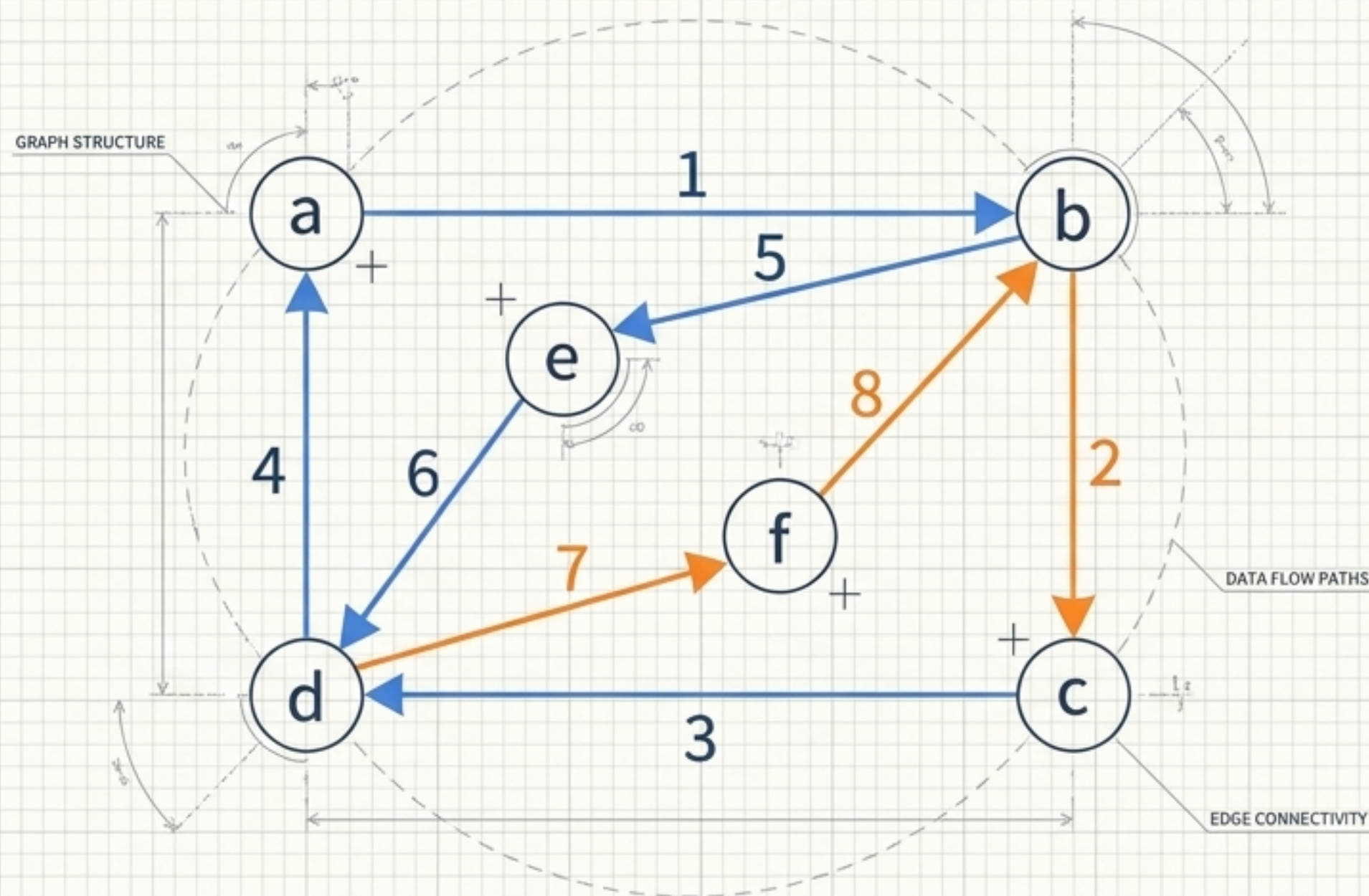


THE ALGORITHM BLUEPRINT

一筆書き探索プログラムの解剖図

令和3年度 秋期 応用情報技術者試験 午後問3「設問1」の視覚的思考プロセス



