

# 知識工学

## 第二回 探索による問題解決

# コメント

パパパコメント. 合言葉は「知識情報工学citns」



<http://papapac.com/post.php?room=知識情報工学citns>



コメントを投稿したい人

教えてもらった部屋名を入れてね

知識情報工学citns

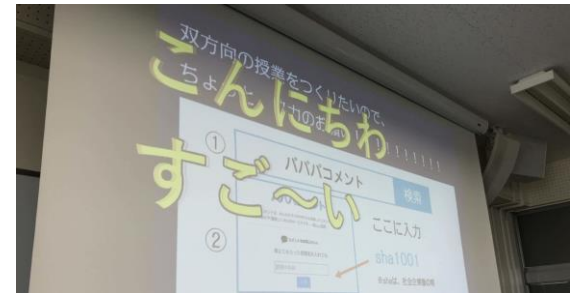
入室



知識情報工学citnsの部屋

コメント送信

コメントは部外者にも見られる可能性があります。個人情報などは送信しないでください。



# 学生の皆さんへのお願い

新型コロナウイルス感染症が社会に蔓延して、1年が経過しました。この間、学生の皆さんもいろんなことを我慢しながら生活してきたと思います。

もう、「そろそろ自由に活動したい」と、強く願っていることと思いますが、皆さんもご存知のとおり、変異株が流行してきており、気を緩められる状況ではありません。

本学でも、これまでに約30名の陽性者が報告されています。学内での濃厚接触者も増加しています。マスクを外しての会話や、学食等での飲食時の会話が濃厚接触者になる最大の原因です。

濃厚接触で「陰性」となっても、2週間は自宅待機を強いられ、不自由な生活を送ることとなります。対面授業の再開とともに、通常に戻りつつある、今の大学生活を維持するには、皆さんの協力が不可欠です。大学内でクラスターが発生すれば、大学を閉鎖しなければなりません。そのようなことにならないためにも、あらためて皆さんにお願いします。学内・学外を問わず、感染防止対策を徹底してください。

- マスクの常時着用
- 手指消毒の徹底
- 食事中は会話をしない
- 友人同士の飲み会、カラオケは自粛する
- 三密回避

そして、体調が少しでもすぐれない日は、無理をせず、自宅で休養してください。

# 前回レビューシート感想

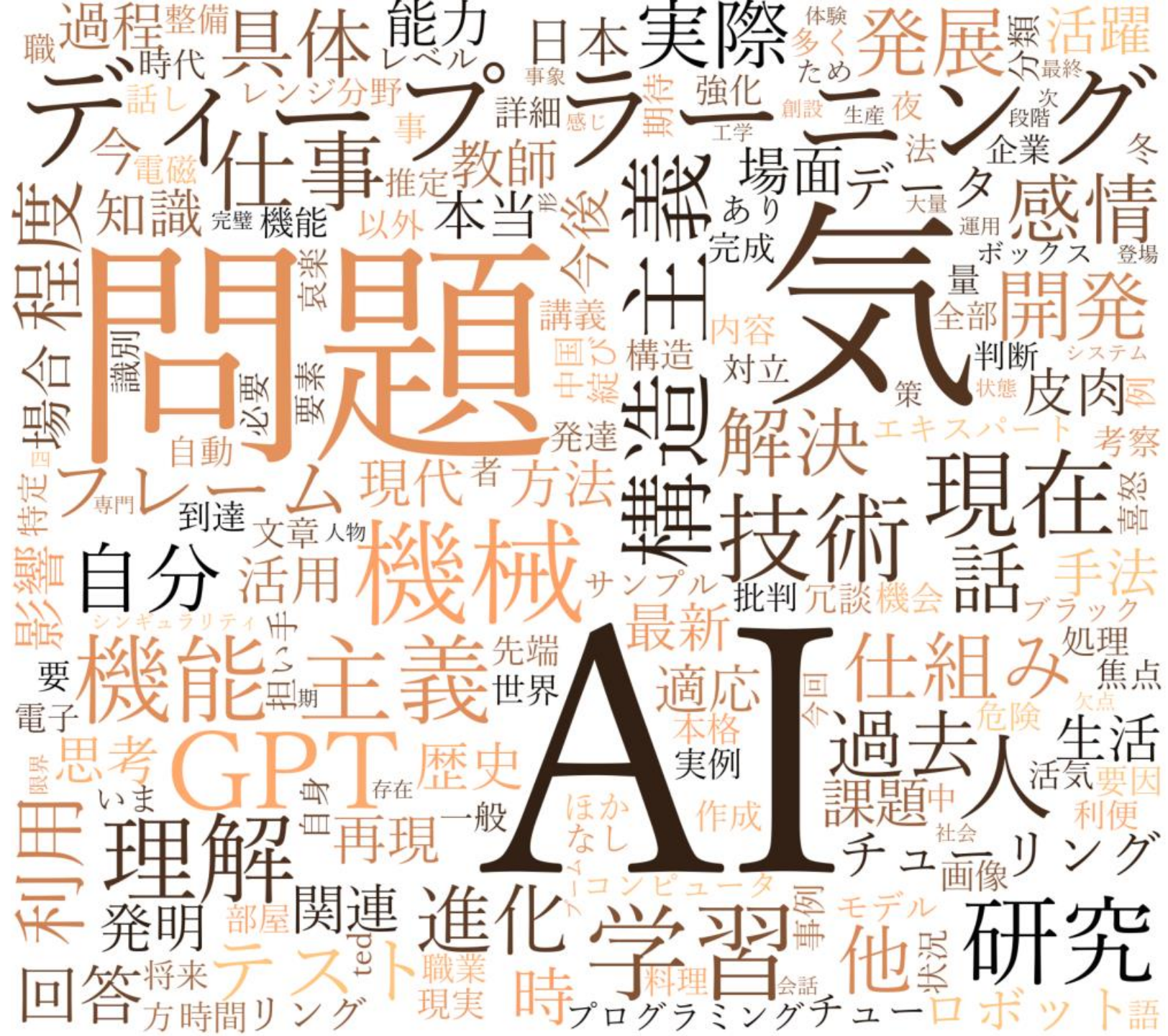
- 人間の仕事を奪われることに不安を感じる.
  - 失業が怖いから、人工知能の研究をしないでほしい
    - 二元的に人間/機械で考えるのではなく、支え合って協働という流れ
- 弱いAIを組み合わせれば強いAIになれるか.
  - 個々のタスクはできるが、日常的な常識は身についていない.

