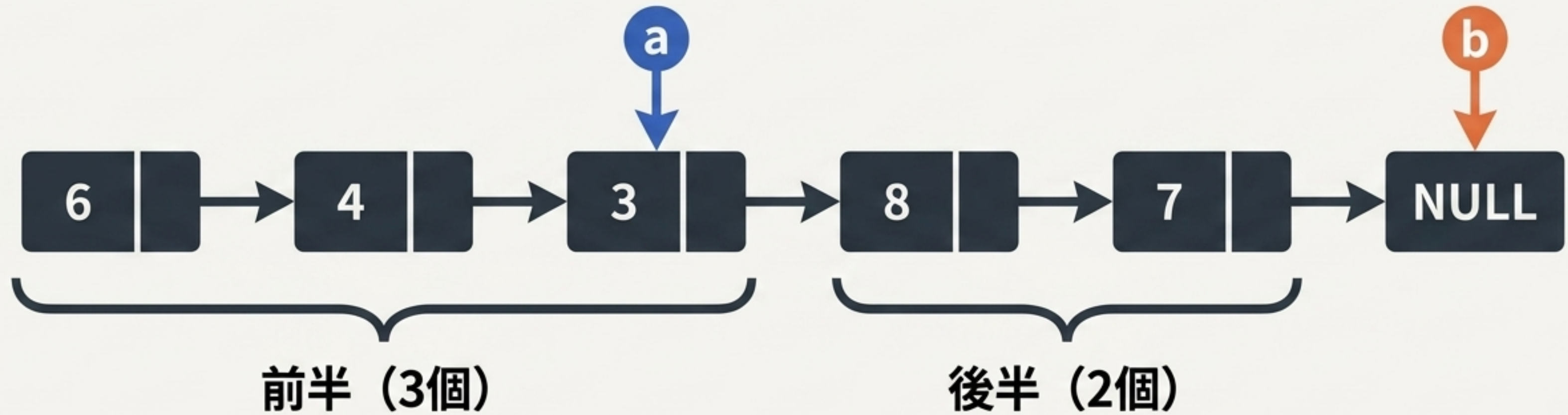
ヒント：前半リストの最後である a の次に「終端 (NULL)」を設定するコードが入ります。

