アルゴリズム講座

第4回 ～幅優先探索編～

2020/07/05 11:00~

@utsubo_21 高澤 栄一

Lesson 4

今回の内容

- 下記のアルゴリズム・データ構造を10回程度に分けて１つずつ解説

Lesson1
- 全探索
- 二分探索
- 深さ優先探索
- 幅優先探索
- 動的計画法
- 累積和
- グラフ・木
- ダイクストラ法
- ワーシャルフロイド法
- クラスカル法
- Union-Find

参考：レッドコーダーが教える、競プロ・AtCoder上達のガイドライン【中級編：目指せ水色コーダー！】
https://qiita.com/e869120/items/eb50fdaece12be418faa

Question

問題

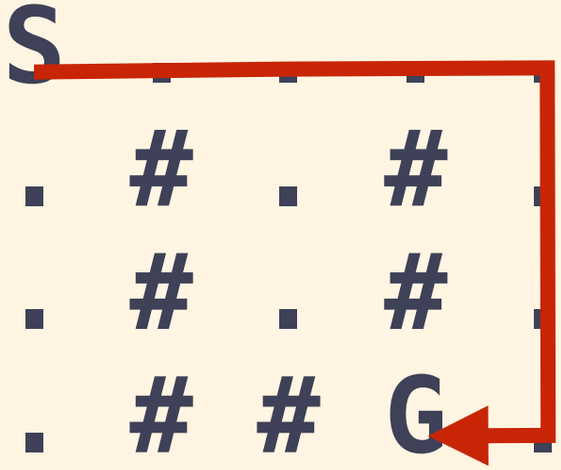
- 迷路が与えられる
- 最短経路長を返しなさい

```
S . . . .
. # . # .
. # . # .
. # # G .
```

```
S : Start
G : Goal
. : Empty(can move)
# : Wall
```

問題

- 迷路が与えられる
- 最短経路長を返しなさい

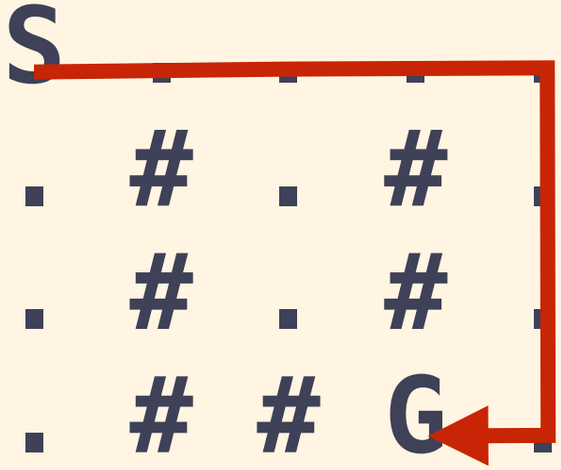



```
S : Start
G : Goal
. : Empty(can move)
# : Wall
```

最短8手で到着

問題

- 迷路が与えられる
- 最短経路長を返しなさい

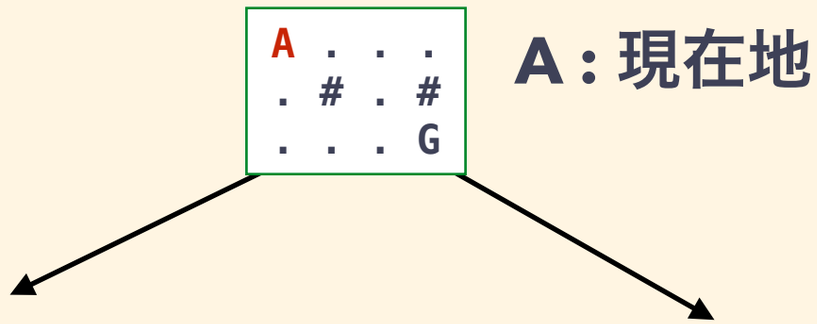



```
S : Start
G : Goal
. : Empty(can move)
# : Wall
```

最短8手で到着

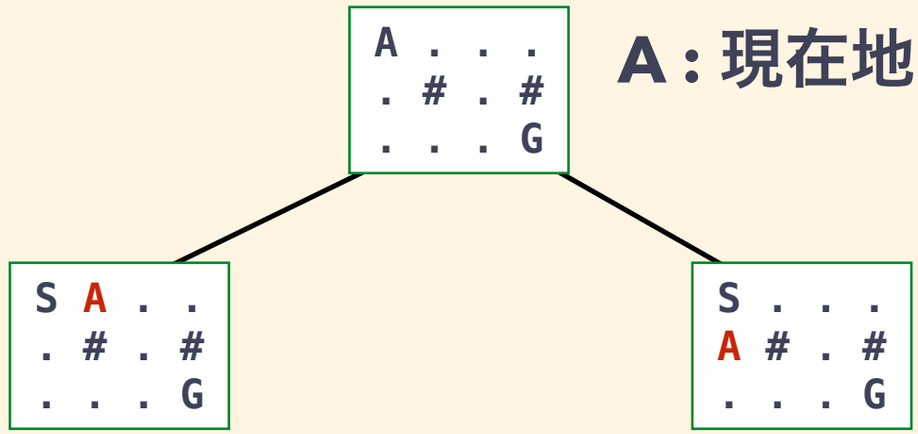
➠ 全ての経路を調べよう！

経路を全列挙しよう



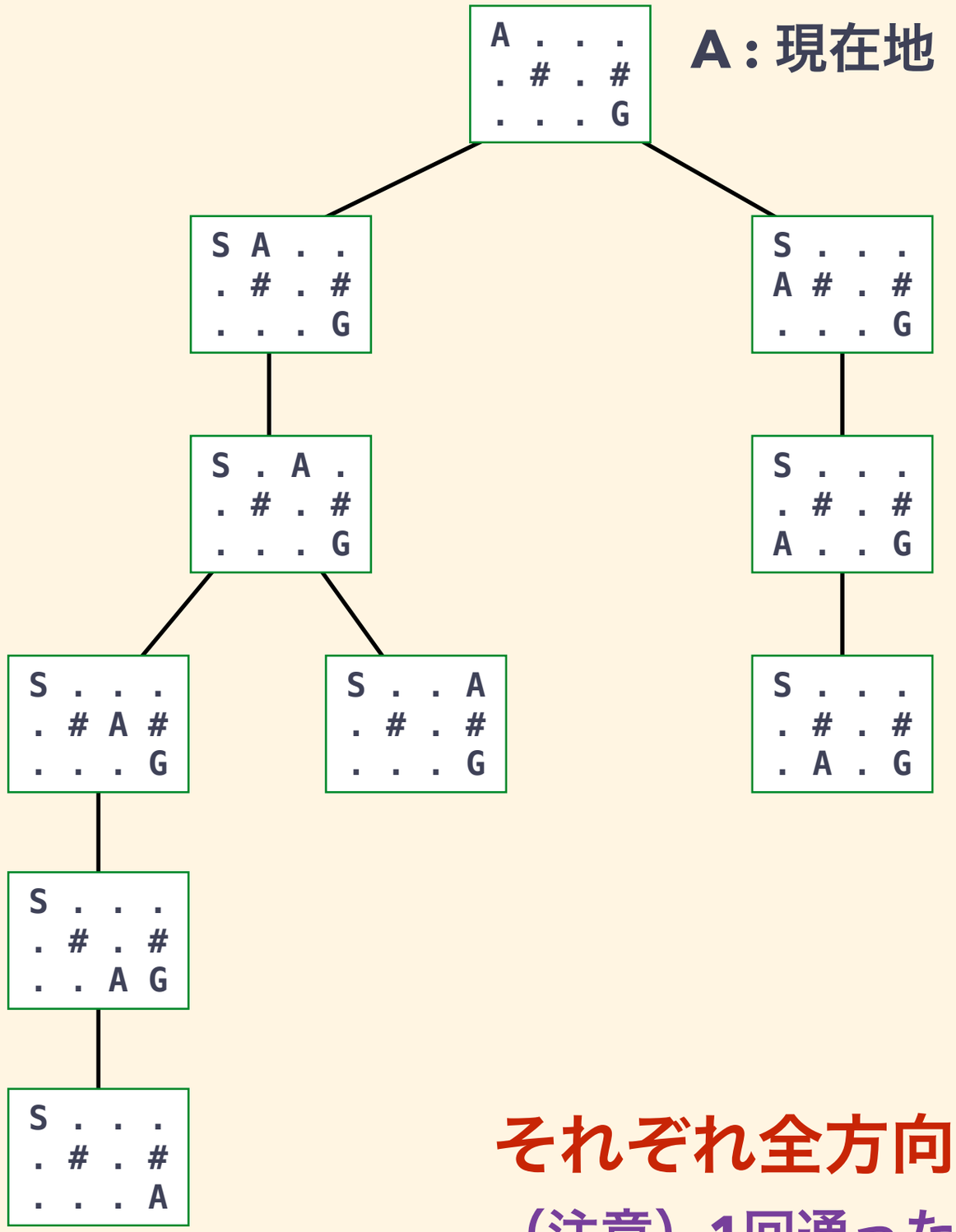
Aの位置からスタート　1マス動かしてみる

経路を全列挙しよう



右と下に移動できる

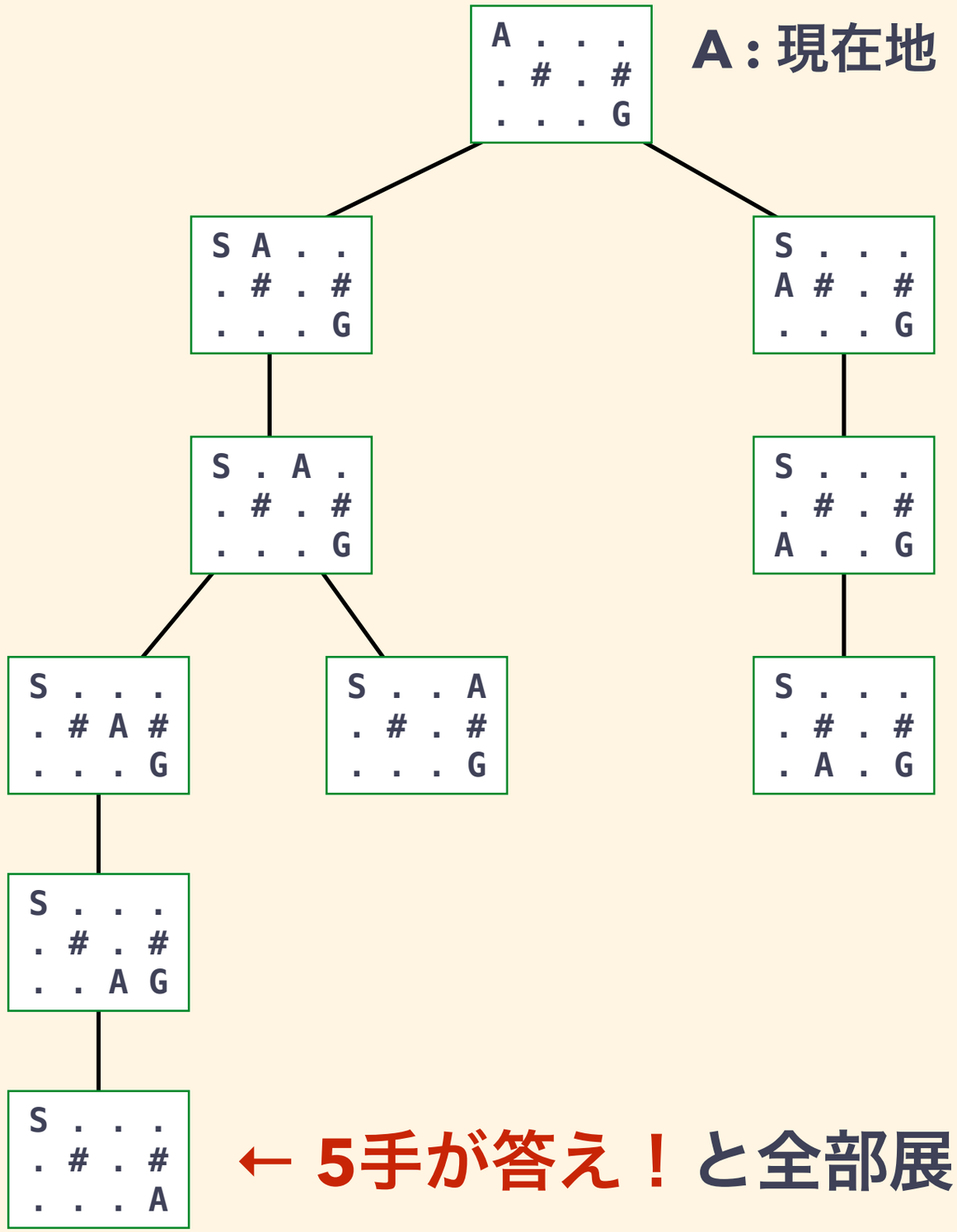
経路を全列挙しよう



それぞれ全方向へ動かしてみると...