「人間」  
「人工知能」  
助詞などは除いてある





もっと知りたい  
(偶数)



[illegible]



「人間」  
「人工知能」  
助詞などは除いてある





# もっと知りたい (奇数)



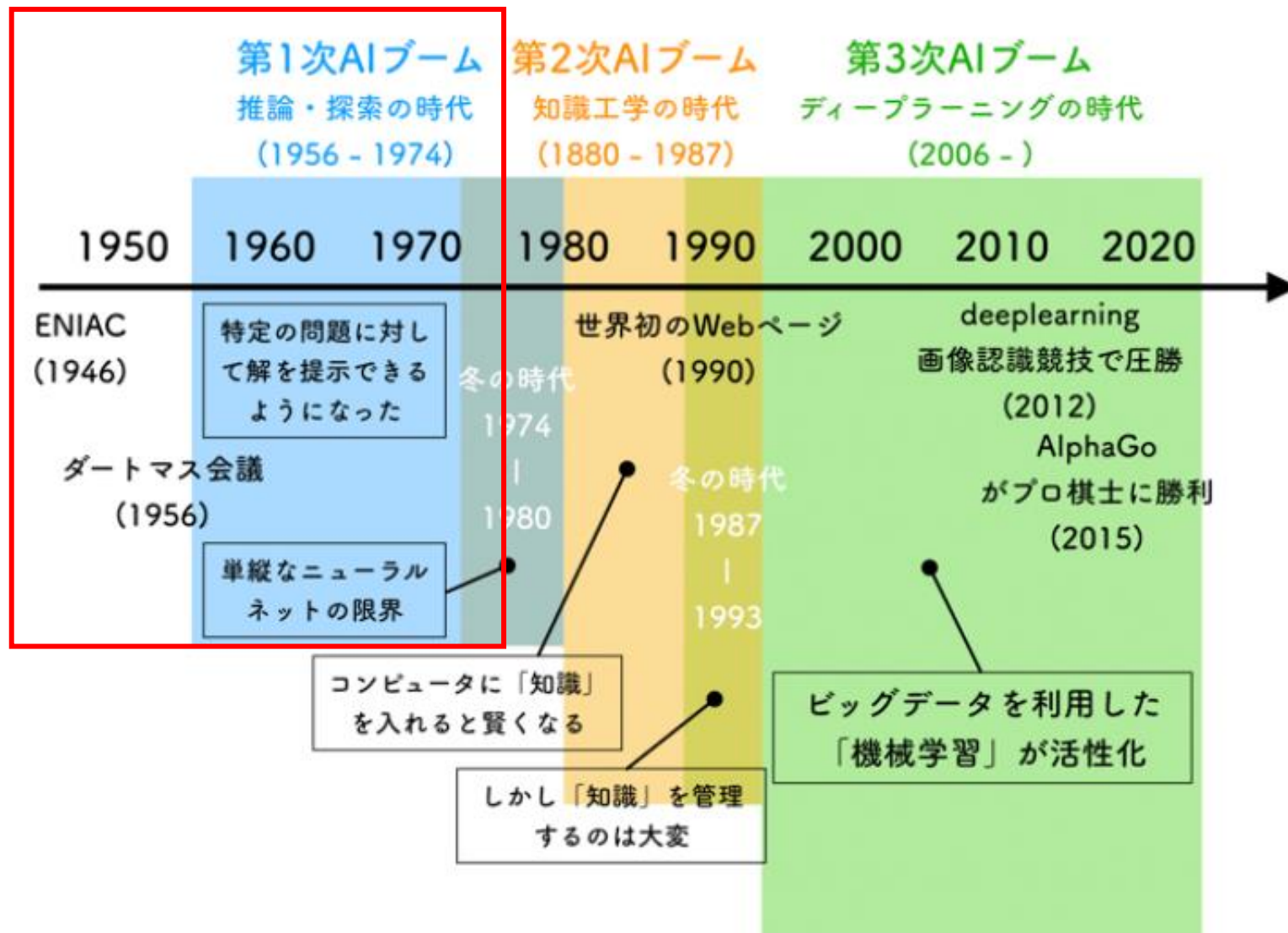
## その他（奇数）





# 今日の内容

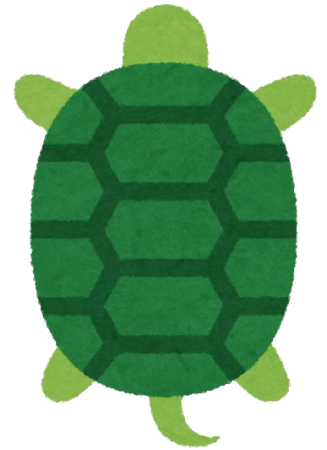
- 1950年代, 60年代あたり
- 単純化された問題を解決するにはどうするかを考える
- 人工知能の基礎の基礎となる問題解決法



# 問題解決の例

- 鶴と亀が合計7匹います。鶴と亀の足の合計は20本です。このとき、鶴と亀はそれぞれ何匹いるのでしょうか？

鶴を $x$ , 亀を $y$  とすると.....





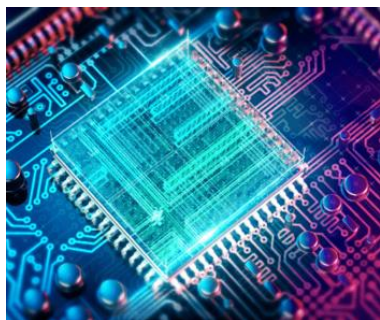
# 解決のために行ったプロセス

- 1. 問題の定式化**：問題の本質的な部分を抽出し，何らかの記法に従って形式的記述を求める
  - 鶴を $x$ , 亀を $y$ と置き換えて方程式を作る
- 2. 形式的処理**：形式的記述を処理して解を求める
  - マトリックスや代入法などを用いて $x$ と $y$ の値を出す
- 3. 問題の対象となる世界での解釈**：上の解を解釈して問題の対象となる世界での意味を求める
  - $(x, y) = (4, 3)$  であるため，鶴は4匹，亀は3匹を意味している

# 人間とコンピュータの世界



1. 何を  $x, y$  と置き, どのように問題を表現するか
3. 求めた  $x, y$  はどのような意味を持つか  
⇒ コンピュータの外の話なので「**外部世界**」



2. 与えられた形式的記述（式）をどのように処理するか  
⇒ コンピュータの中の話なので「**内部世界**」

# コンピュータが得意な部分

外部  
世界

内部  
世界

**1. 問題の定式化**：問題の本質的な部分を抽出し，何らかの記法に従って形式的記述を求める

- 鶴を $x$ , 亀を $y$ と置き換えて方程式を作る



**2. 形式的処理**：形式的記述を処理して解を求める

- 代入法などを用いて $x$ と $y$ の値を出す



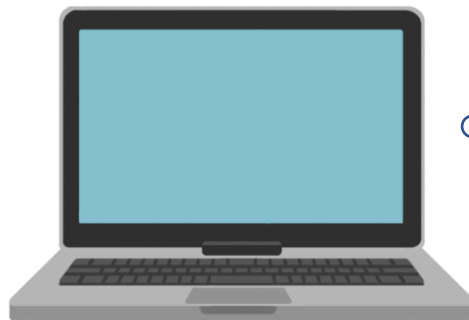
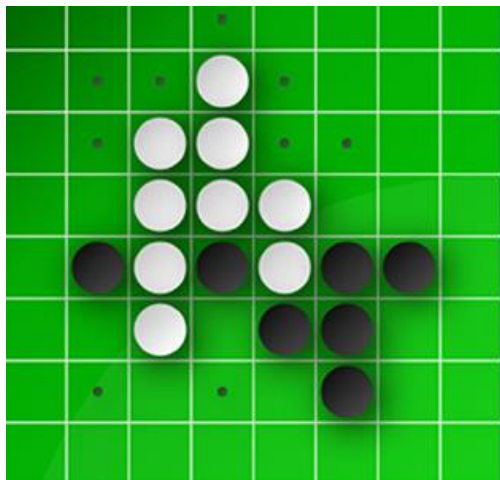
**3. 問題の対象となる世界での解釈**：上の解を解釈して問題の対象となる世界での意味を求める

