競プロ勉強会 180619

17 - baton

計算量

- 計算量とは、演算の性能を表す指標に使われるもの

- 時間計算量(処理時間の指標) ← “計算量”と言われたら基本こっち

  空間計算量(メモリ使用量の指標)

- 入力の値の変化に対してどの程度、計算時間orメモリ使用量が増えるのかがわかる

表記の仕方

- $O(1), O(\log N), O(N), O(N+M), O(N^2), O(NM^2), O(2^N)$
  のような形で表記する
- 読み方は、『オーダー〇〇』や『ビックオー〇〇』
- 3,4Qの微積でやるスモールオーとは、微妙に定義が異なる

例

- これは、$O(N^2)$

```
for(int i=0;i<N;i++){
  for(int j=0;j<N;j++){
    for(int k=0;k<3;k++){
      sum += s[i] * t[k][j];
```

定数倍(ここでは3)は無視する

例

- これは、$O(\log N)$

```
while(N%3 == 0){
    N/=3;
    counter++;
}
```

$\log_3 N = \log N / \log 3$　... $\log 3$は定数

例

- これは、$O(N^2 + M)$

```
for(int i=0;i<N*N;i++) sum += s[i];
for(int i=0;i<N;i++) sum += t[i];
for(int i=0;i<M;i++) sum += u[i];
```

$$O(N^2) + O(N) + O(M) = O(N^2 + M)$$

競プロにおける計算量

- 競プロの問題は、TL 1sec, 2secが多い
- 最大の制約を代入してみて、
  - 10^7 ・・・だいたい間に合う
  - 10^8 ・・・定数倍が軽ければ間に合う

制約からの予測

- N <= 20　・・・ O(N * 2^N)
- N <= 100　・・・ O(N^3)
- N <= 100000 ・・・ O(N log N)
- N <= 10^18　・・・ O(log N)か、O(1)

有名アルゴリズムの計算量

$O(N^3)$

- ワーシャルフロイド(全点間最短距離)
- 行列積

$O(N^2)$

- バブルソート・挿入ソート

有名アルゴリズムの計算量

$O(N \log N)$

- クイックソート・マージソート

$O(N \log \log N)$

- エラトステネスの篩

有名アルゴリズムの計算量

$O(N)$

- 線形探索

$O(\sqrt{N})$

- 素数判定

有名アルゴリズムの計算量

$O(\log N)$

- 二分探索

- ユークリッドの互除法

- Union Find (頑張るとアッカーマン関数の逆関数)

- BIT・セグツリー

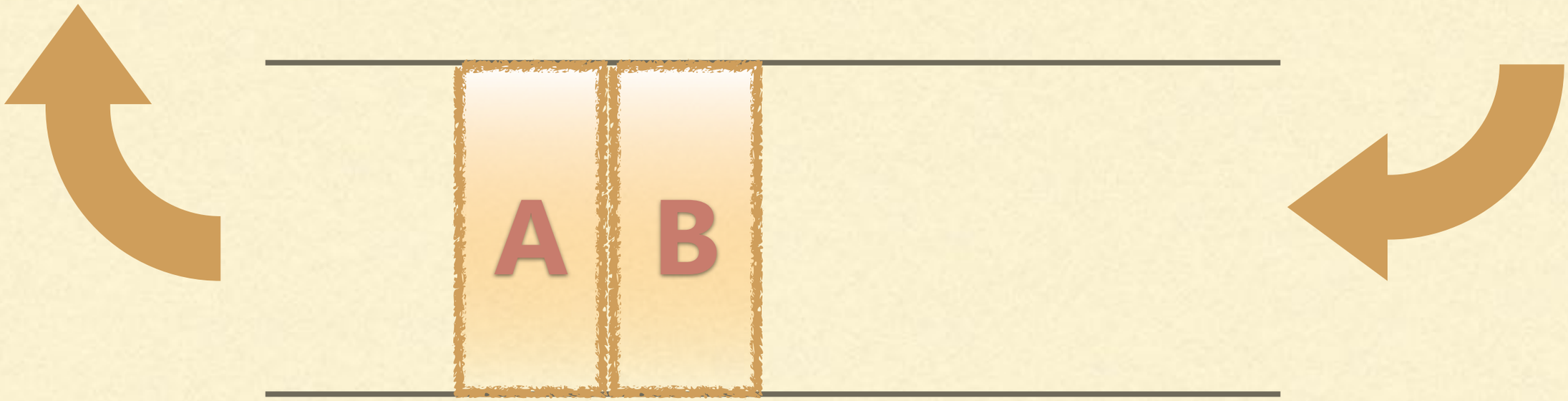
QUEUE(キュー)

C++はstd:queue

QUEUE(キュー)

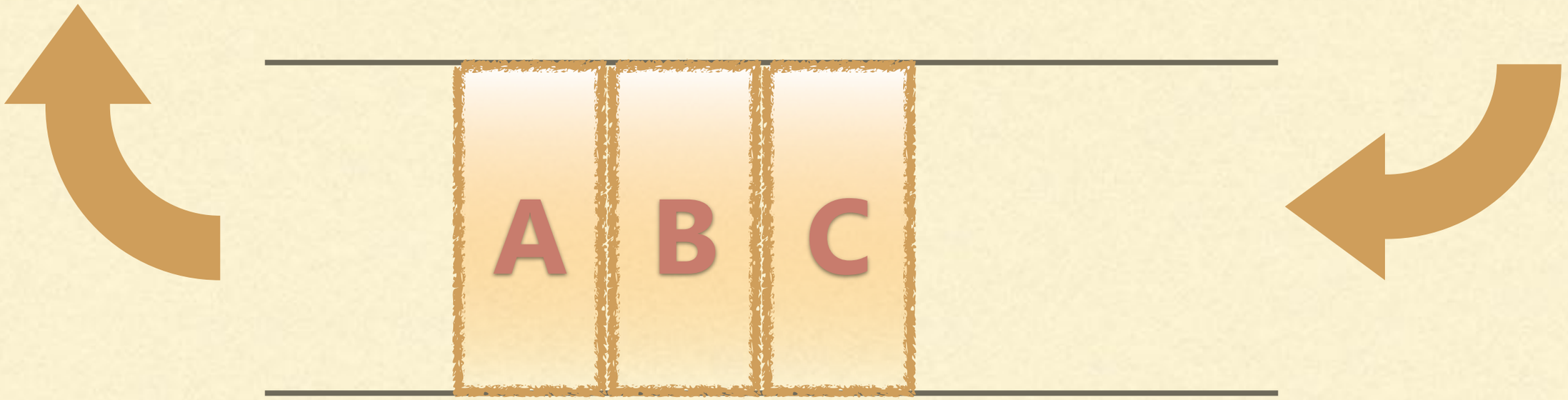
C++はstd:queue

QUEUE(キュー)



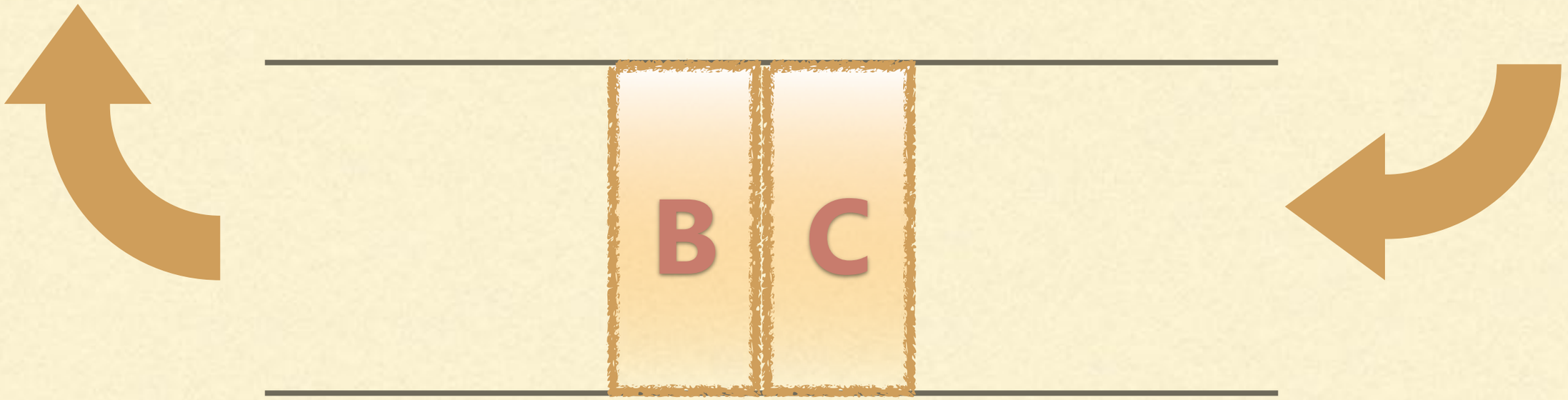
C++はstd:queue

QUEUE(キュー)



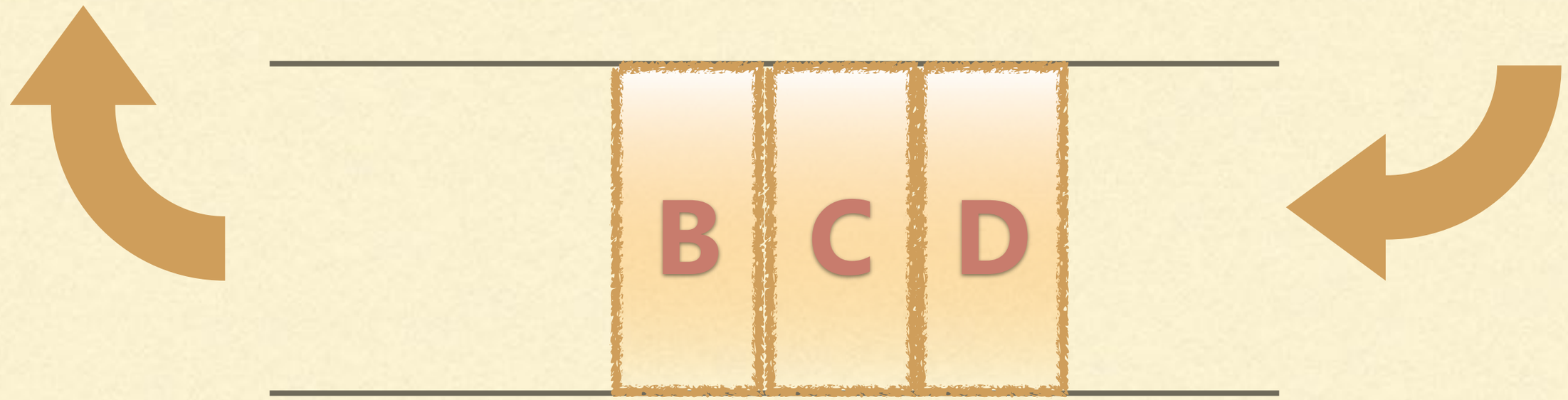
C++はstd:queue

QUEUE(キュー)



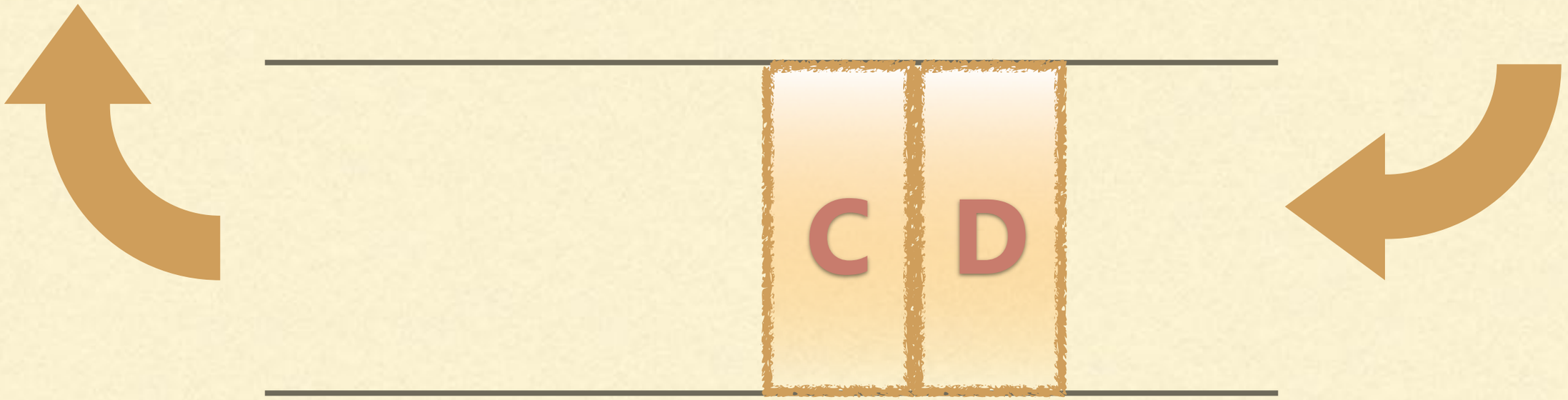
C++はstd:queue

QUEUE(キュー)



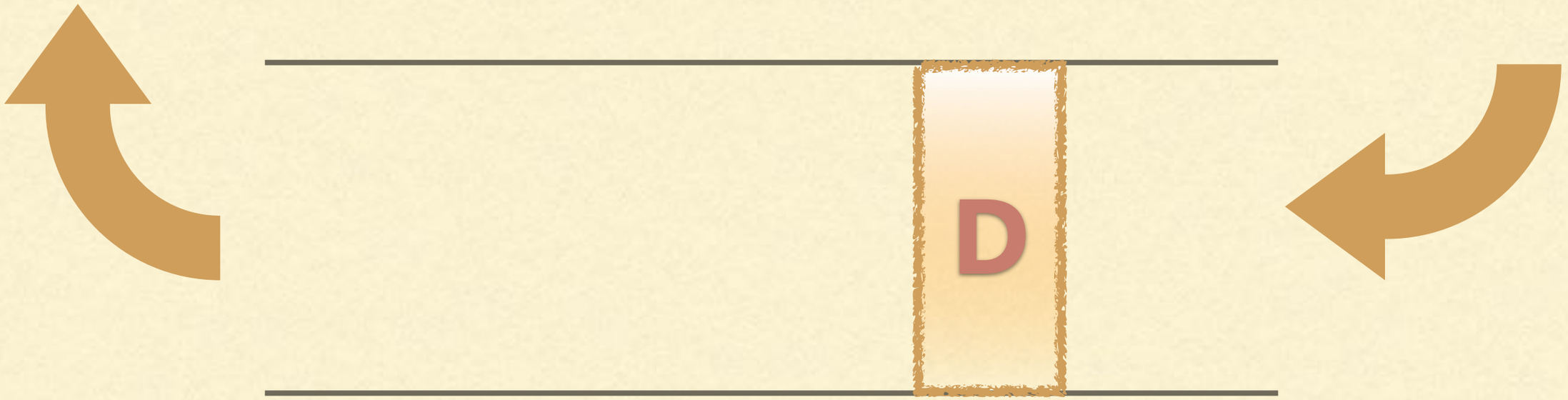
C++はstd:queue

QUEUE(キュー)



C++はstd:queue

QUEUE(キュー)

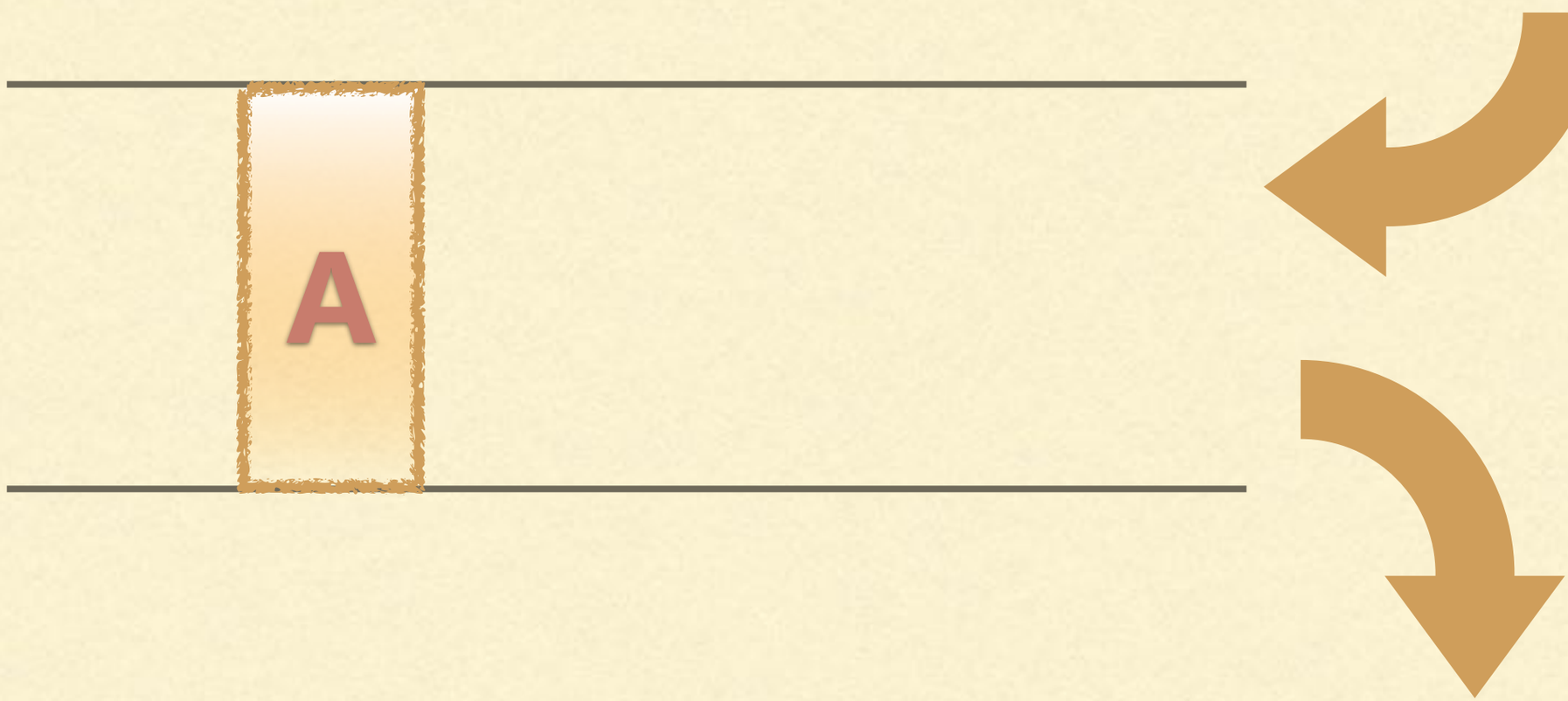


C++はstd:queue

STACK(スタック)

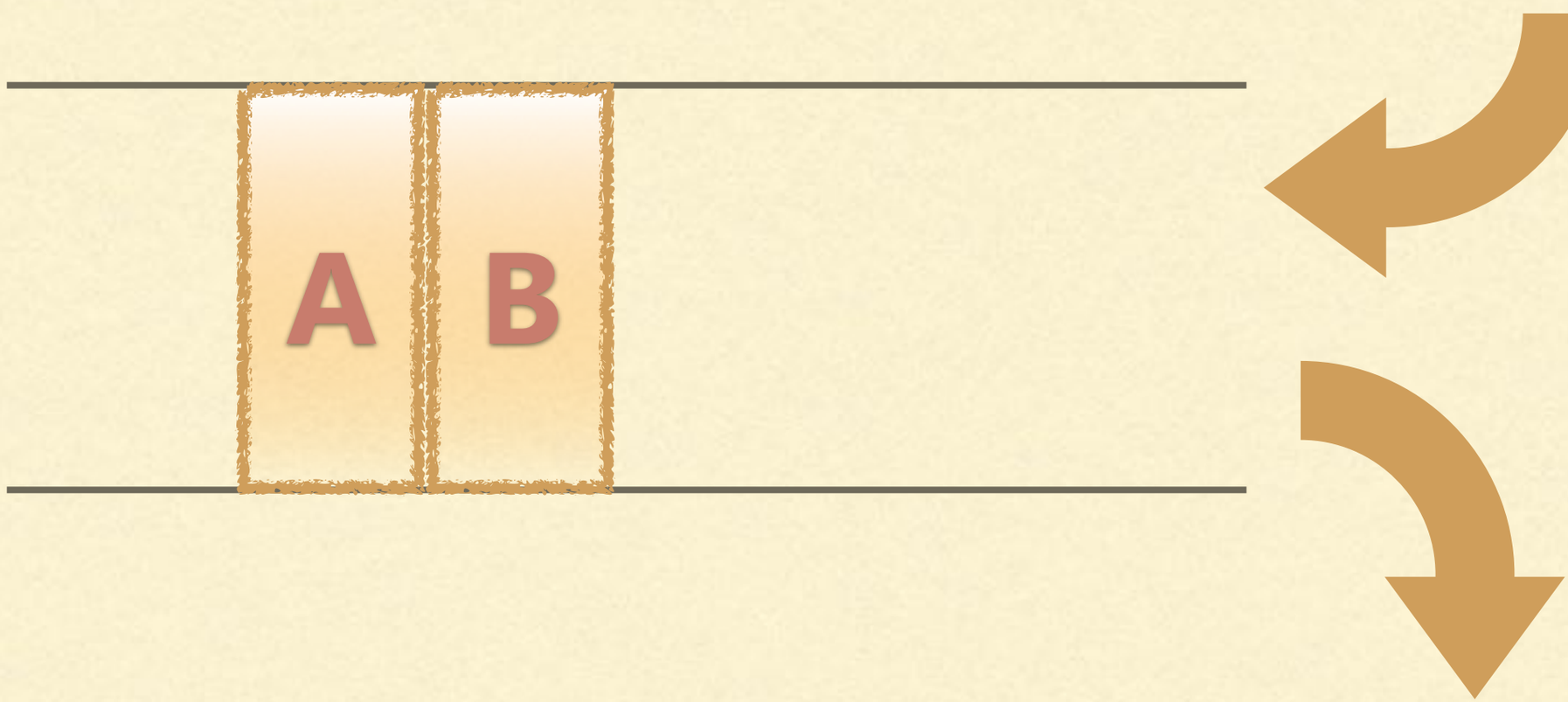
C++はstd:stack

STACK(スタック)