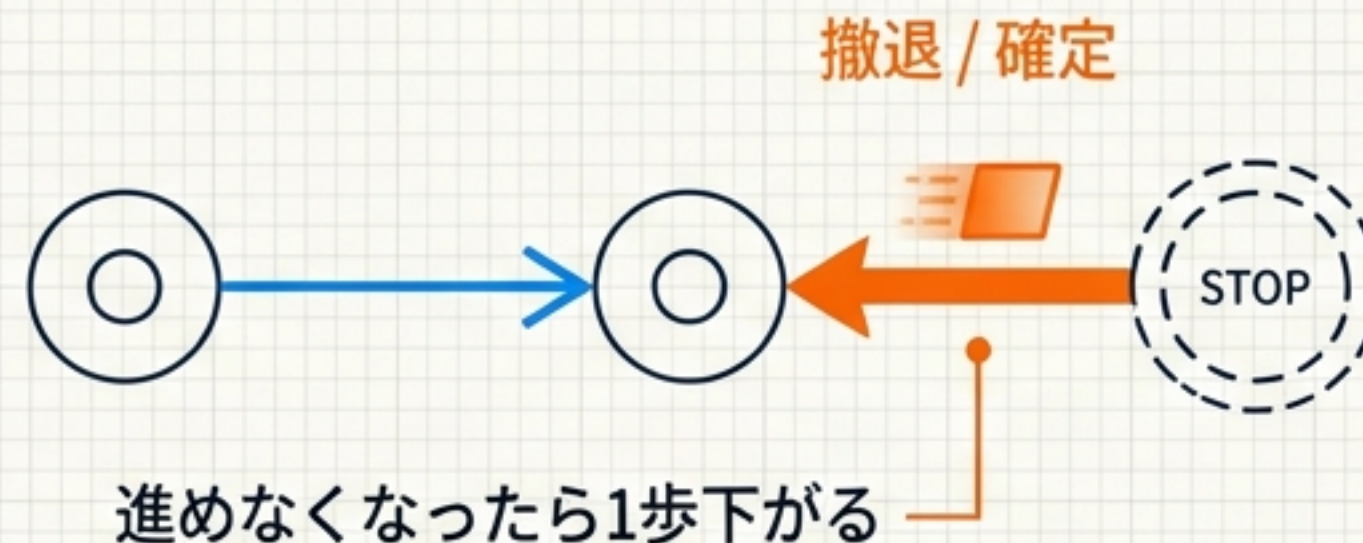
探索の基本戦略：進めるところまで進み、行き止まりで戻る

この一筆書きアルゴリズム（バックトラック法）は、2つの基本動作の繰り返しで構成されています。未探索のルートを青（探索）、行き止まりからの撤退をオレンジ（確定）として捉えましょう。