- $(x, y) = (4, 3)$  であるため，鶴は4匹，亀は3匹を意味している

# 問題解決のための探索

- 現実でもゲームでも，明確な答えはひと目では分からない
- 取れる手段の中から，良さそうなものを探る（**探索する**）ことが必要になる

※ 現実的な問題を扱うと，指数関数的に探索空間が増大して組合せ爆発になるため，初期は小規模な問題を扱わざるを得なかった。（**トイ問題**）



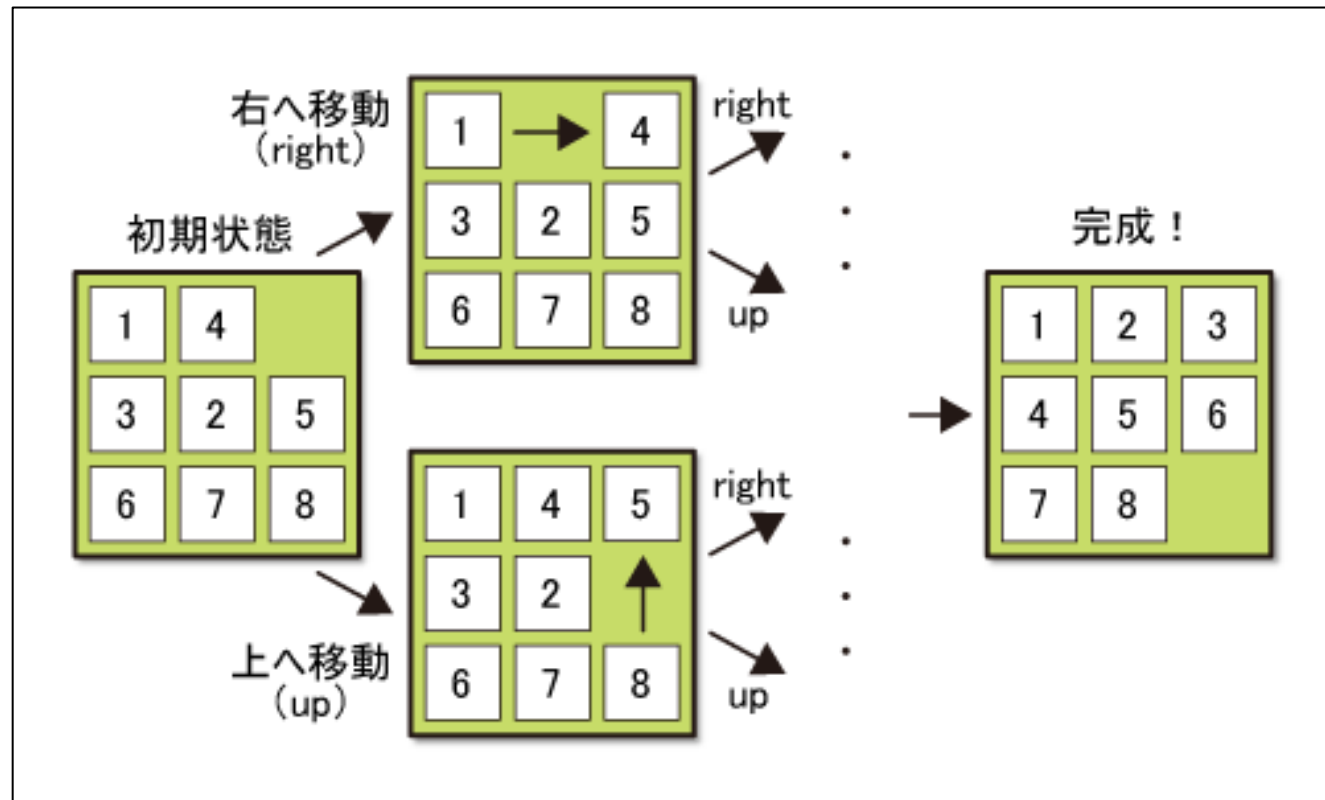
複数場所に置ける  
けど，勝つ可能性  
が高いのは.....？



# 8パズル

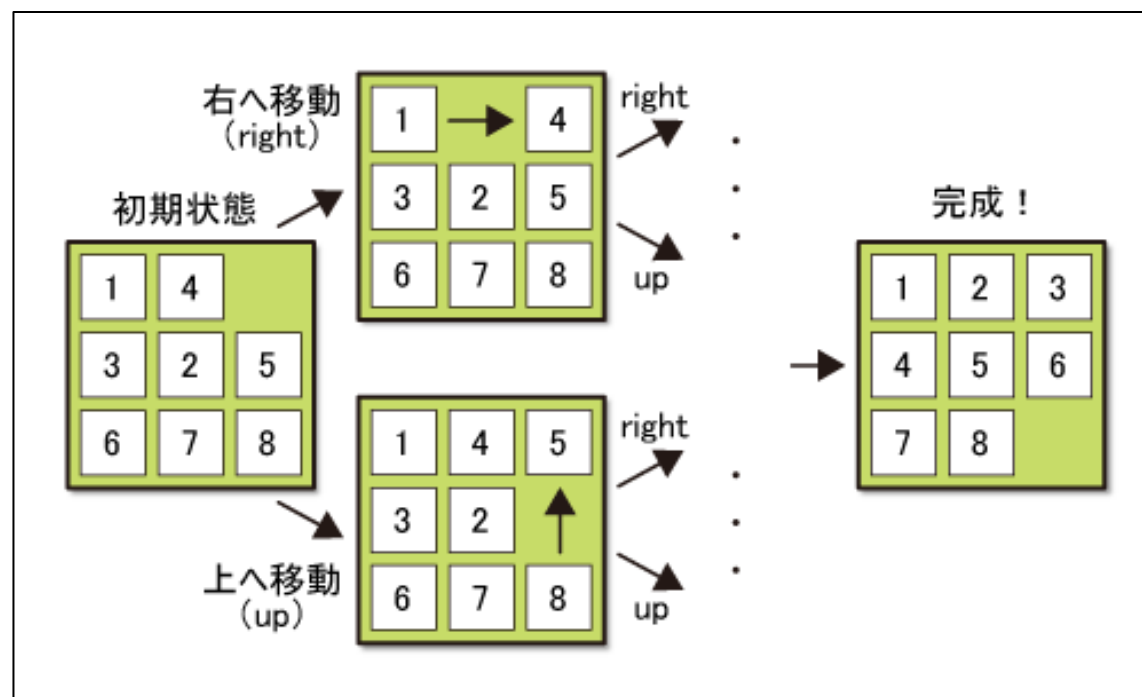
一番右の状態になるように、  
パネルを移動させる

空いているマスへ何を動かせば  
よいかを探索する



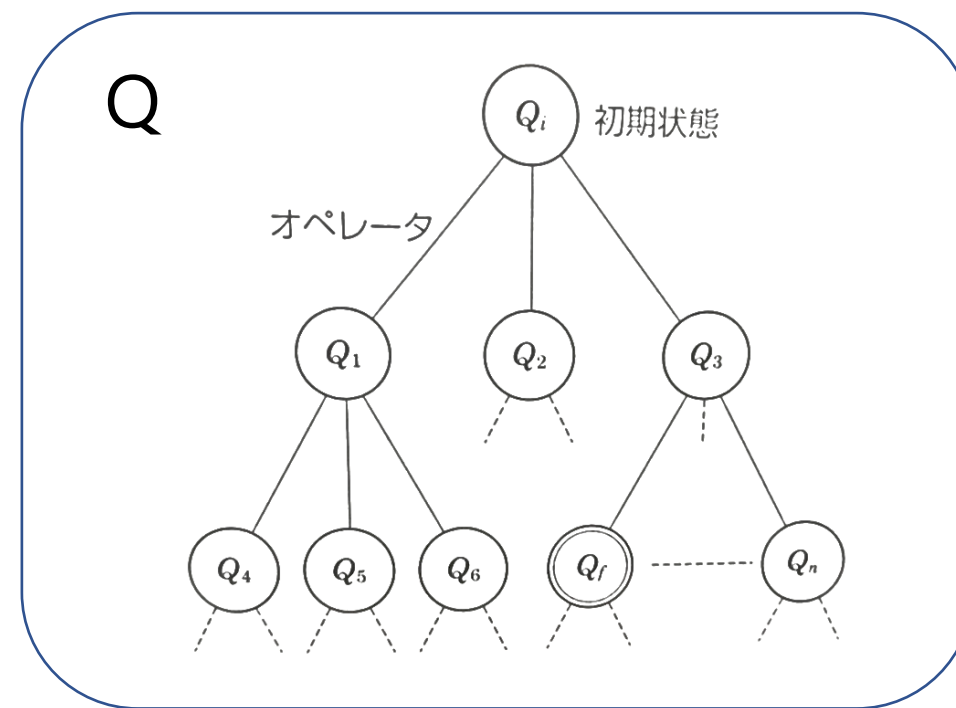
# 定式化の手法 1 状態空間法

- 状態空間集合 $Q$  : 問題となる全ての分岐ルート集合
- オペレータ集合 $O$  : マスを動かす方法
- 状態遷移関数 $\Psi : Q \times O \rightarrow Q$  オペレータを適用する関数
- 初期状態 $Q_i$  : 一番左の図
- 終了状態 $Q_f$  : 一番右の図



# 定式化の手法 1 状態空間法

- 状態空間集合 $Q$ ：問題となる全ての分岐ルート集合
- オペレータ集合 $O$ ：マス进行動かす方法
- 状態遷移関数 $\Psi : Q \times O \rightarrow Q$ . オペレータを適用する関数