設問1(2) ポインタ追跡ワークスペース



bがNULLに到達するまで、a（1歩）とb（2歩）の軌跡をたどってみましょう。

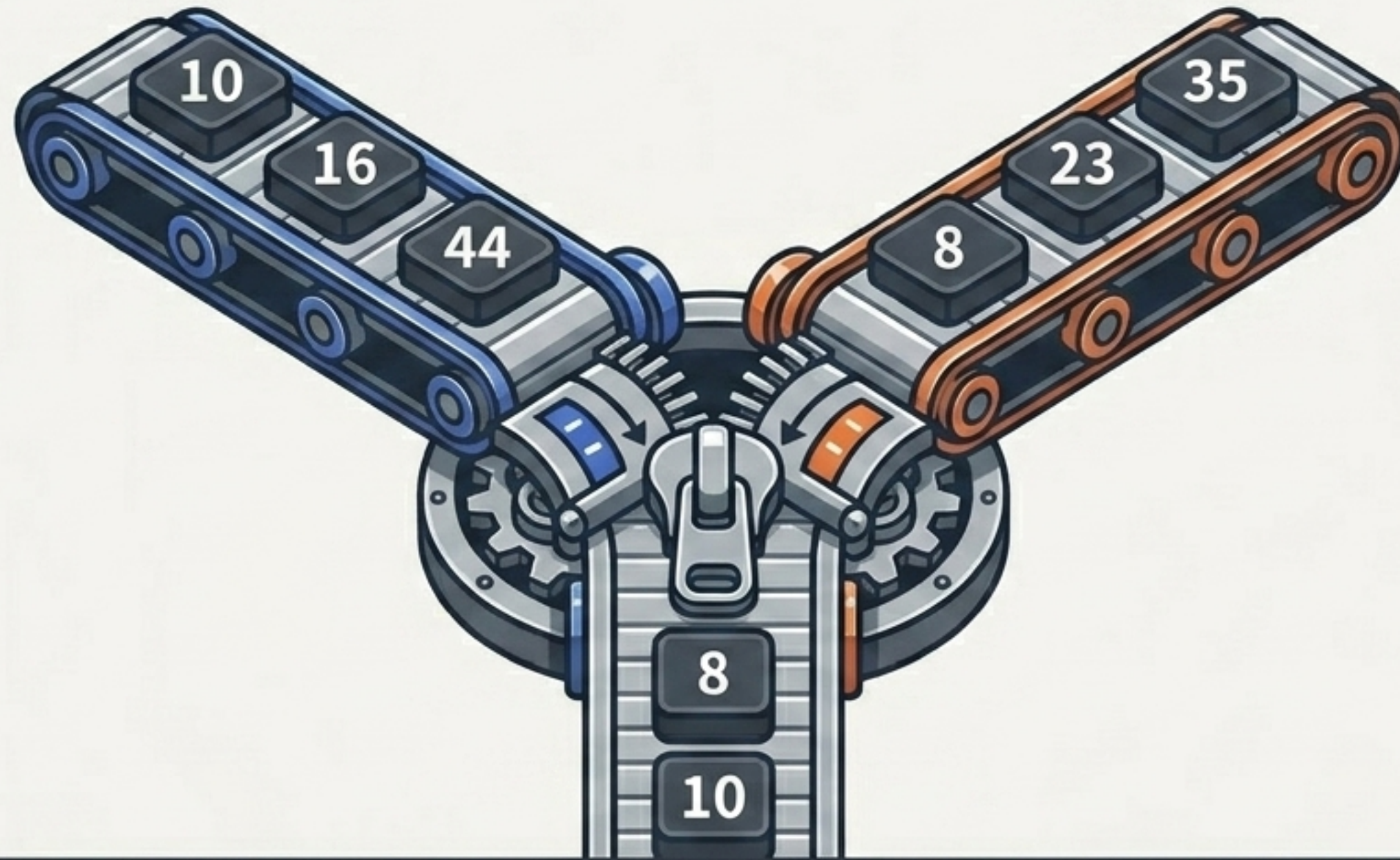
設問1(3) ヒント：奇数個($2N+1$)の分割



分割結果：前半は [6, 4, 3] (3個)、後半は [8, 7] (2個)。

数式化： $N=2$ のとき、前半は3個なので「 $N+1$ 」。後半は2個なので「 N 」。奇数の場合、常に前半が1つ多くなります。

第2フェーズ：リスト の併合 (Merge)