前進 - Advance

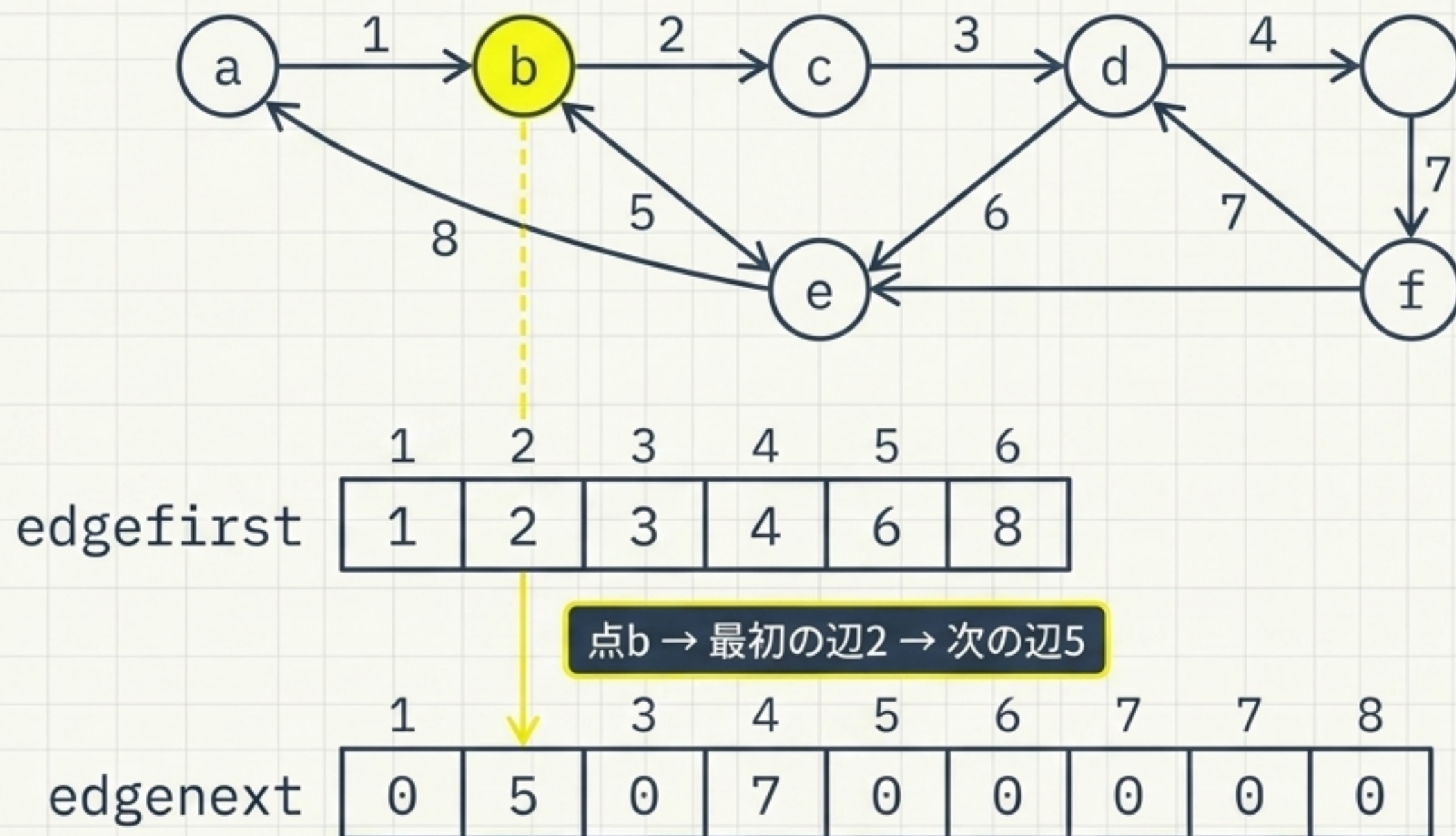


後退 - Backtrack



グラフ空間からデータ空間へのマッピング

プログラムは図形を見ることができません。代わりに、edgefirst（最初の道）と edgenext（次の道）という2つの配列をポインタとして繋ぎ合わせることで、グラフの構造を把握します。



エンジンの心臓部：if-else 分岐

空欄のすべては、探索をコントロールする if-else の内部に存在します。

「道があるから進む (if)」 「道がないから戻る (else)」という役割を意識して解剖を進めます。

while ループ

進む処理 (探索)

if (current[x] が [ア] でない)

[ア] 🔍

[イ] 🔍

戻る処理 (遡り)