- 初期状態 $Q_i$ ：一番左の図
- 終了状態 $Q_f$ ：一番右の図

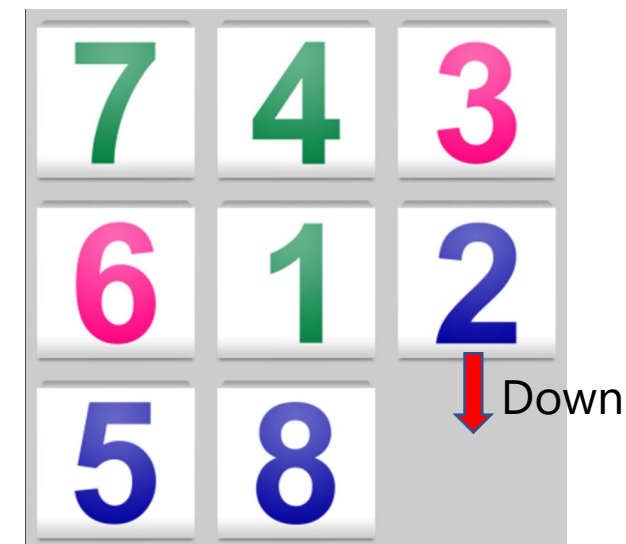


# 定式化の手法 1 状態空間法

- 状態空間集合 $Q$  : 問題となる全ての分岐ルート集合
- オペレータ集合 $O$  : マスを動かす方法
- 状態遷移関数 $\Psi : Q \times O \rightarrow Q$  オペレータを適用する関数
- 初期状態 $Q_i$  : 一番左の図
- 終了状態 $Q_f$  : 一番右の図

$O$

UP	: if 空所の下に駒がある	then 下の駒を上を動かす
Down	: if 空所の上に駒がある	then 上の駒を下を動かす
Right	: if 空所の左に駒がある	then 左の駒を右を動かす
Left	: if 空所の右に駒がある	then 右の駒を左を動かす





# 定式化の手法 1 状態空間法

- 状態空間集合 $Q$  : 問題となる全ての分岐ルート of 集合
- オペレータ集合 $O$  : マスを動かす方法
- 状態遷移関数 $\psi : Q \times O \rightarrow Q$  オペレータを適用する関数.
- 初期状態 $Q_i$  : 一番左の図
- 終了状態 $Q_f$  : 一番右の図

7	4	3
6	1	2
5	8	

$\times \psi \rightarrow$   
Down

7	4	3
6	1	
5	8	2

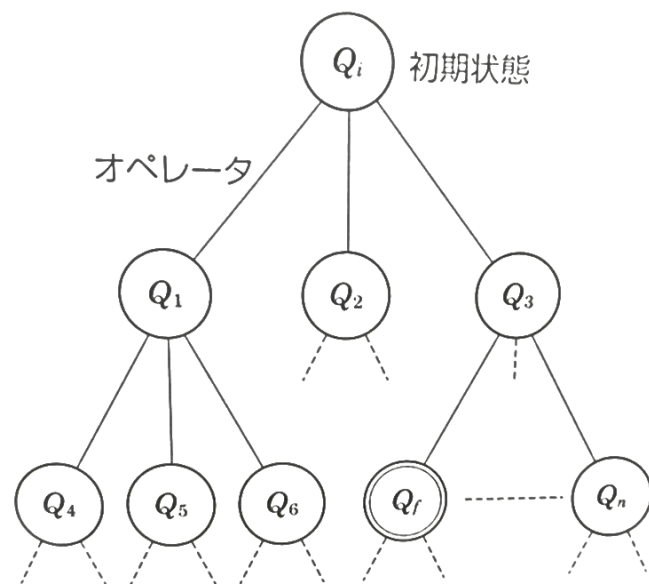
$[[7, 4, 3],$   
 $[6, 1, 2],$   
 $[5, 8, 0]]$