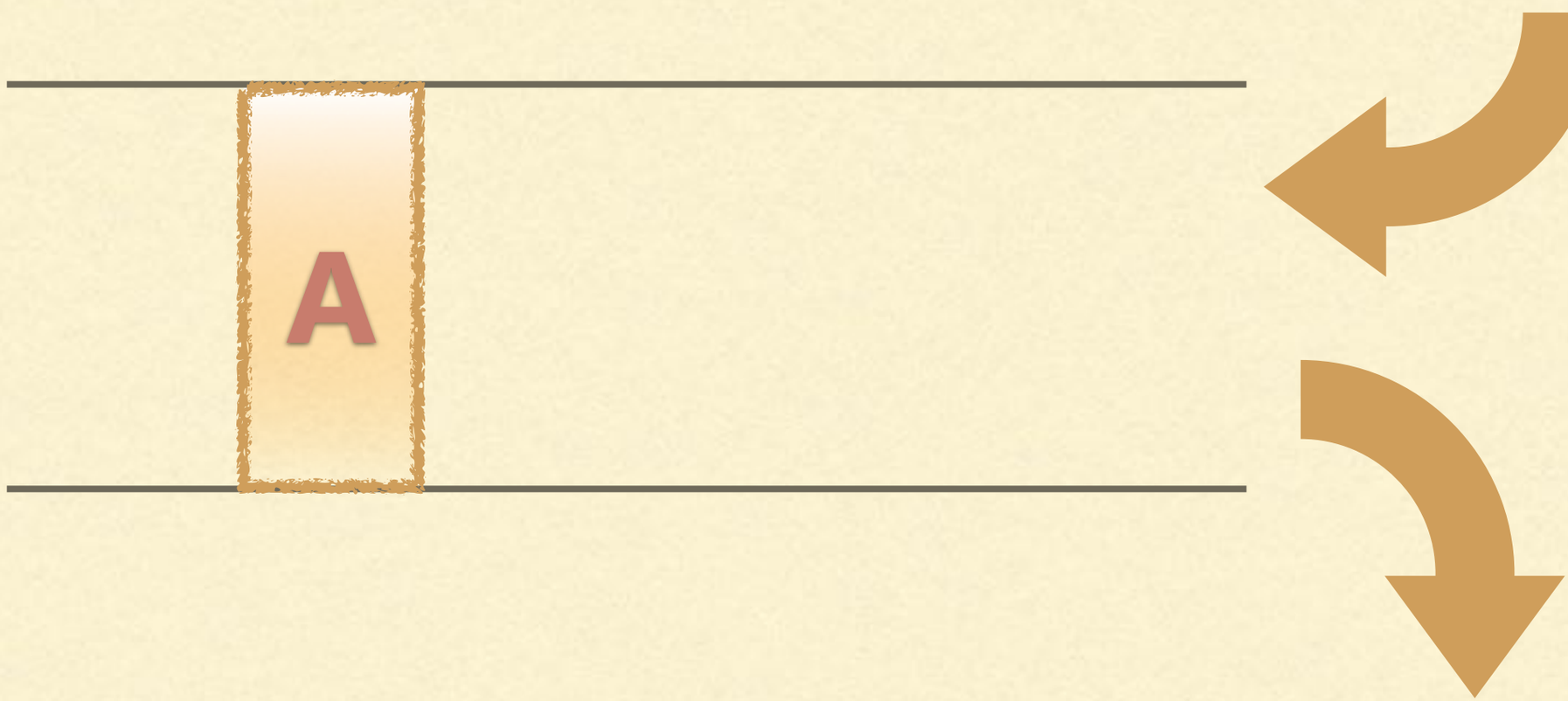
C++はstd:stack

STACK(スタック)



C++はstd:stack

STACK(スタック)

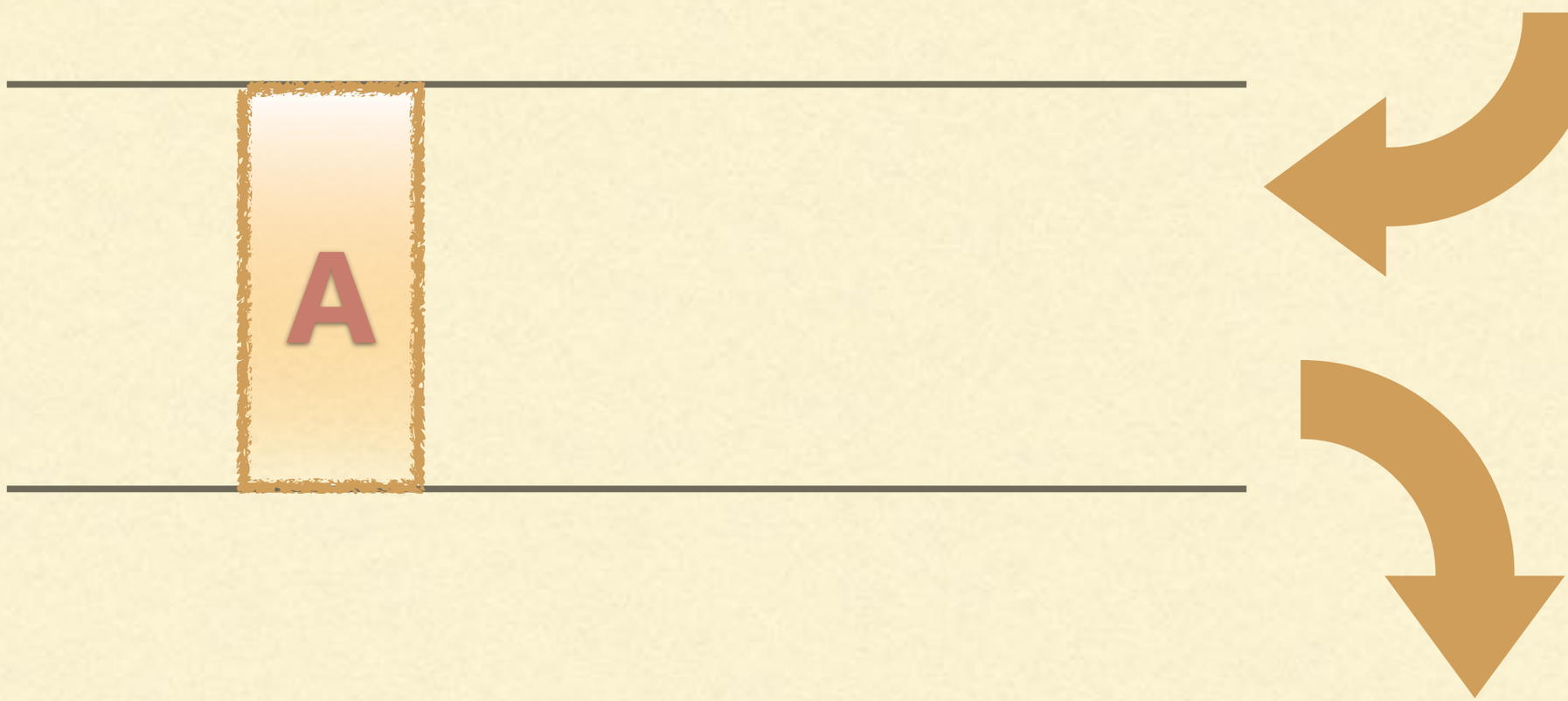


C++はstd:stack

STACK(スタック)

C++はstd:stack

STACK(スタック)



C++はstd:stack

STACK(スタック)

C++はstd:stack

STACK(スタック)

C++はstd:stack

STACK(スタック)

C++はstd:stack

STACKの実装

```
int stack[SIZE], top = 0;

void push(int x) { stack[top++] = x; }
void pop() { assert(top > 0); top--; }
int getTop(){ assert(top > 0); return stack[top-1];}
```

SIZE = stackが保持する最大数

C++はstd:stack

QUEUEの実装

```
int queue[SIZE], back = 0, front = 0;

void push(int x) { queue[back++] = x; }
void pop() { assert(front < back); front++; }
int getFront() { assert(front < back); return queue[front]; }
```

SIZE = queueにpushする要素数

(循環型にすると最大数になる)

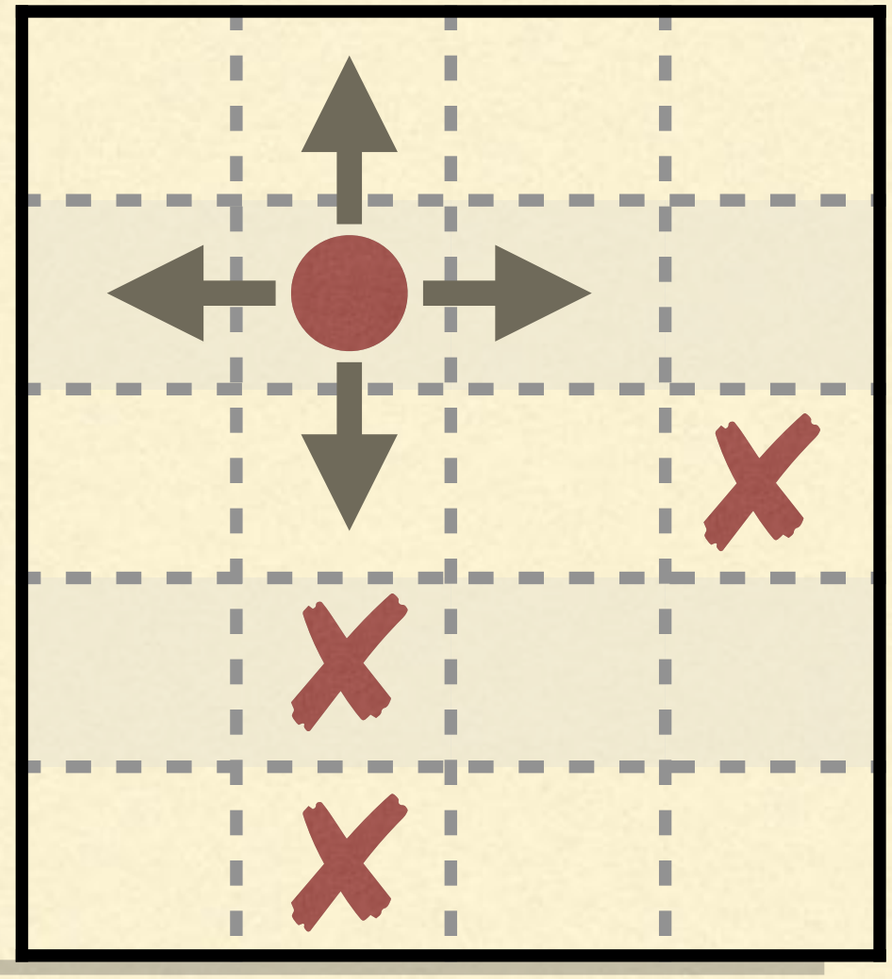
C++はstd:queue

幅優先探索

(BREADTH FIRST SEARCH)

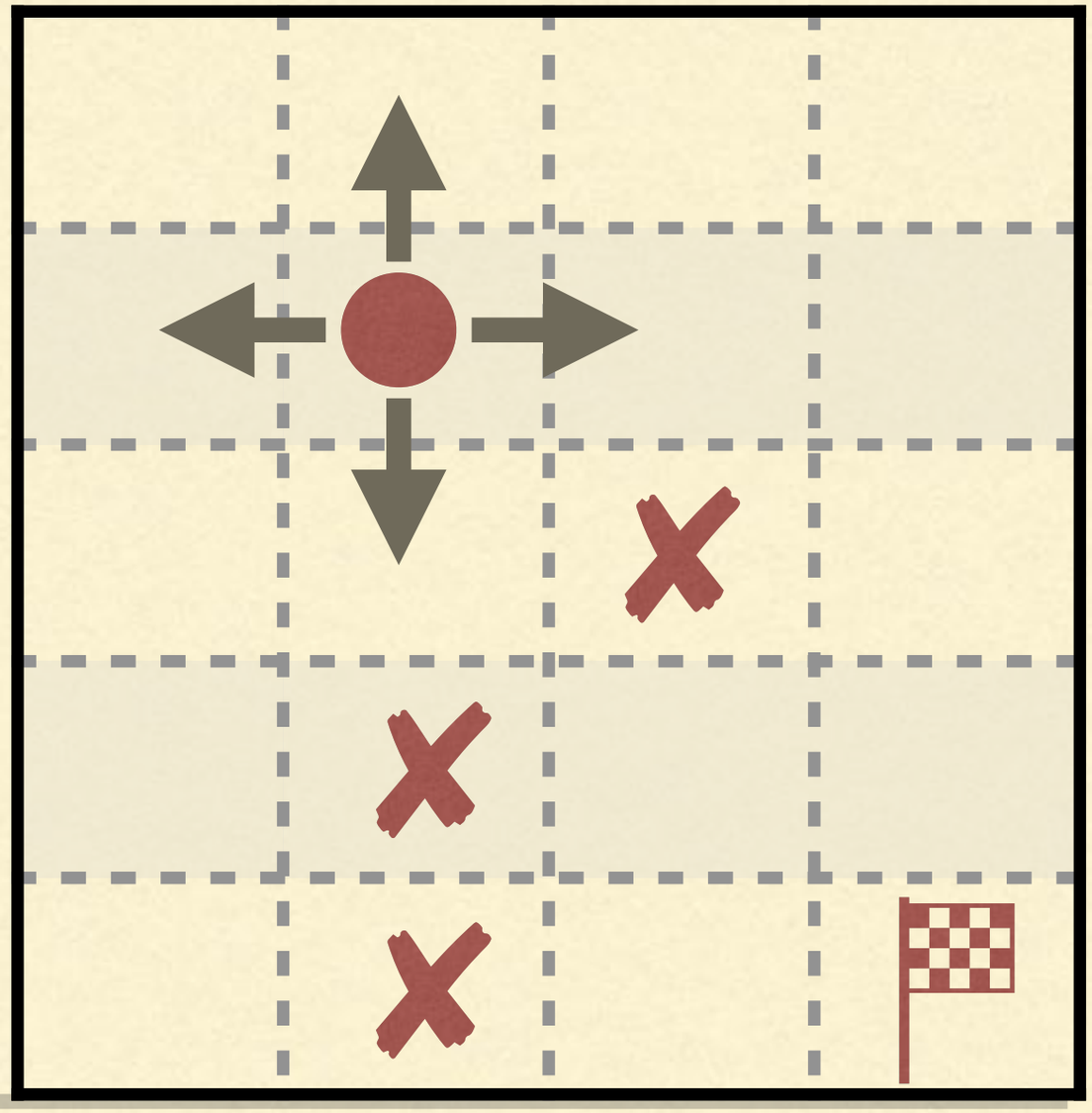
問題設定

- H×Wの格子上の盤面があり、その盤面上で
  コマを動かすことを考えます。
- コマは、1回の移動で左右上下の
  隣接マスへ動かすことが可能です。
- 盤面上に設定されている障害物のマス
  と盤面外には移動できません。



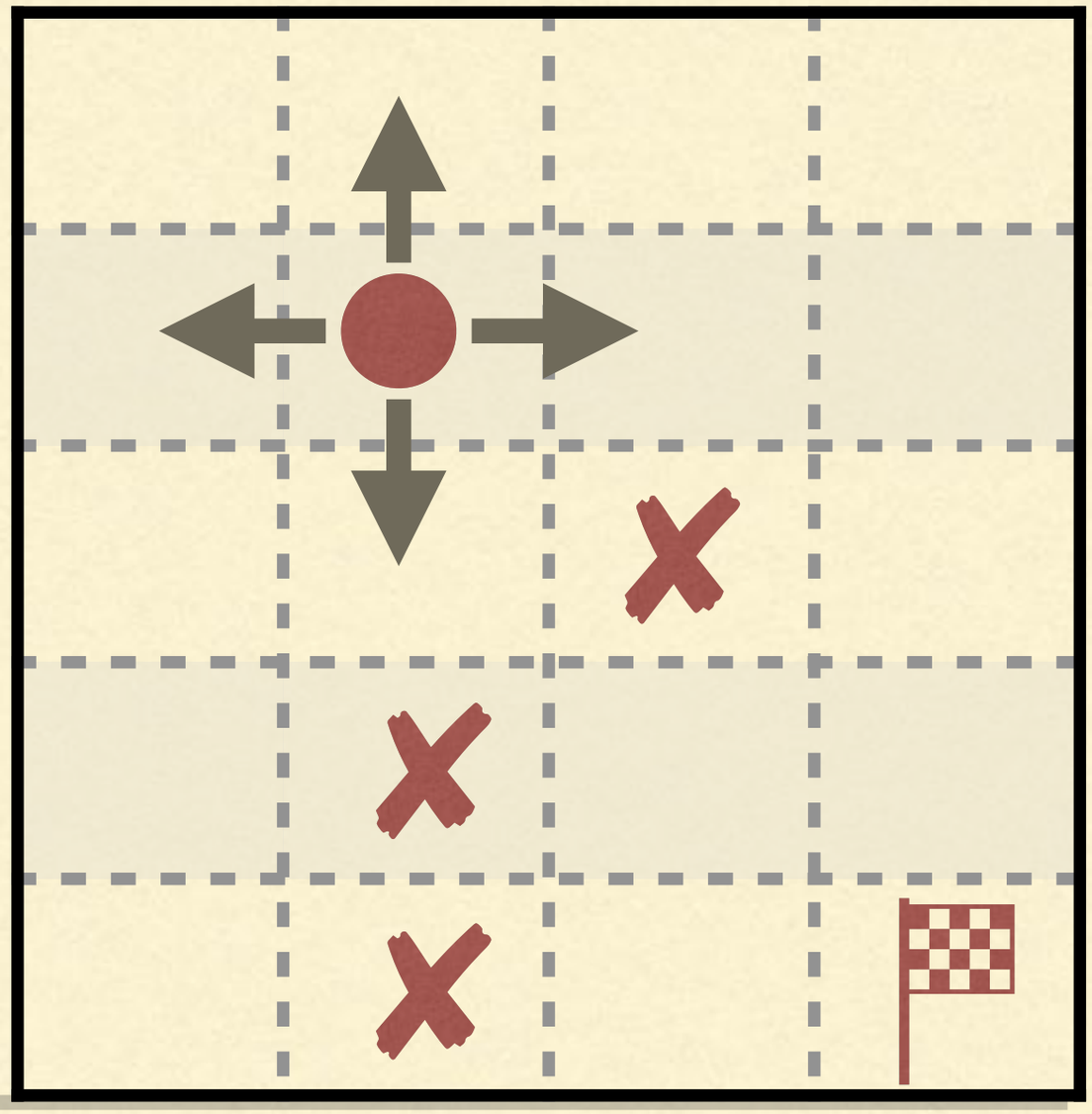
問題1-1

- 盤面状にスタートマスとゴールマス
  が設定されています。
- スタートからゴールまでの
  最小移動回数を求めて下さい。
- H≤2000,W≤2000



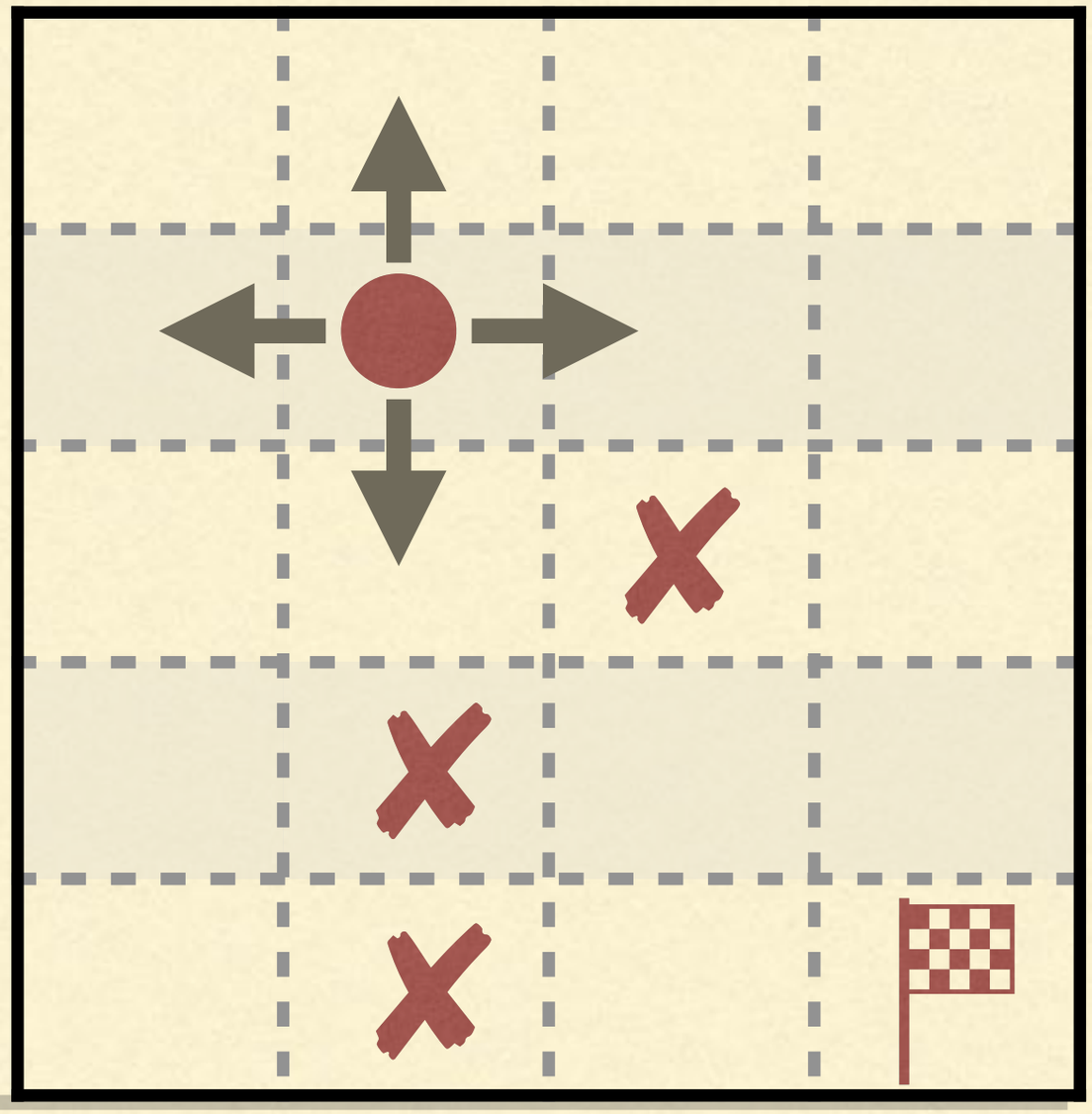
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