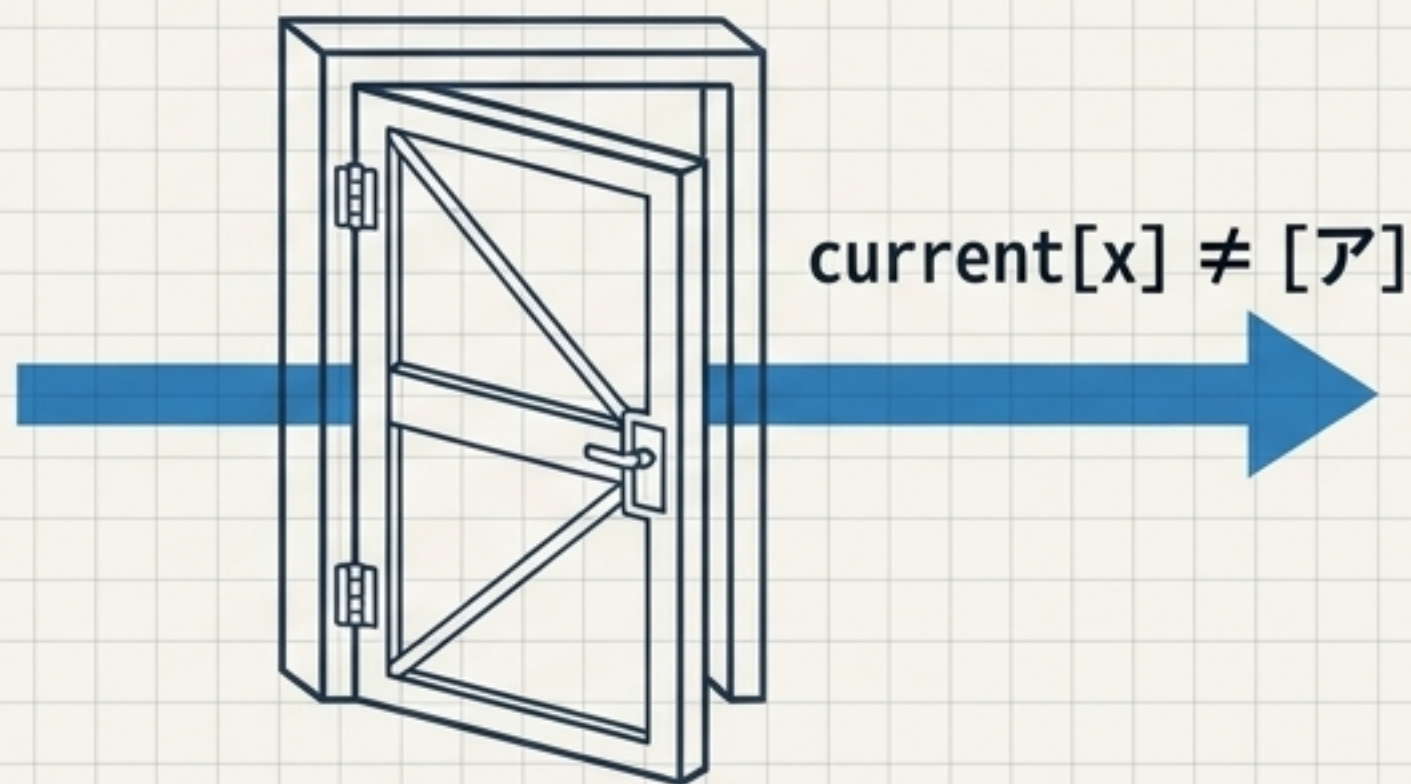
[ウ] 🔍

[エ] 🔍

[ア] 前進のゲートキーパー

if の中に入る（進む）ためには、「未探索の辺が存在する」必要があります。
定義表を参照し、「辺がない」ことを示す無効値は何であることを確認しましょう。

if (current[x] が [ア] でない)

