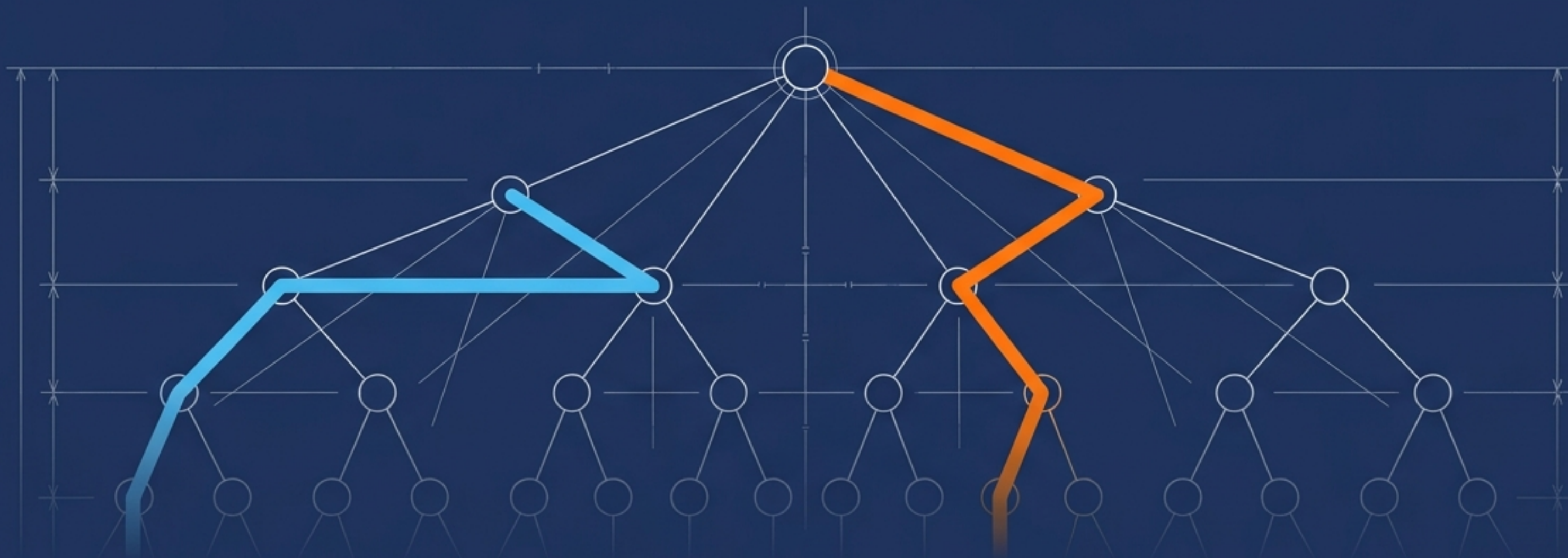


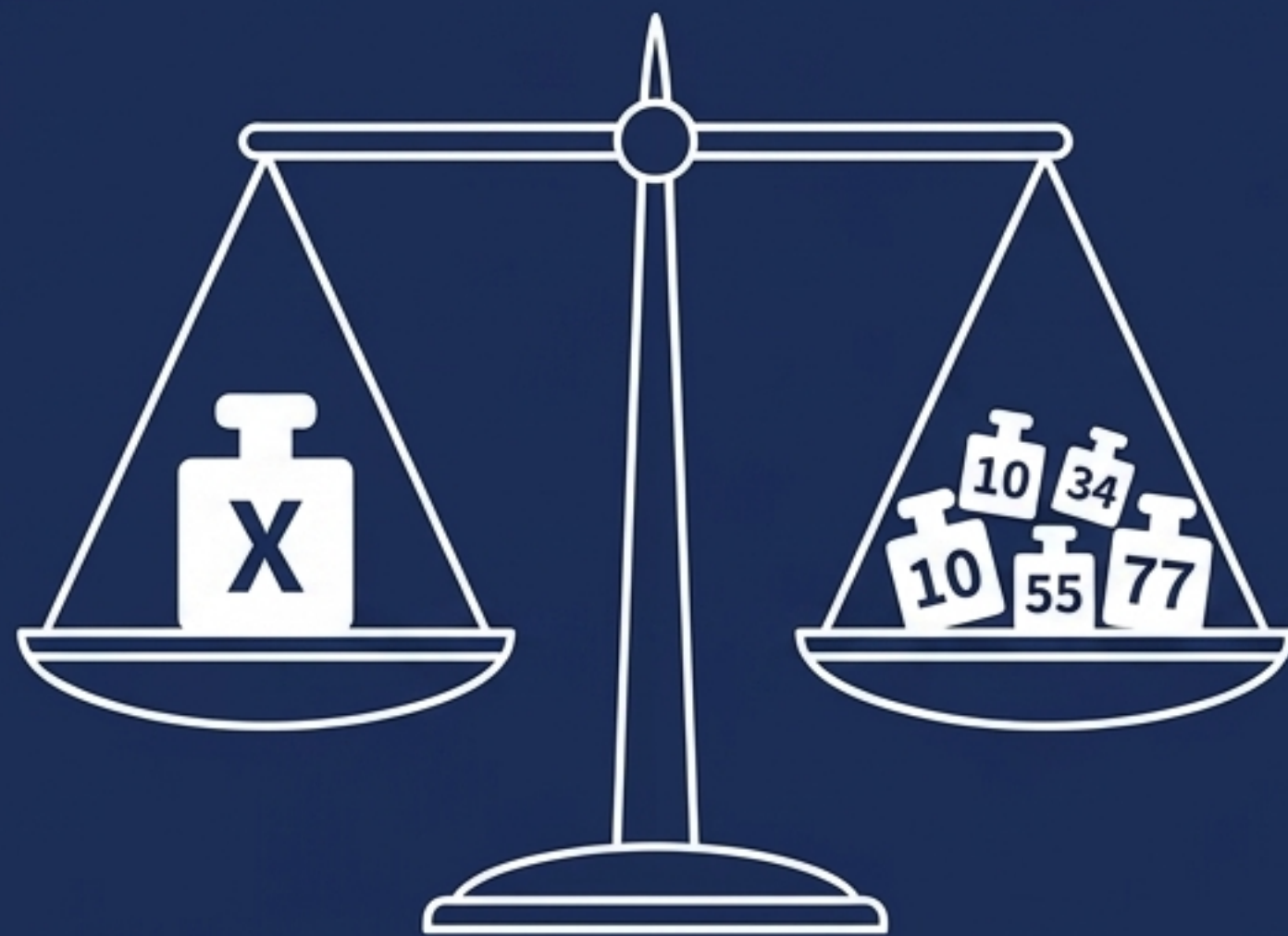
応用情報技術者試験から読み解く、アルゴリズムのナビゲーション構造

探索者の設計図：キューとスタック

なぜ木を探索するのか？ どう迷わずに進むのか？



課題：目標値「X」に最も近い組み合わせを見つけよ