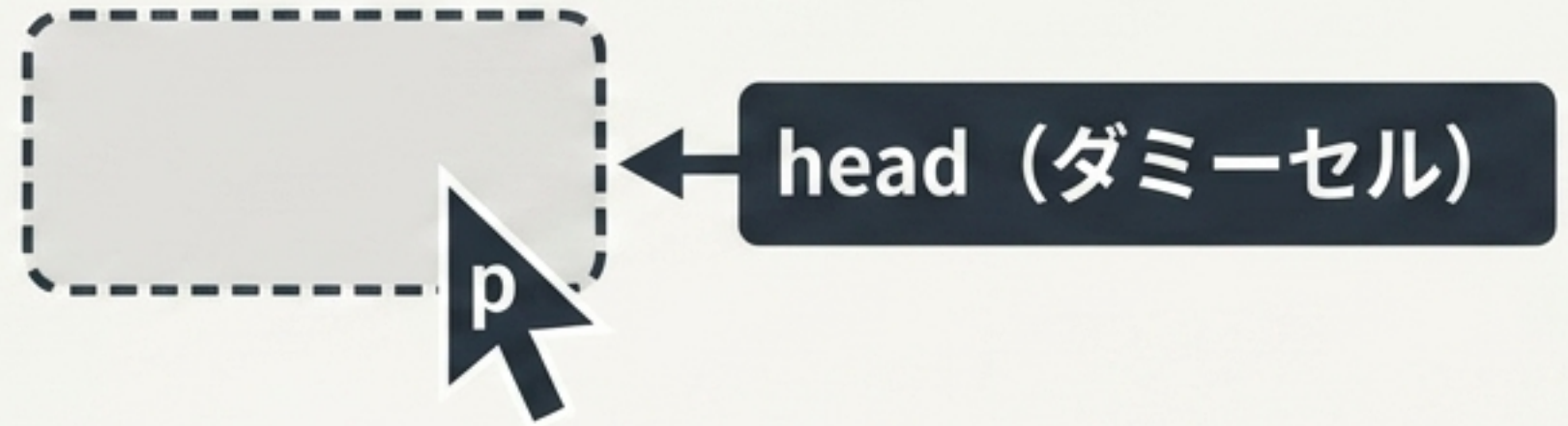


2つのリストの「先頭」同士を比較し、小さい方を新しいリストに引き抜いていく。これをどちらかが空になるまで繰り返す。

併合のテクニック：「ダミーセル」の活用

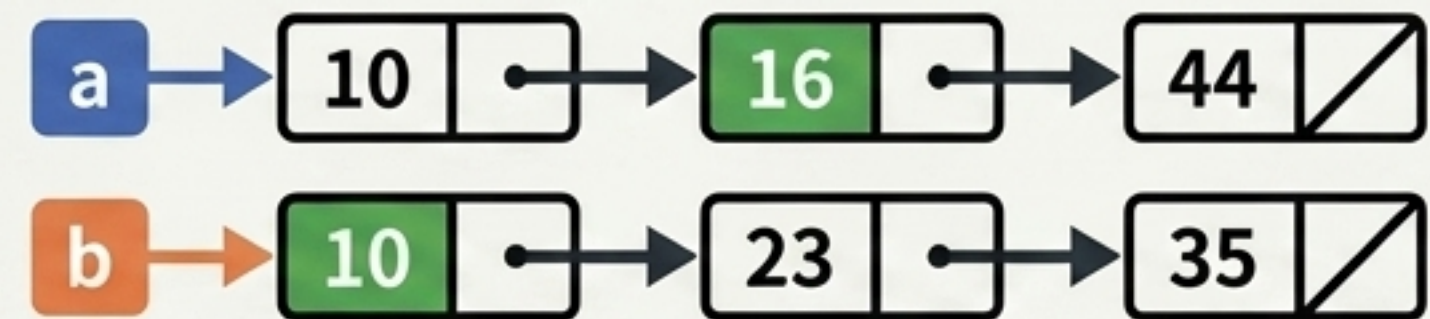