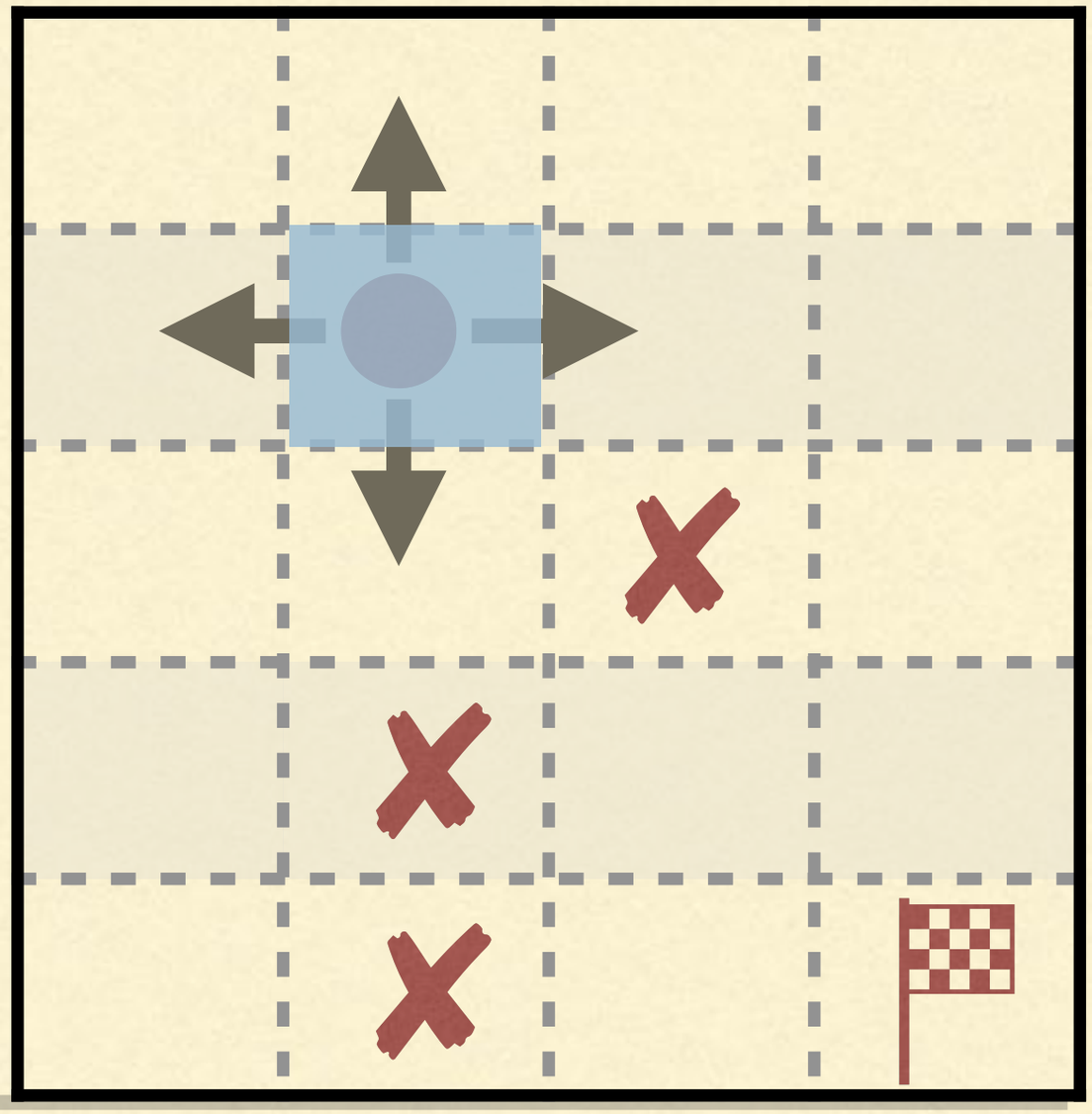
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



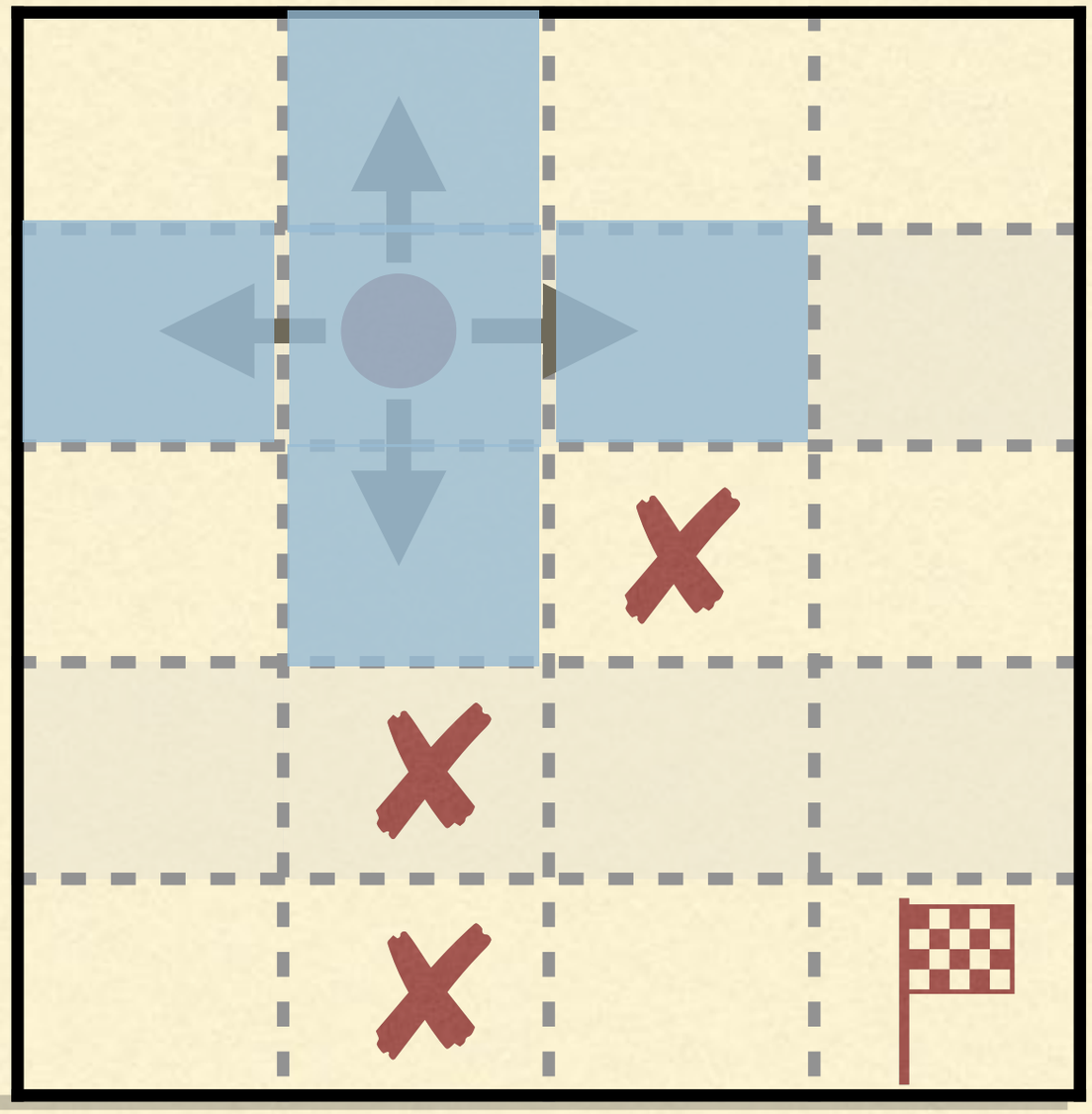
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



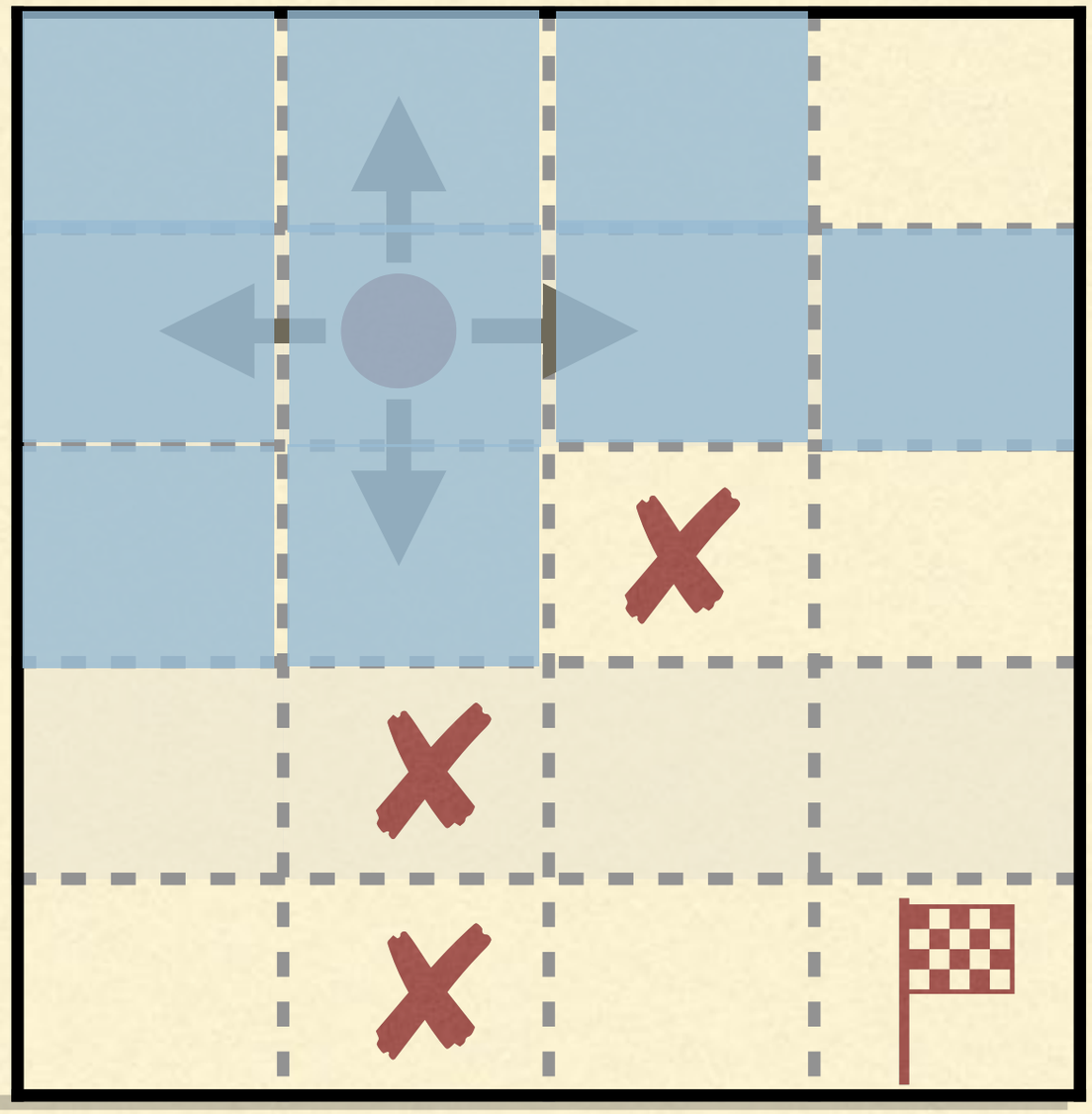
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



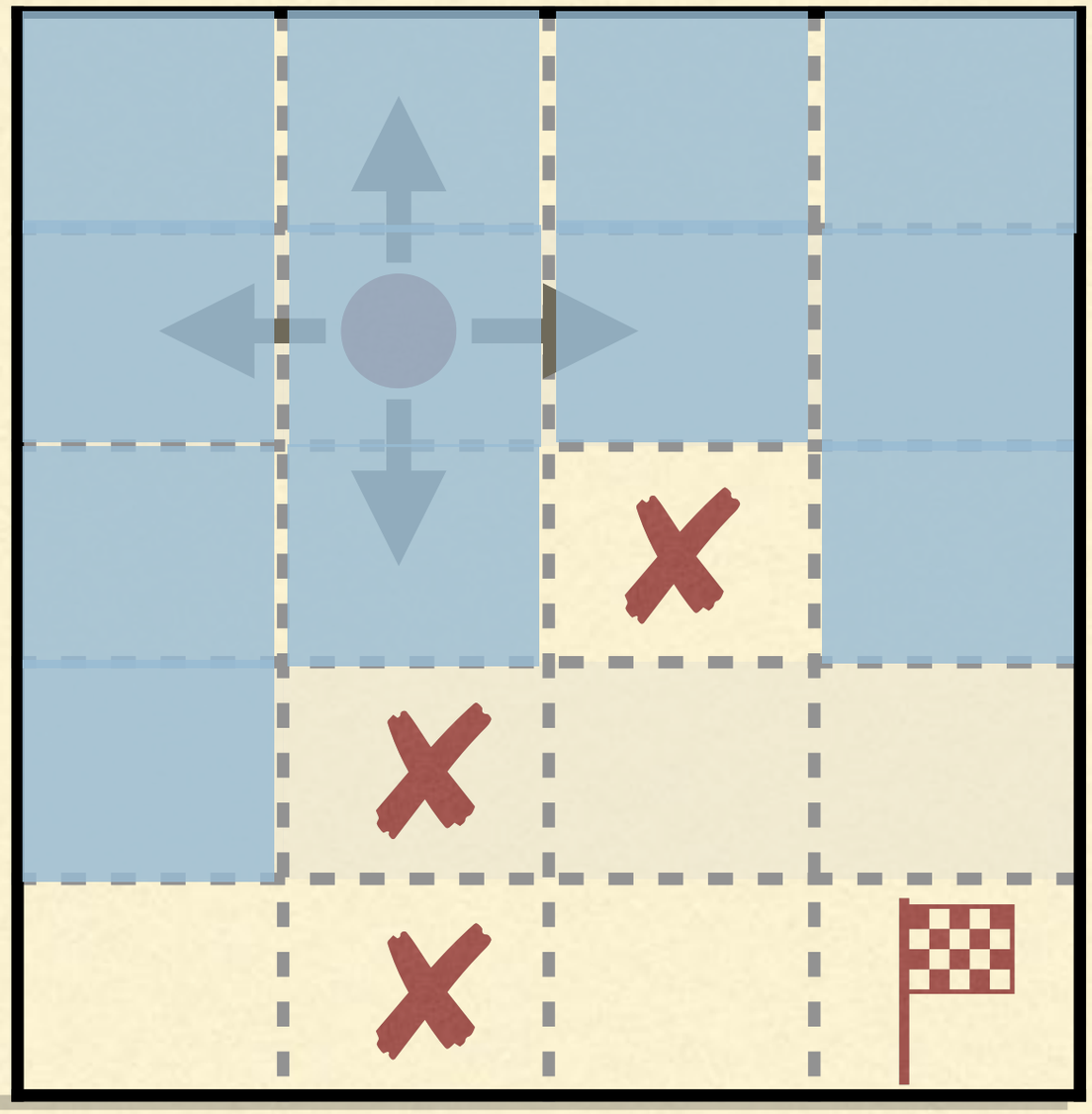
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



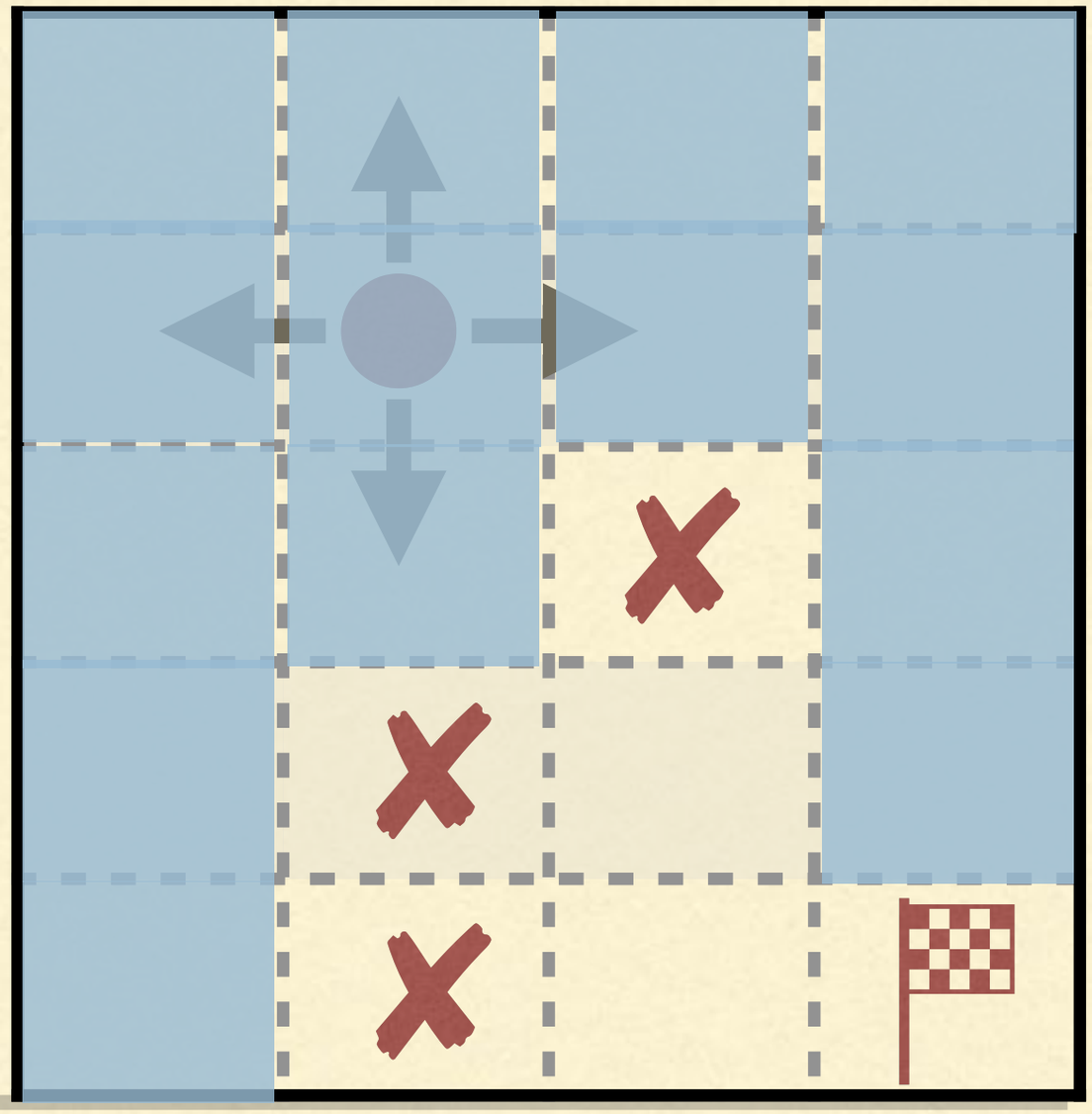
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



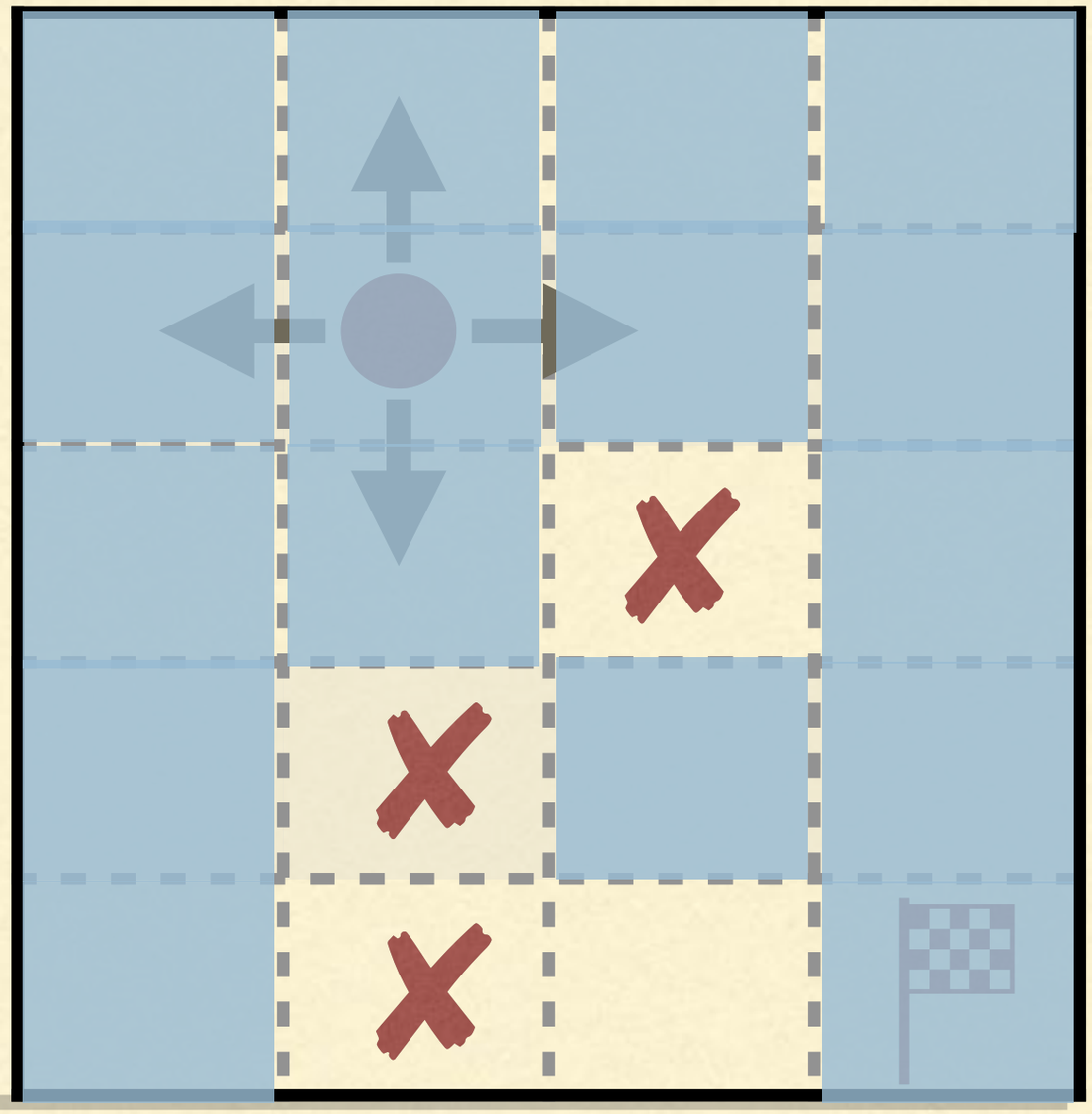
BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)



BFSとは、

- 距離が近いところから
  順番に見る手法
- キュー (待ち行列) を使用
- 計算量 O(HW)