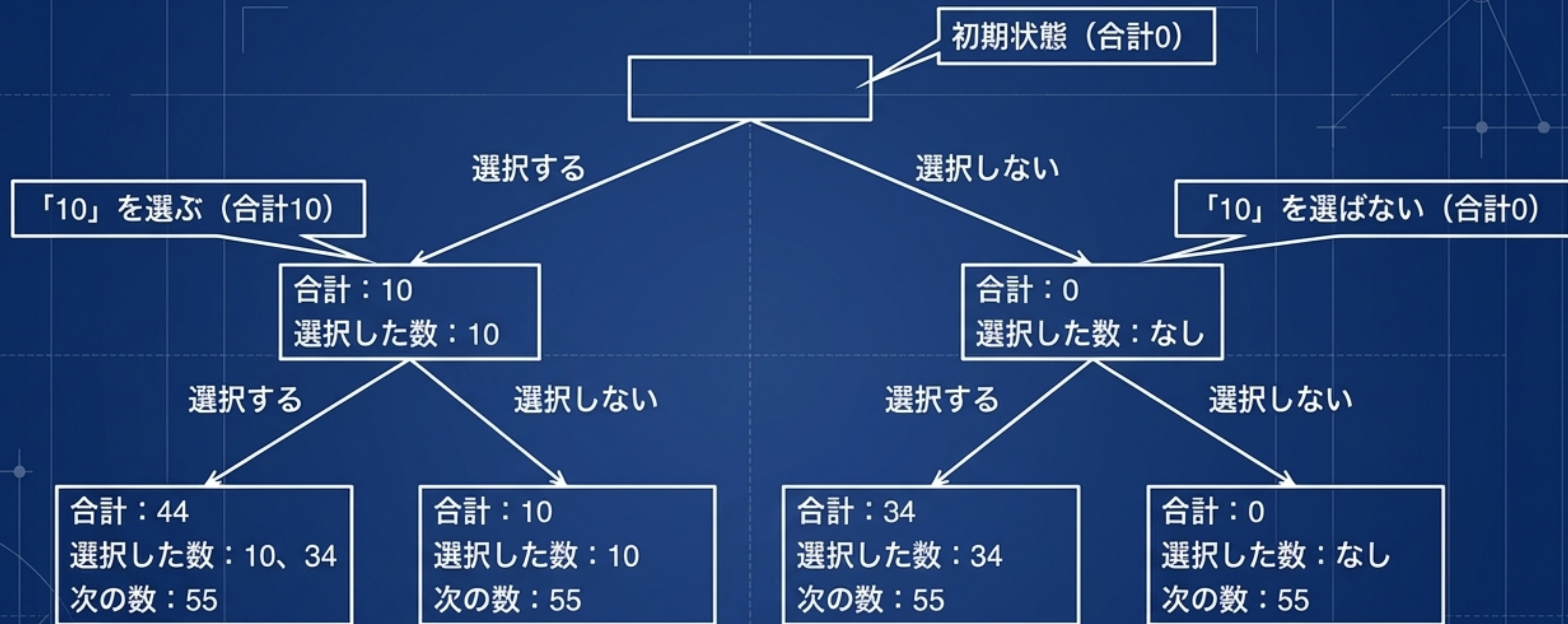


n 個の異なる数から任意の個数を選び、合計を目標 X に近づけるミッション。

〔問題〕

指定された n 個の異なる数（自然数）の中から任意の個数の数を選択し、それらの合計が指定された目標 X に最も近くなる数の組合せを 1 組選択する。その際、合計は X より大きくても小さくてもよい。ただし、同じ数は 1 回しか選択できないものとする。