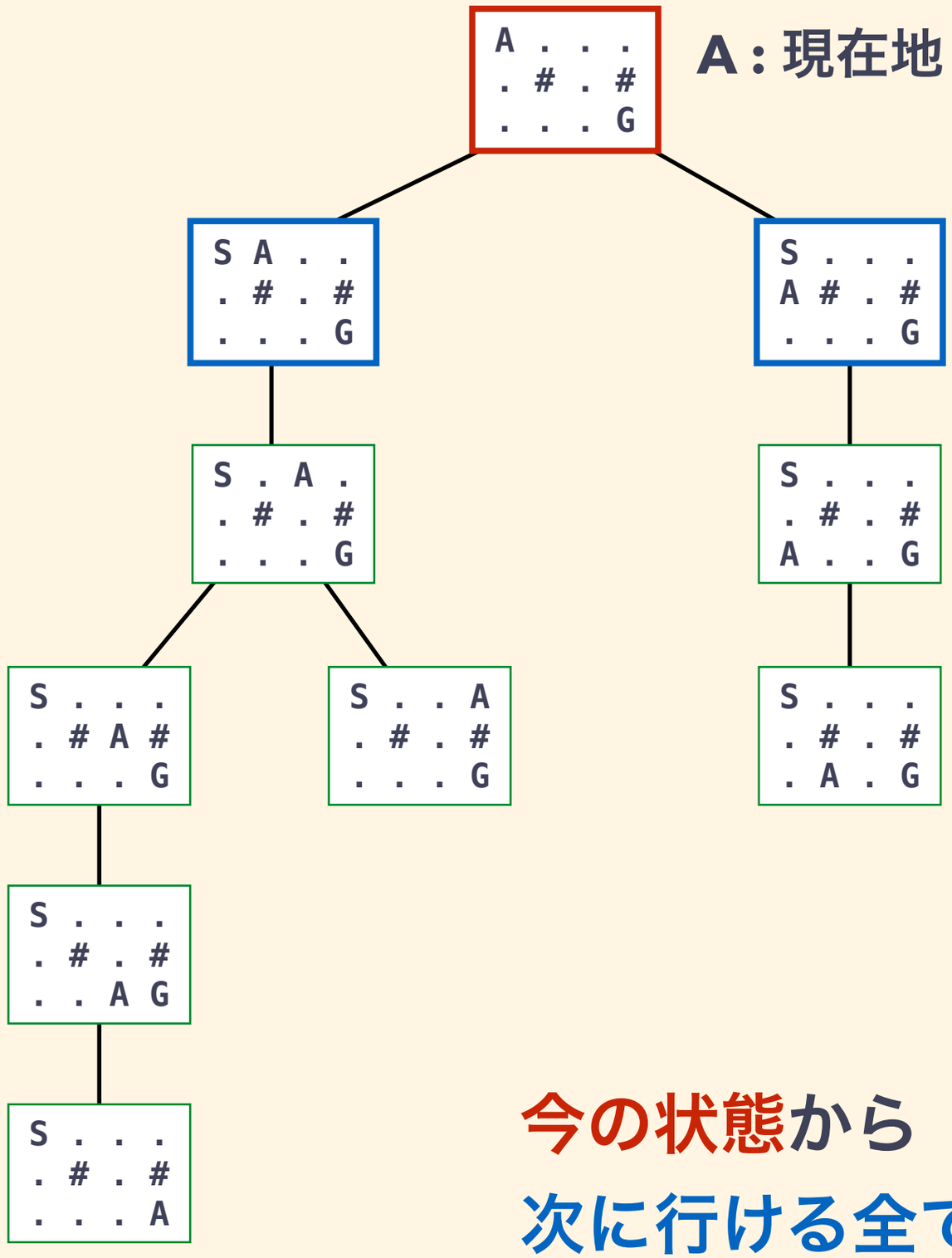
（注意）1回通った場所は通らないように

経路を全列挙しよう



← 5手が答え！と全部展開してやれば分かる

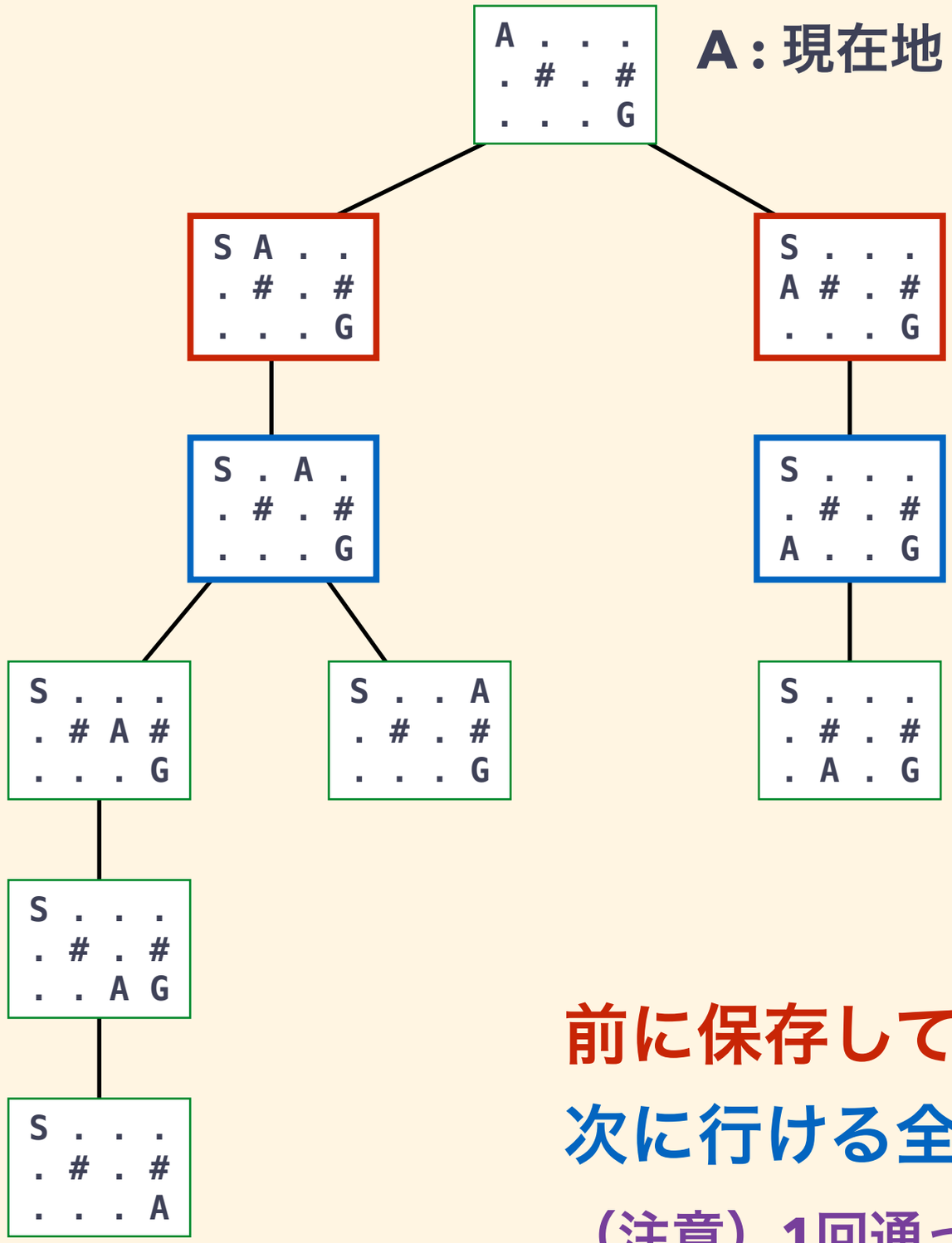
どのように全列挙するか？



今の状態から

次に行ける全ての状態を生成し保存

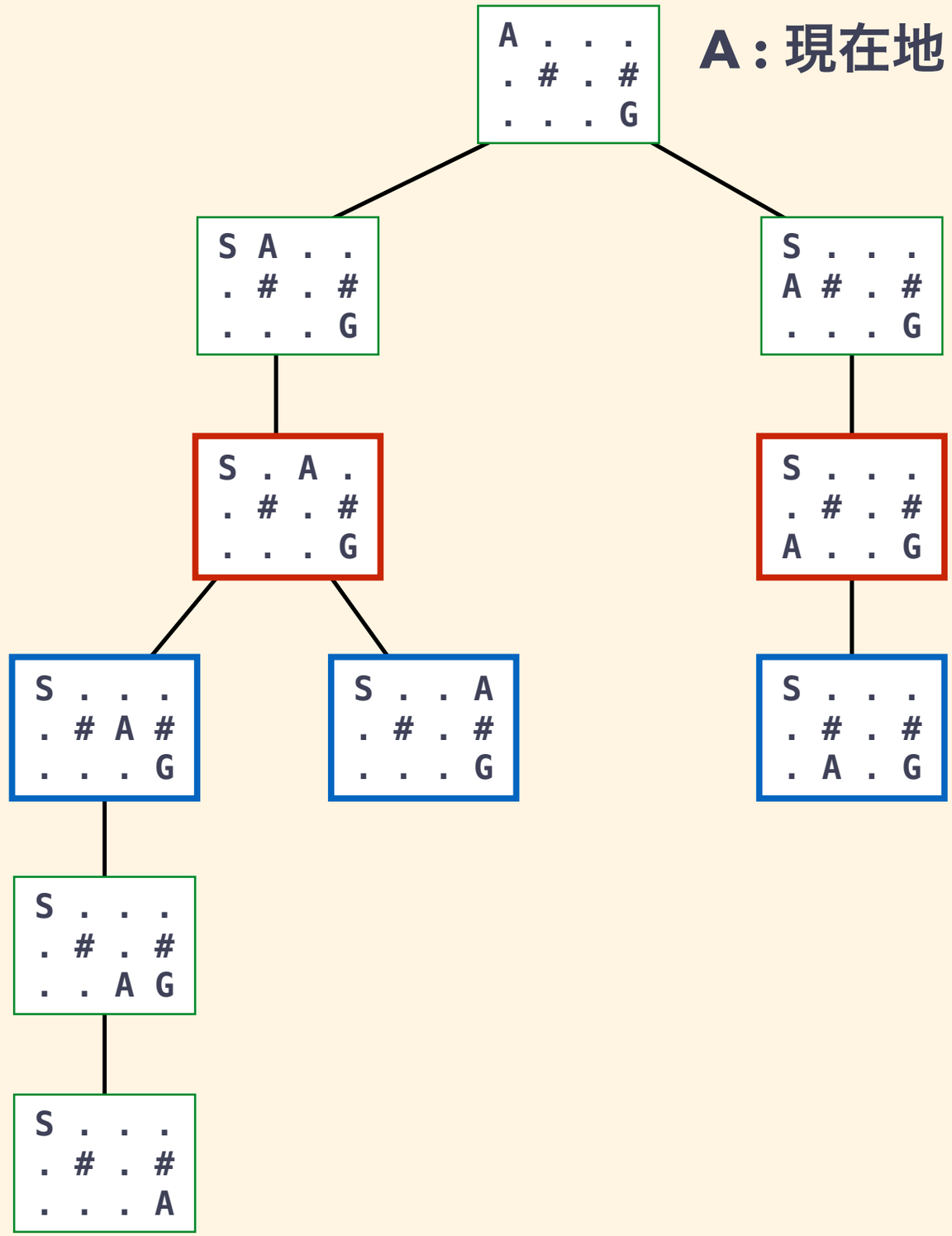
どのように全列挙するか？



前に保存しておいた状態から
次に行ける全ての状態を生成

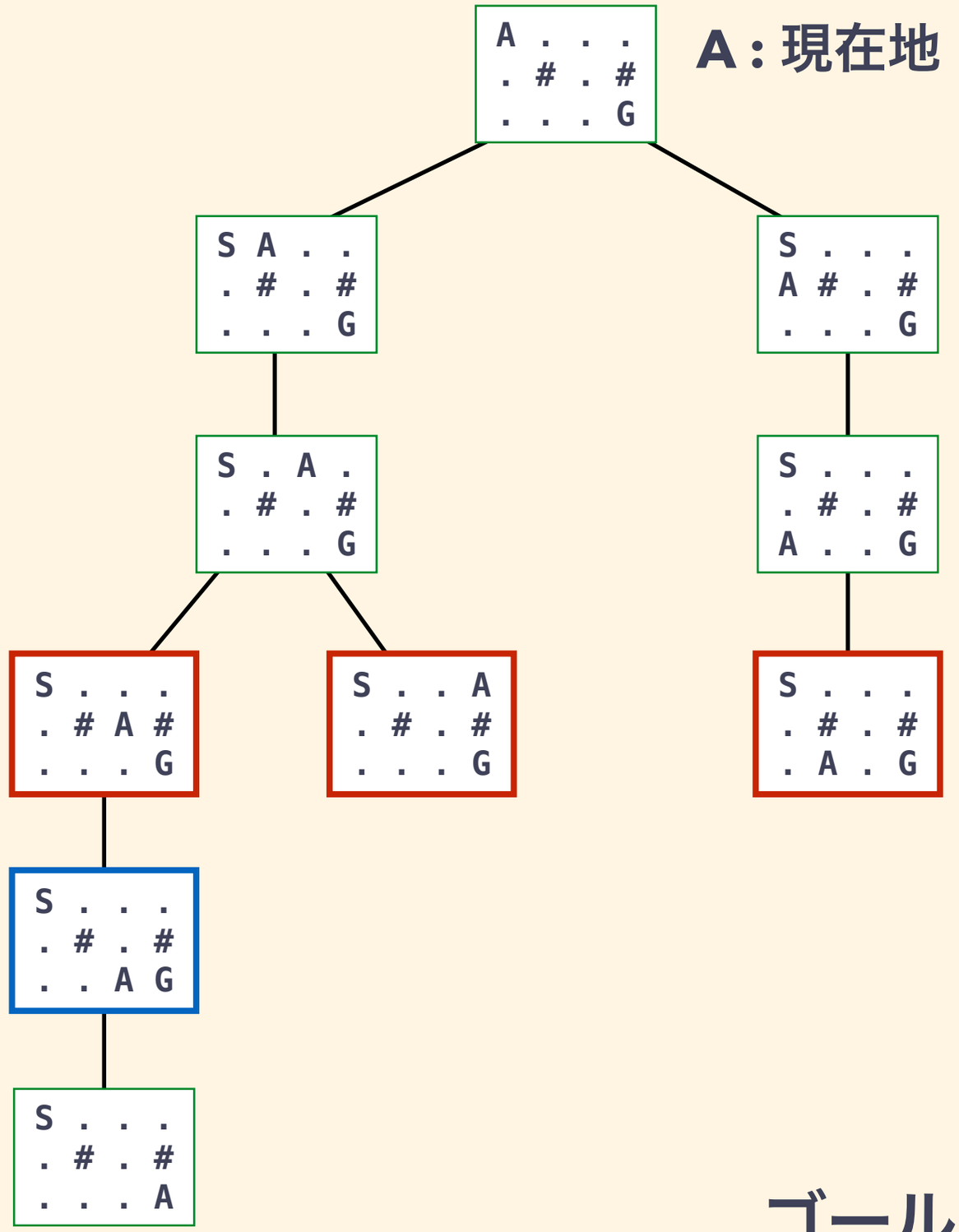
（注意）1回通った場所は通らないように

どのように全列挙するか？



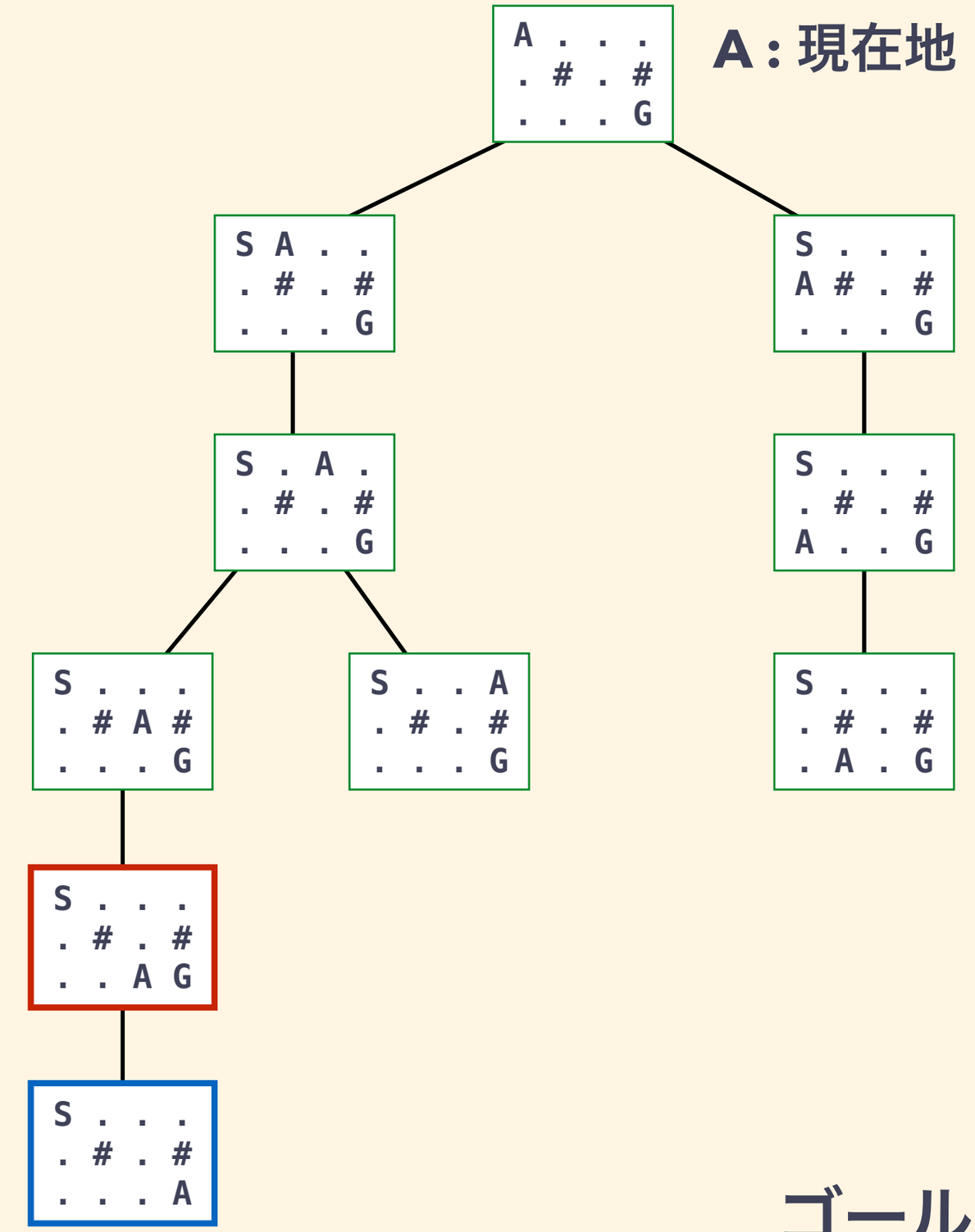
同じように

どのように全列挙するか？



ゴールに着くまで繰り返す

どのように全列挙するか？



ゴールに着くまで繰り返す

どう実装するのか？

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持

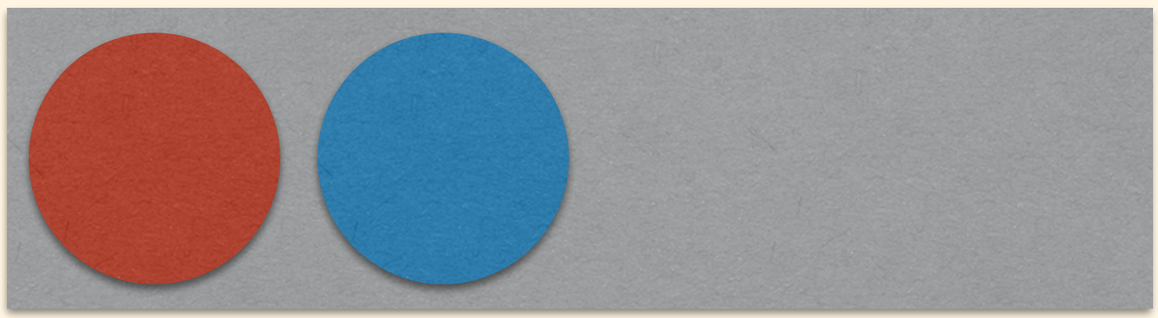
キュー

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持



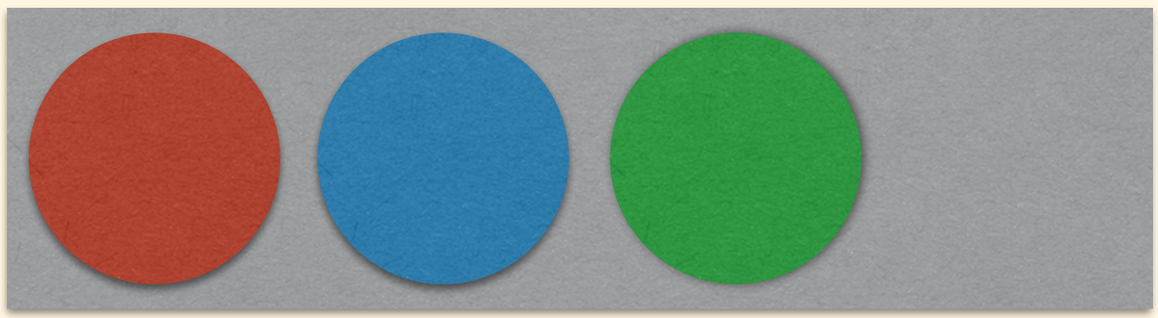
キュー

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持

キュー

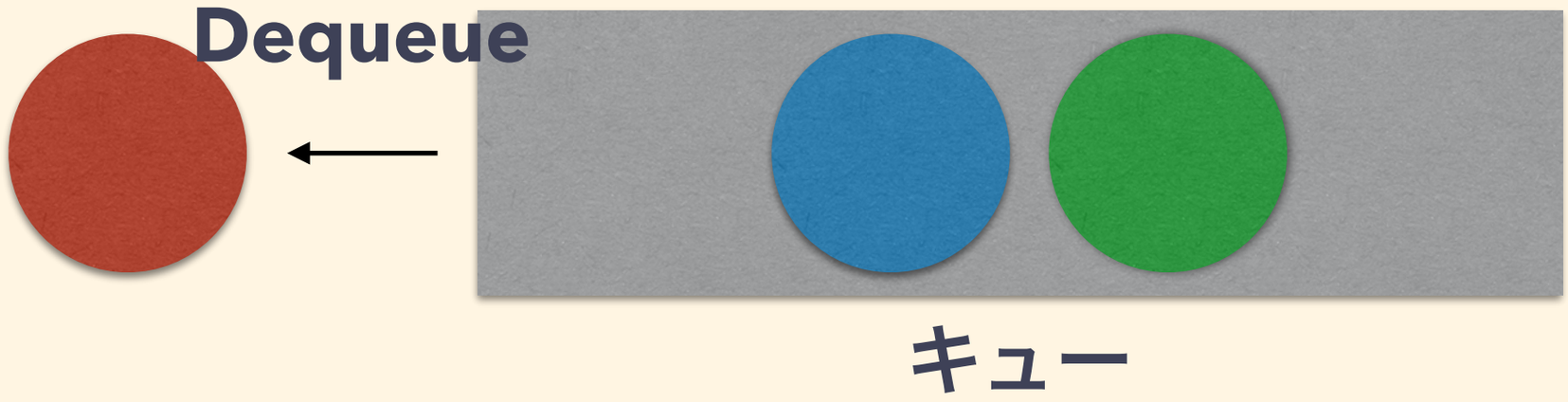
- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持