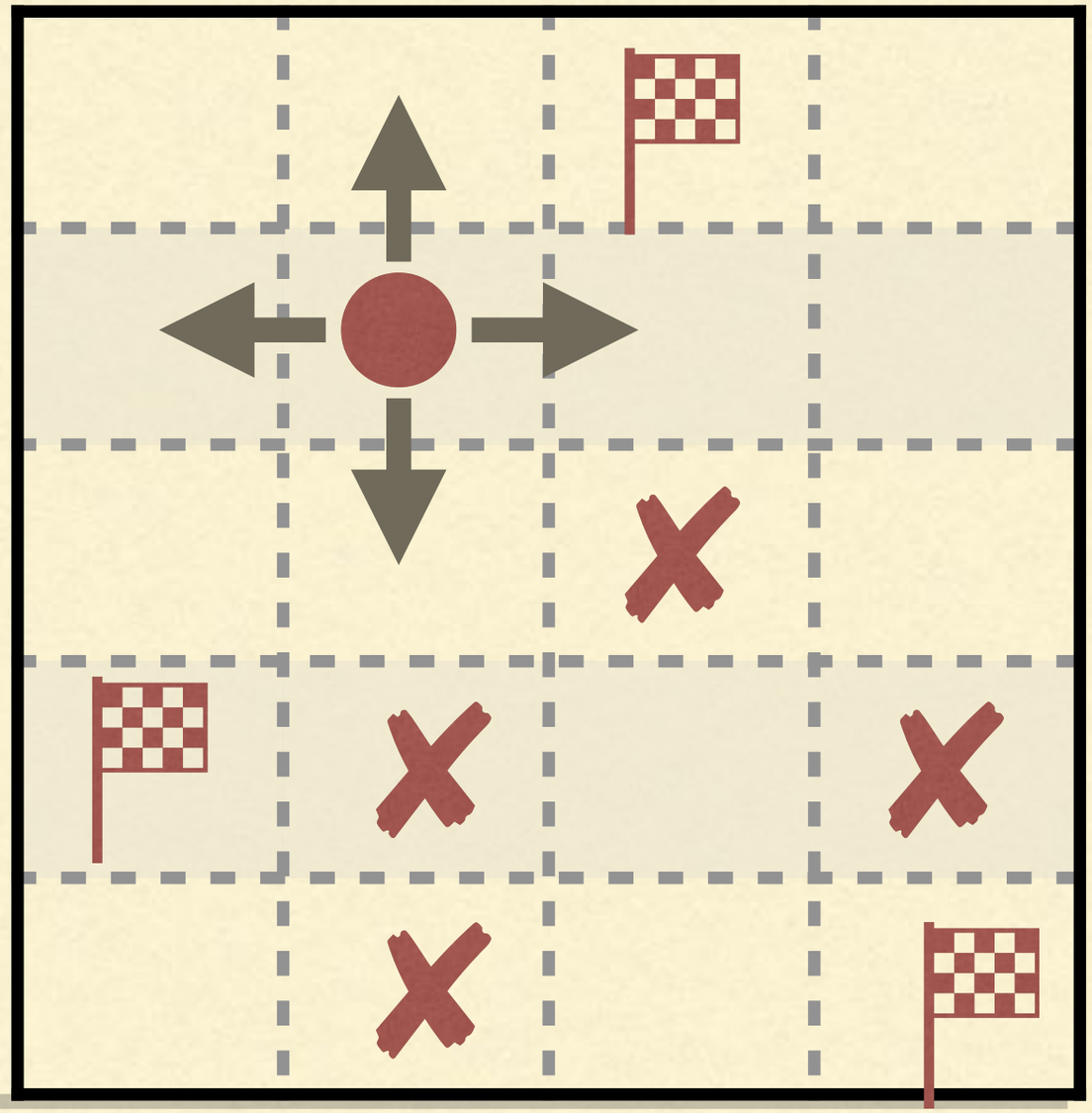


BFSの擬似コード

- while キューから一つ取る
    - すでに訪れている場合は continue;
    - 訪れたフラグを立てる
    - ゴールである break;
    - その場所から移動できる全ての地点をキューに入れる

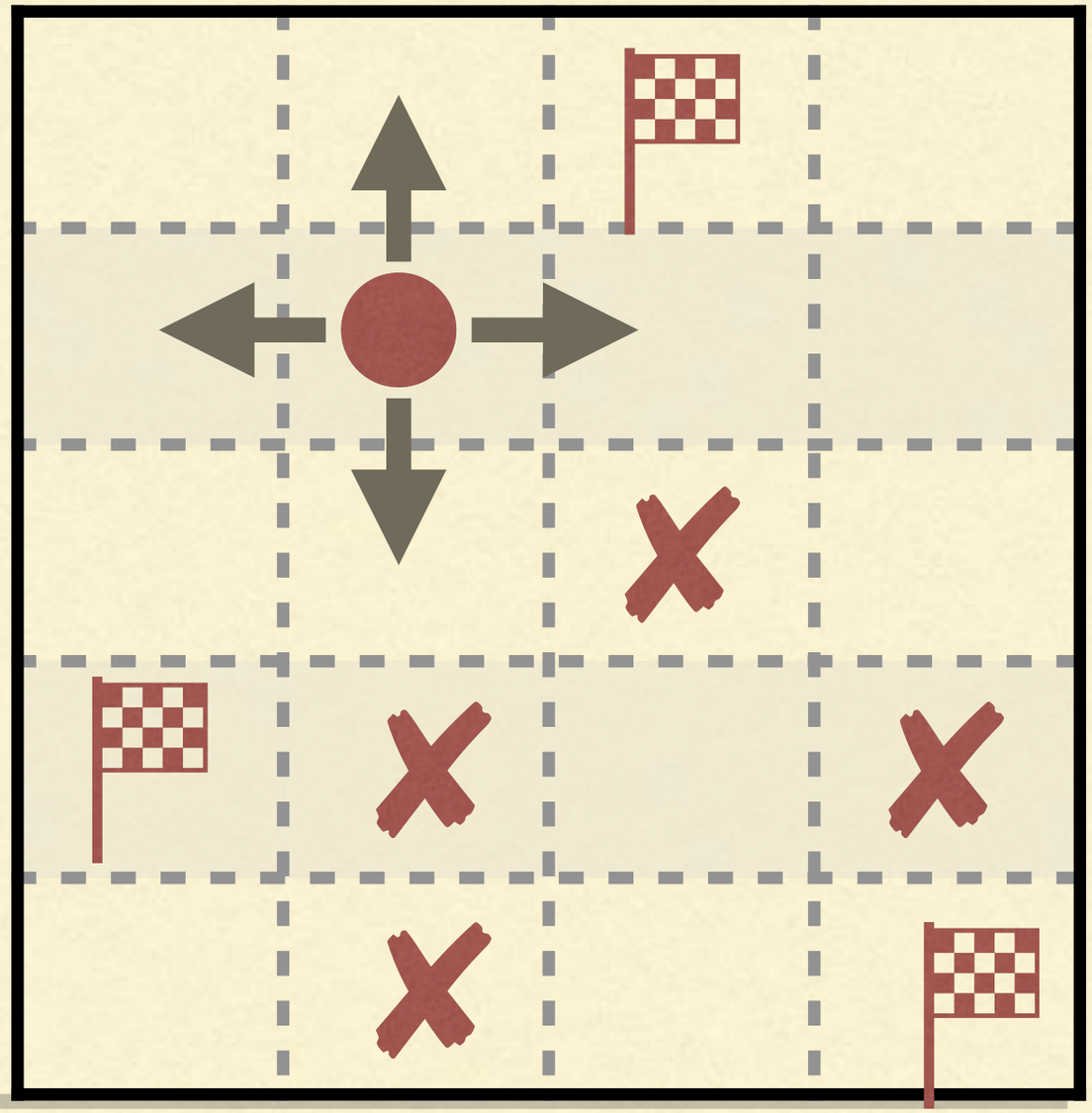
問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000



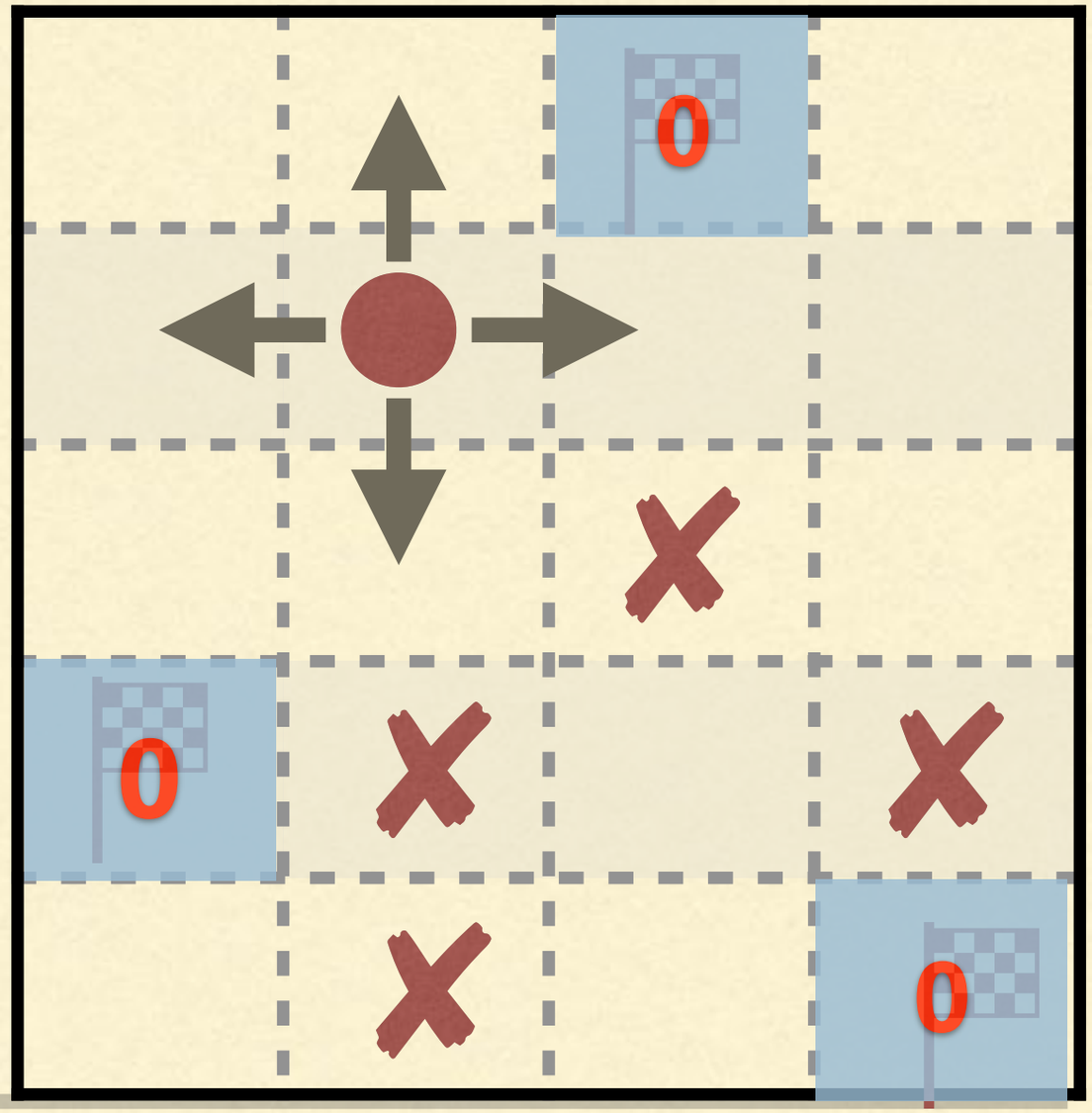
問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000



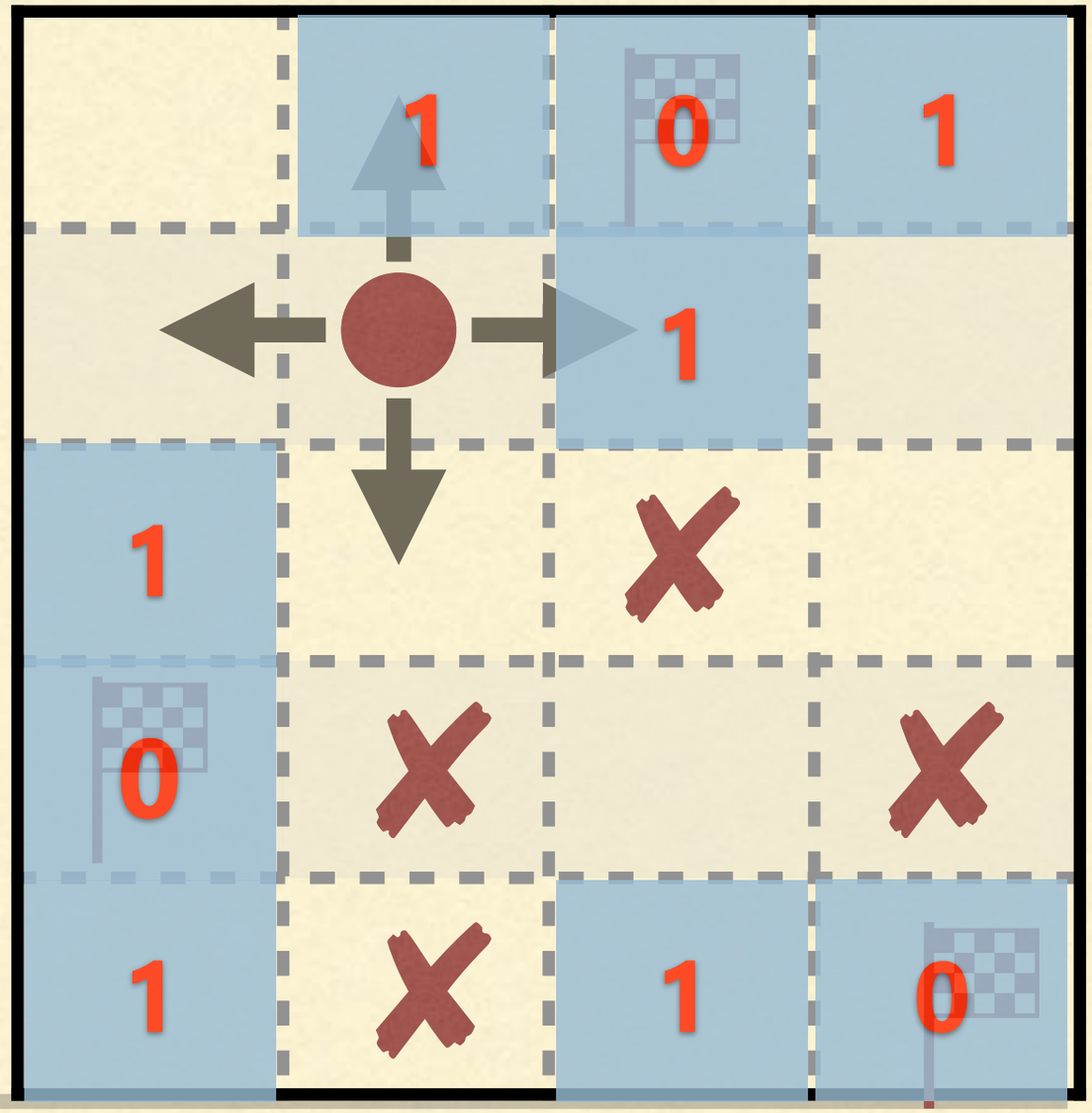
問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000



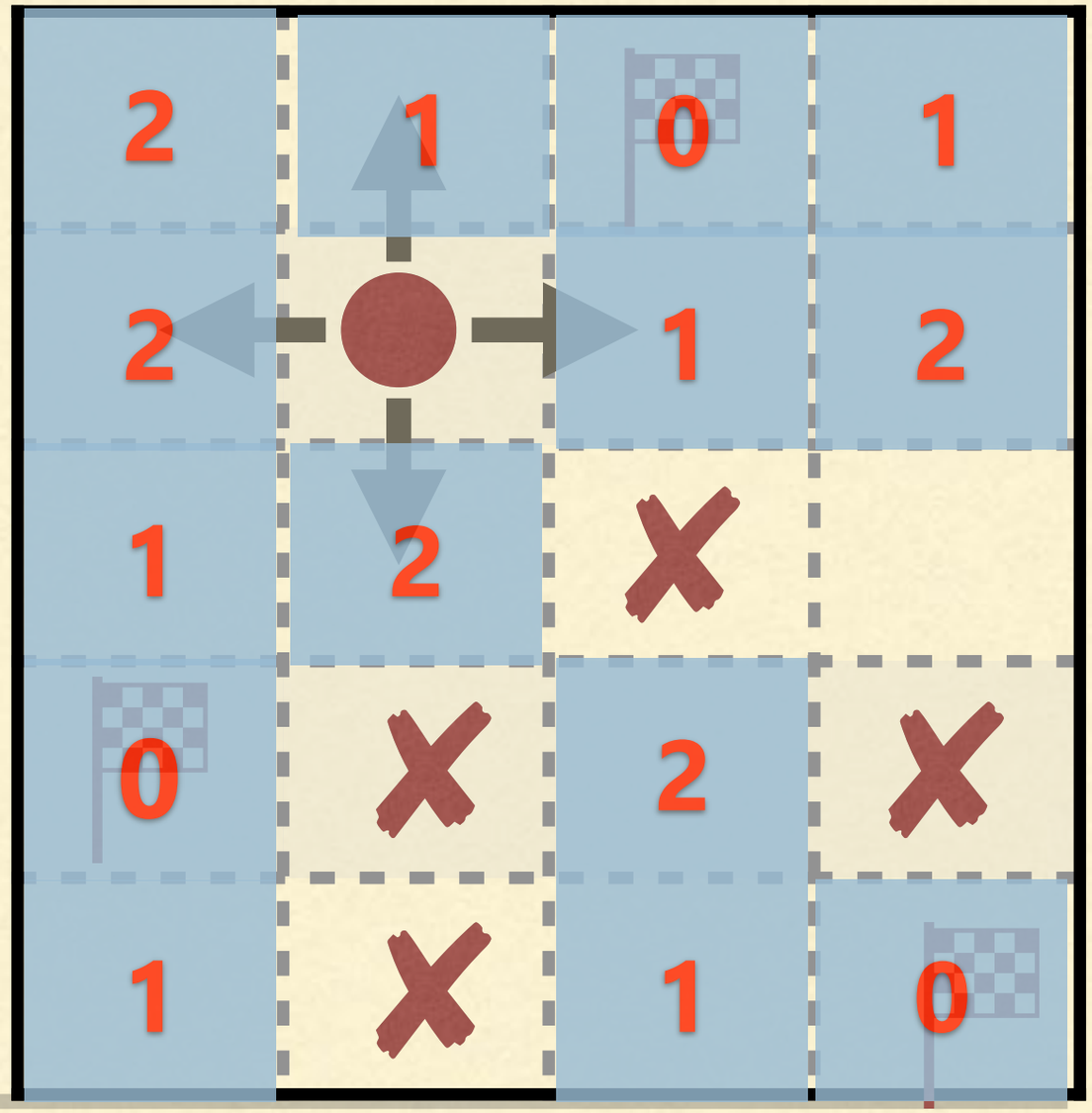
問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000



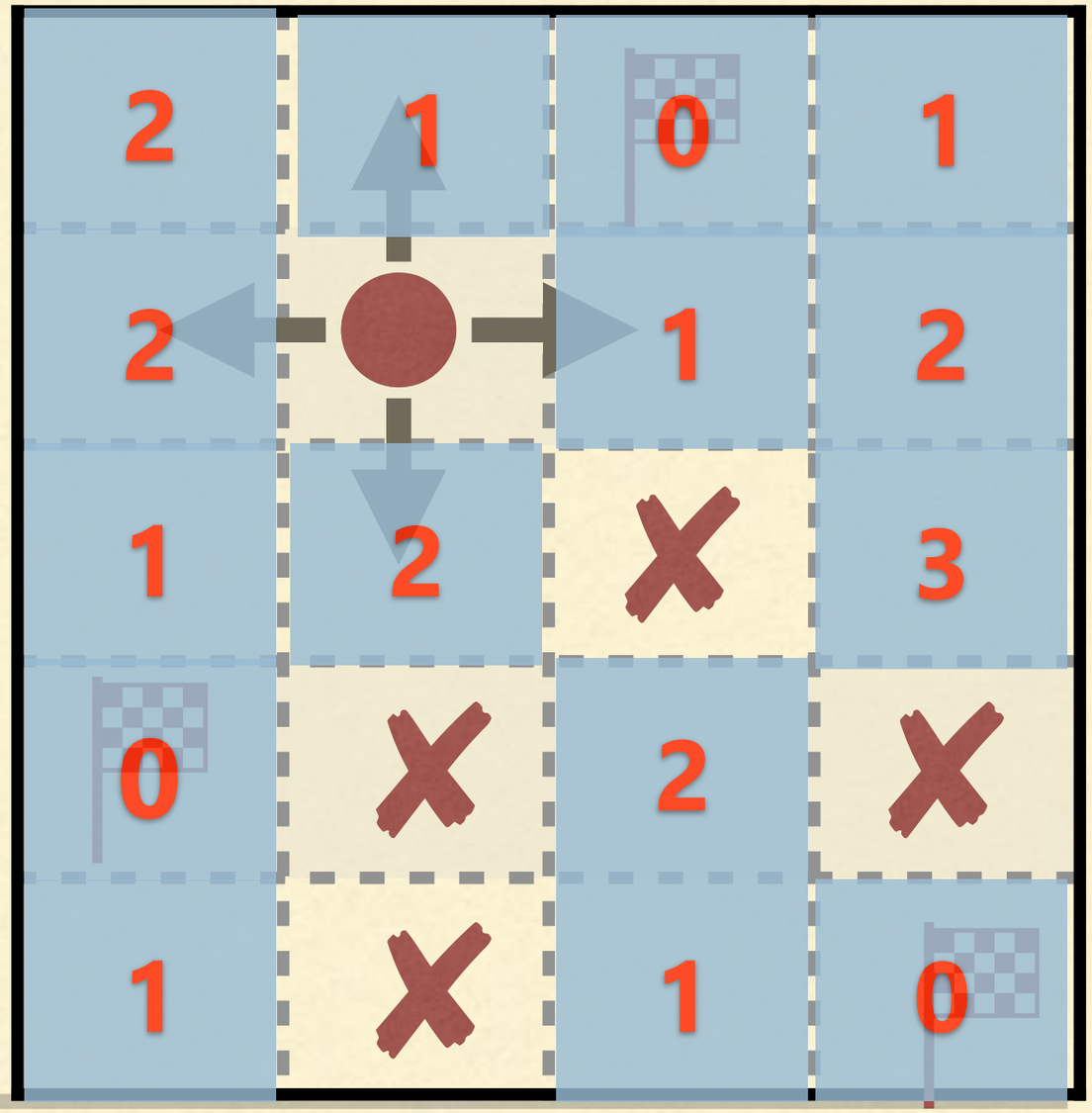
問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000



問題1-2

- 盤面状にN個の異なるゴールマスg_iがある。
- 各マスに対して最も近いゴールとその距離を求めて下さい。
- 複数のゴールへの距離が同じ場合は、ゴールマスの番号が早いものを答えとして下さい。
- N≤HW,H≤2000,W≤2000