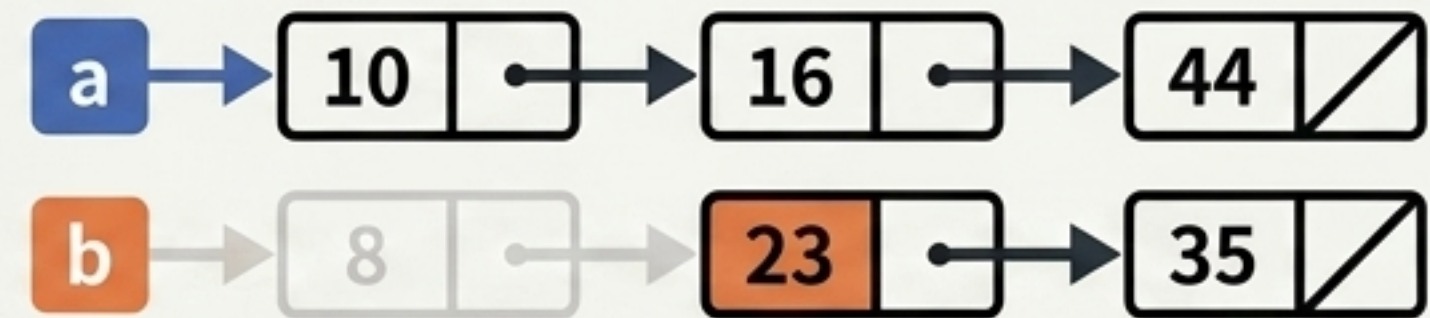
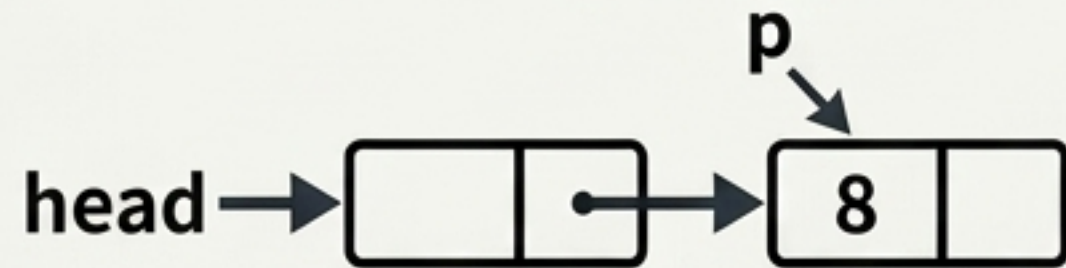
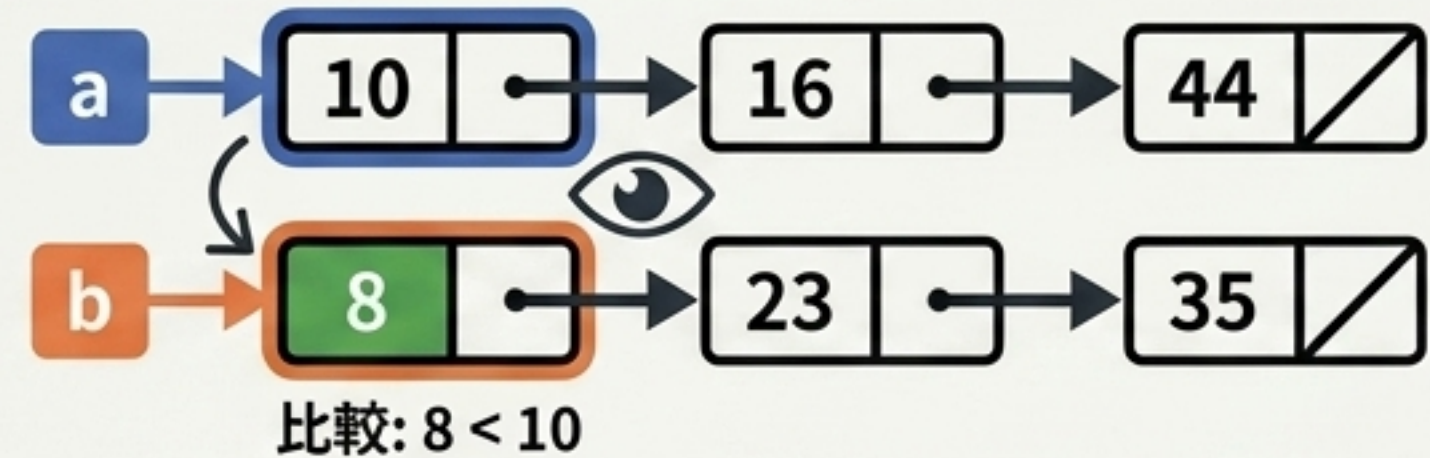
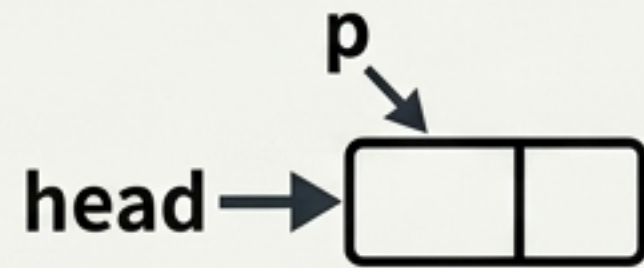