キュー

キュー

- とは？
  - 基本的なデータ構造
  - データを先入れ先出しのリスト構造で保持



最初に入れたものが最初に出てくる

データを追加すること　：Enqueue

データを取り出すこと　：Dequeue

キュー

- 各言語でのライブラリ
  - C++　　: std::queue<T>
  - Python : Queue
  - Java　　: Deque<T> (実装はArrayDeque<T>等)

*** データを追加/取り出す関数の名前が各言語で違うので注意**

（Enqueue, Dequeueではない名前になっている）

キュー

- コード例 (Python)

コード

```
que = queue.Queue()
que.put('A')
que.put('B')
que.put('C')

que.get()
que.get()

que.put('D')
```

内部の雰囲気

```
[]
[A]           in  A
[A, B]        in  B
[A, B, C]     in  C

[B, C]        out A
[C]           out B

[C, D]        in D
```

で，キューを使って
どう実装するのか？

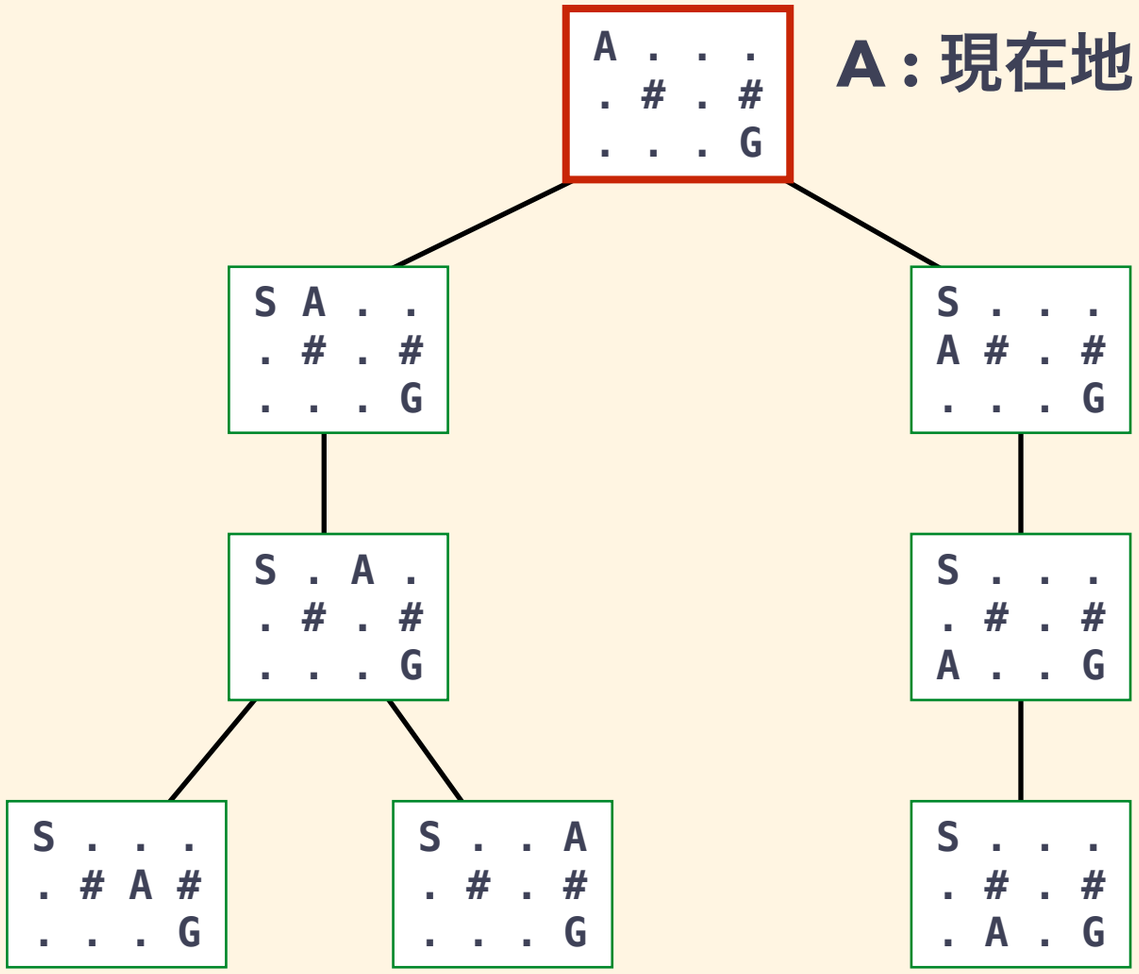
幅優先探索を書く

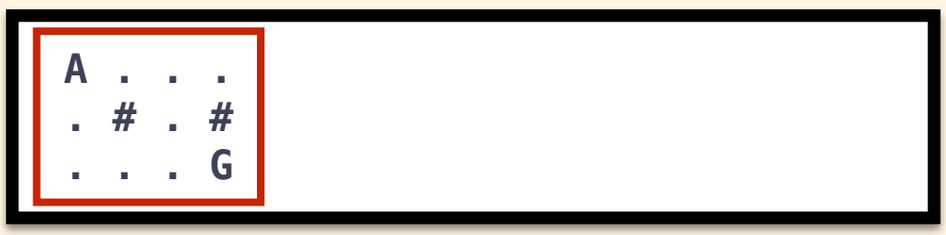
- 擬似コード

```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる

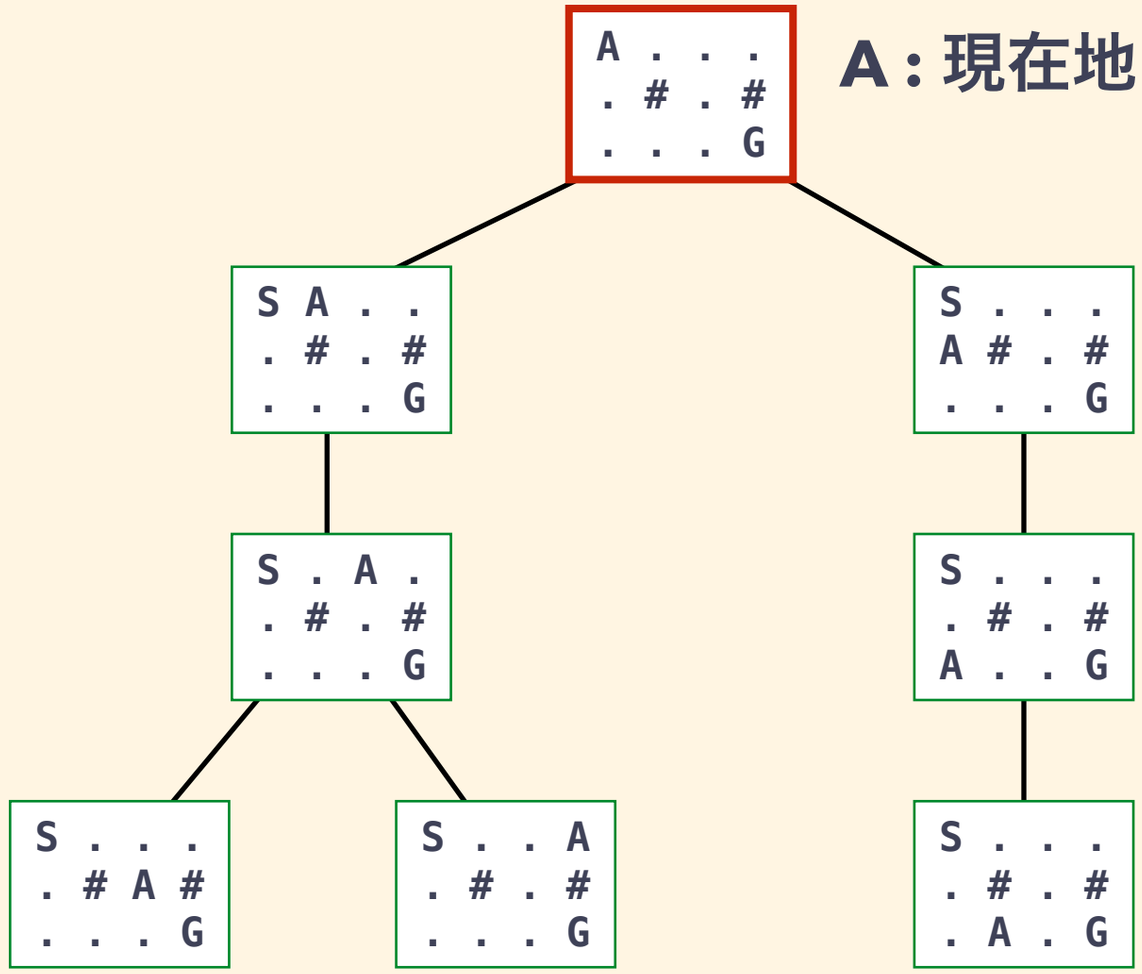


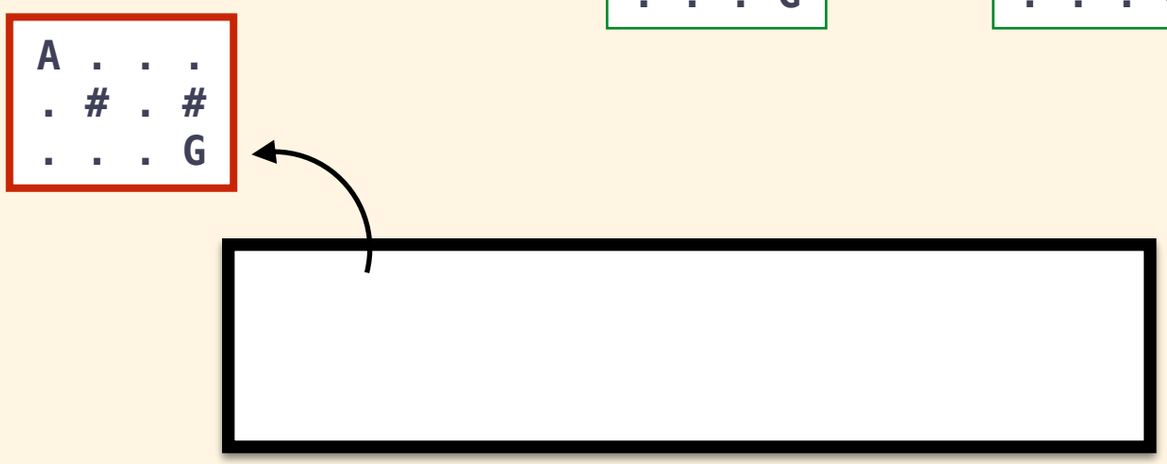