分岐の連鎖：問題は「木」として視覚化できる



すべての可能性を網羅する＝この広大な木のすべてのノードを「訪問」すること。

探索の必須ツール：「次に行く場所のメモ帳」



To-Do List（次に見るべき状態のメモ帳）

- 木構造は分岐のたびに選択枝が増える。
- 人間の記憶（や単純なプログラム）では、どこまで探索したか迷子になる。

**結論：未探索のノードを一時的に保存する
「構造化されたメモ帳」が不可欠。**

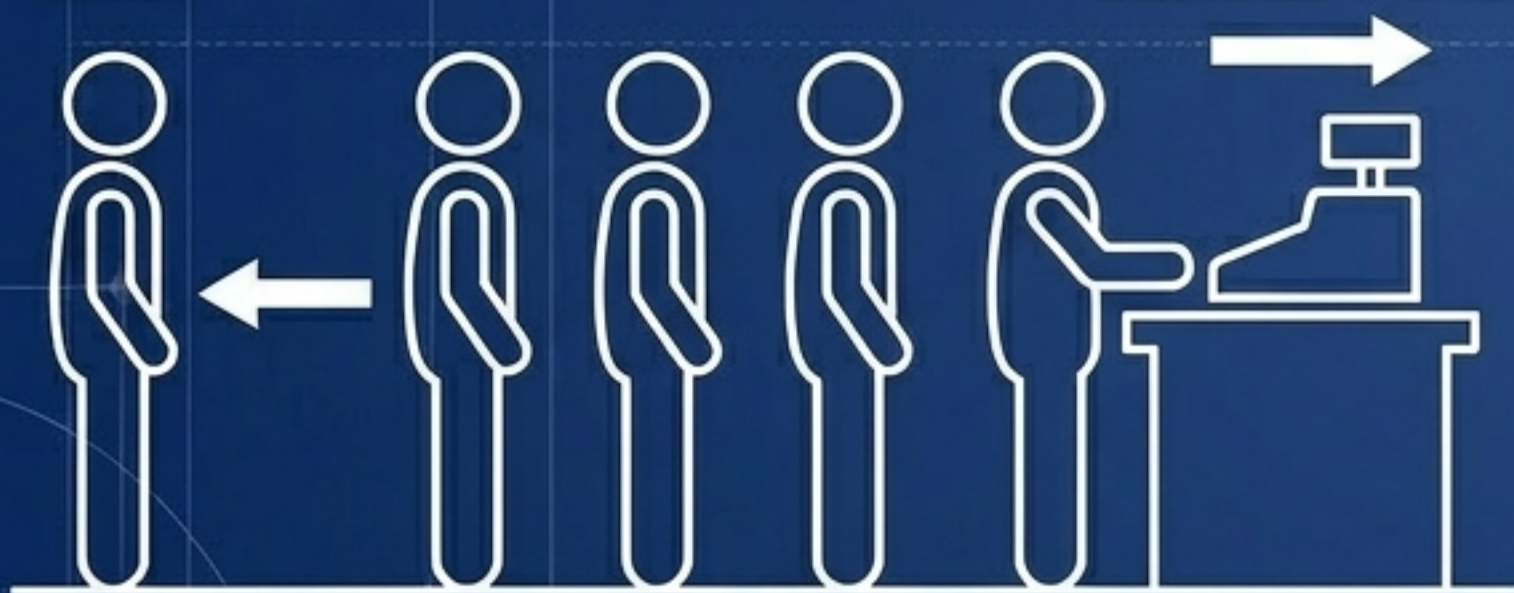
メモ帳のルール A：キュー (Queue)