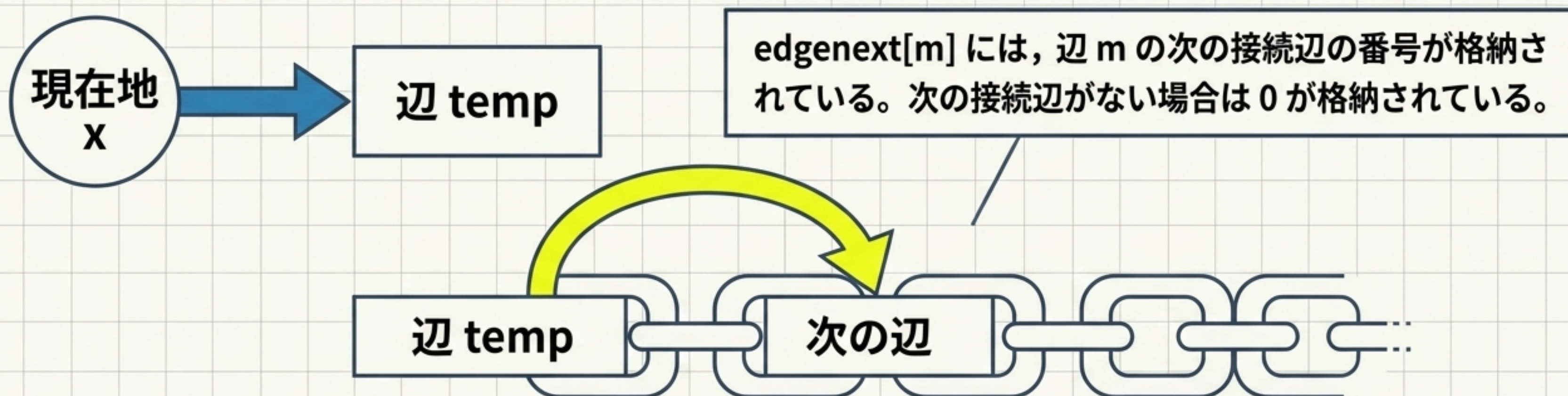


名称	内容
current[n]	current[n] には、点 n を始点とする未探索の辺の中で最小の番号を格納する。点 n を始点とする未探索の辺がない場合は 0 を格納する。