```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる

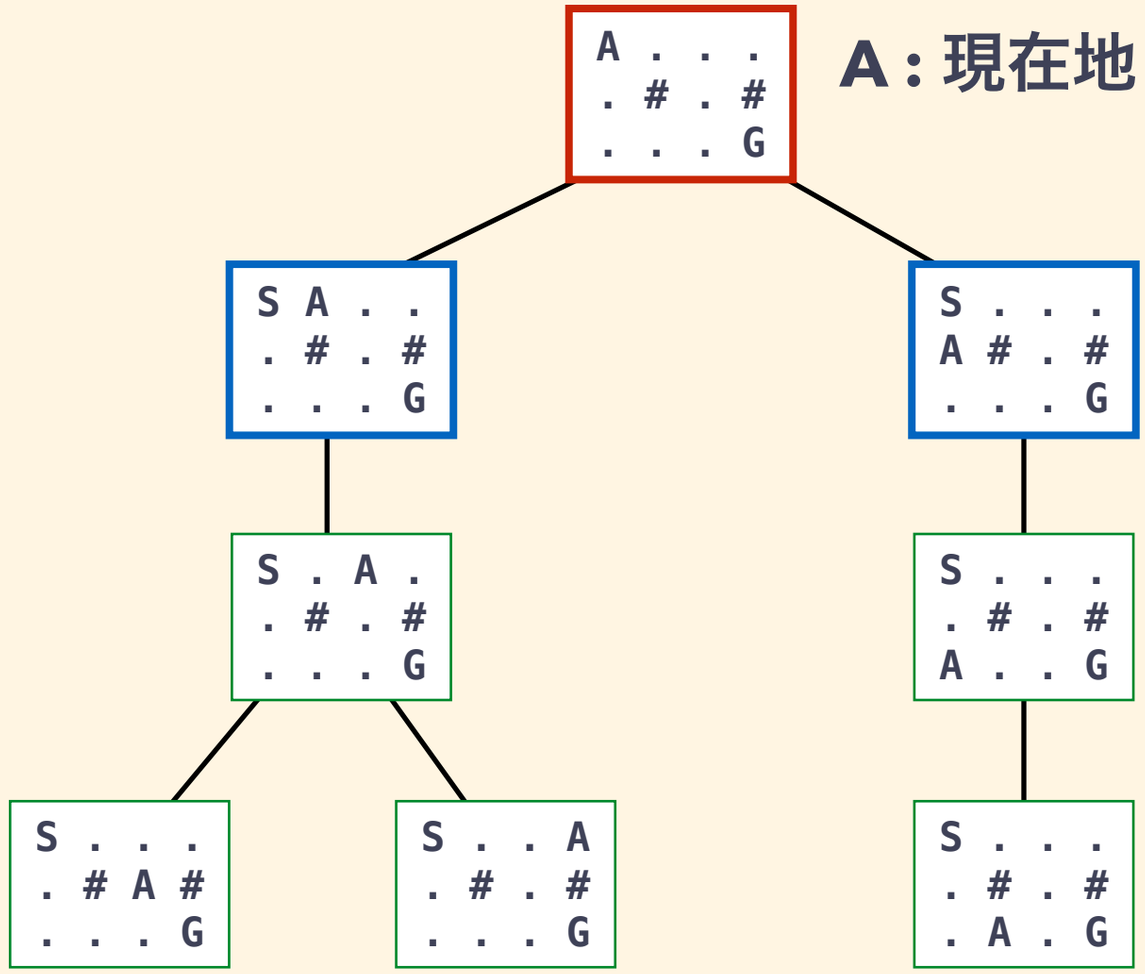


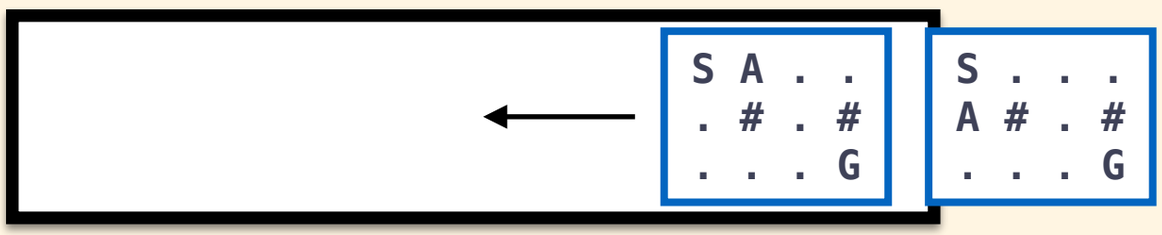



```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる

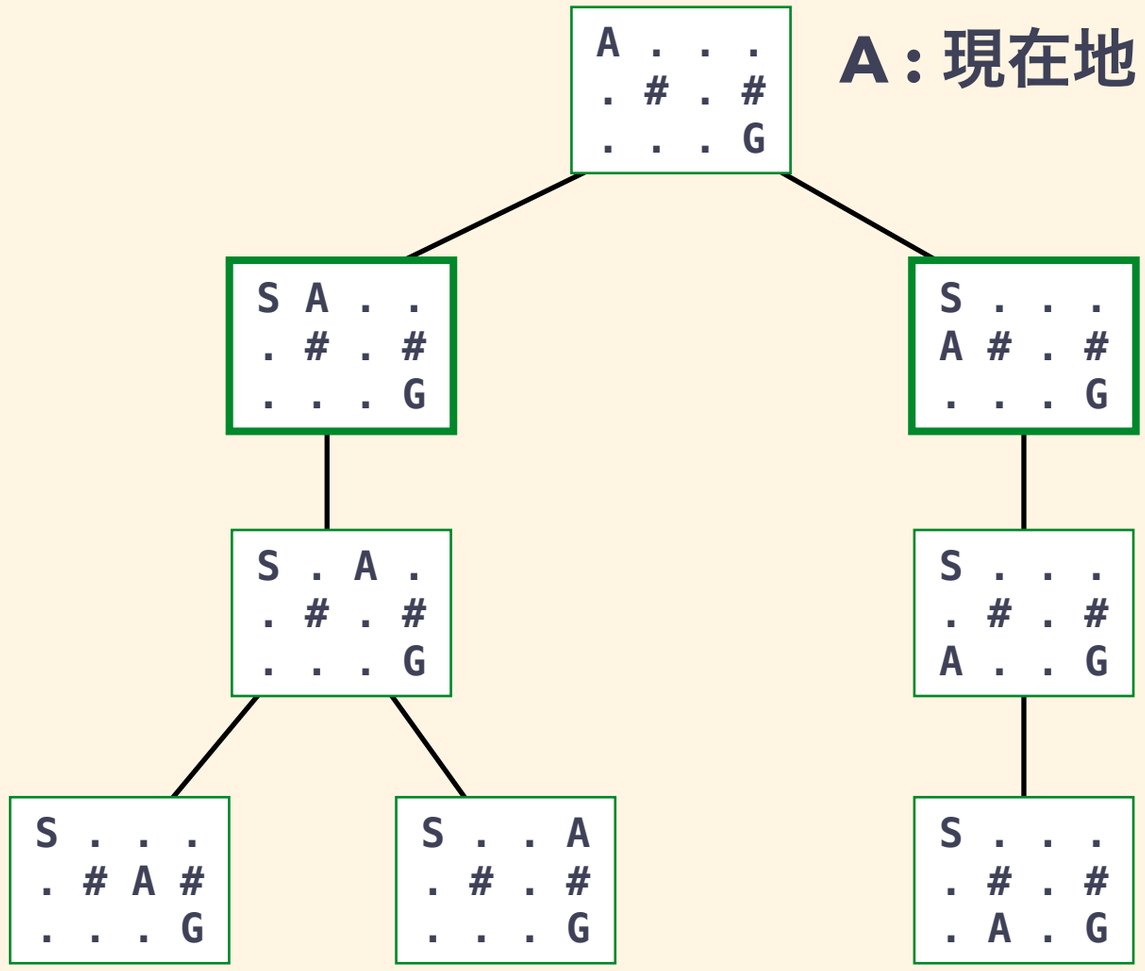


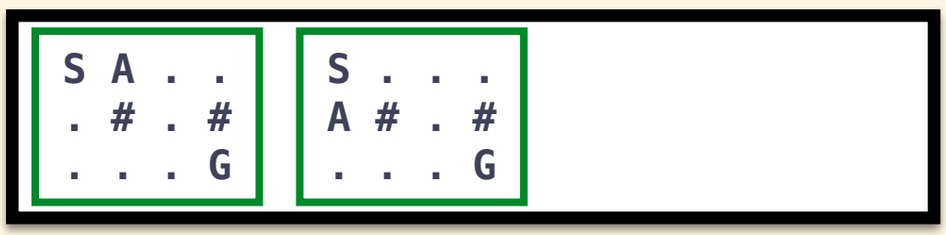



```
que = queue.Queue()
que.put(init_state)    // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる



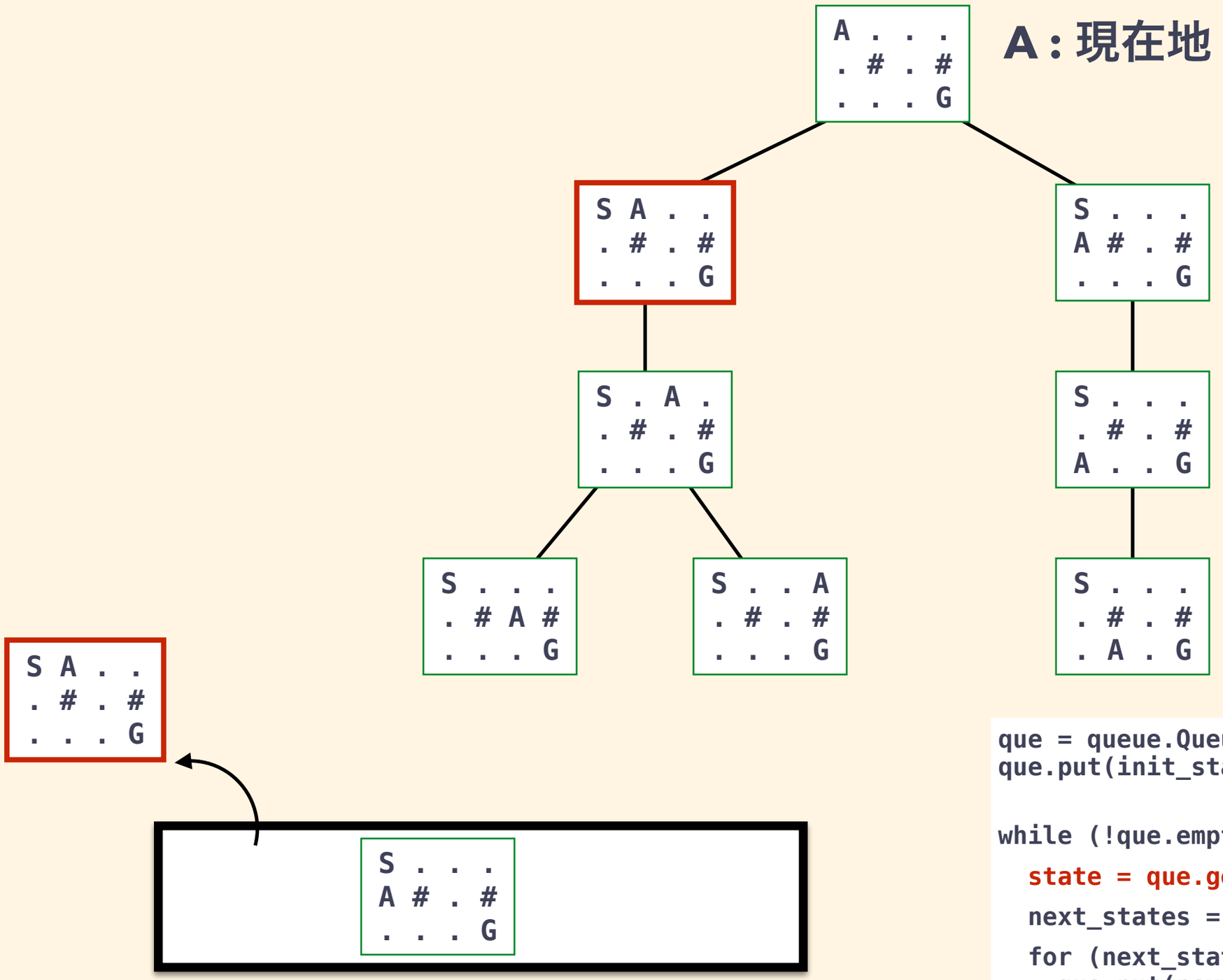


1手動かした状態が入っている

```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

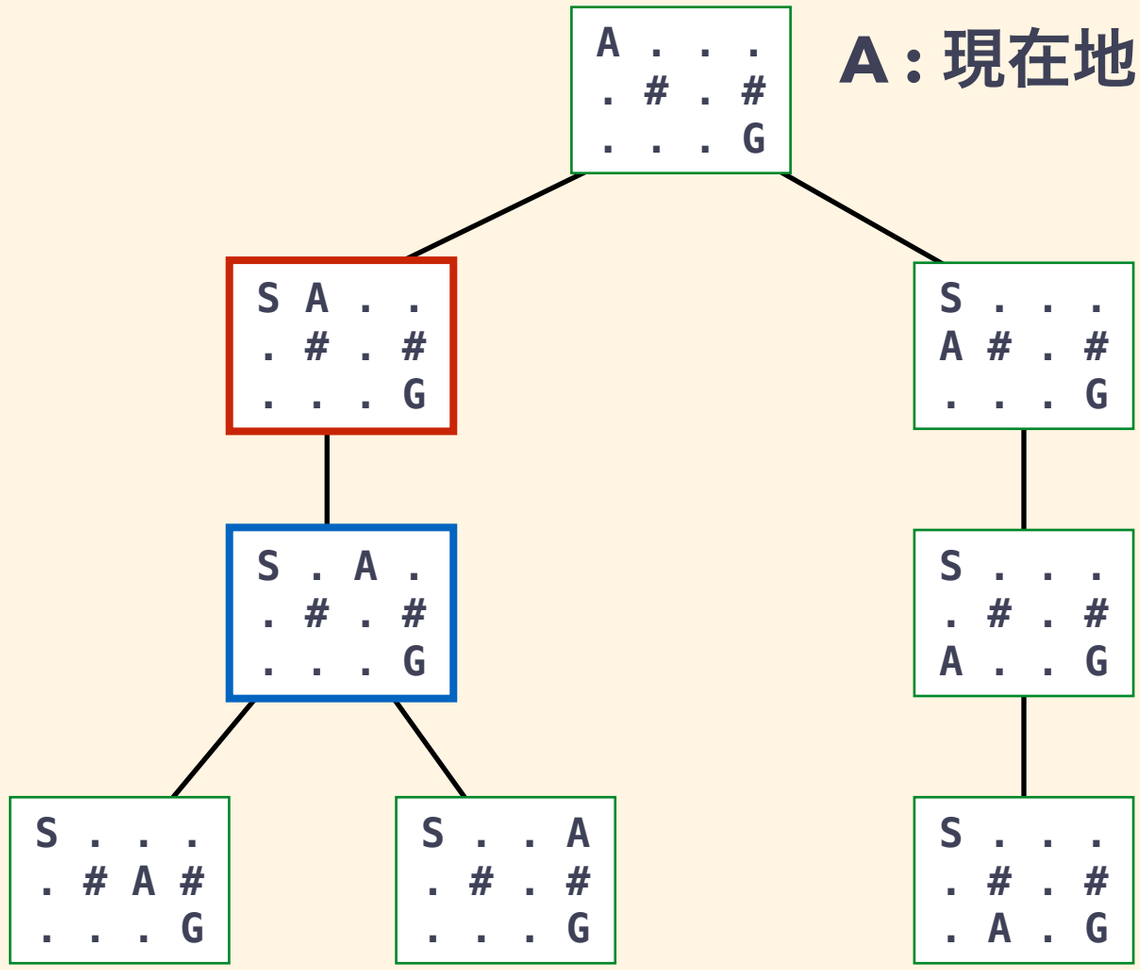
処理の流れを追ってみる

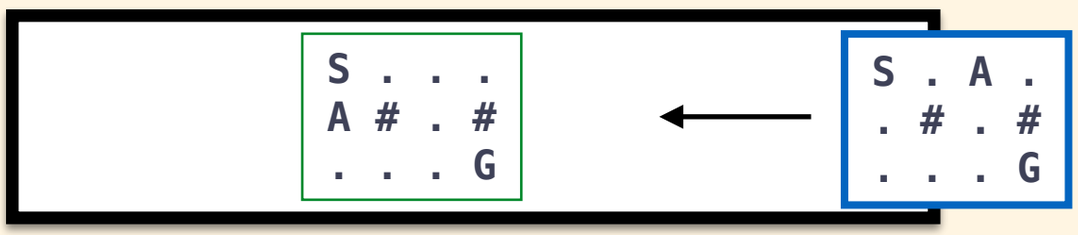



```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる

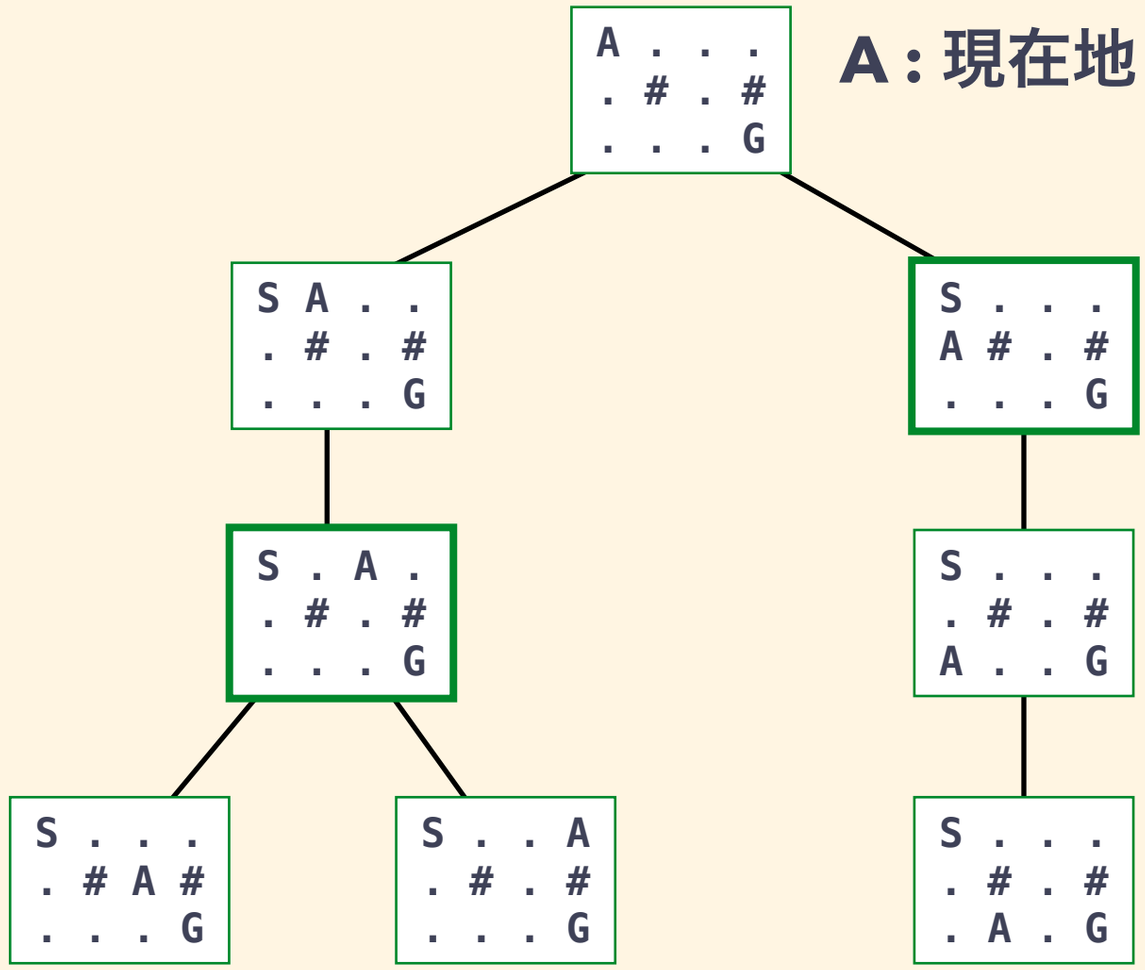


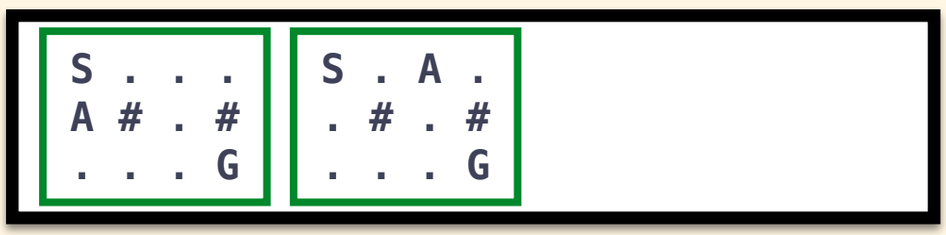



```
que = queue.Queue()
que.put(init_state)    // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる

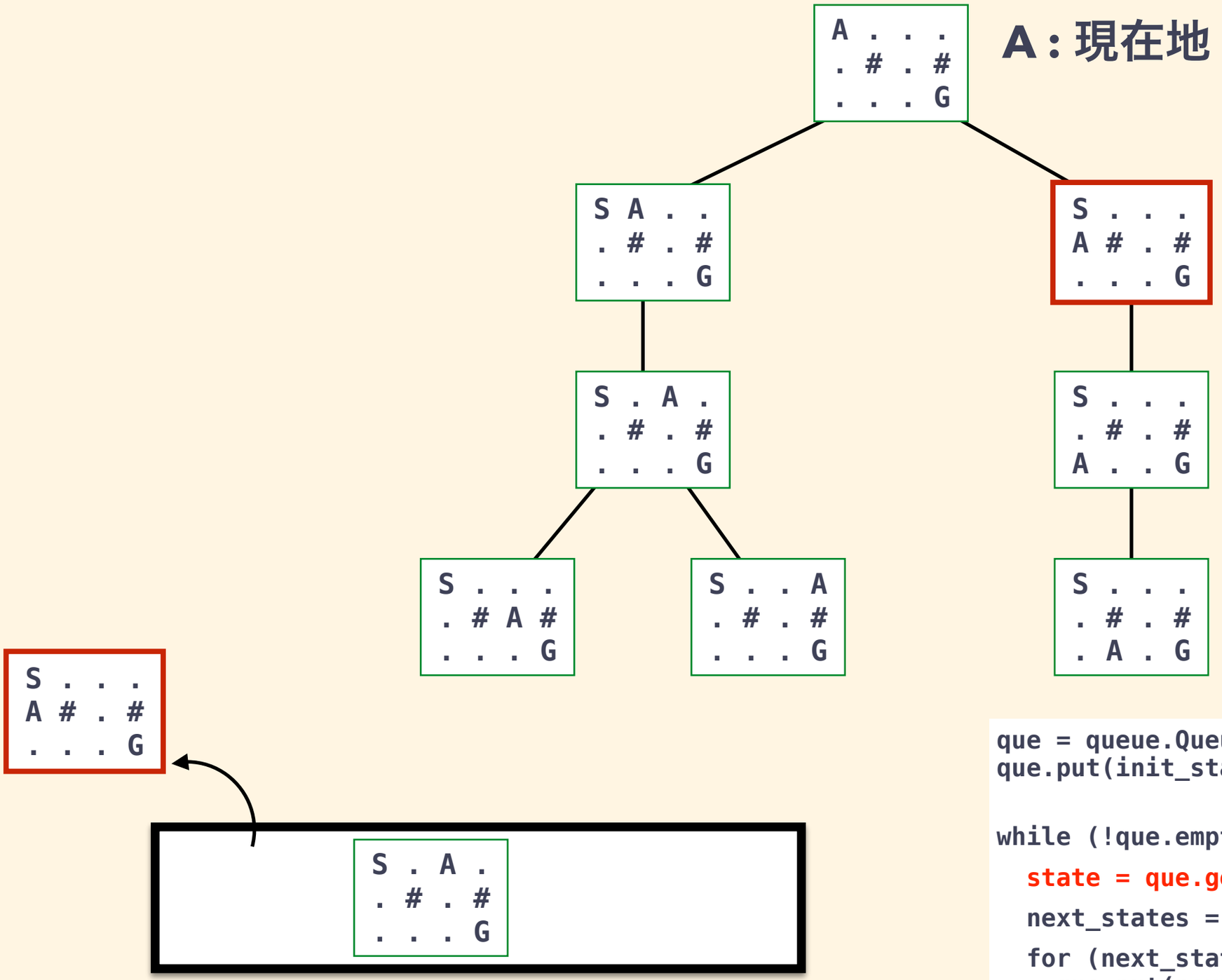