FIFO (First-In, First-Out) / 先入れ先出し

Metaphor

JOIN

SERVED FIRST



Data Structure

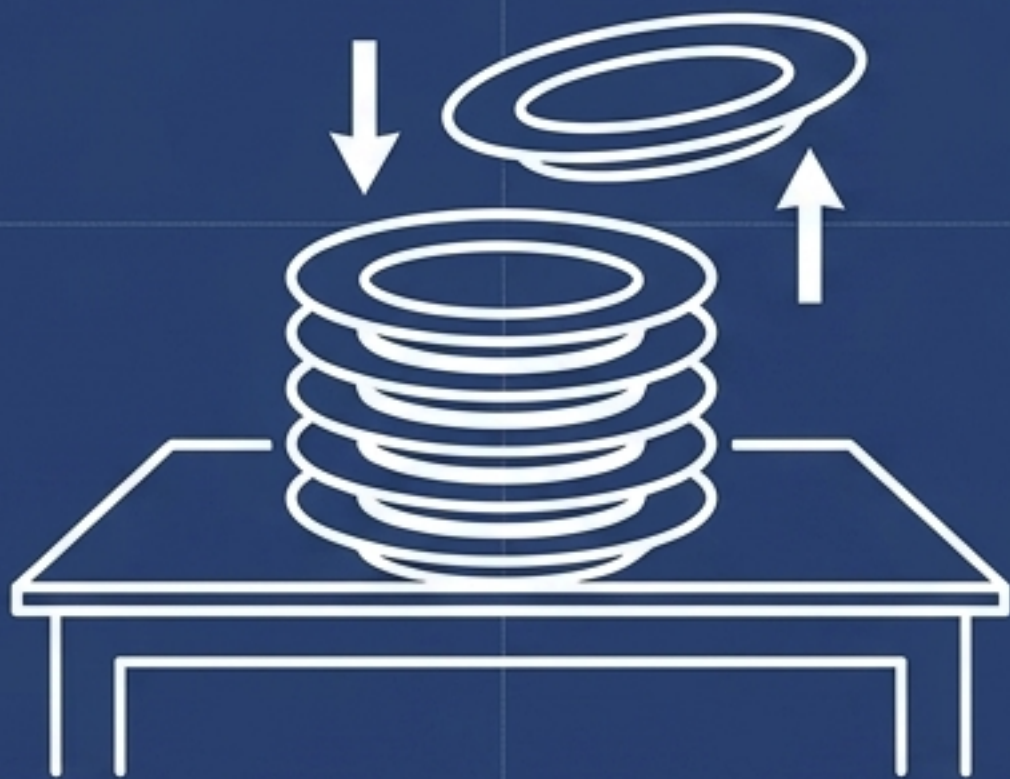


「レジ待ちの列」。先に追加された古いデータから順に処理される。順番を厳密に守る、平等なルール。

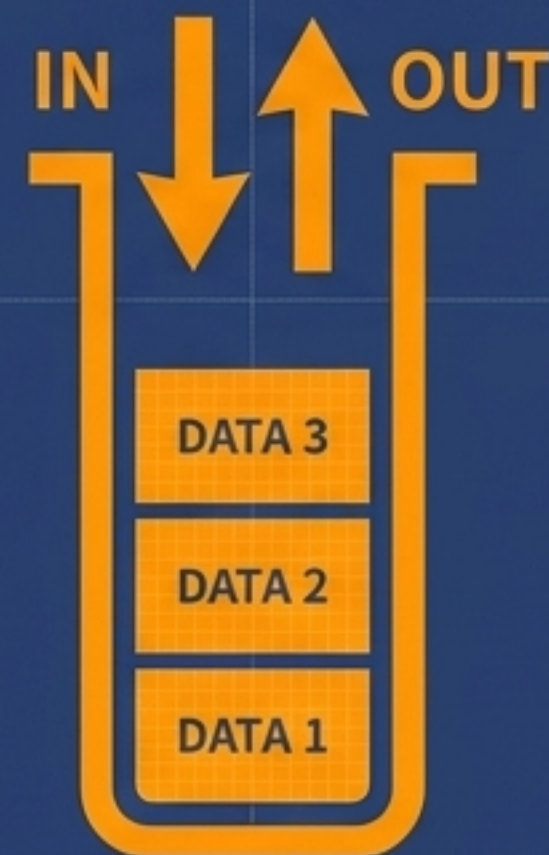
メモ帳のルール B : スタック (Stack)