$\times \psi \rightarrow$   
Down

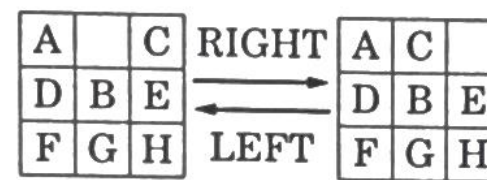
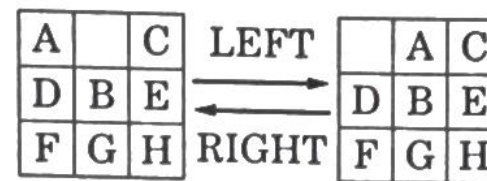
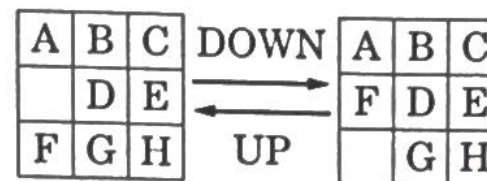
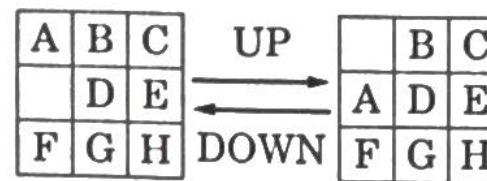
$[[7, 4, 3],$   
 $[6, 1, 0],$   
 $[5, 8, 2]]$