```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

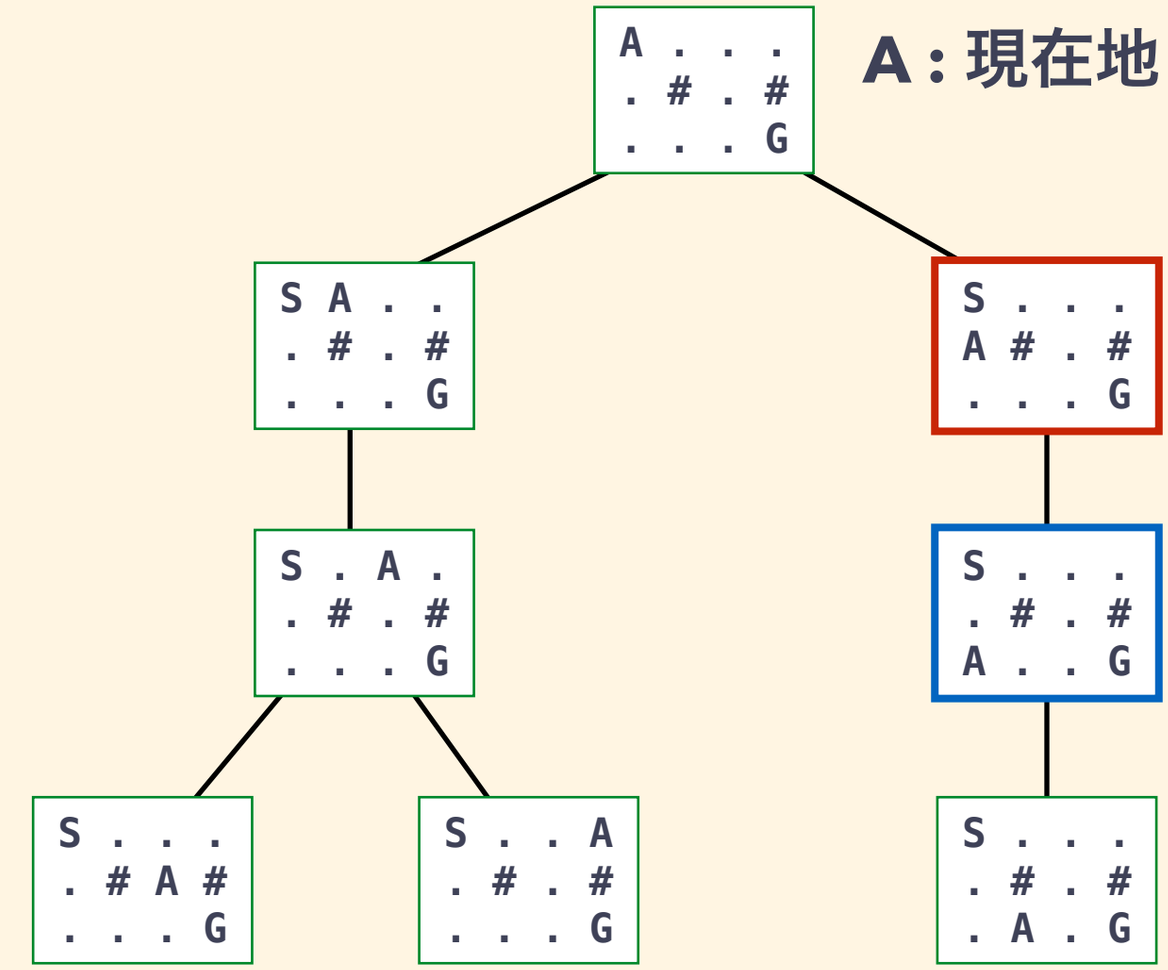
処理の流れを追ってみる




```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

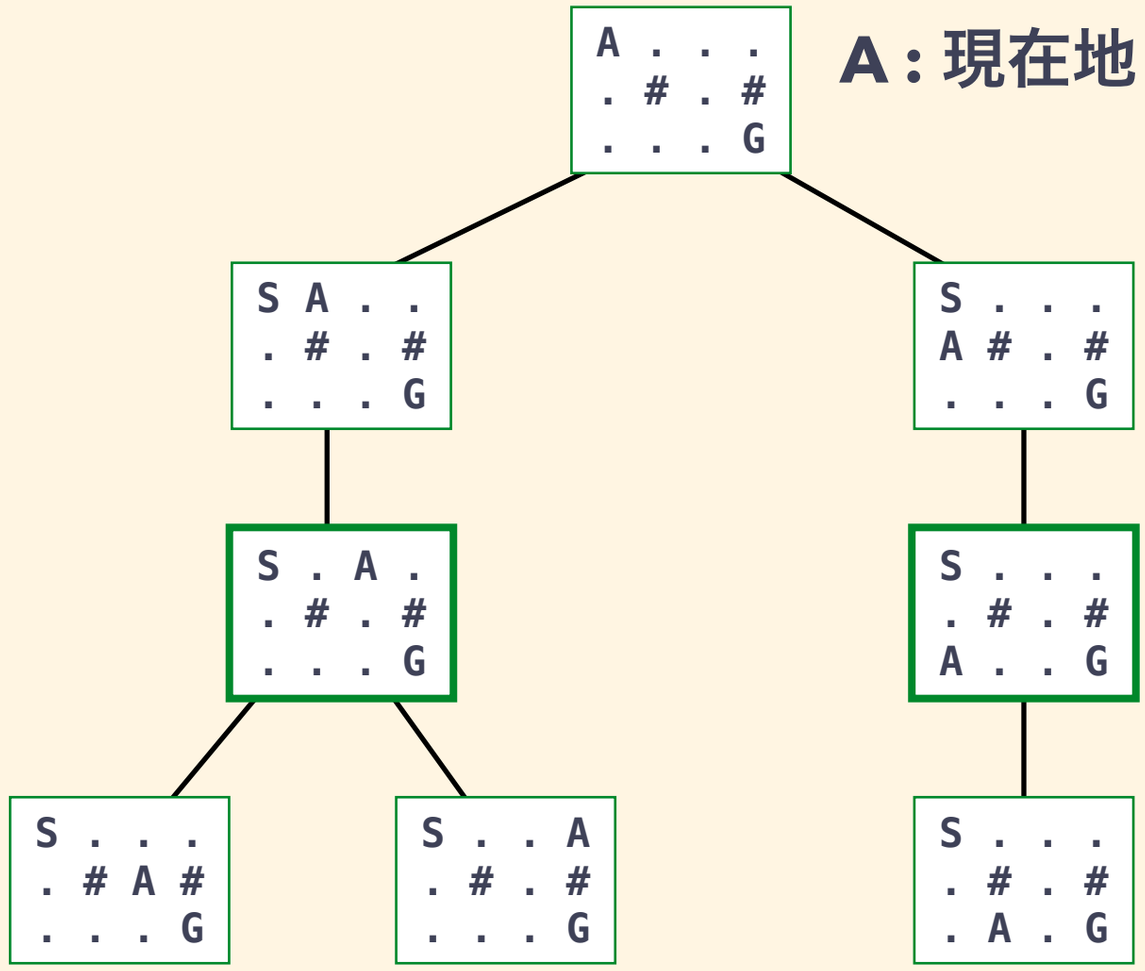
処理の流れを追ってみる

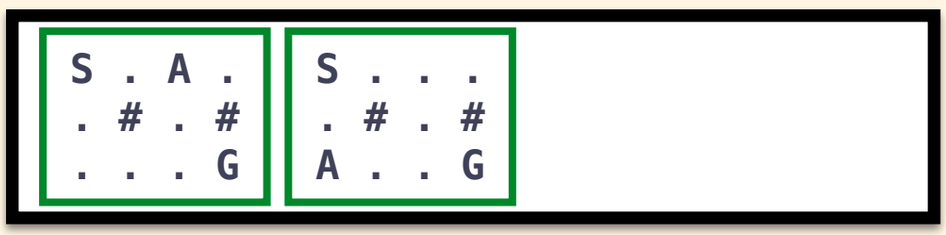



```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる



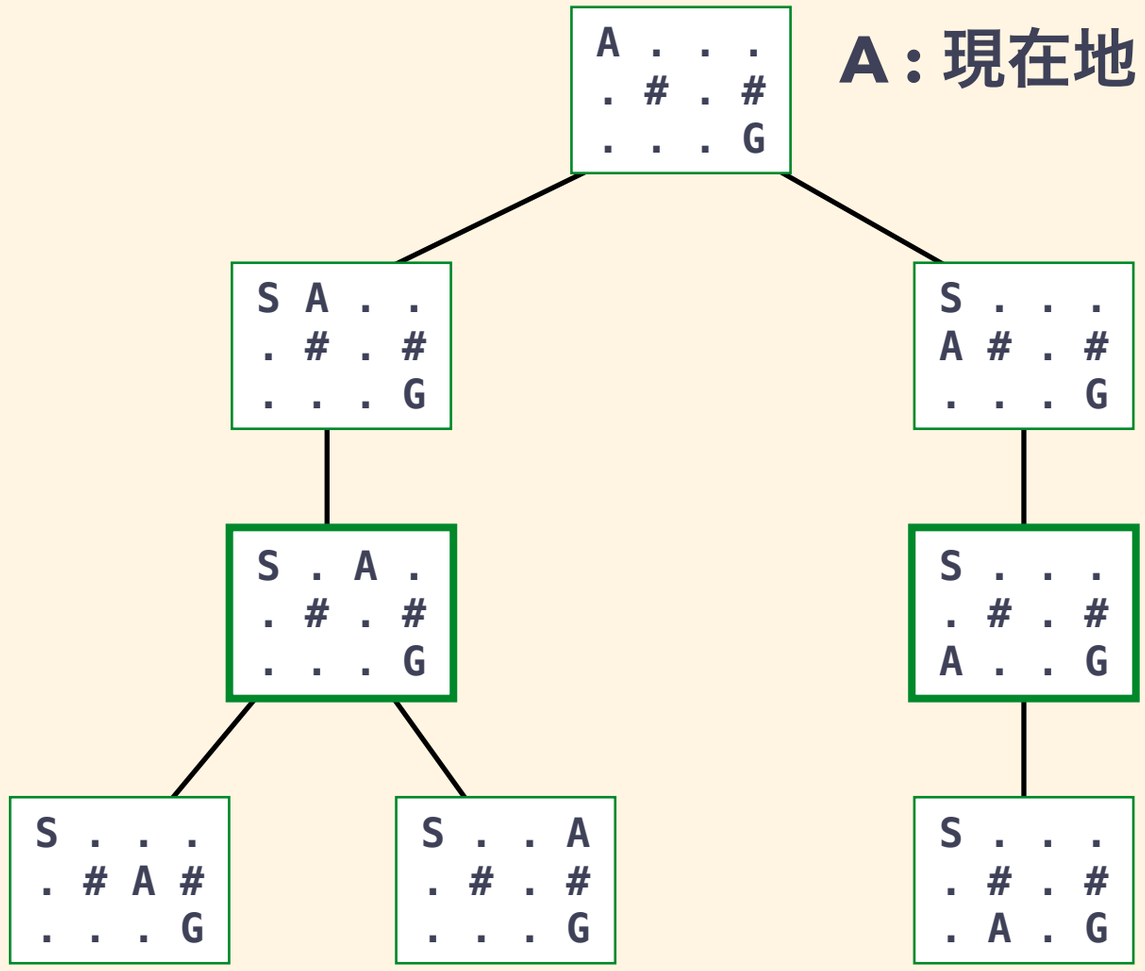


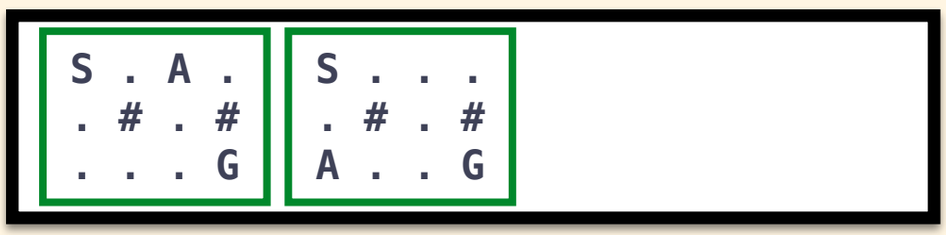
2手動かした状態が入っている

```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

処理の流れを追ってみる





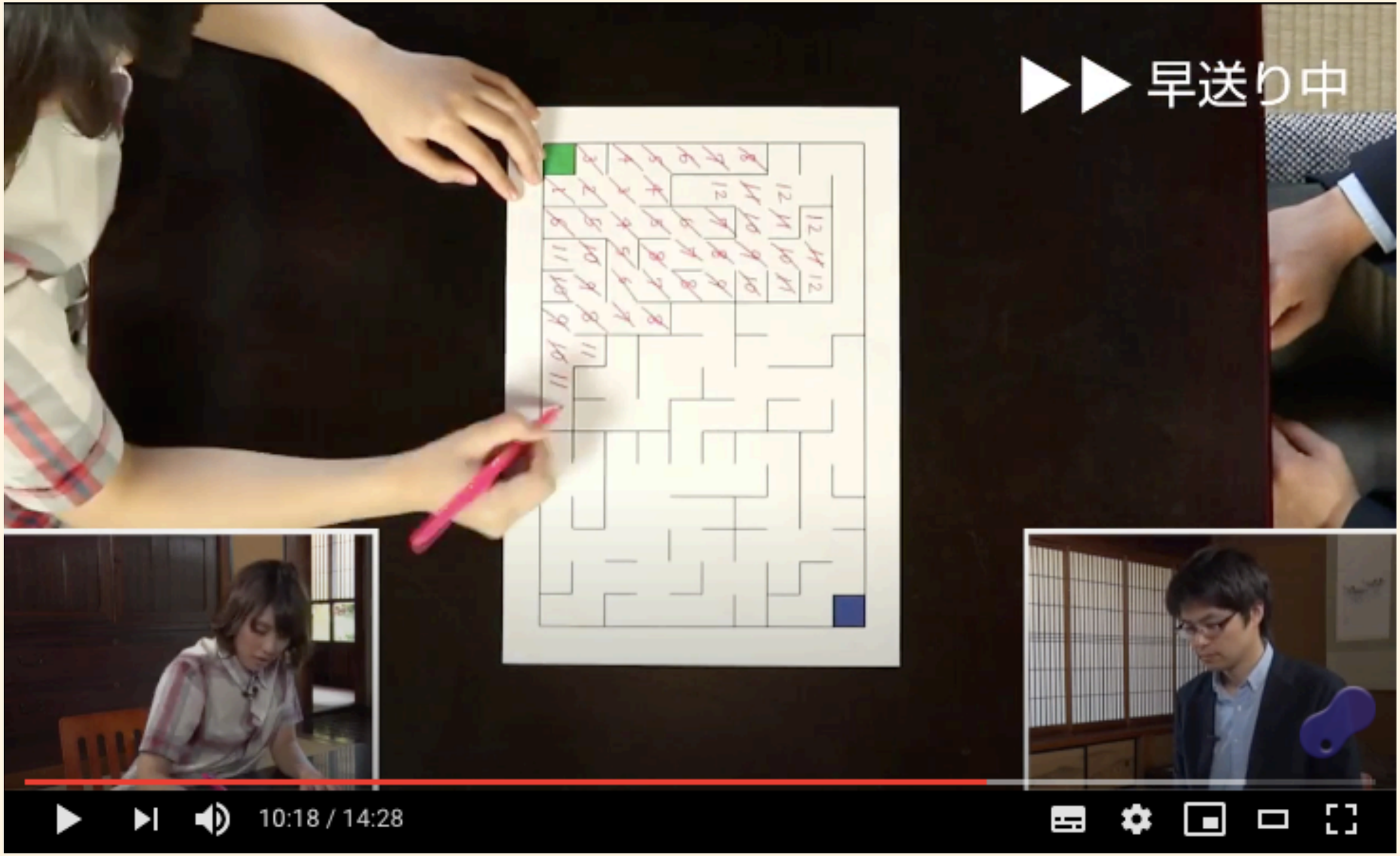
ゴールを見つけるまで繰り返す