[イ] 次の探索候補のセットアップ

辺 temp を探索済みにした直後、配列 `current[x]` を更新します。

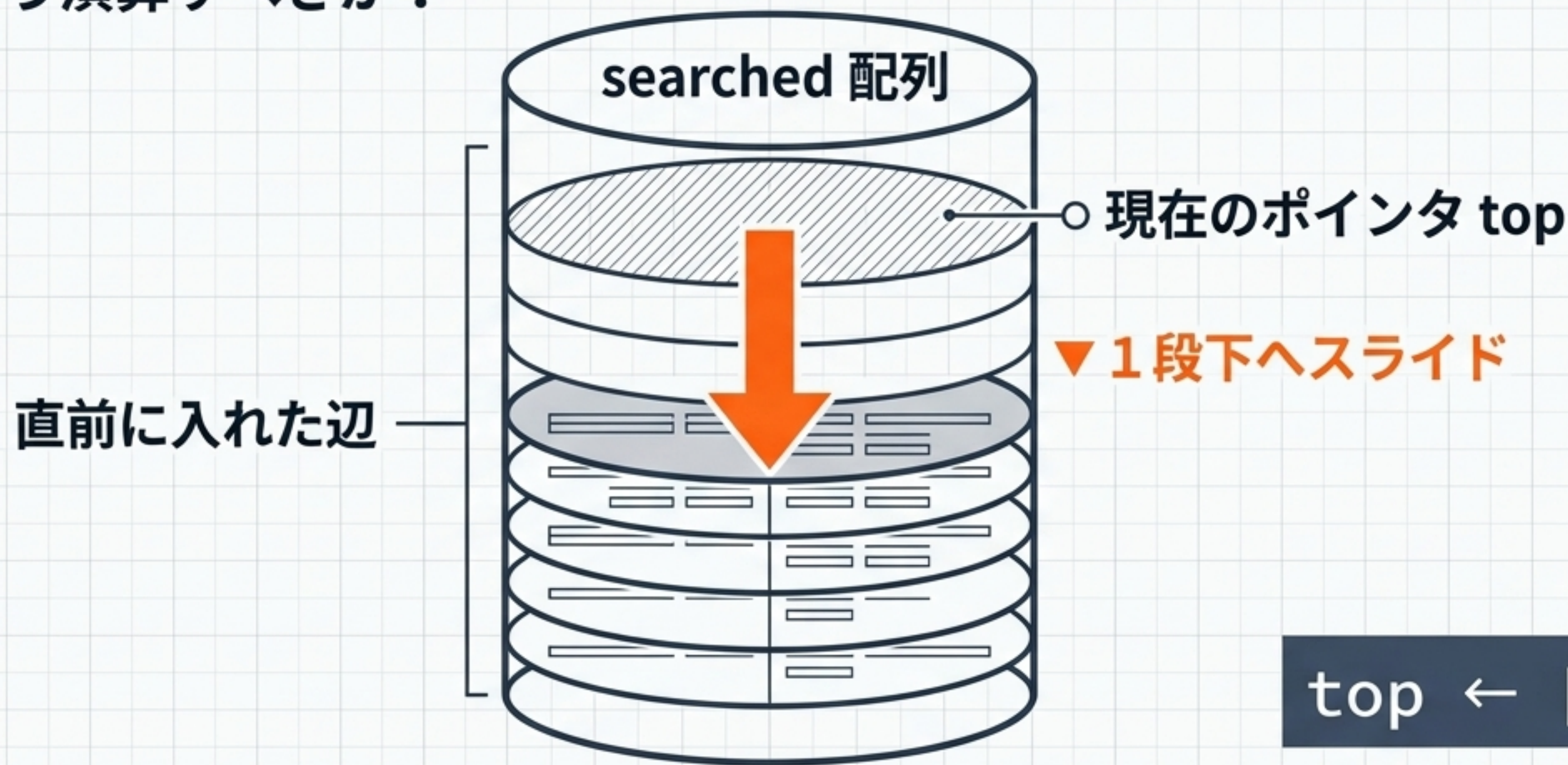
「辺 temp の、その次に大きい接続辺」が格納されている配列はどれでしょうか？