最初にどちらのリストから要素を引き抜くか未確定なため、新しいリストの「先頭」を固定できない。



値を持たない空の箱を用意し、そのnextに次々と小さい要素を繋げていく。これにより、最初の要素接続条件分岐（if文）を排除できる。

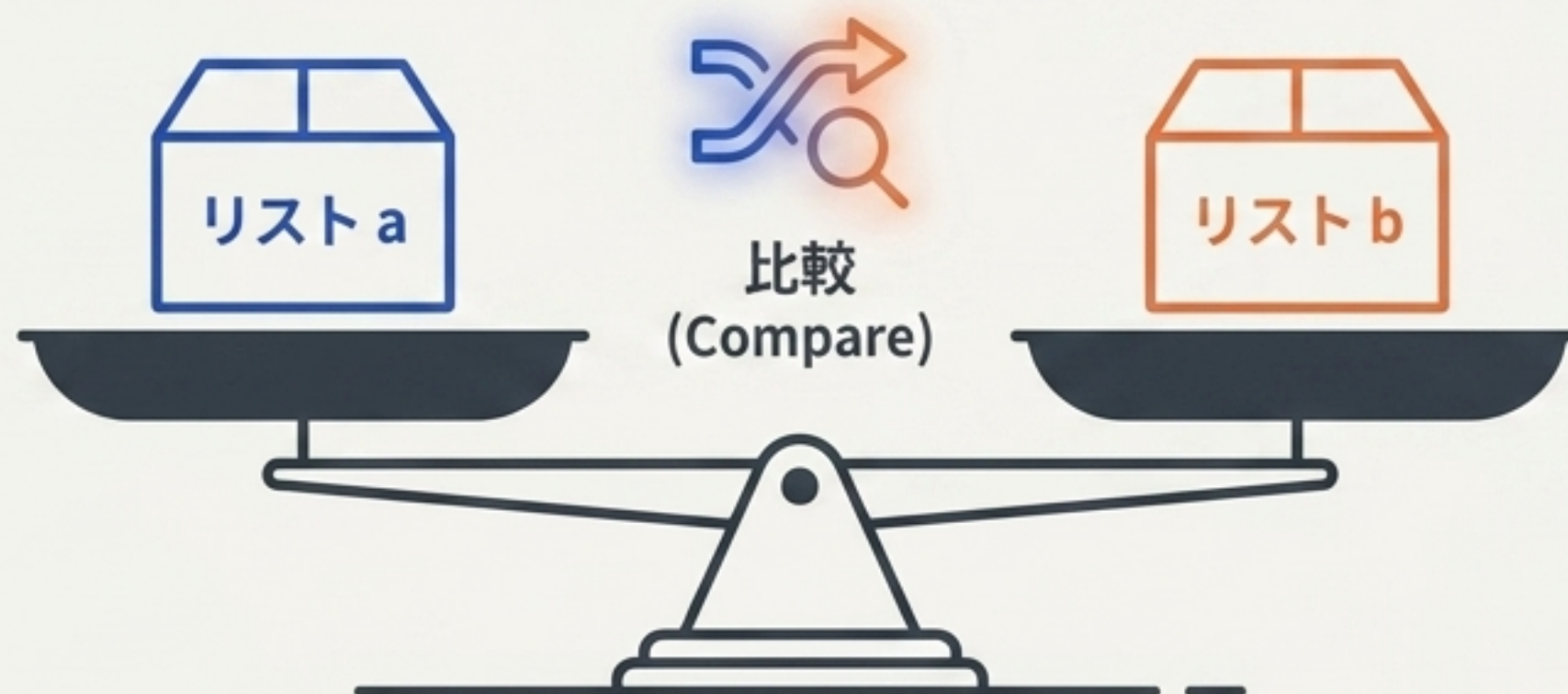
シミュレーション：比較と連結



小さい方を採用し、採用された側のポインタだけが次に進む。

設問2 ヒントタイム：空欄 [エ]

```
while ( [ エ ], かつ, bがNULLと等しくない )
```



前提:

a->value と b->value を
「比較」する処理。

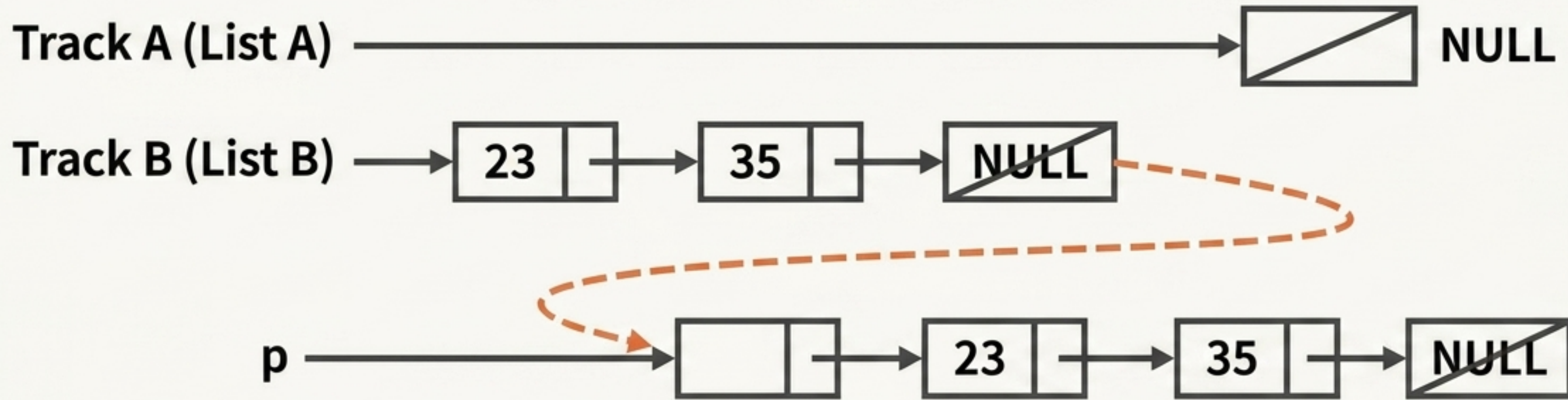
→ 事実:

比較を行うためには、両方のリストに要素が存在していなければならない。どちらか一方が空 (NULL) になったら、比較は不可能になる。

→ 結論:

したがって、bだけでなく、aも空ではない（要素がある）ことが継続条件になります。

設問2 ヒントタイム：空欄 [オ] (残余処理)

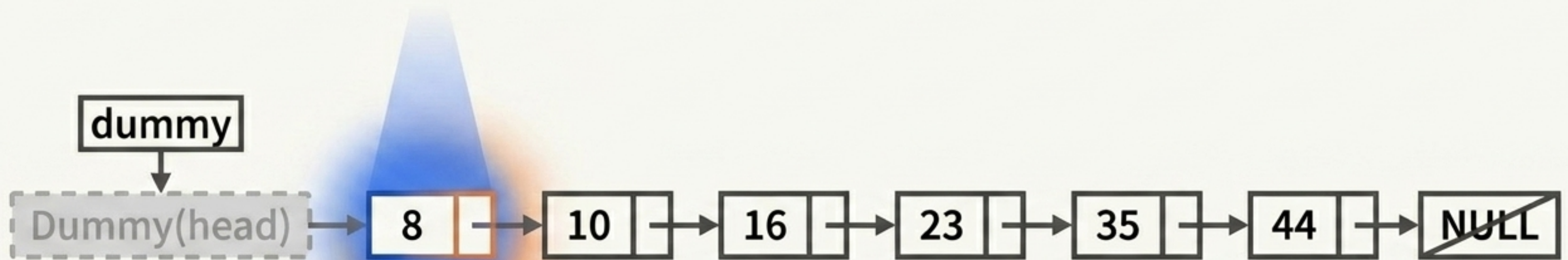


```
if ( [ オ ] ) {  
    p->next = b  
}
```