LIFO (Last-In, First-Out) / 後入れ先出し

Metaphor



Data Structure

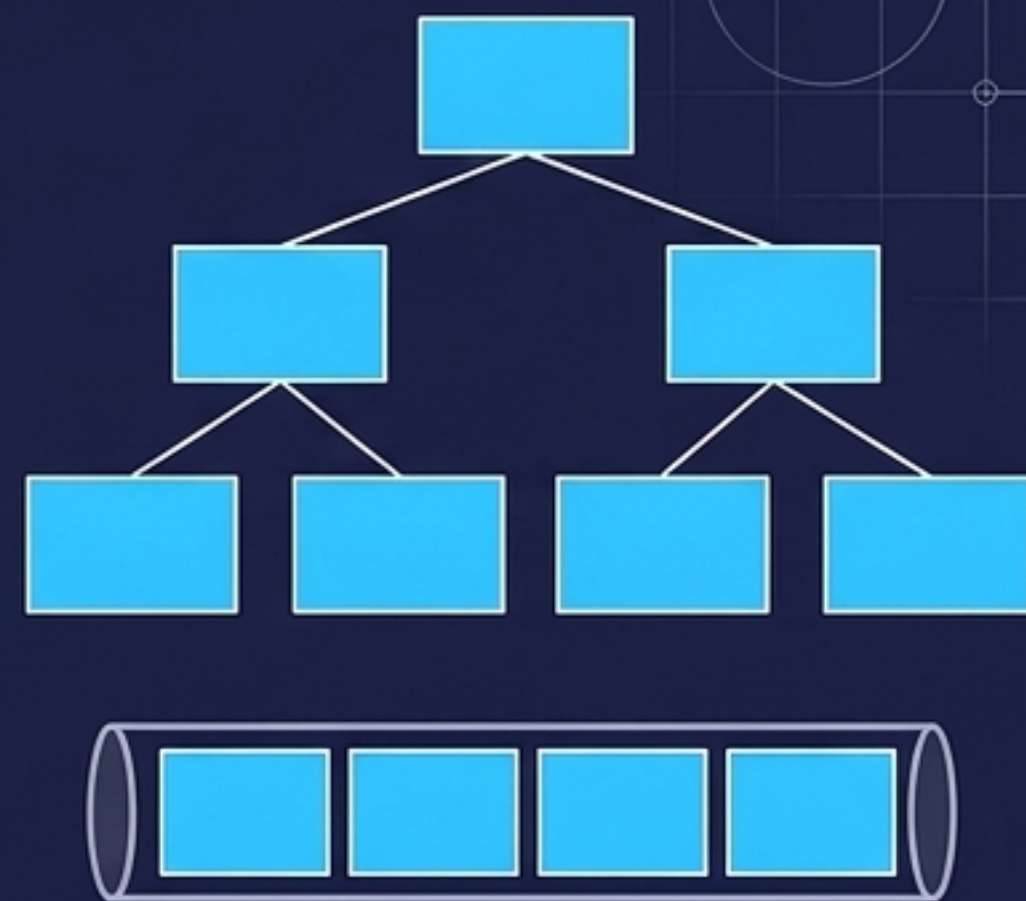
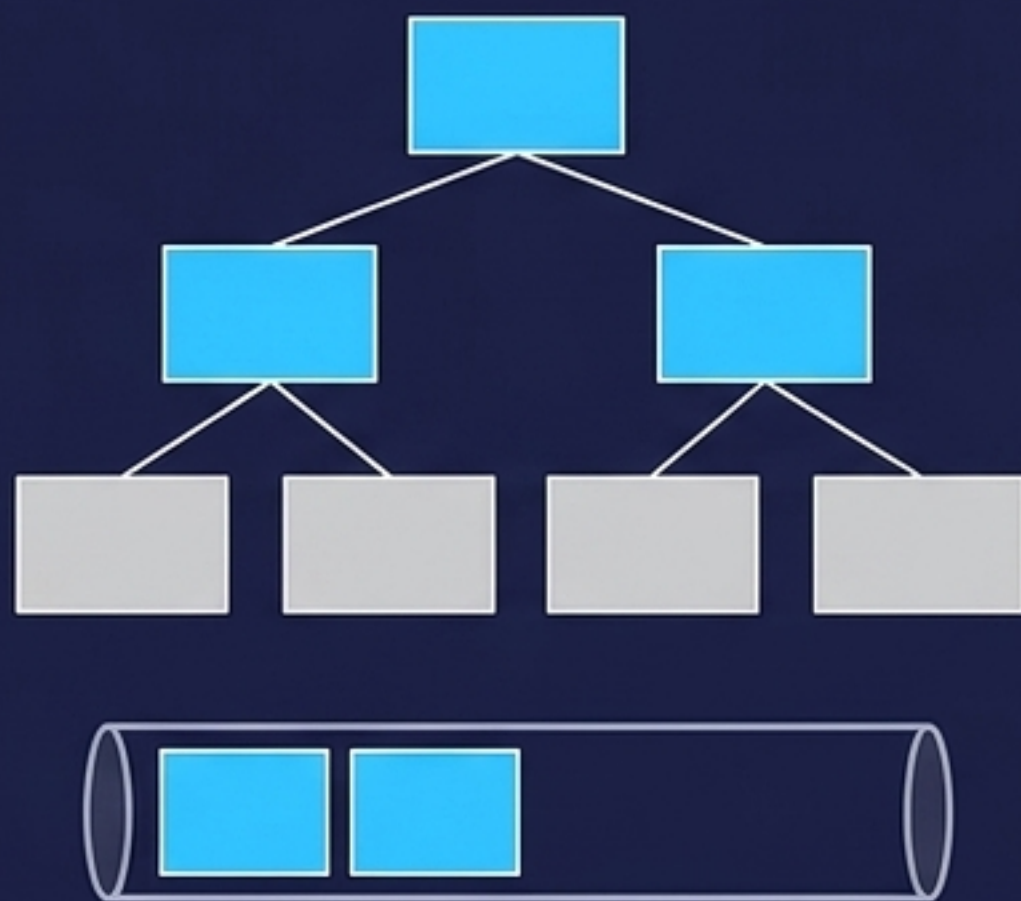
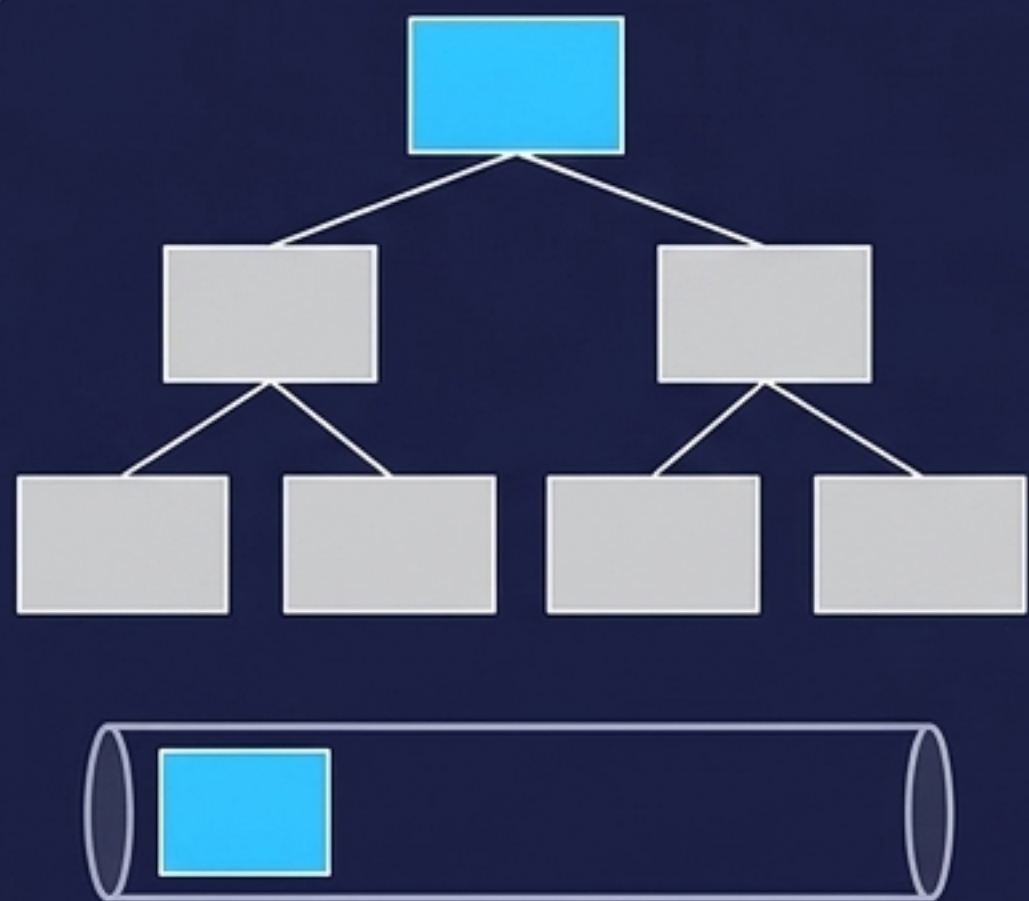


Concept: 「積み上げられた皿」。最後に追加された最新のデータから真っ先に処理される。
最新の文脈を優先するルール。

データ構造の特性比較

キュー (Queue)	スタック (Stack)
 レジ待ちの列	 積み上げられた皿
FIFO (先入れ先出し)	LIFO (後入れ先出し)
Enqueue / Dequeue	Push / Pop
「古さ・順番」を優先	「新しさ・深さ」を優先