# 状態表現と状態遷移

問題の状態を何らかの形で表現したものを**状態表現**といい、  
どのように移り変わるかを示したものを**状態遷移**という



木による状態表現



空白の操作から見た状態遷移ルール

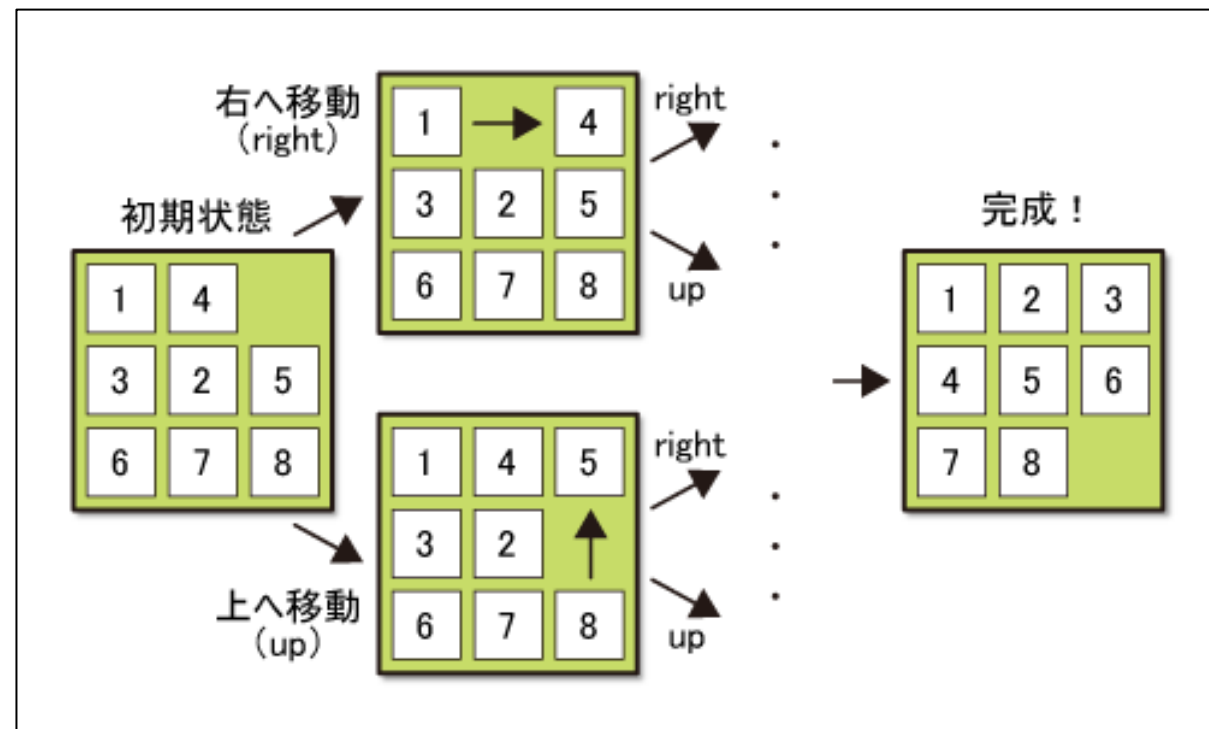
# 前向き探索と後ろ向き探索

- **前向き探索**

- 初期状態から考える

- **後ろ向き探索**

- 最終状態から探っていく
- 終わりの手前を考える



# 定式化の手法 2 問題分割法

- 東京から大阪まで行きたいとき、複数の選択肢が考えられる
1. 飛行機で行く
  2. 車で行く
  3. フェリーで行く
  4. 鉄道で行く





# 定式化の手法 2 問題分割法

- 東京から大阪まで行きたいとき、複数の選択肢が考えられる

1. 飛行機で行く
2. 車で行く
3. フェリーで行く
4. 鉄道で行く

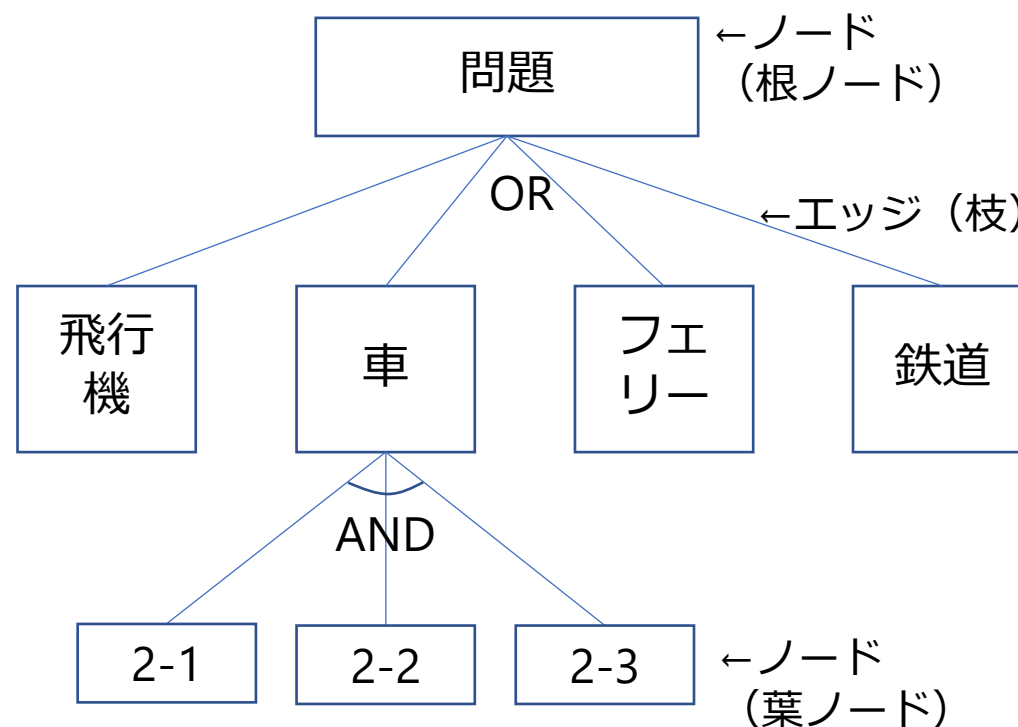


- 2-1 東京から静岡まで行く
- 2-2 静岡から名古屋へ行く
- 2-3 名古屋から大阪へ行く



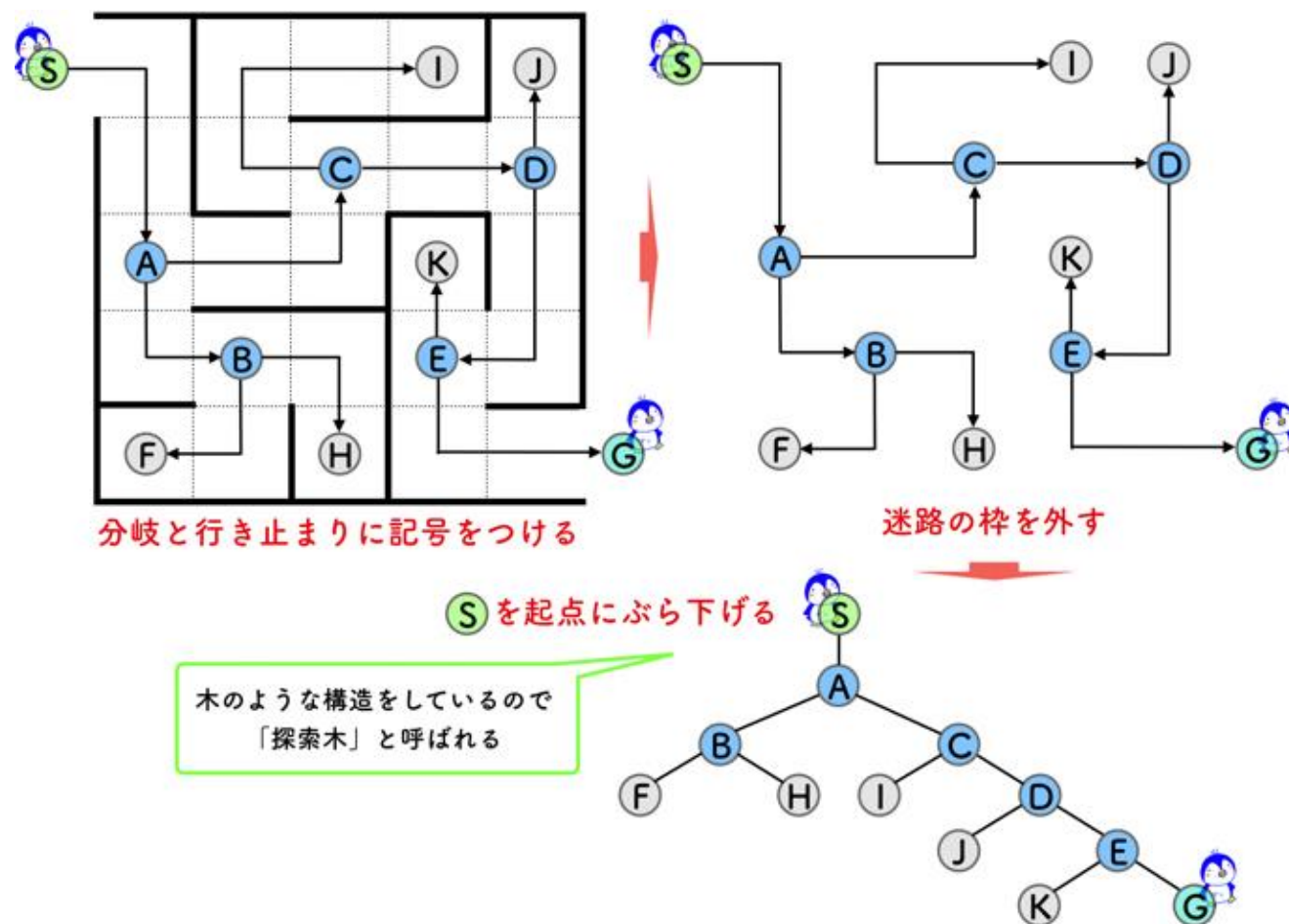
# 問題分割法の考え方

- 分割した小さな問題を考え、それぞれの解を合わせれば問題が解ける
- ある節点がOR節点を持つとき、そのうちのどれかが解決されていれば節点は解決されている。
- ある節点がAND節点（枝に弧がある）を持つとき、全てのAND節点が解決されていればその節点は解決されている。



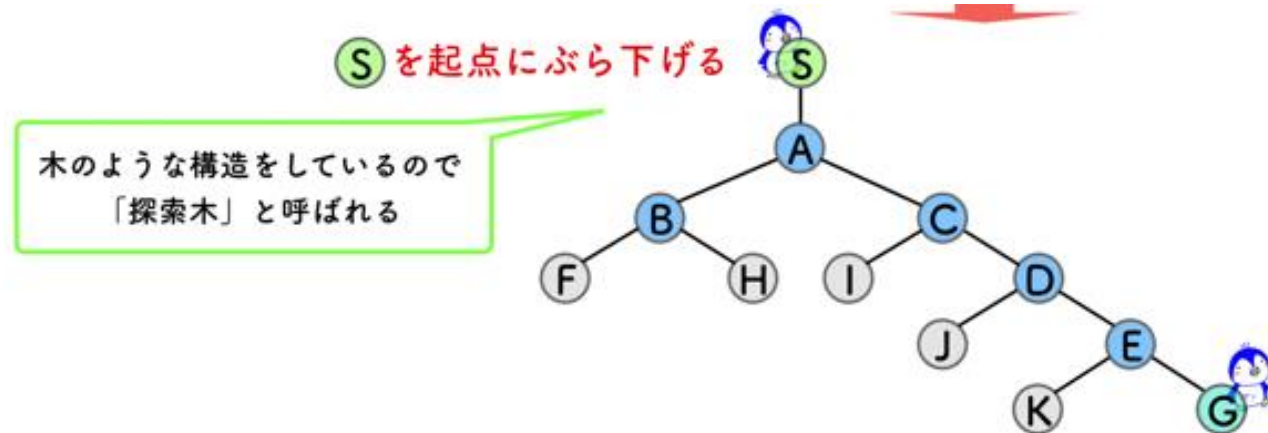
# 探索の手法

- 最初に与えられた状態から，目的の状態に至るまでの状態の変化を，場合分けを行いながら探し出すにはどうすれば効果的か．



- 右の場合，Sを初期状態としてGまで到達したい．

# ブラインド探索



ただ力任せに探索

- **縦型探索**：一つ一つの節を深く見ていく.