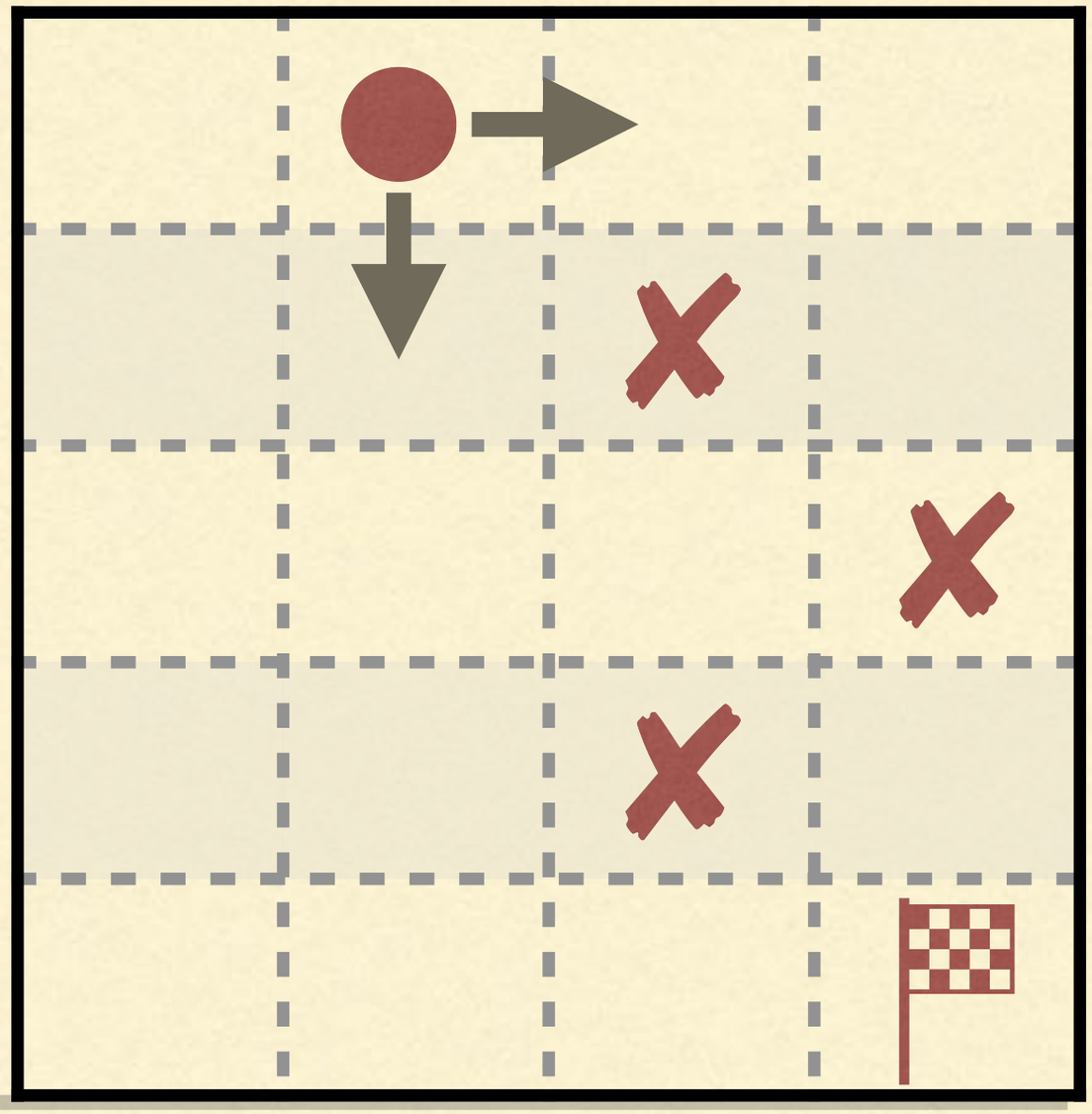


深さ優先探索

(DEPTH FIRST SEARCH)

問題2-1

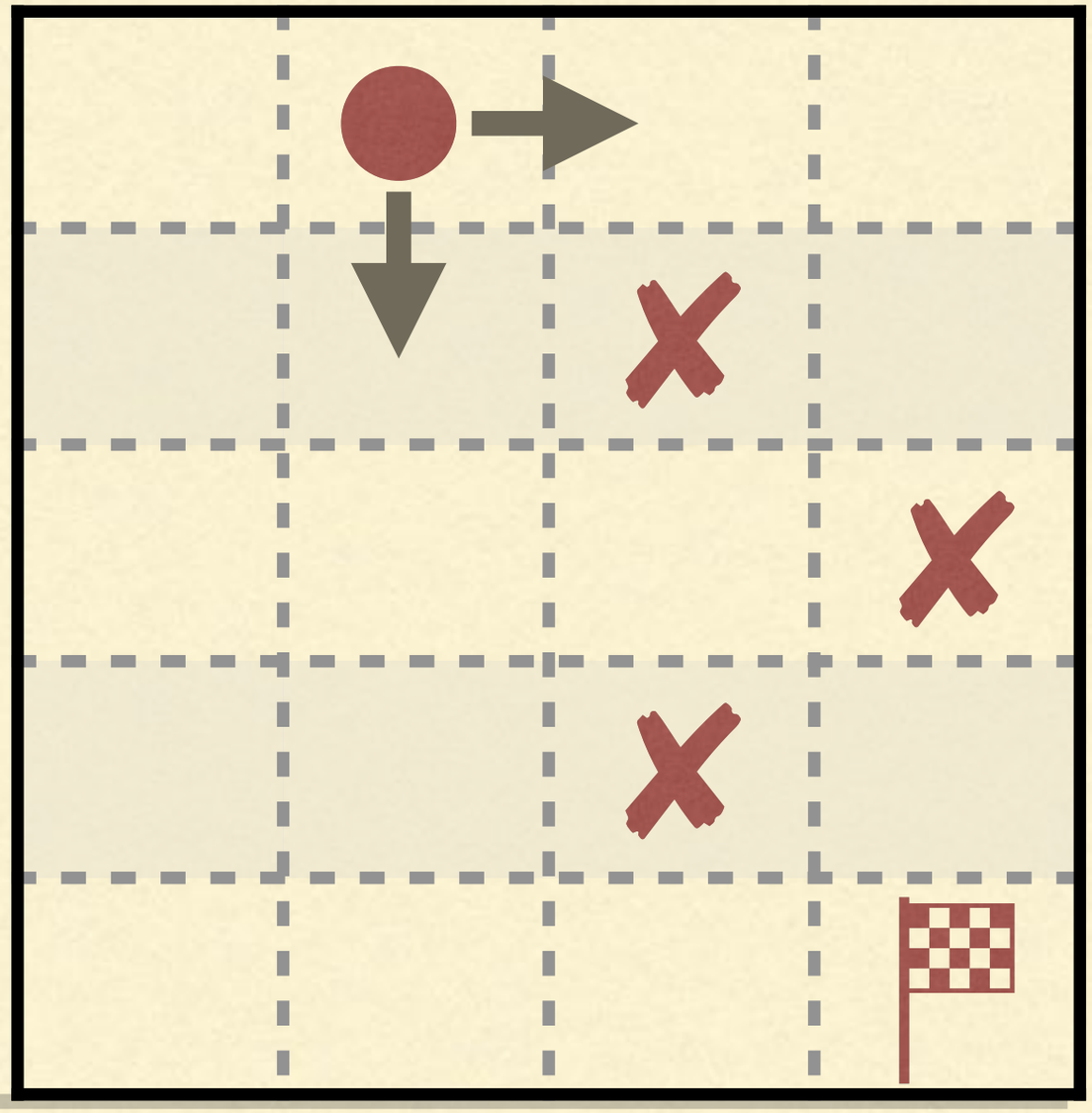
- 盤面状にスタートマスとゴールマス
  が設定されています。
- スタートからゴールまでの
  経路を一つ求めて下さい。
- H≤11,W≤11



DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って
  できなくなったら戻る)
- スタックか再帰を使う



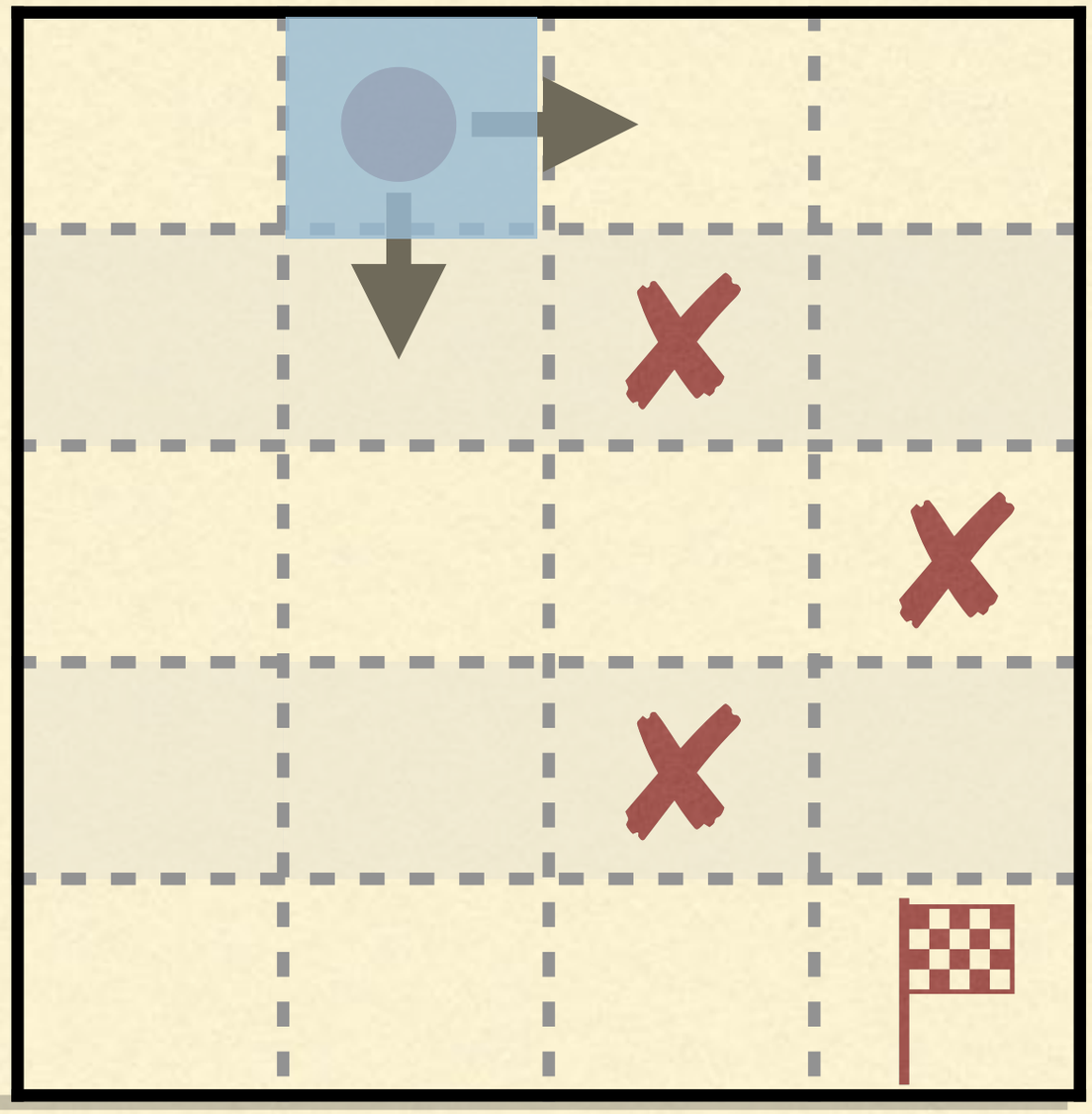
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



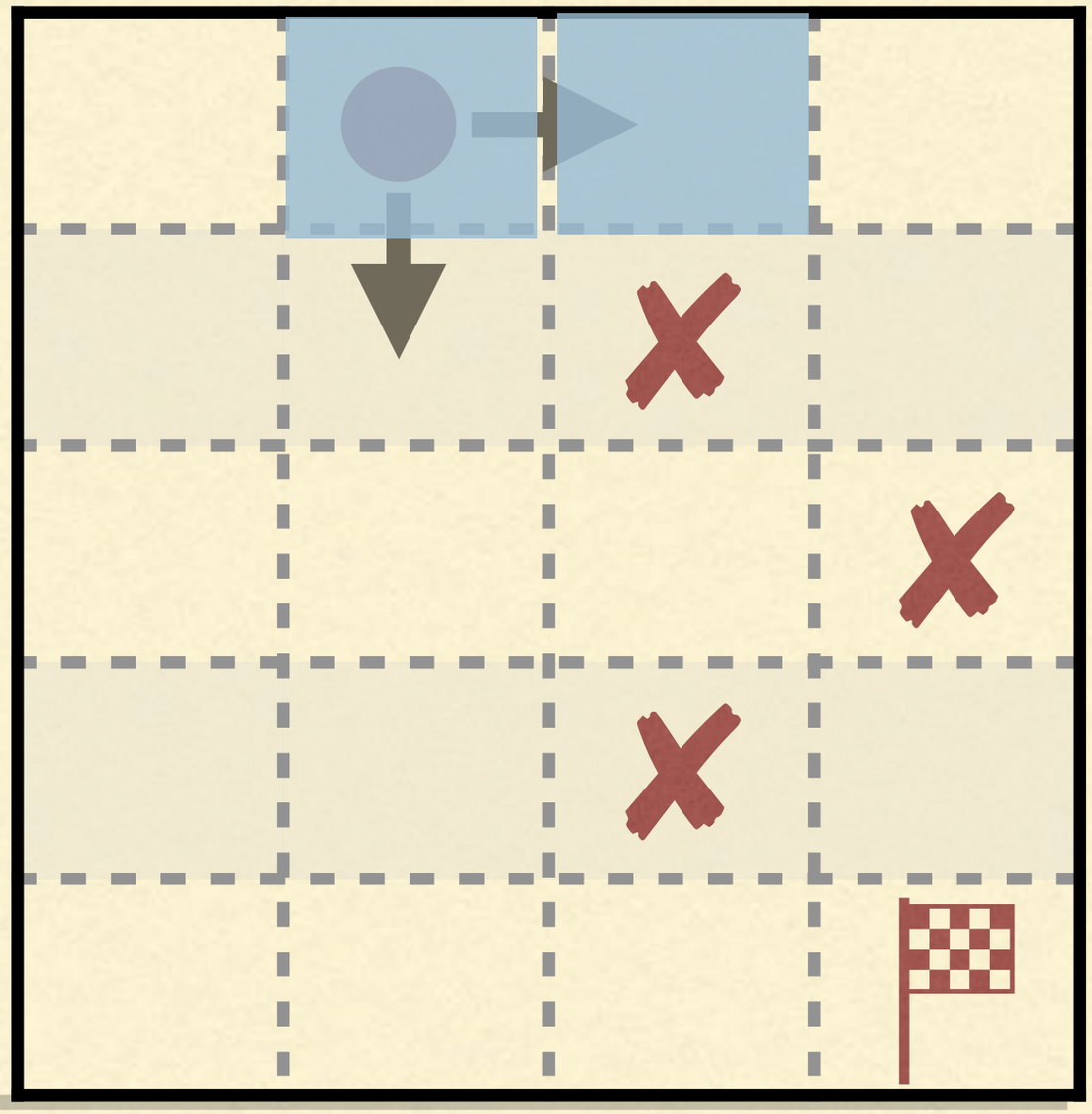
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



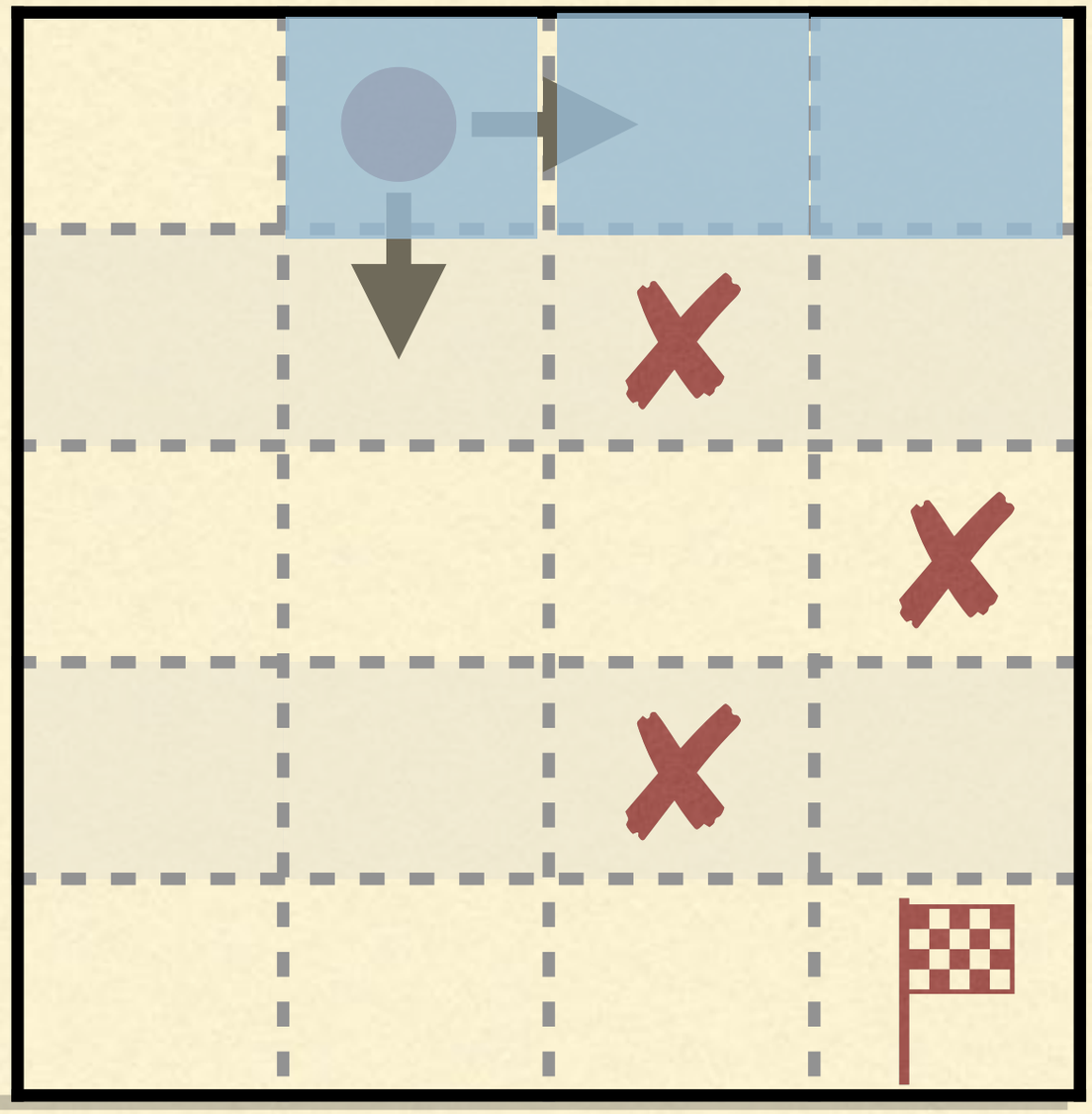
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



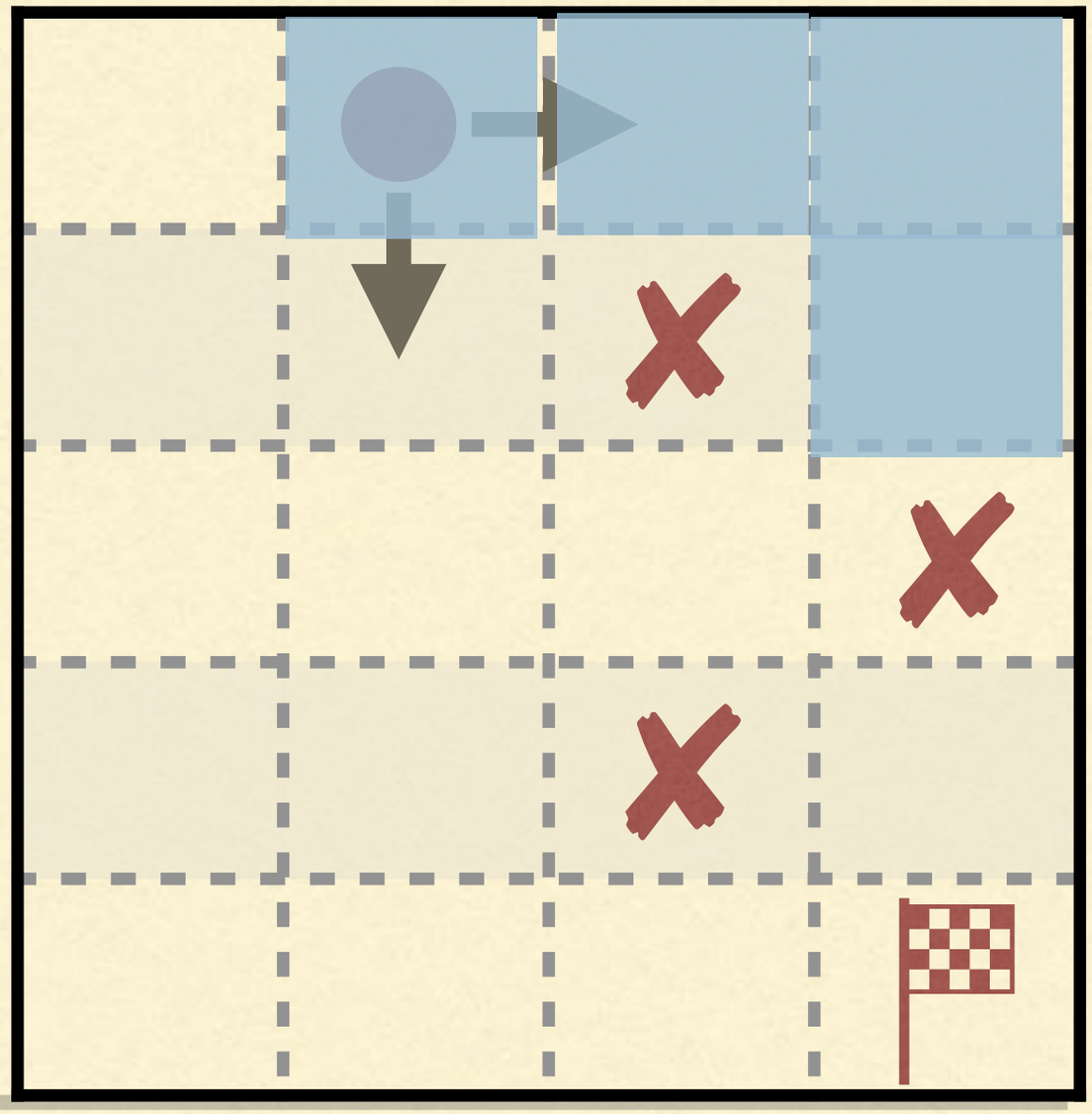
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