ヒント:

このif文は「リストbにまだ要素が残っている場合」の処理です。
ループを抜けた時点で「bに要素があるか」を判定する条件式を考えましょう。

設問2 ヒントタイム：空欄[力]（戻り値）



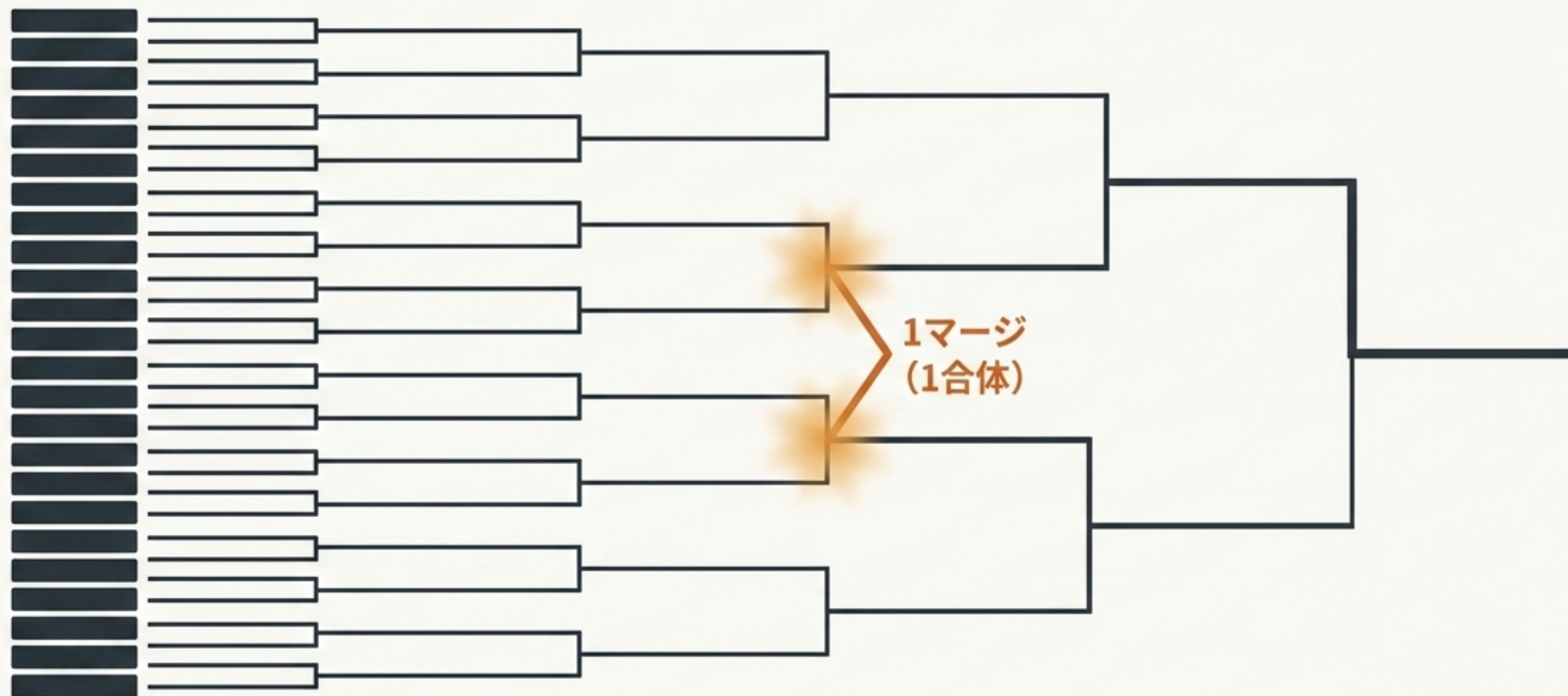
陥りやすい罠 (The Trap)