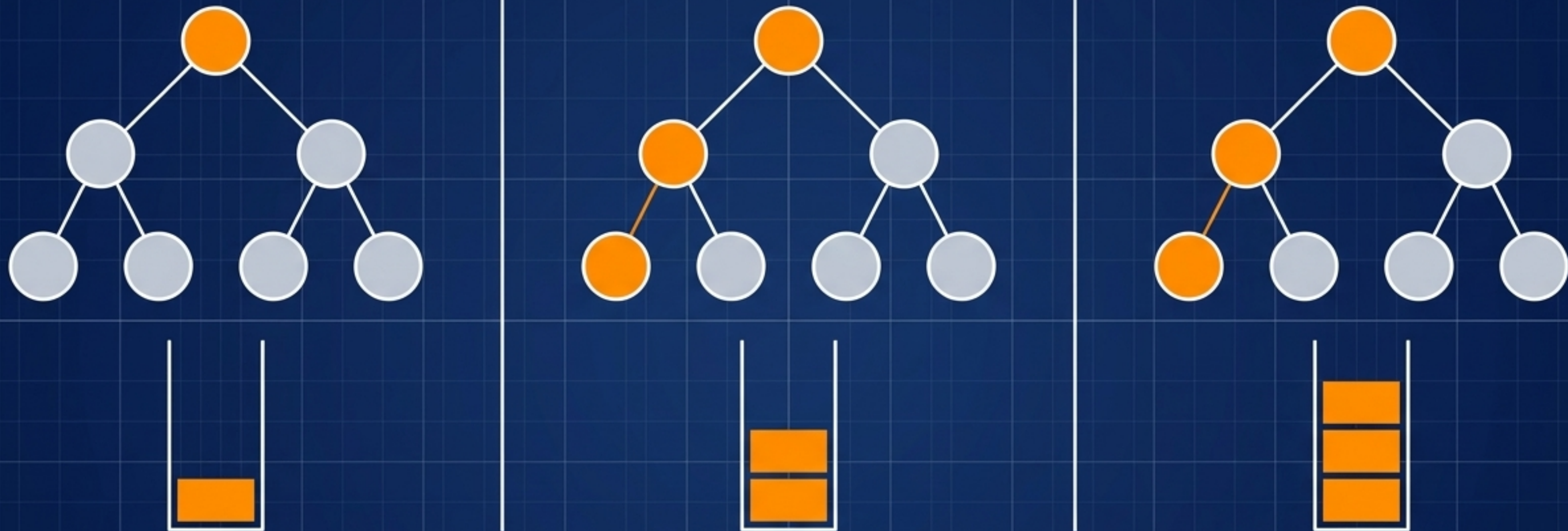
キューを使った探索：幅優先探索（BFS）



FIFOのルールにより、「同じ階層」のノードがすべてキューに登録され、古い順（=上の階層順）に消化される。

軌跡: 木を「横へ、横へ」と地層のように探索していく。

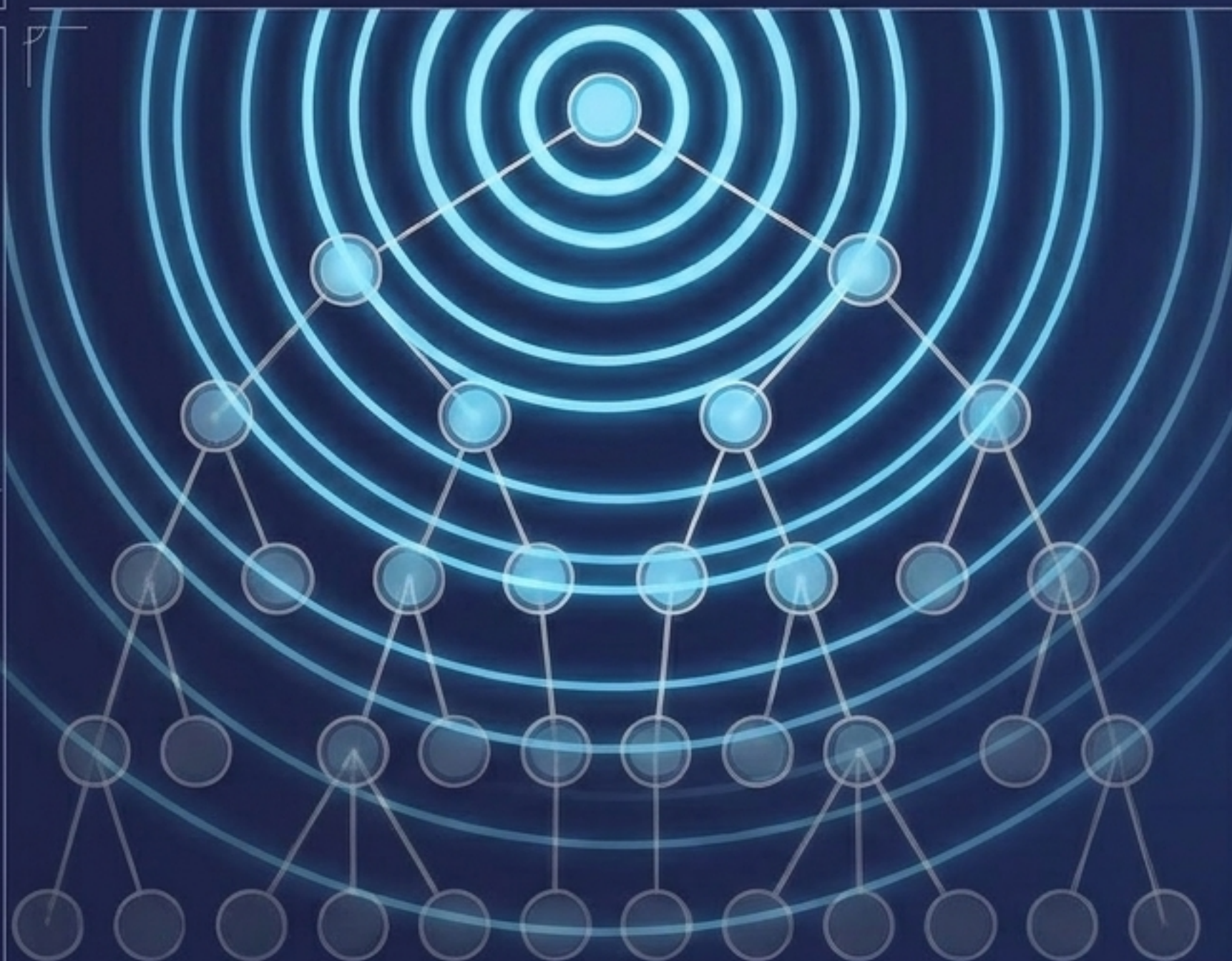
スタックを使った探索：深さ優先探索 (DFS)