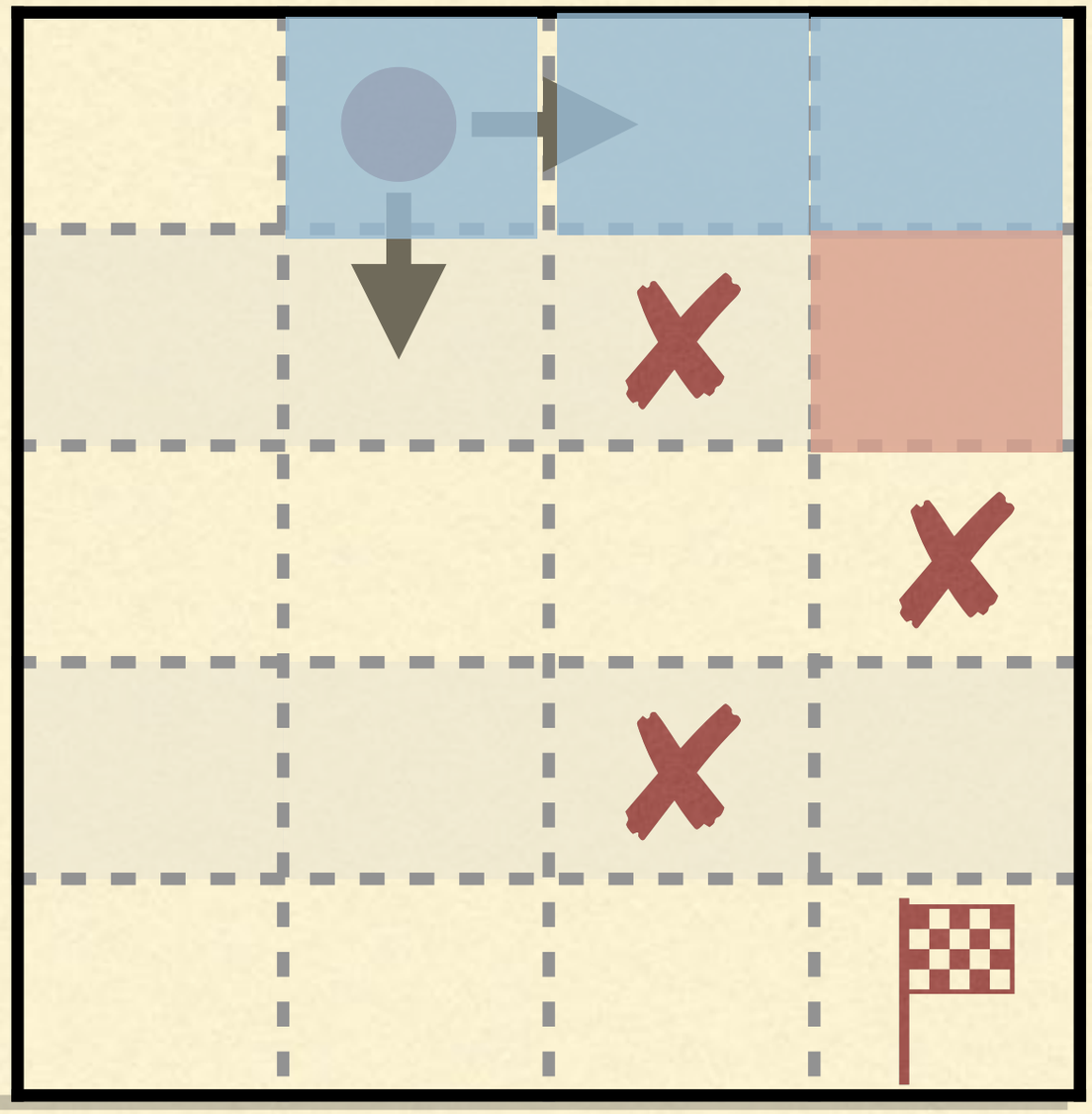
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



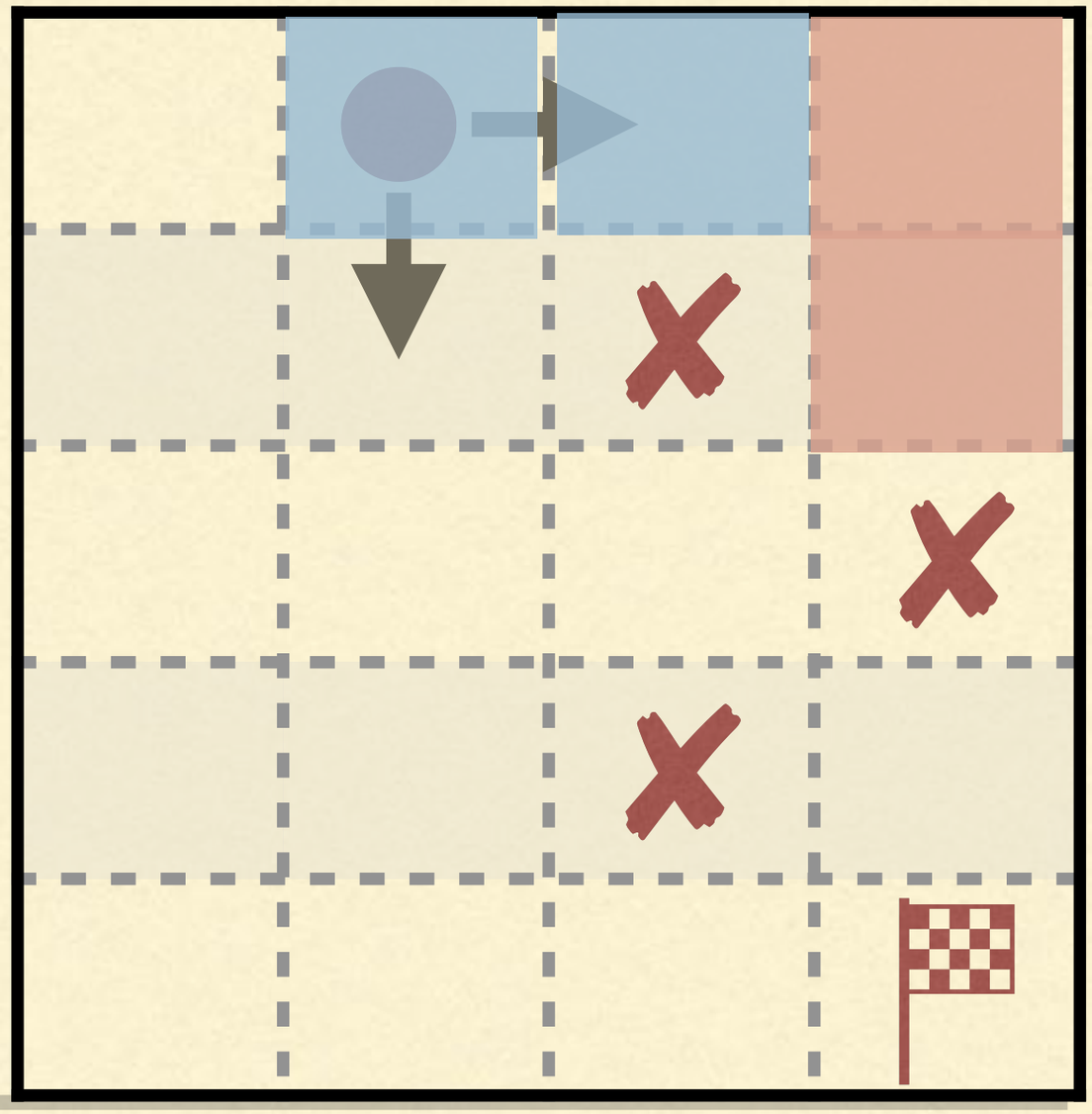
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



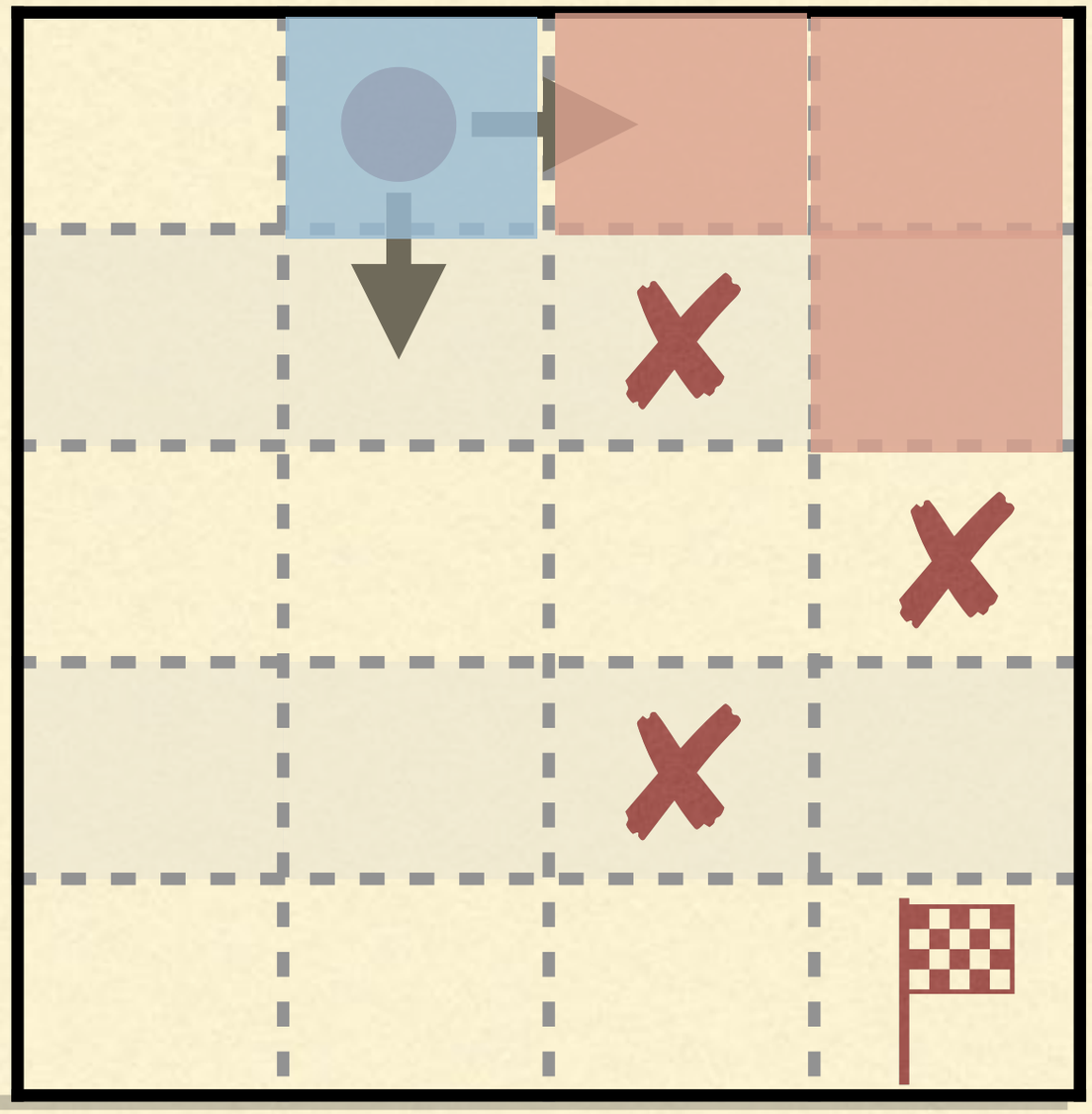
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



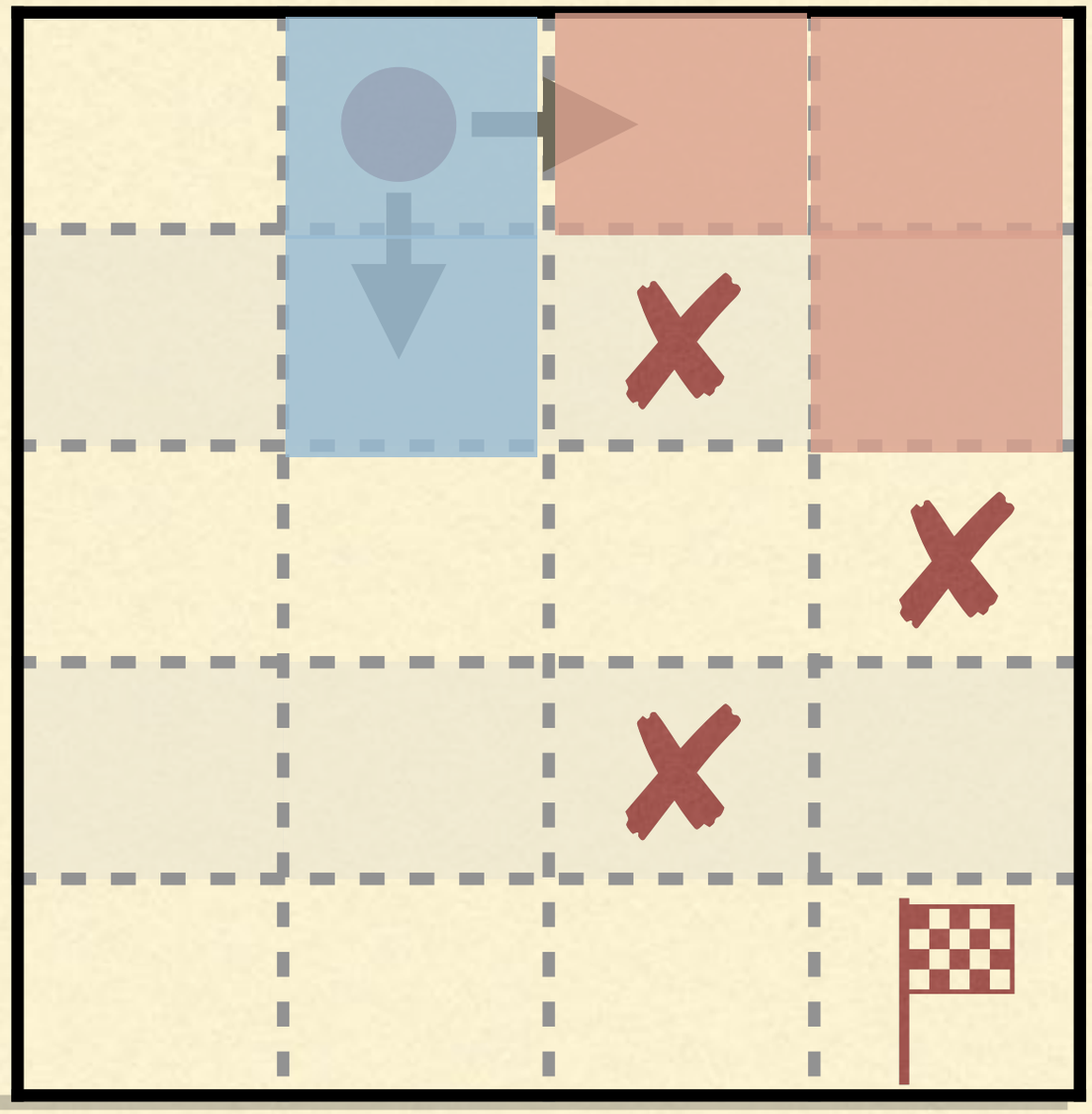
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



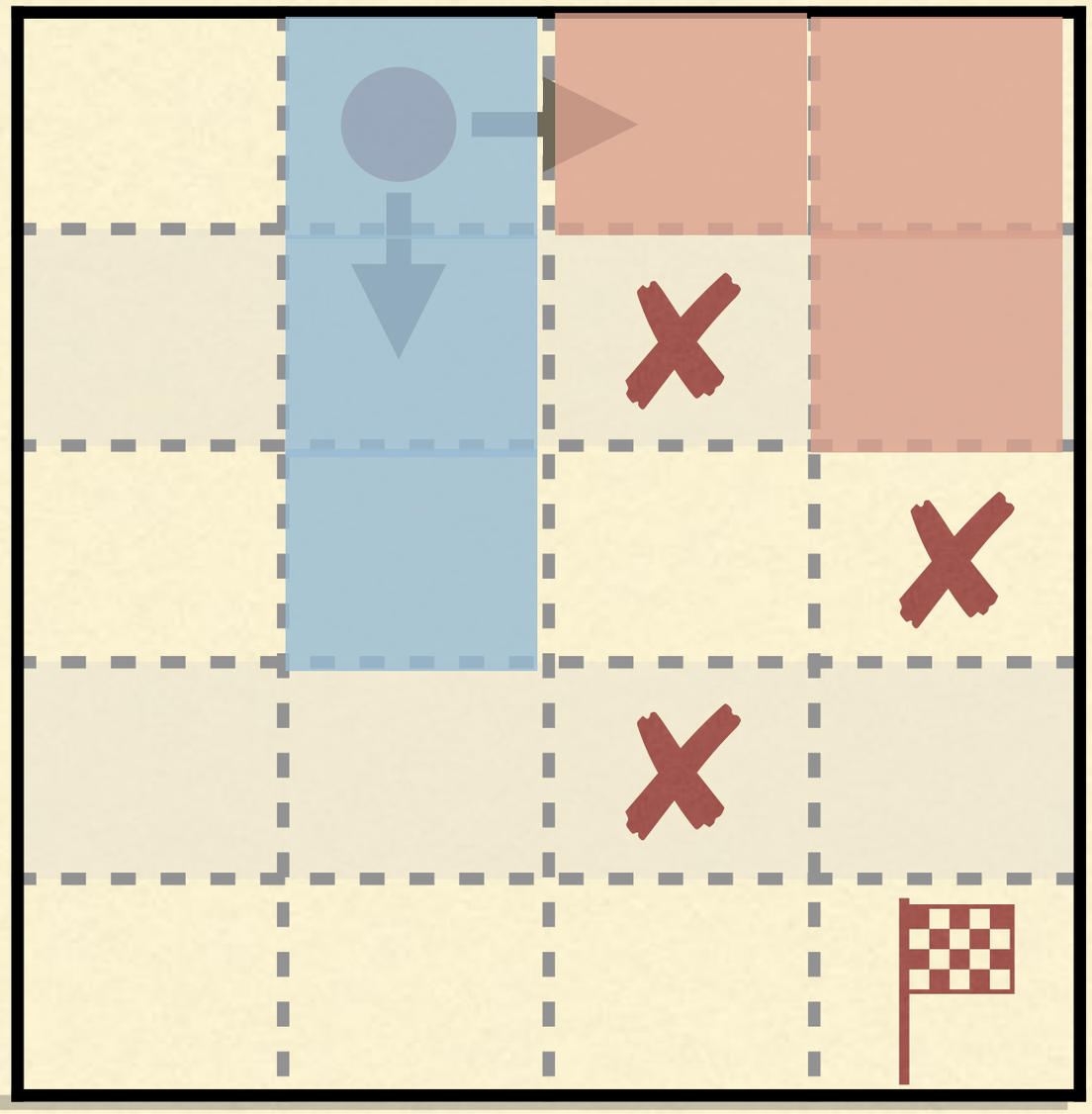
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



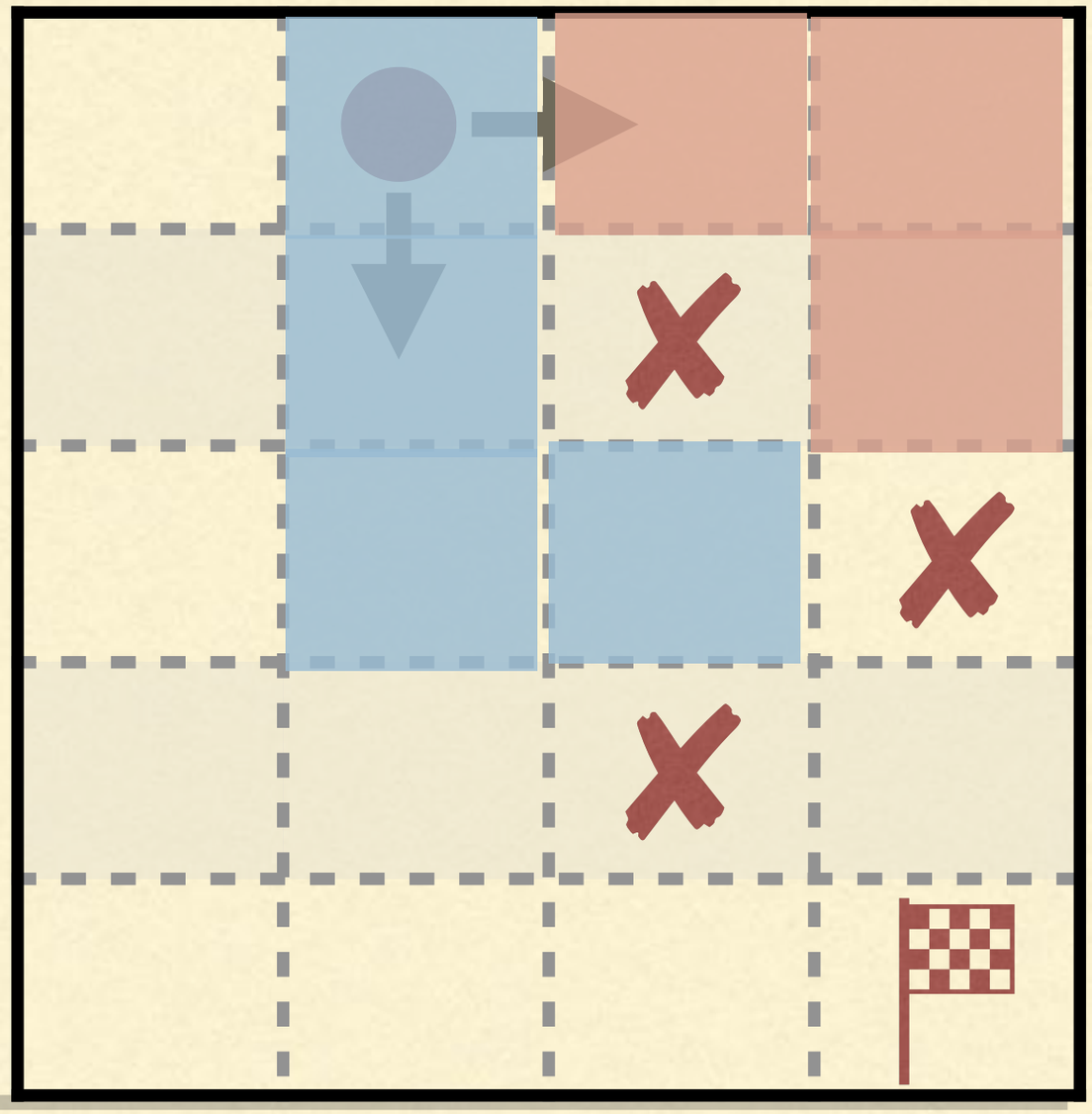
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