S, A, B, F, H, C, I, D, J, E, K, G

- **横型探索**：同じ階層を全て見てから次の階層に行く

S, A, B, C, F, H, I, D, J, E, K, G

# キューとスタック



- **スタック** : **LIFO** (Last-in first-out) 最後に入ったものが先に処理される  
→ お皿を重ねて上から使う
- **キュー** : **FIFO** (First-in first-out) 最初に入ったデータが先に処理される  
→ お店の行列で早く並んだ人から受付する

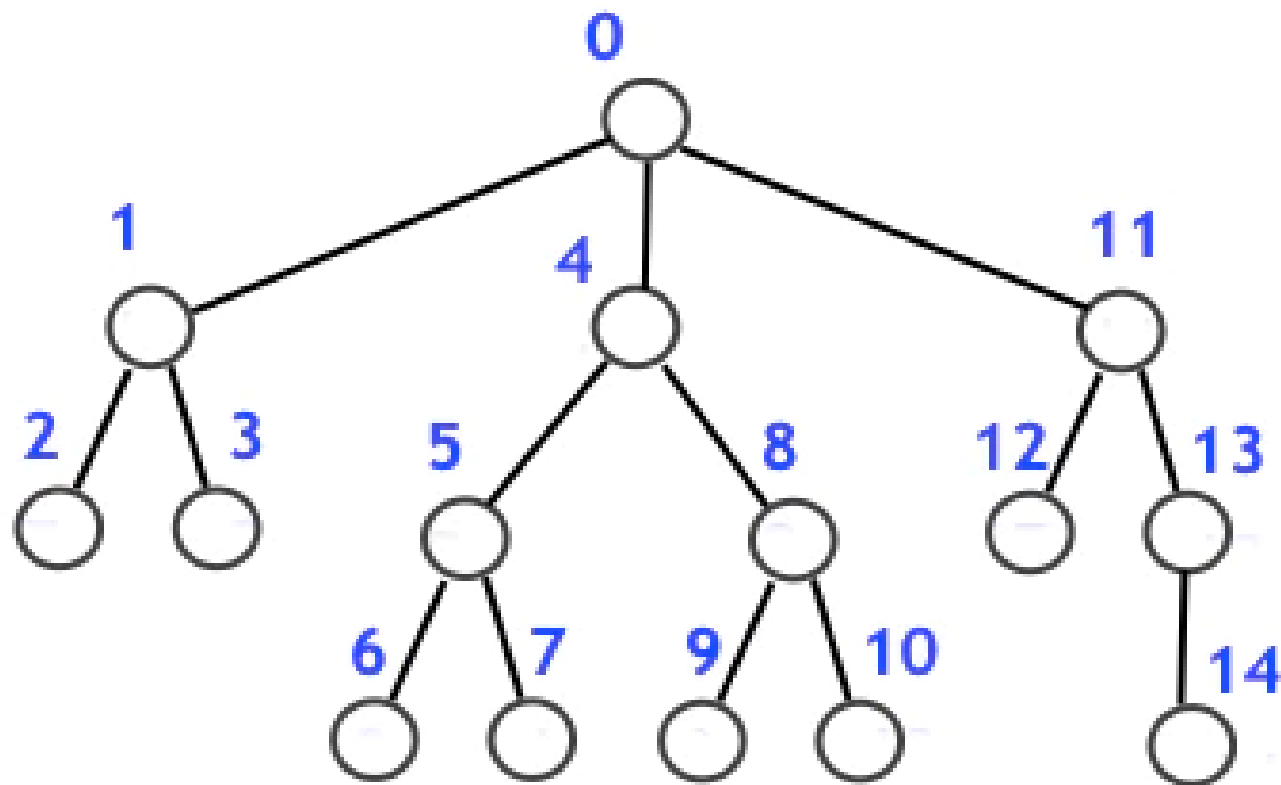




# 縦型（深さ優先）探索アルゴリズム

1. 根ノードを空のスタックに加える.
2. ノードをスタックの先頭から取り出し, 以下の処理を行う.
3. ノードが探索対象であれば, 探索をやめ結果を返す (成功)
4. そうでない場合, ノードの子で未探索のものを全てスタックに追加する.
5. もしスタックが空ならば, グラフ内の全てのノードに対して処理が行われたので, 探索をやめ"not found"と結果を返す (失敗)
6. 2に戻る.

# 縦型探索アルゴリズム

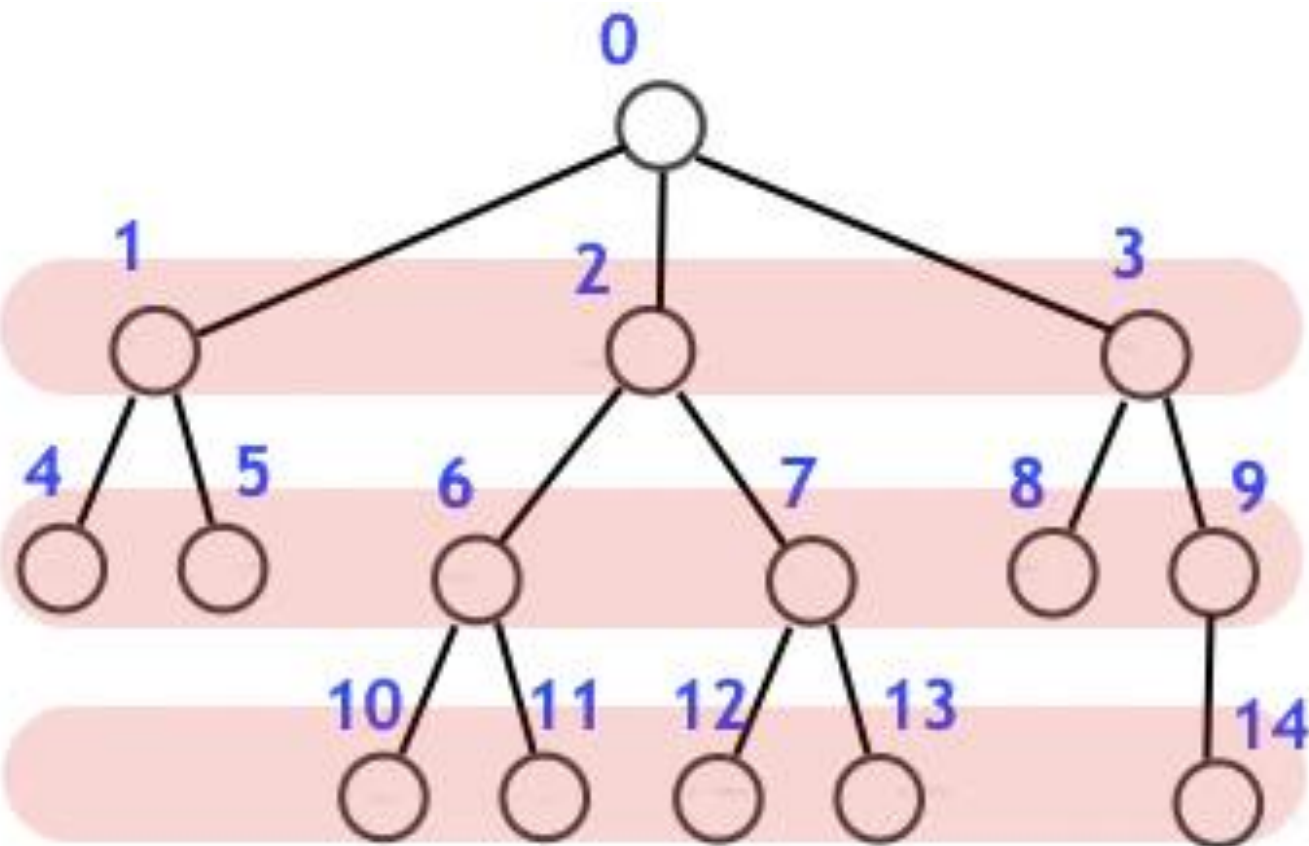


1. 根ノードを空のスタックに加える.
2. ノードをスタックの先頭から取り出し, 以下の処理を行う.
3. ノードが探索対象であれば, 探索をやめ結果を返す (成功)
4. そうでない場合, ノードの子で未探索のものを全てスタックに追加する.
5. もしスタックが空ならば, グラフ内の全てのノードに対して処理が行われたので, 探索をやめ"not found"と結果を返す (失敗)
6. 2に戻る.