併合後のリストの先頭は head ですが、これは値を持たない「ダミー」です。

解決策 (The Solution)

呼び出し元に返す「本当のデータの先頭」は、ダミーの次のセルです。
ダミーの next を返す記述を考えましょう。

第3フェーズ：マージ回数の算出



32個のバラバラな要素は、32チームが参加するトーナメント戦と同じ構造です。
2つのリストを併合（マージ）する処理は、トーナメントの「1試合（合体）」に相当します。

設問3 ヒント：32要素の総マージ回数

1段目: 32個を2個ずつペアに → 16回マージ (16本になる)

2段目: 16本をペアに → 8回マージ (8本になる)

3段目: 8本をペアに → 4回マージ (4本になる)

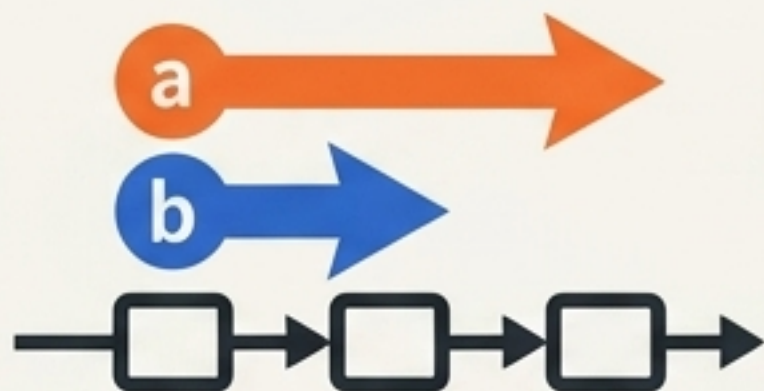
4段目: 4本をペアに → 2回マージ (2本になる)

5段目: 最後の2本を → 1回マージ (完成！)

ヒント：これらの合計を足し合わせれば、答えが導き出せます。

まとめ：連結リスト攻略の2大テクニック

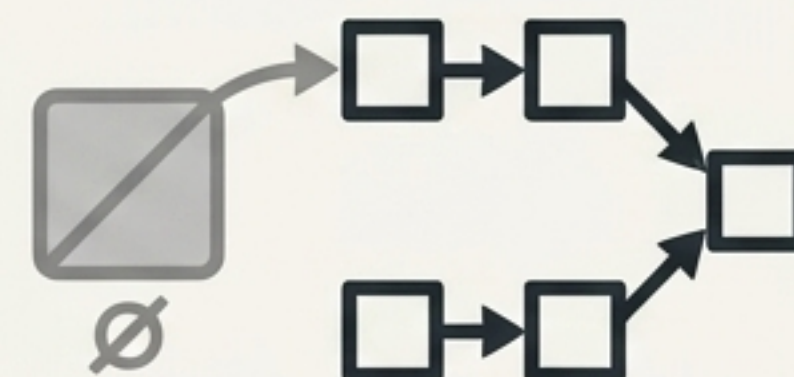
探索のテクニック (The Divide)



ウサギとカメのポインタ (2倍速)

長さ不明のリストの中央を特定するための定石。ポインタ操作の応用力の証。

構築のテクニック (The Merge)



ダミーセルの活用

複雑な先頭処理の分岐を排除し、コードを美しく保つための先人の知恵。

この2つの視覚的イメージを持っていれば、本番のアルゴリズム問題でもポインタを見失うことはありません。自信を持って試験に臨んでください！