```
que = queue.Queue()
que.put(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.get()    // キューから状態の取り出し
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.put(next_state) // キューへ次の状態を追加
  }
}
```

迷路を解く流れを実演（Python）

大きな迷路ではこんな風に動きます



はじめてのP：第4週： アルゴリズム入門　―プログラミングの理論を体験で学ぼう！2　　https://youtu.be/k1iBqyAf4As?t=521

応用編

8パズルを
解いてみよう

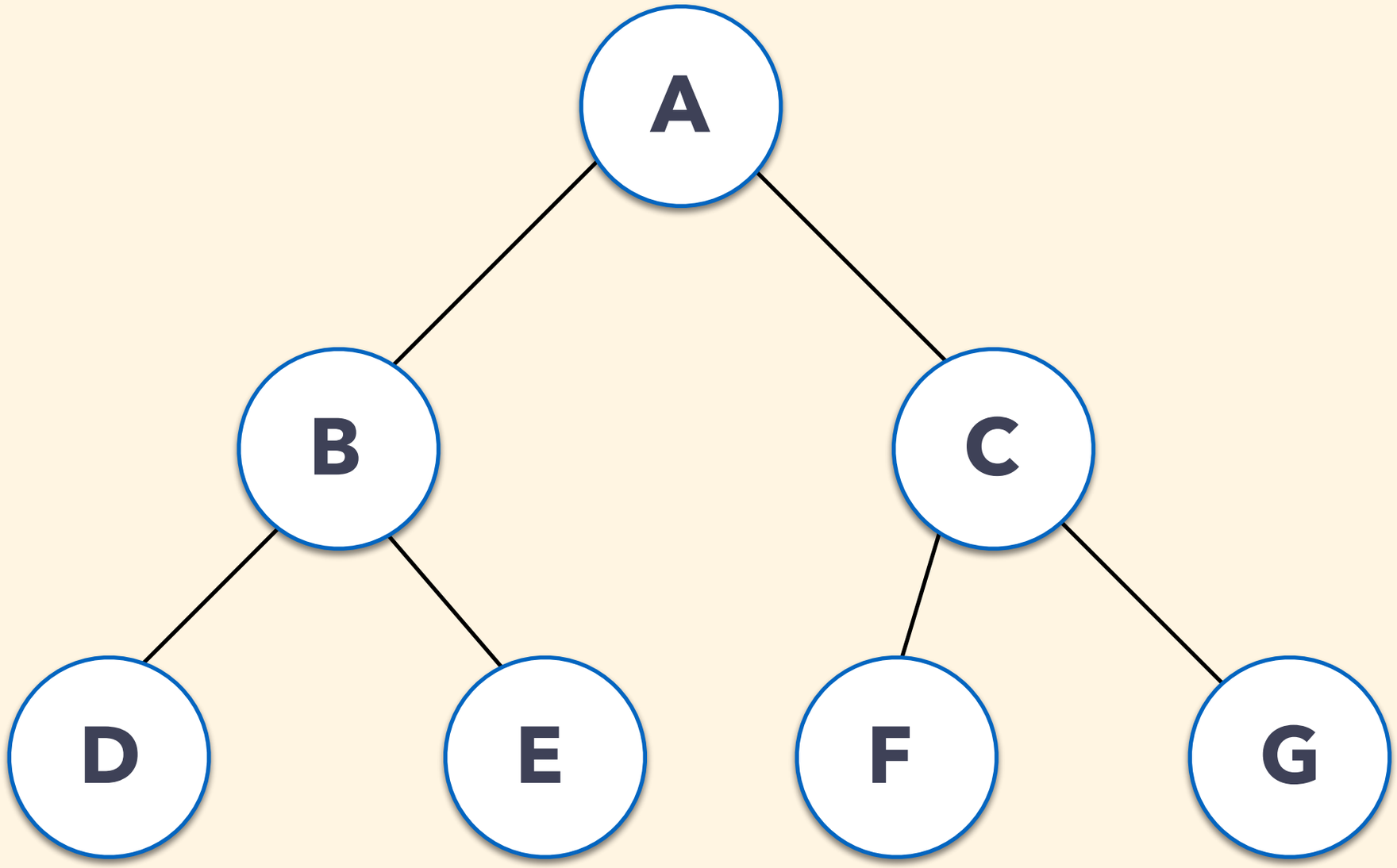
穴埋め問題にしました

- 穴埋め問題 (C++)
  - https://wandbox.org/permlink/nCDU0Tffbq7uMeTf
- 答え
  - https://wandbox.org/permlink/t8NhHj7g9IqQtGd8

アンケート

以下，没スライド

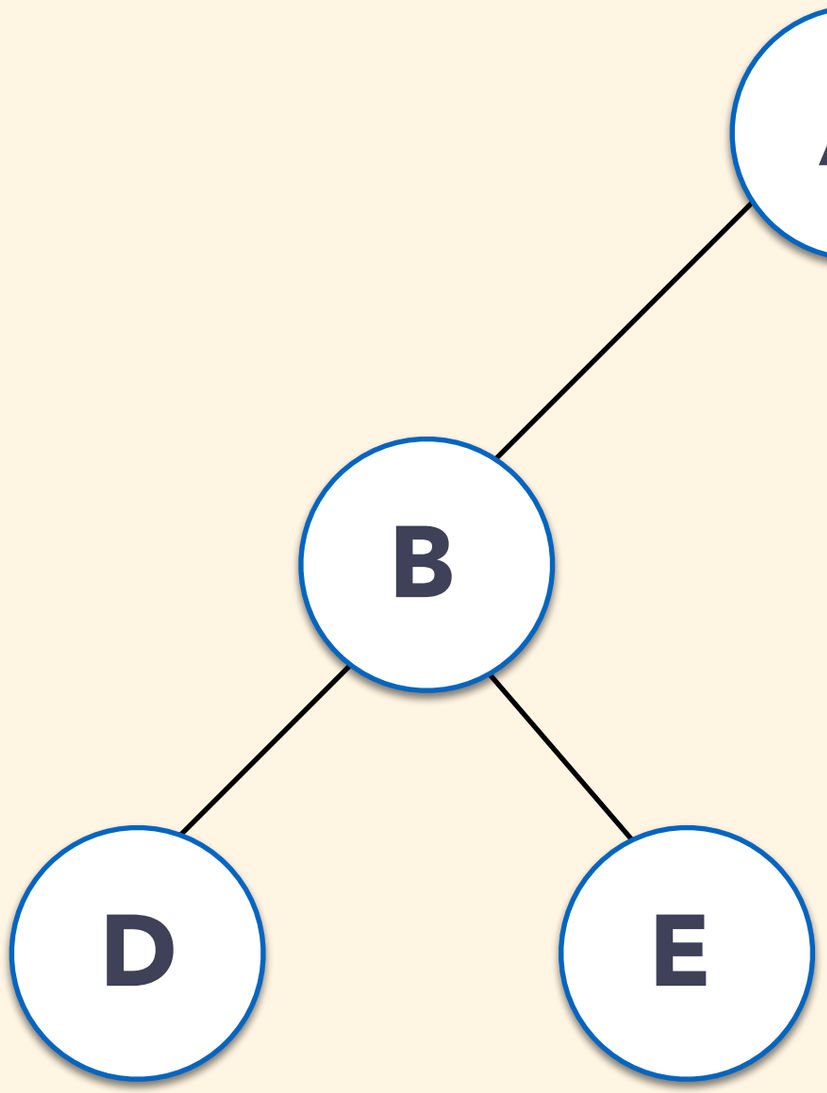
処理の流れを追ってみる

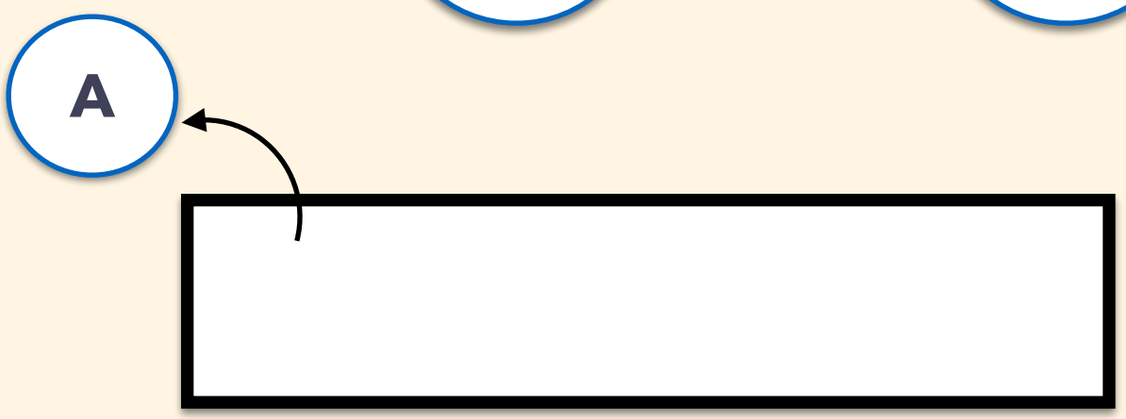



```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

処理の流れを追ってみる

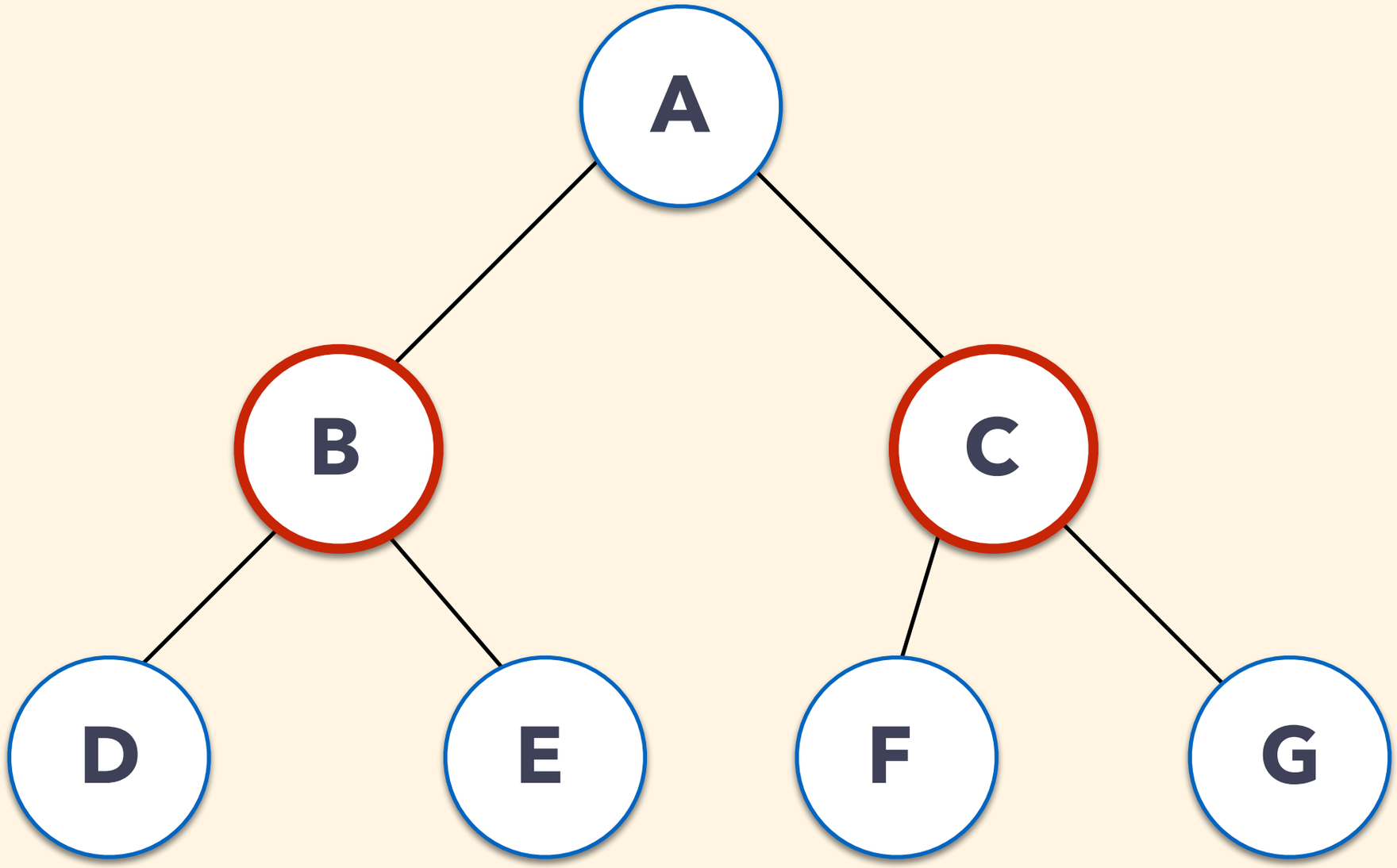





```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

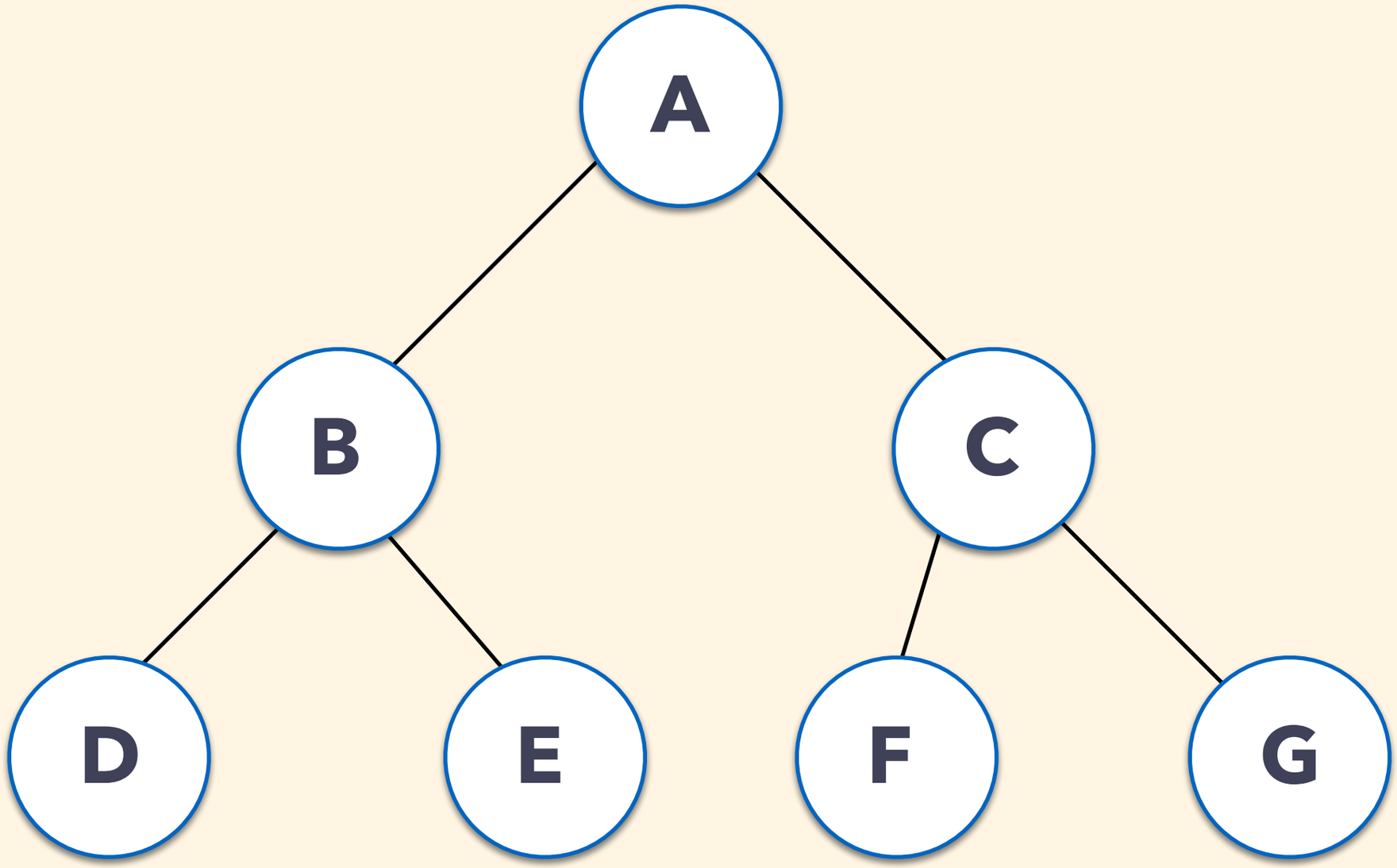
処理の流れを追ってみる

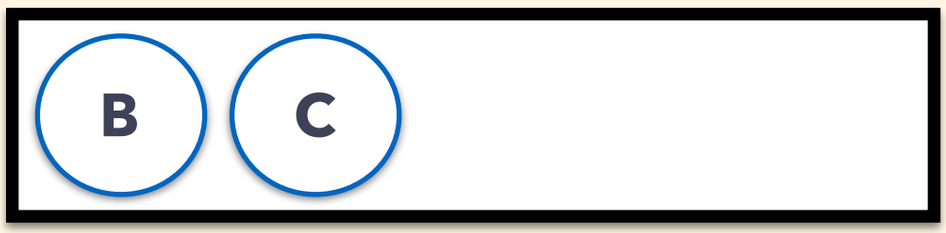