`current[x] ← [ イ ]`

[ウ] スタックからの巻き戻し開始

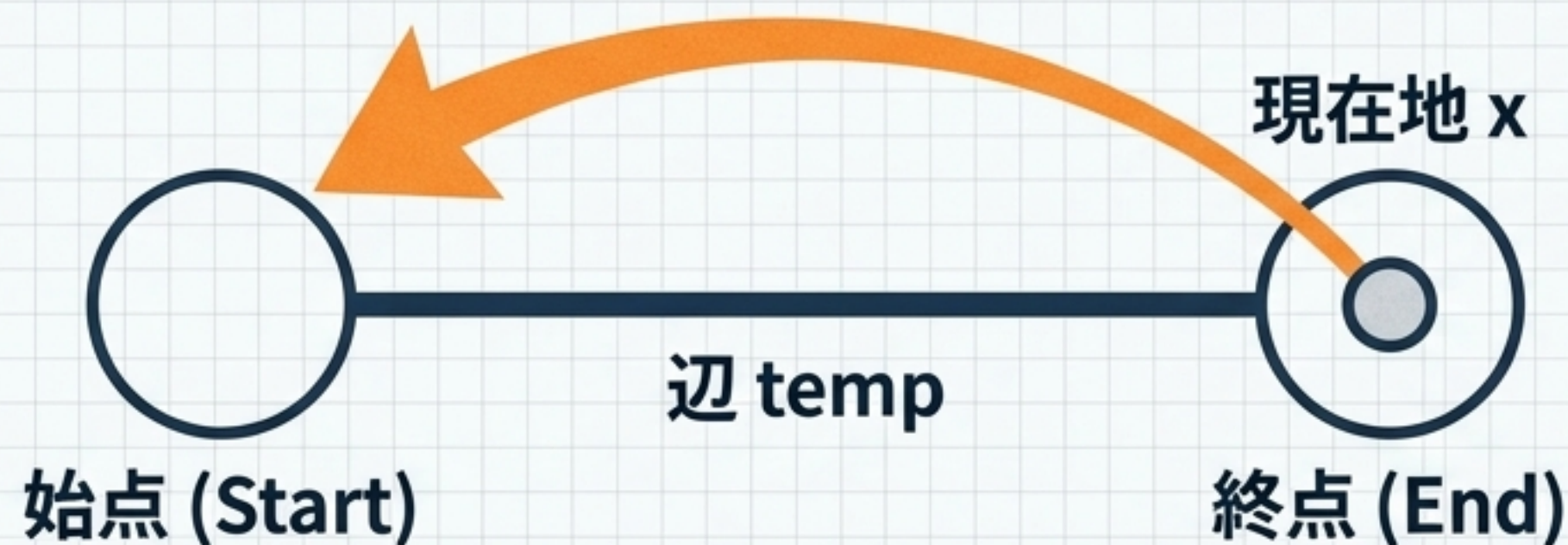
行き止まりに到達したため、直前に通った辺を searched から取り出します。
ポインタ top は現在「次の空き場所」を指しているため、直前のデータにアクセスするにはどう演算すべきか？



top ← [ウ]

[工] 現在地の物理的な後退

辺 temp を遡るということは、現在地 x を「辺の終点」から「辺の始点」へ戻すことを意味します。辺の始点番号を管理している配列を参照しましょう。



1	start[m] : 辺 m の始点の番号
2	end[m] : 辺 m の終点の番号

$x \leftarrow [\text{工}]$

対称性から見るアルゴリズムの構造

前進と後退の処理は、鏡合わせのように対称的な操作を行っています。このマトリクス構造を意識することで、プログラムの空白が論理的な必然性をもって埋まっていきます。

	【進む処理 (if)】	【戻る処理 (else)】
スタックの操作	searched に辺を入れる (top増)	searched から辺を出す (top減)
現在地 x の移動	辺の終点へ移動 (end)	辺の始点へ移動 (start)
役割	未探索ルートの開拓	確定済みルート (path) への登録

思考のプロセスから、正答へ

各行がグラフ上で「どの動き」に対応しているのか。このブループリント（解剖図）と問題文の定義表を照らし合わせれば、自ずと格納すべき変数や数値が見えてくるはずです。アルゴリズムは、可視化することで最も美しいロジックを見せます。

```
function directedE()  
  for ( i を 1 から N まで 1 ずつ増やす )  
    current[i] ← edgefirst[i]  
  endfor  
  top ← 1  
  last ← M  
  x ← 1  
  while ( ①last が 1 以上 )  
    if ( current[x] が 【無効値 (道なし)】 でない )  
      temp ← current[x]  
      searched[top] ← temp  
      current[x] ← 【辺tempの次の接続辺】  
      x ← end[temp]  
      top ← top + 1  
    else  
      top ← 【ポインタを1つ手前へ】  
      temp ← searched[top]  
      path[last] ← temp  
      x ← 【辺tempの始点】  
      last ← last - 1  
    endif  
  endwhile  
endfunction
```

// 各点での未探索の辺の番号を初期化

// 探索済の経路の辺の格納位置を初期化
// 確定済の経路の辺の格納位置を初期化
// 出発点は点1

// 点 x からたどる接続辺は temp
// 接続辺 temp を探索済の経路に登録
// 点 x から次にたどる未探索の辺を格納
// 接続辺 temp の終点を点 x にする

// 探索済の辺を遡る
// 遡った辺は temp
// 辺 temp を確定済にする