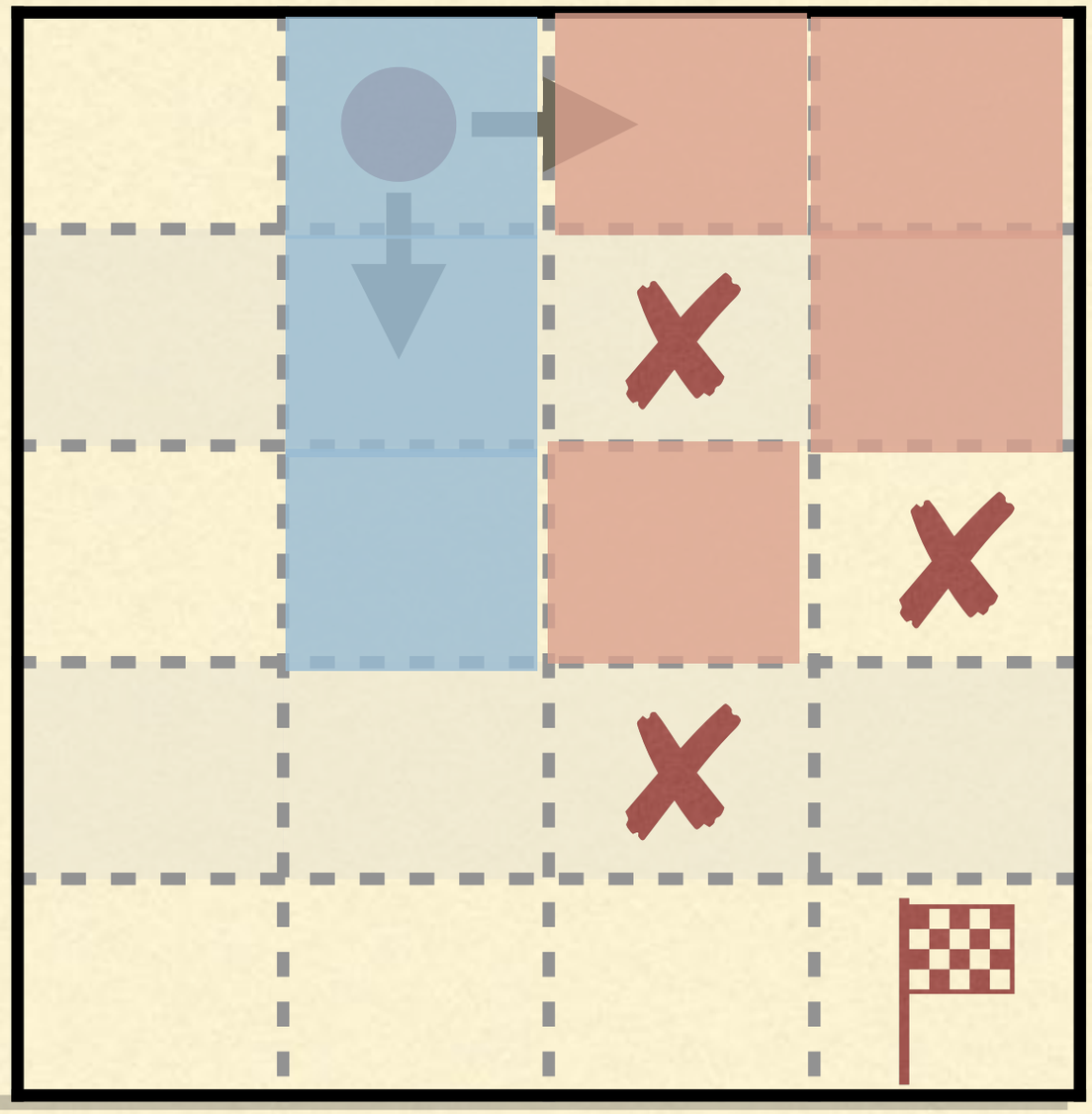
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



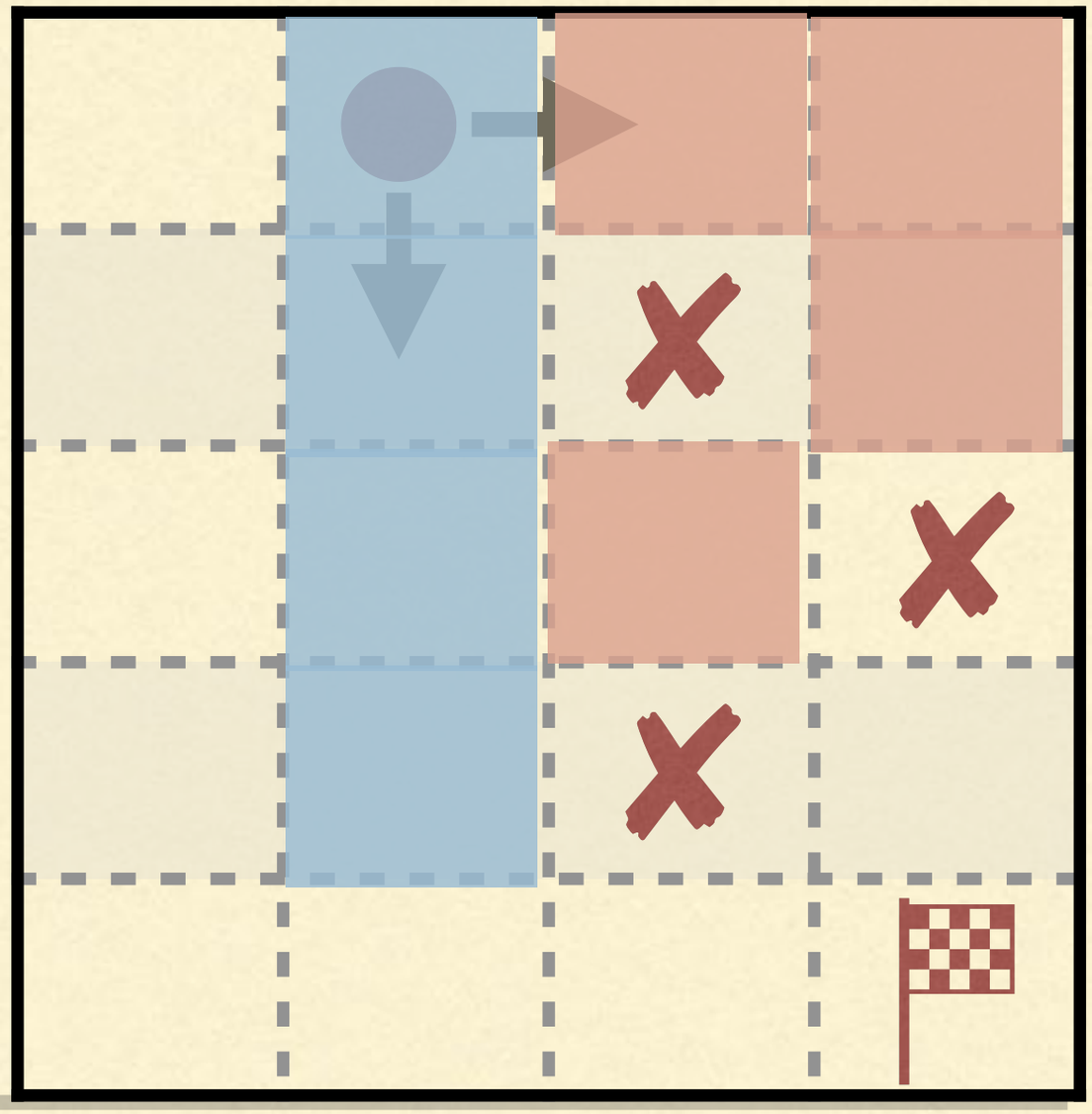
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



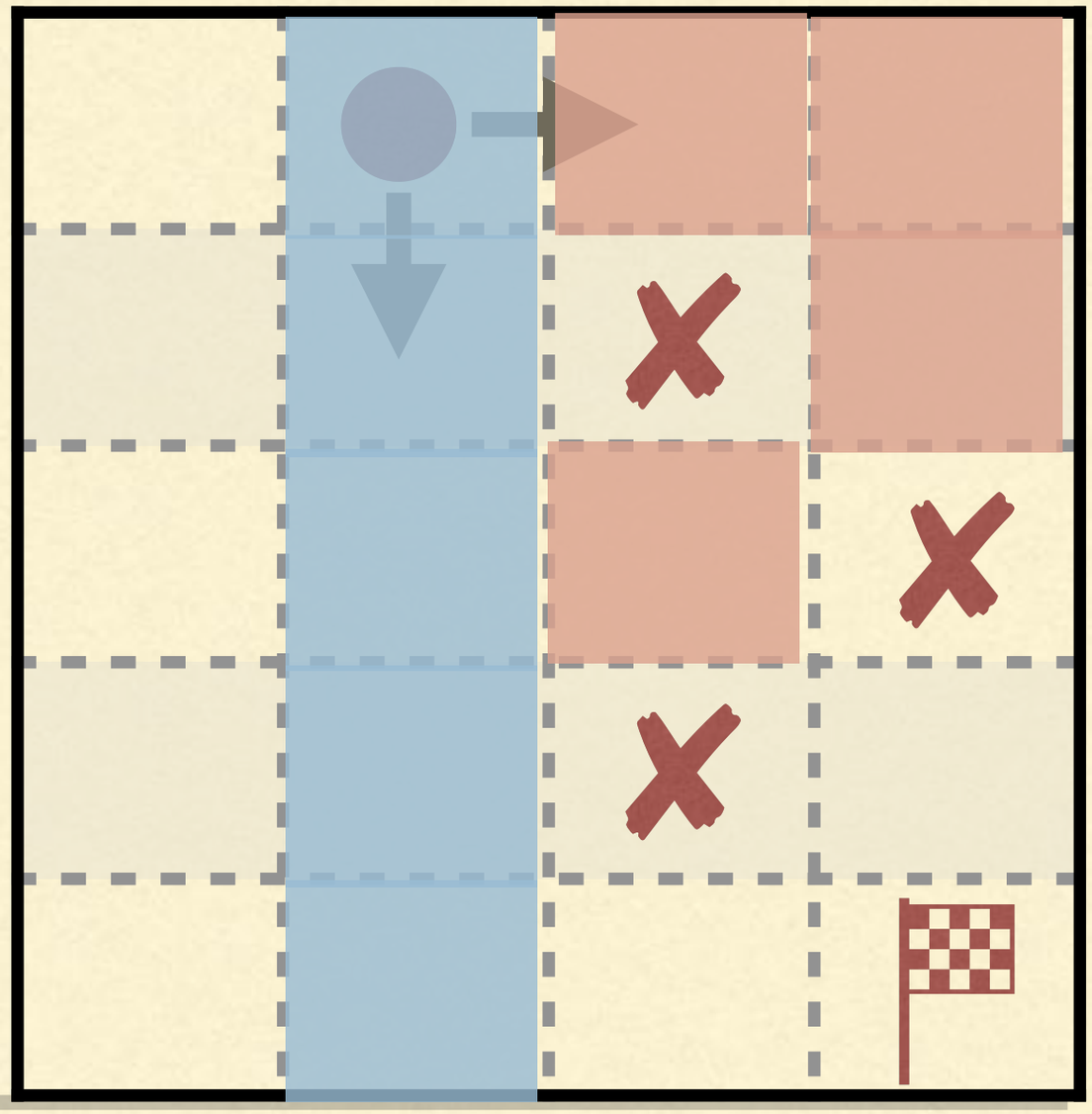
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



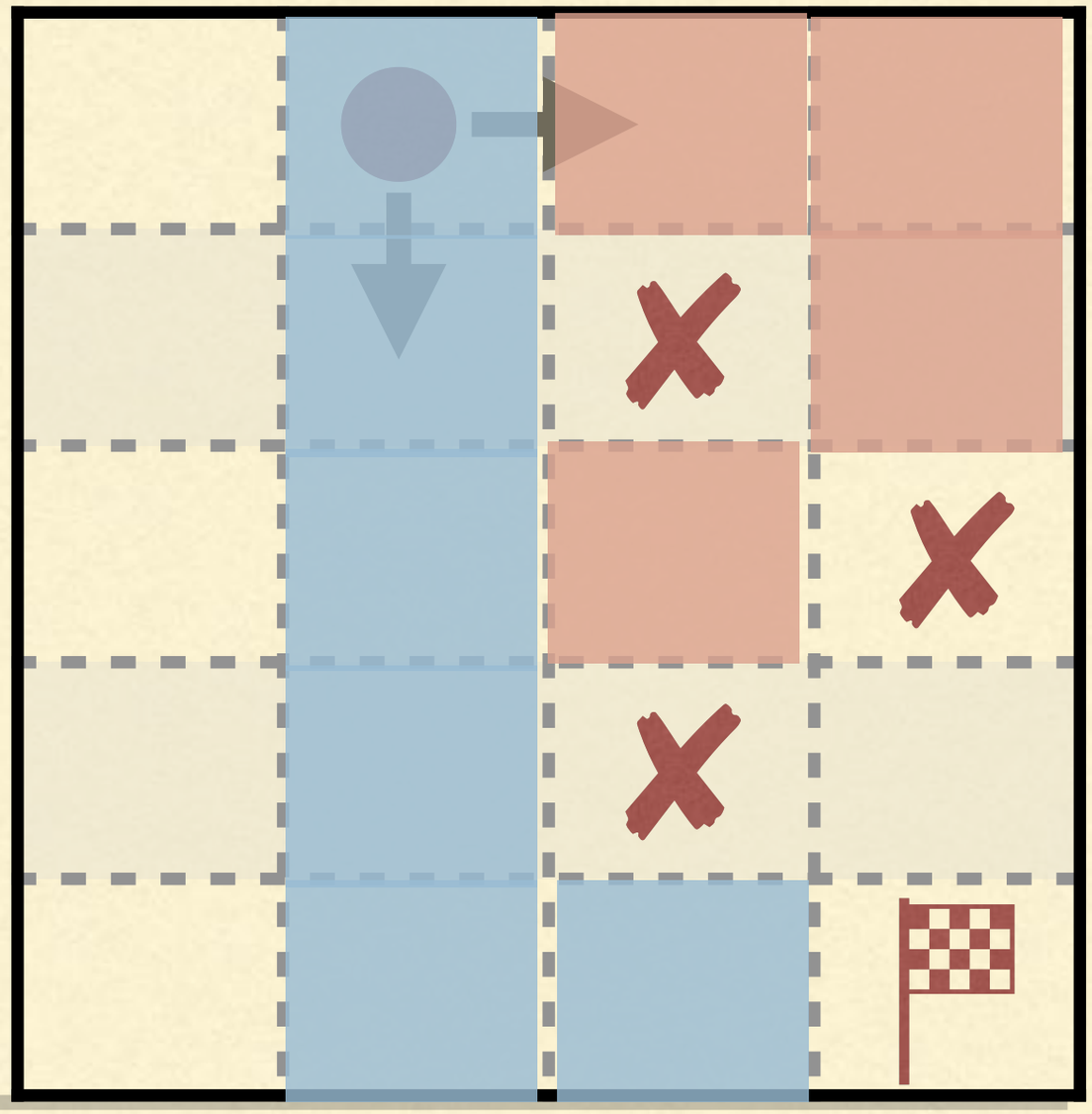
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う



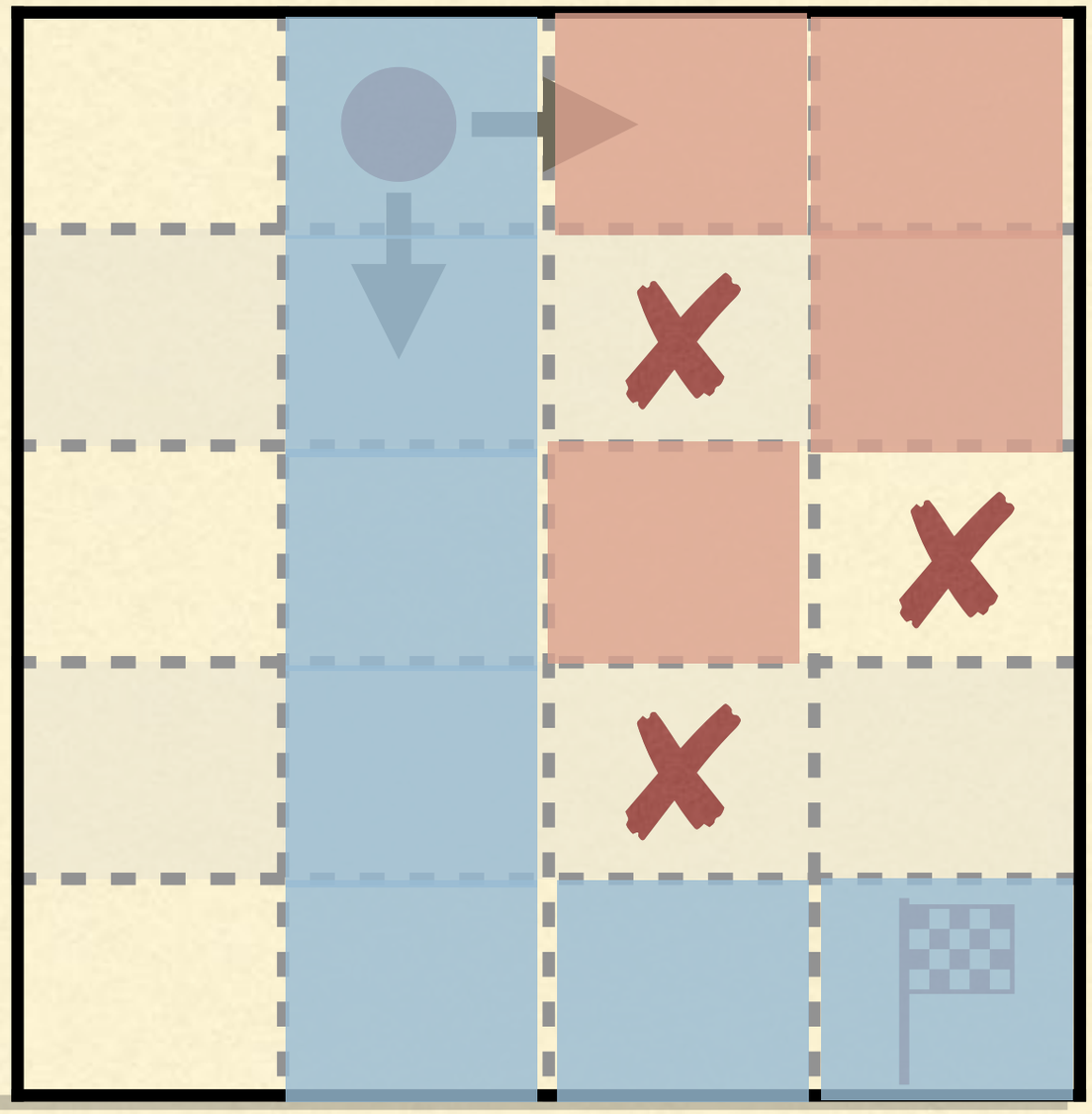
DFSとは、

- 人間が行える探索方法

  (探索できるところまで行って

  できなくなったら戻る)