LIFOのルールにより、今見つけたばかりの「最新の子ノード」が優先して処理される。

軌跡: 木を「縦へ、深く」と一直線に底まで突き進む。

軌跡の統合：全く同じ木でも、見え方が変わる

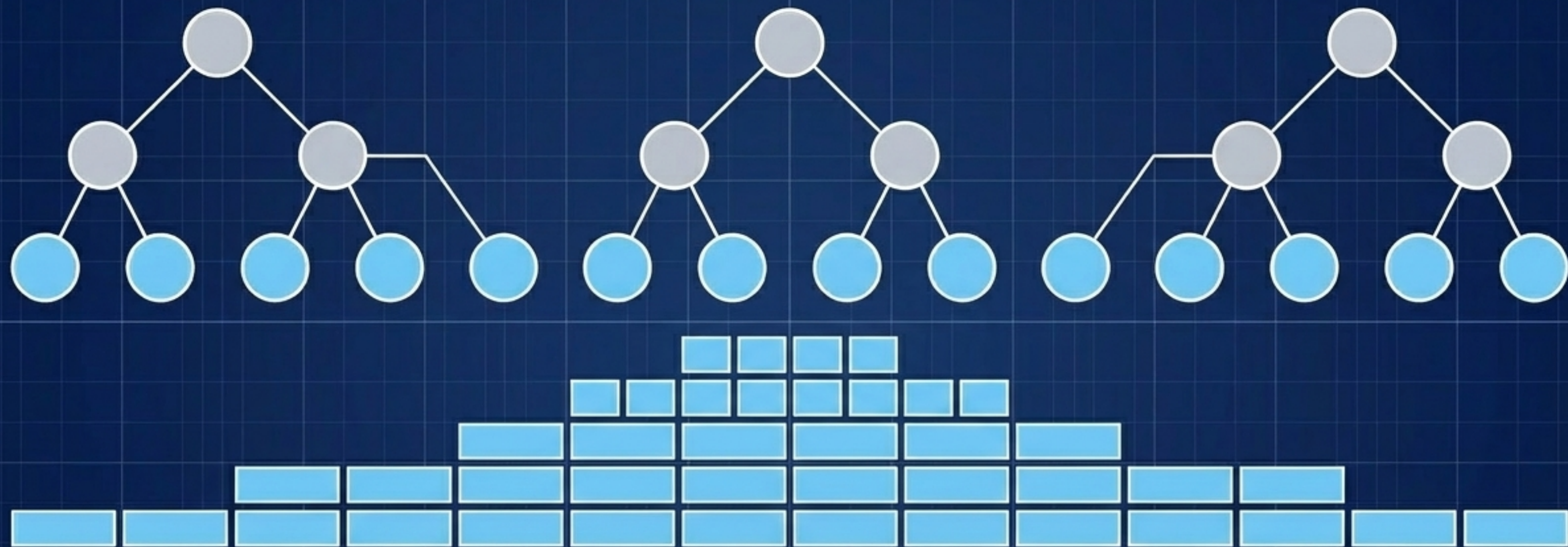


「波紋」のように全体を均等に把握する (BFS)



「掘削」のように一直線に深淵を目指す (DFS)