```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

処理の流れを追ってみる

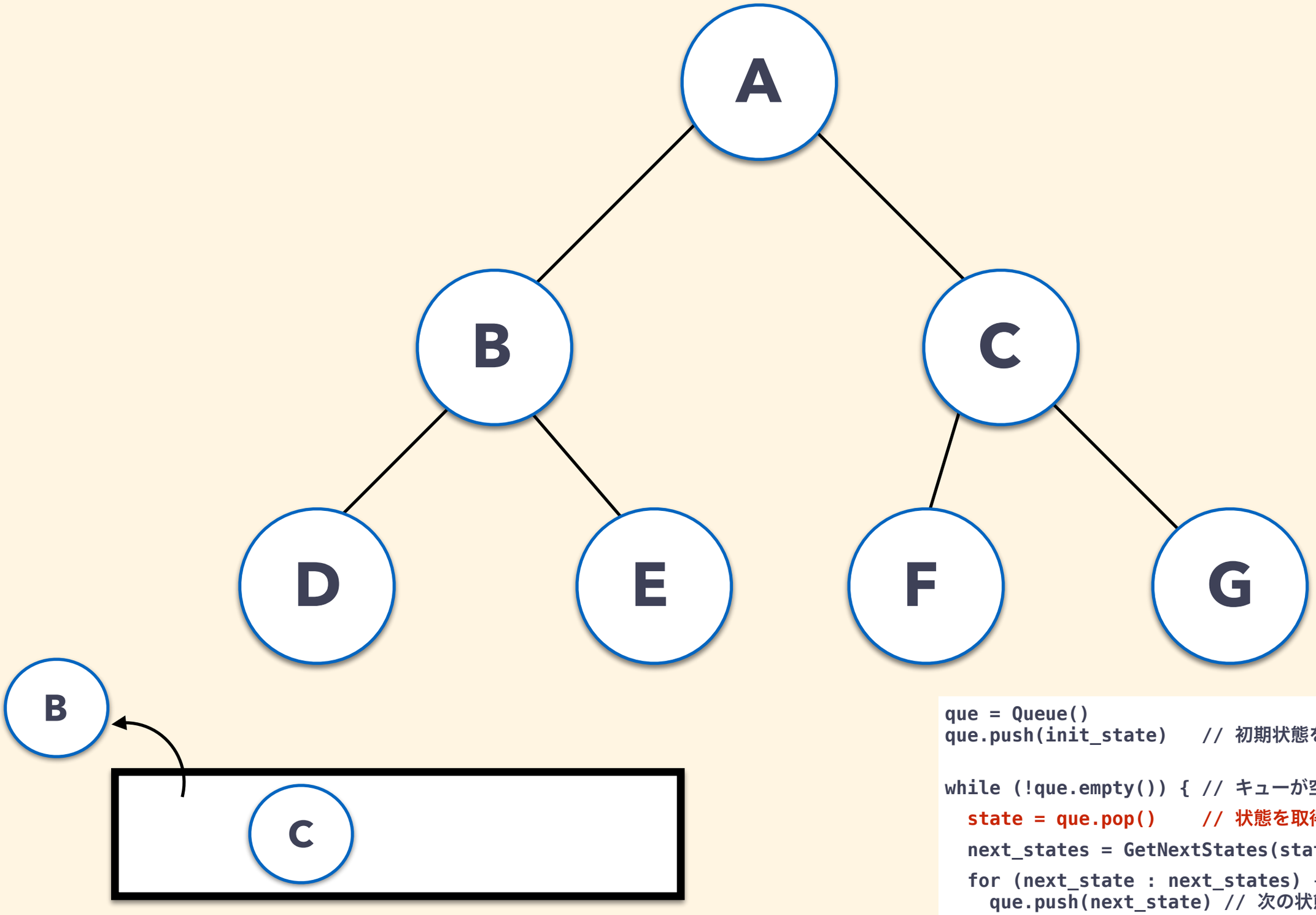





```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

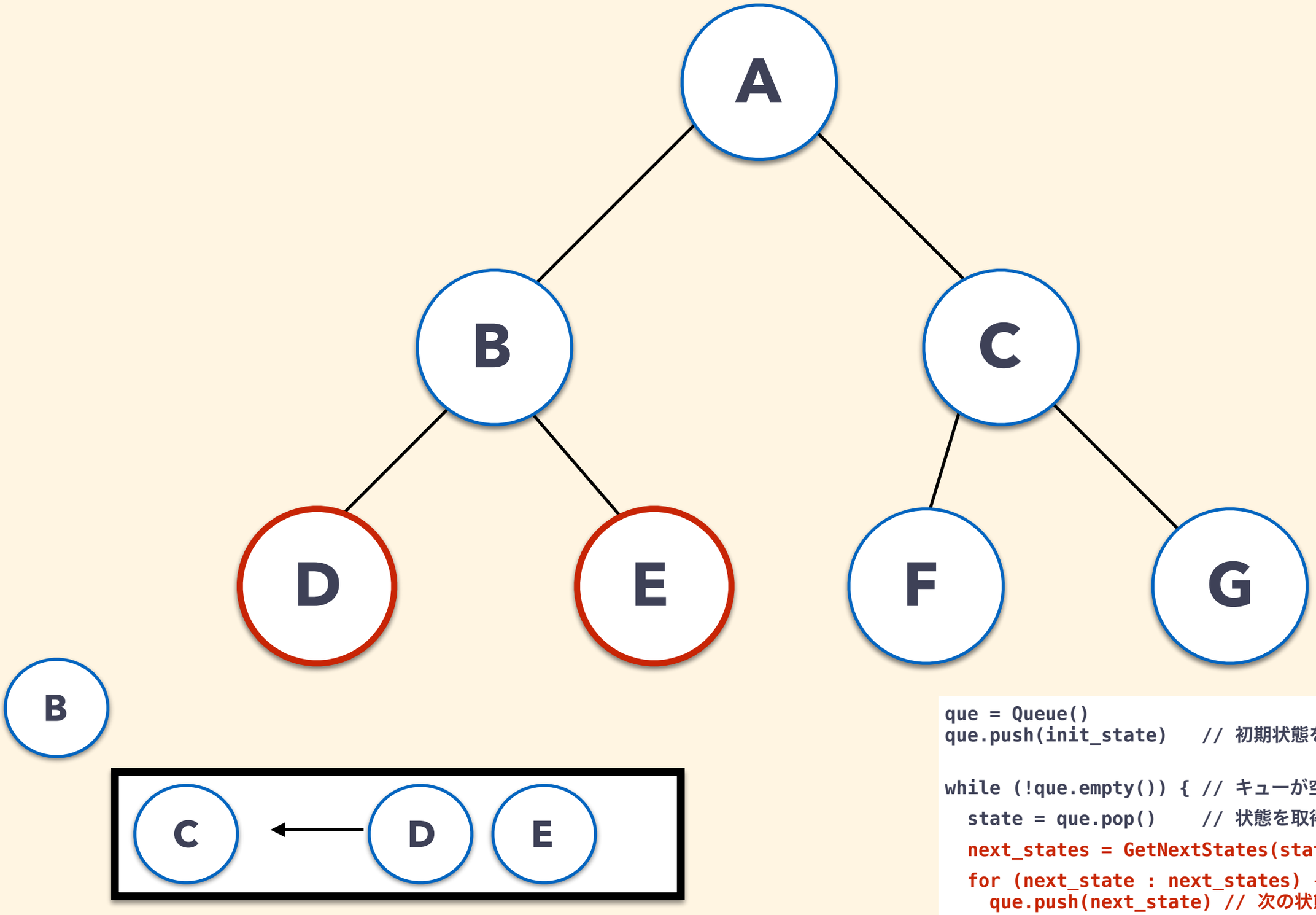
処理の流れを追ってみる




```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

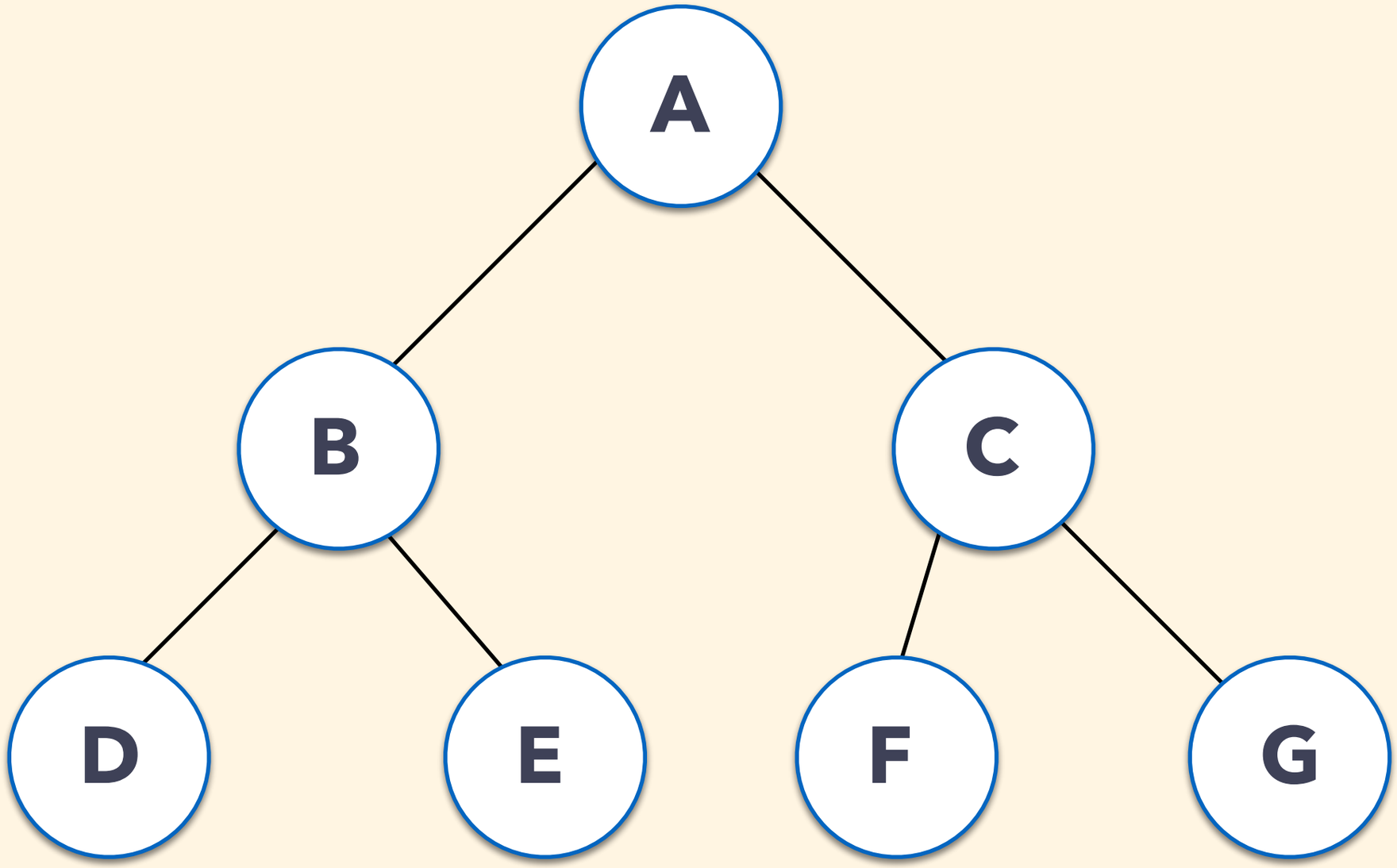
処理の流れを追ってみる

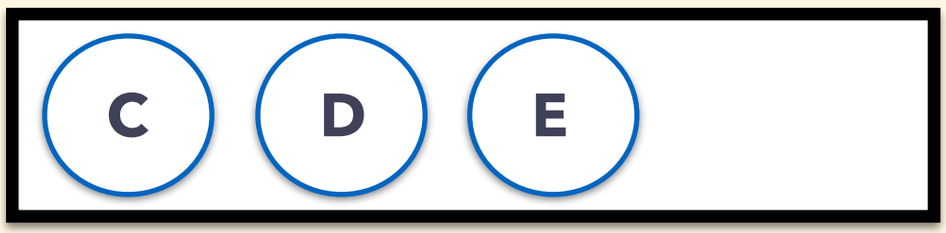



```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```

処理の流れを追ってみる






```
que = Queue()
que.push(init_state)   // 初期状態をキューに追加

while (!que.empty()) { // キューが空でない時
  state = que.pop()    // 状態を取得
  next_states = GetNextStates(state) // 次の状態一覧
  for (next_state : next_states) {
    que.push(next_state) // 次の状態を追加
  }
}
```