- スタックか再帰を使う

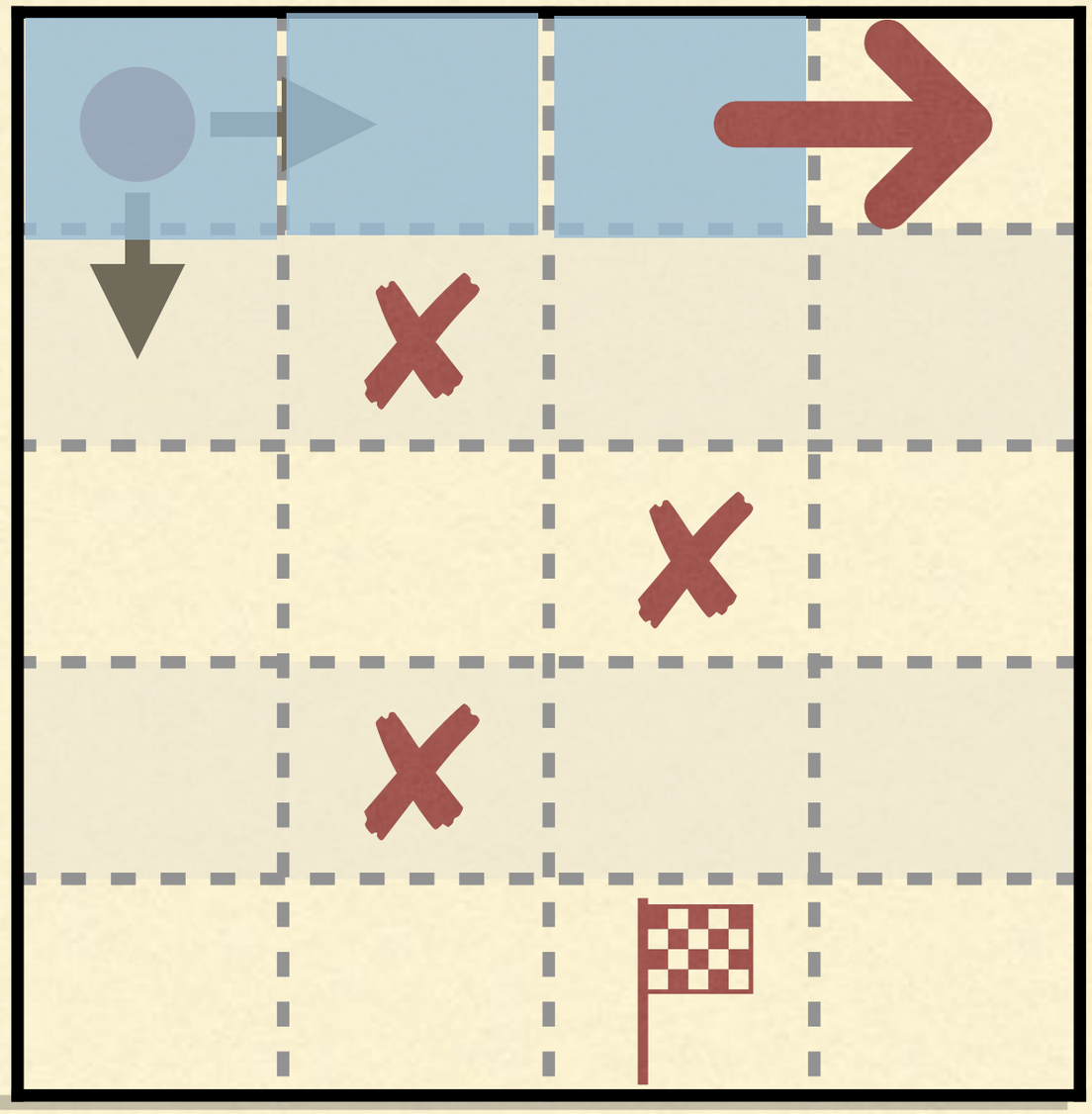


DFSの擬似コード

- while スタックから一つ取る
  - 現在地がゴール break;
  - 移動先がある場合は、その場所から移動できる全ての地点をスタックに入れる

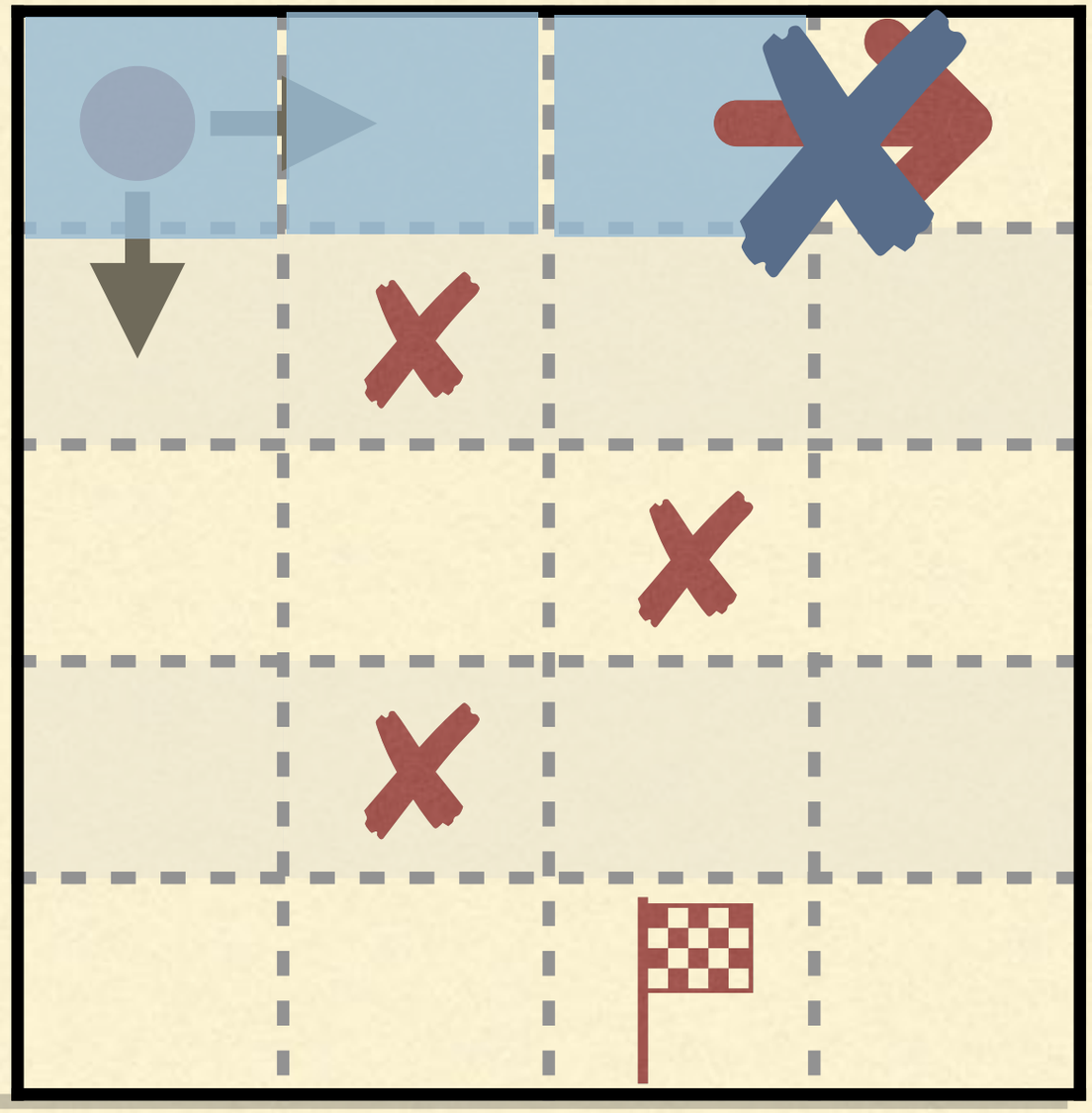
枝刈り

- DFS中にこれ以上探索しても、明らかに答えがないことが分かる場合はその先を求めずに戻ること

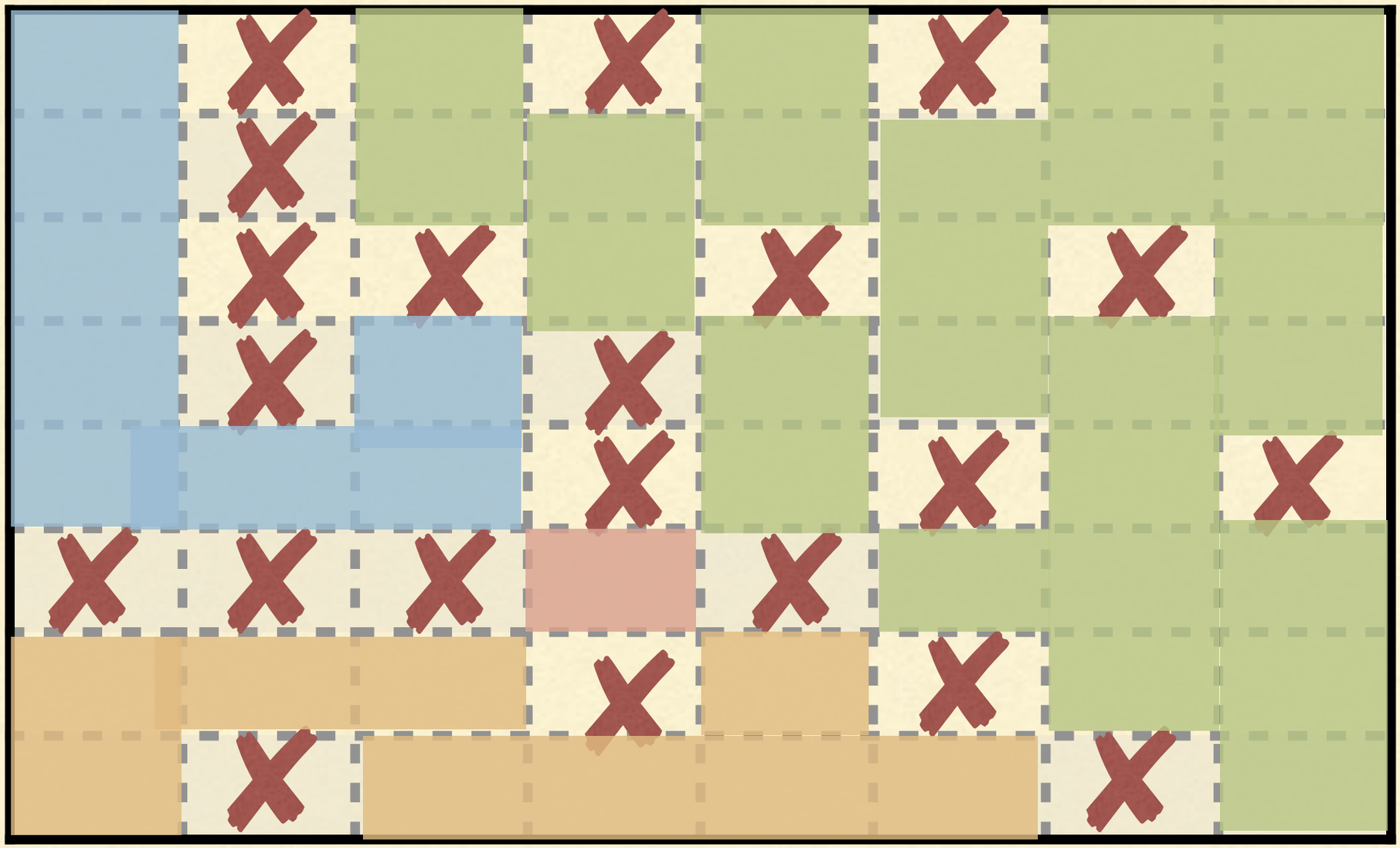


枝刈り

- DFS中にこれ以上探索しても、明らかに答えがないことが分かる場合はその先を求めずに戻ること

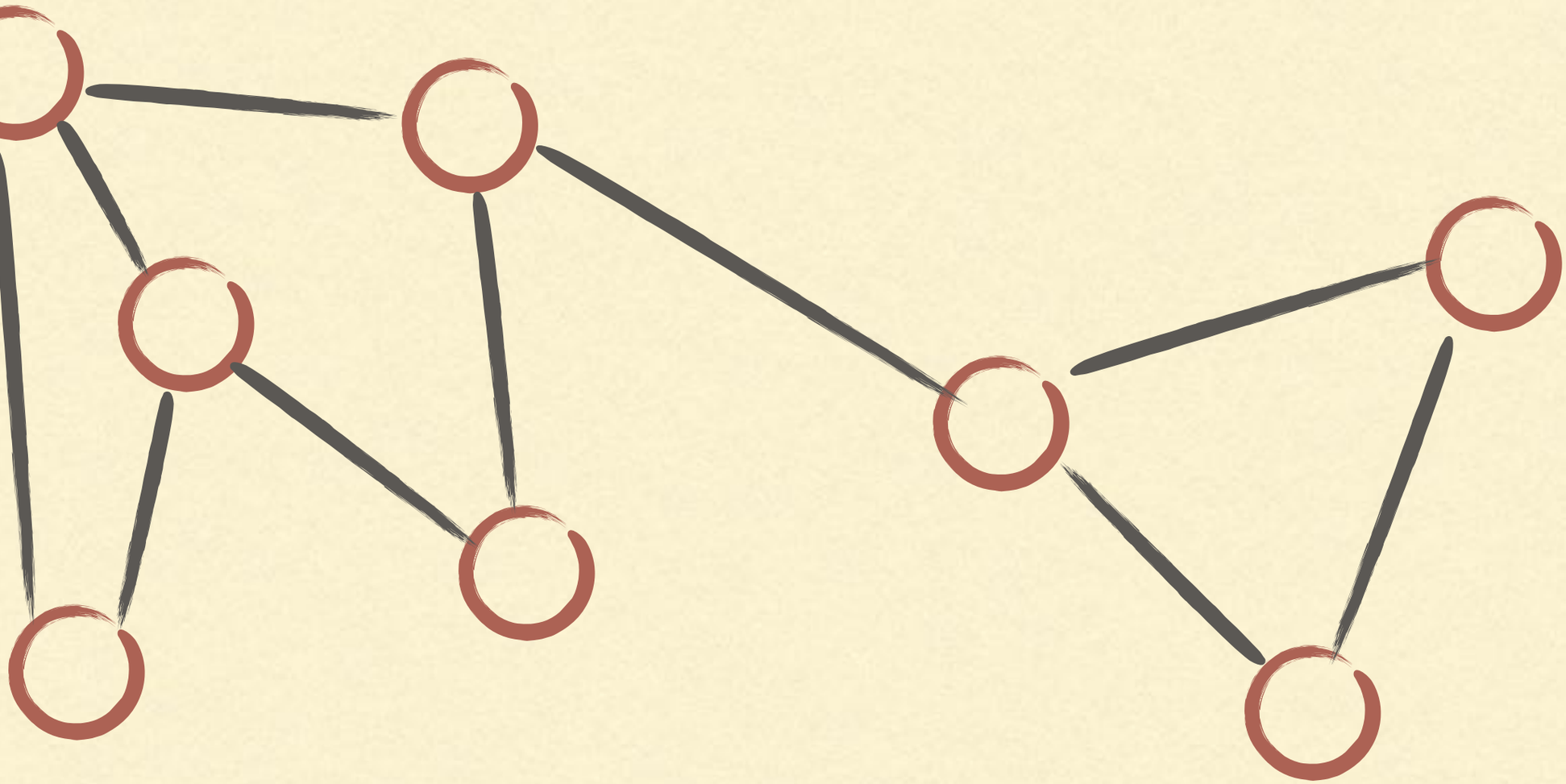


DFS/BFSで連結成分数を数える

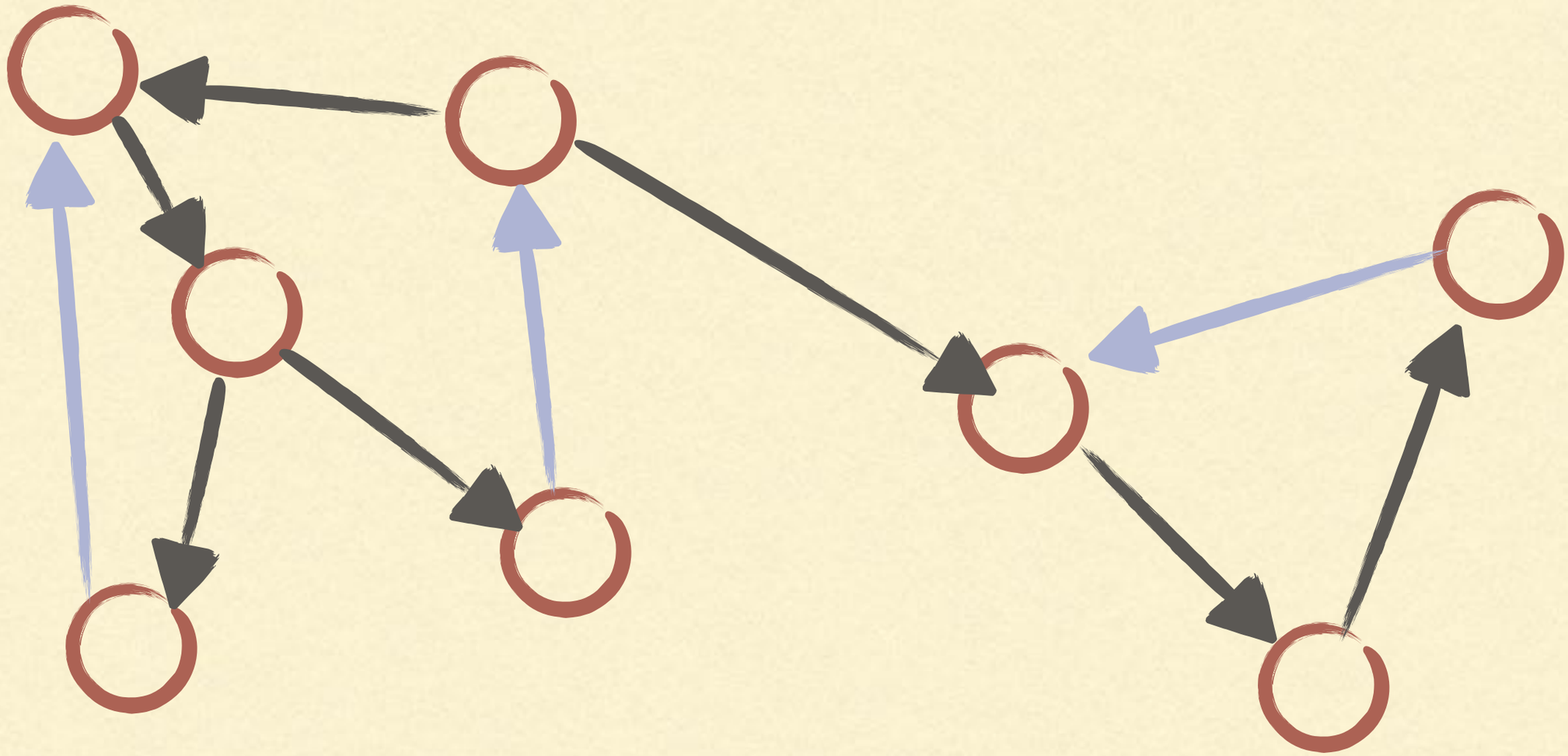


「まだ未訪問の地点があれば、そこからDFS/BFS」を繰り返す

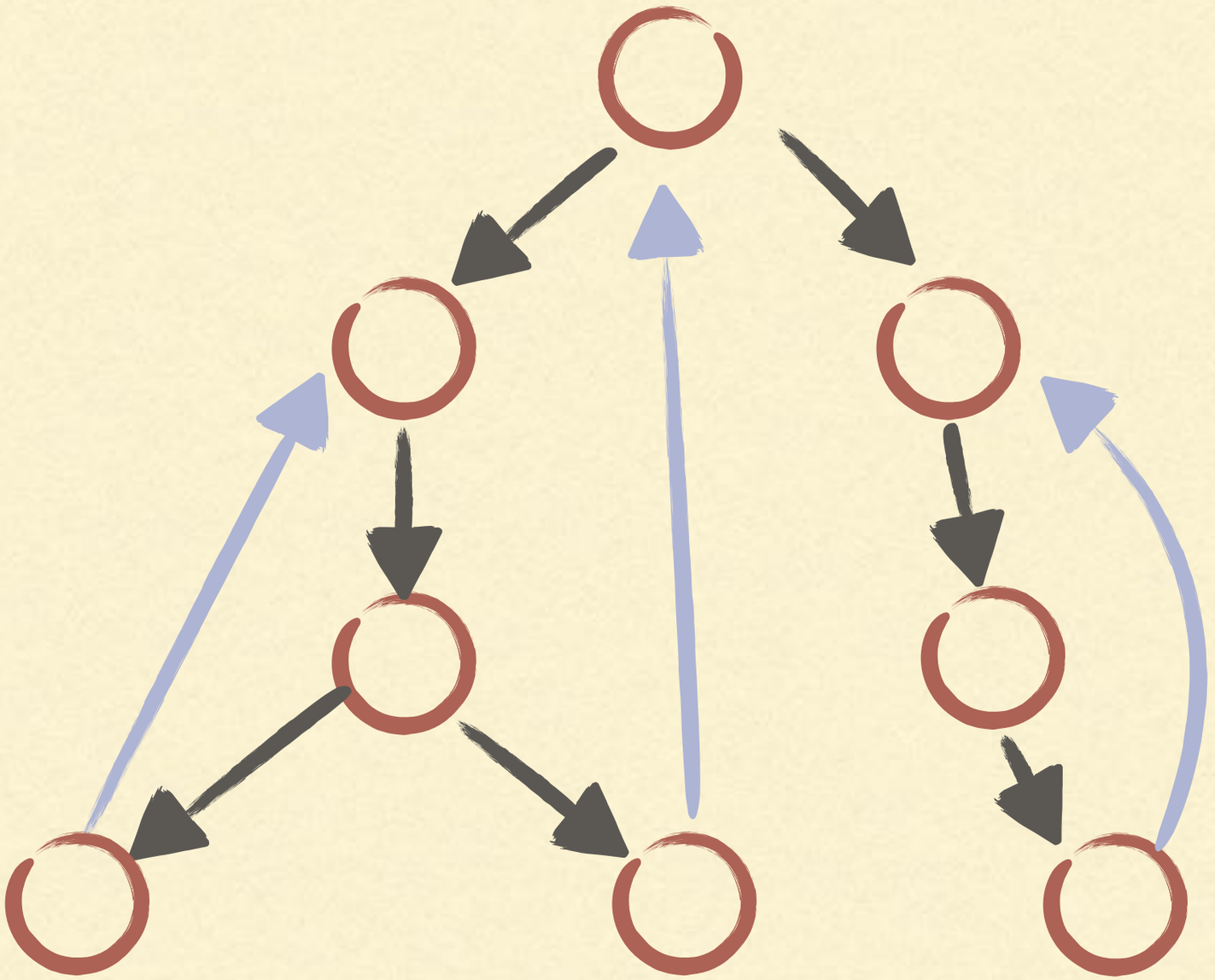
DFS応用 - DFS木



DFS応用 - DFS木

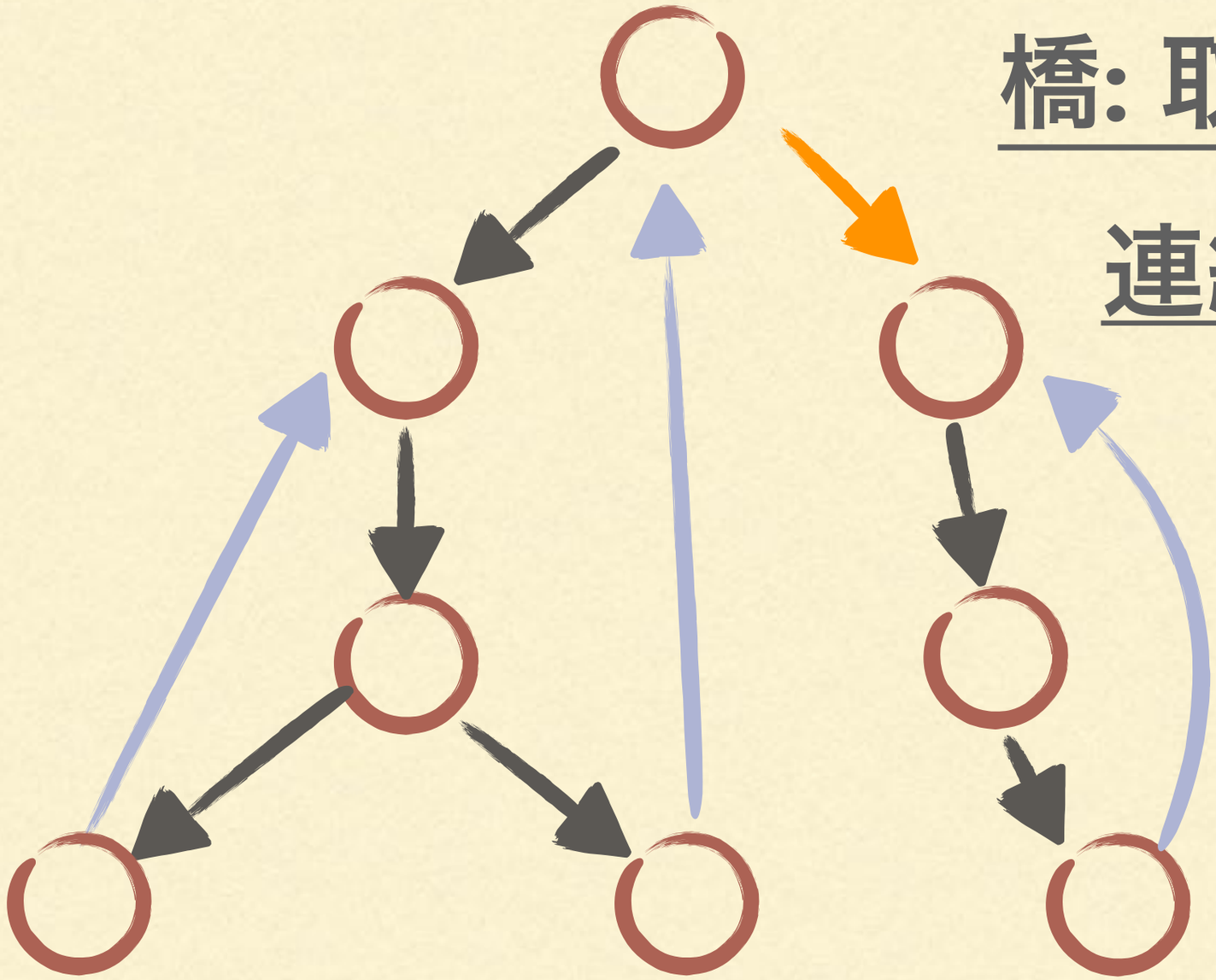


DFS応用 - DFS木



後退辺

DFS応用 - DFS木



橋: 取り除くと

連結成分が増える辺

ICPCの競技環境について

- 基本的に演習室のiMacです
- エディタ/コンパイラ等はすでに入っているものは使用可能

  また、管理者権限を必要としなければ入れられる(VSCなど)
- 情リテでの端末と基本同じなので試して下さい
- C/C++を使っている人は、

  Clang: `gcc/g++`　　GCC: `gcc-7 / g++-7`