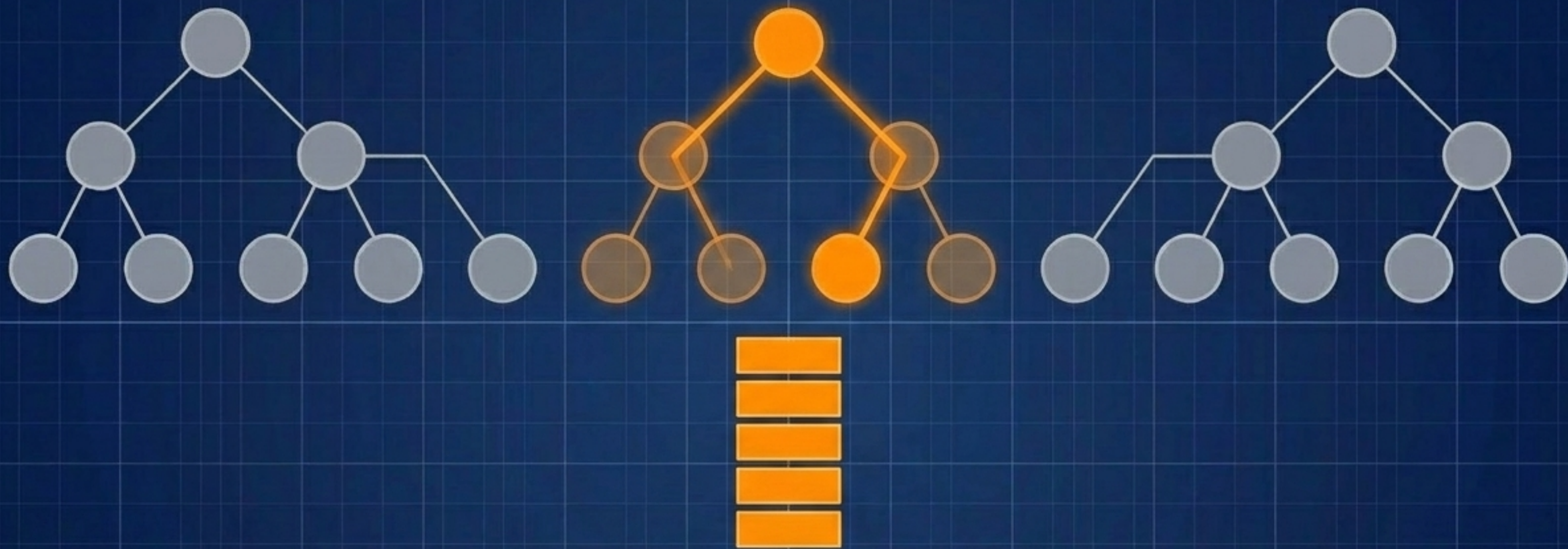
パフォーマンスへの影響：キューのメモリ消費



キューは横に広がるため、ある階層のノードを「すべて同時に」記憶しておく必要がある。

Insight: 階層が n 段階ある場合、最も広い層（最下層）のノード数は 2^n 個。未探索メモの最大値は、この「木の最大幅」に比例して爆発的に増大する。

パフォーマンスへの影響：スタックのメモリ消費



スタックは縦に進むため、現在の「ルート(道筋)」と、その横道(兄弟ノード)だけを記憶しておけばよい。

Insight: 階層が n 段階の場合、記憶する最大ノード数は深さに比例し、およそ n 個程度で済む。メモリ空間を劇的に節約できる。

試験問題の核心：メモリ使用量の最大値

設問4: データ構造にキューを使用した場合に、キューが必要とするメモリ使用量の最大値として適切なものは？



ア $2^n m$

イ $2nm$

ウ nm

エ $n^2 m$

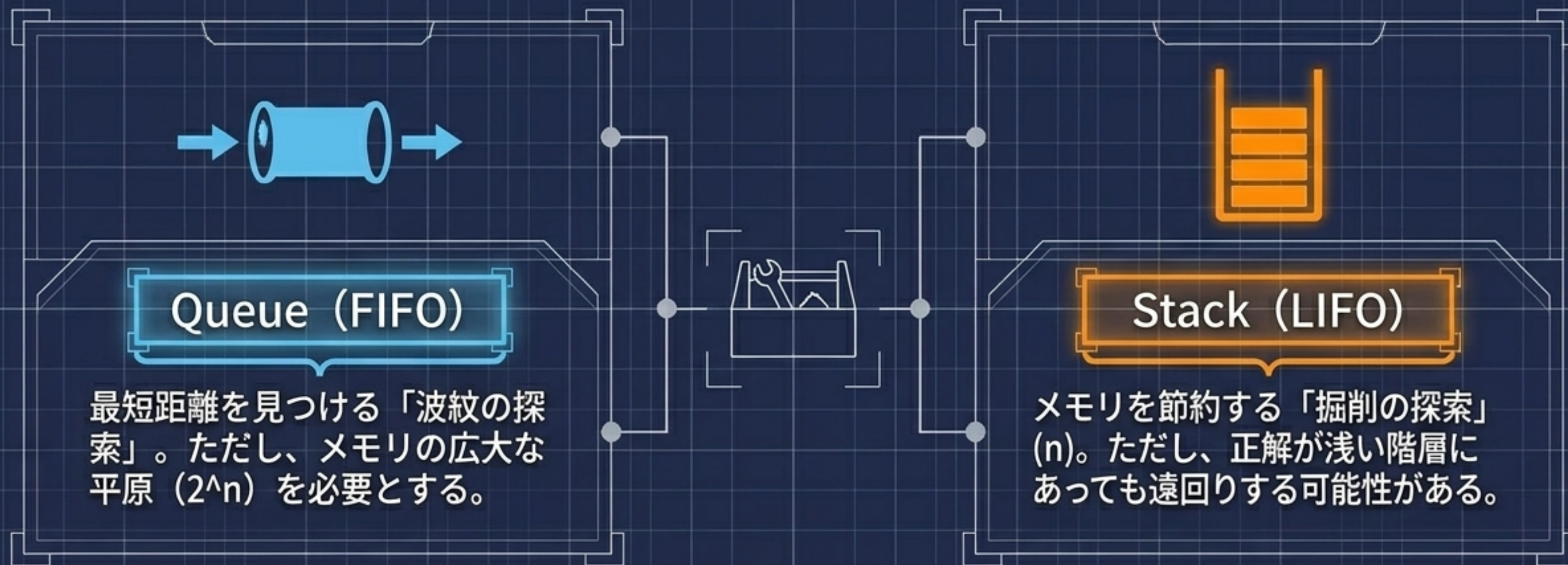
オ $(n+1)m$



Logical Deduction:

1. キュー = 幅優先探索
2. 最もノード数が多い階層を「すべて」同時に保持する。
3. 深さ n の二分木の最大幅は 2^n 個。
4. 1状態あたりのメモリが m ならば、最大値は $2^n \times m$

探索者の設計図：まとめ



アルゴリズムの選択とは、単なるコードの違いではない。
限られたリソース(メモリ)と目的(探索順)を天秤にかける「設計の美学」である。