# 横型（幅優先）探索アルゴリズム

1. 根ノードを空のキューに加える.
2. ノードをキューの先頭から取り出し、以下の処理を行う.
3. ノードが探索対象であれば、探索をやめ結果を返す.
4. そうでない場合、ノードの子で未探索のものを全てキューに追加する.
5. もしキューが空ならば、グラフ内の全てのノードに対して処理が行われたので、探索をやめ“not found”と結果を返す（失敗）
6. 2に戻る.

# 横型（幅優先）探索アルゴリズム



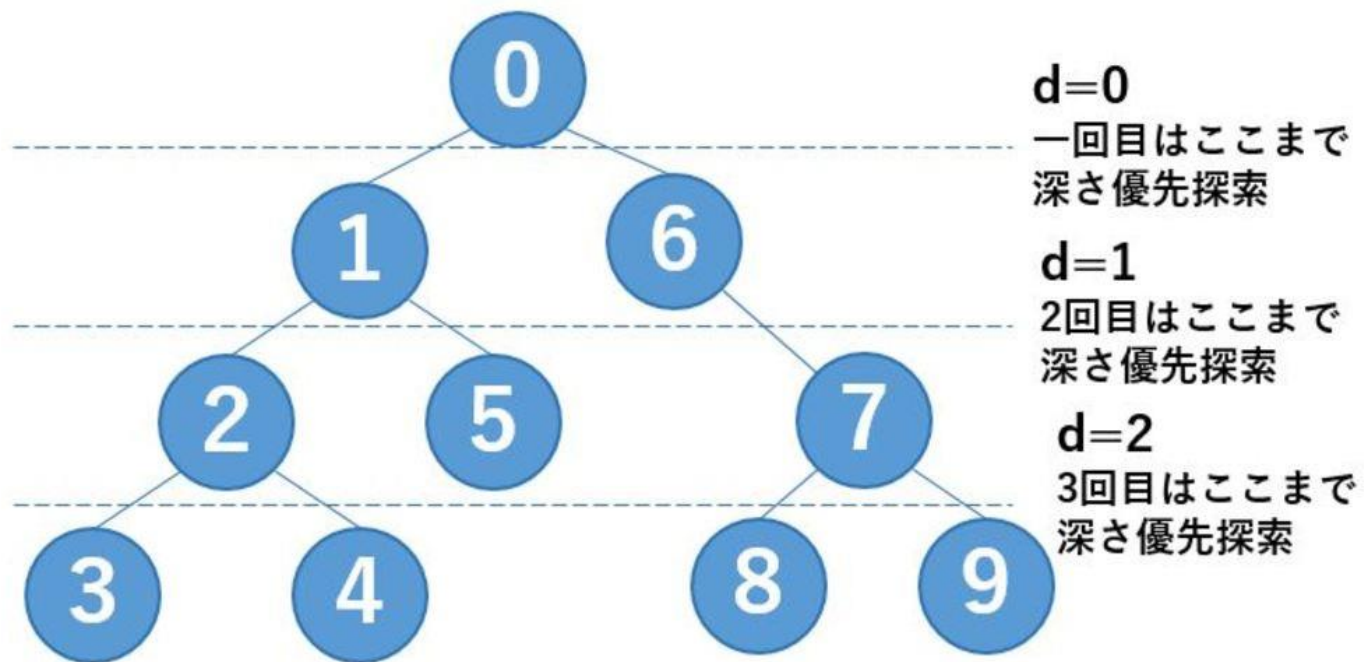
1. 根ノードを空のキューに加える.
2. ノードをキューの先頭から取り出し、以下の処理を行う.
3. ノードが探索対象であれば、探索をやめ結果を返す（成功）
4. そうでない場合、ノードの子で未探索のものを全てキューに追加する.
5. もしキューが空ならば、グラフ内の全てのノードに対して処理が行われたので、探索をやめ“not found”と結果を返す（失敗）
6. 2に戻る.

# 反復深化探索

- 先程の探索法よりも, もう少しスマートに探したい
- 縦型と横型のいいところ取りをしてはどうか.
- ある階層から決められた深さまで見て, 解がなければ1段階深くしてもう一度見ていく



# 反復深化探索の仕組み



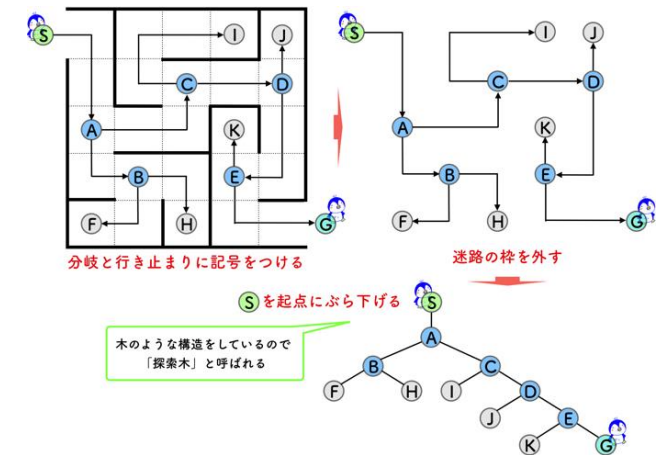
深さを制限しつつ縦型探索を行う。

解がなければ、深さを一つ増やして再度探索する。











左の図はd=3まで行ったあとの順番

# 4つの評価基準

- 完全性：解が存在するときに必ず見つけるか？
- 最適性：最小経路コストの解を最初に見つけられるか？
- 時間計算量：解を見つけるための計算時間
- 空間計算量：必要なメモリ量



# それぞれの特徴

	完全性	最適性	時間計算量	領域計算量
縦型探索	 深すぎると見つ けられないかも	 より深いゴール を見つける恐れ	 最大の深さに関して 指数関数的. $O(b^m)$	 計算量は $O(bm)$
横型探索	 解があれば必ず 見つける	 最も浅いゴール を見つける	 深さに関して指数関 数的. $O(b^d)$	 深さに関して指数 関数的. $O(b^d)$
反復深化法	 解があれば必ず 見つける	 最も浅いゴール を見つける	 深さに関して指数関 数的. $O(b^d)$	 計算量は $O(bd)$

b: 分枝度（子の平均数）, d: 解の深さ, m: 探索木の最大の深さ

5 分休憩



# ヒューリスティック探索

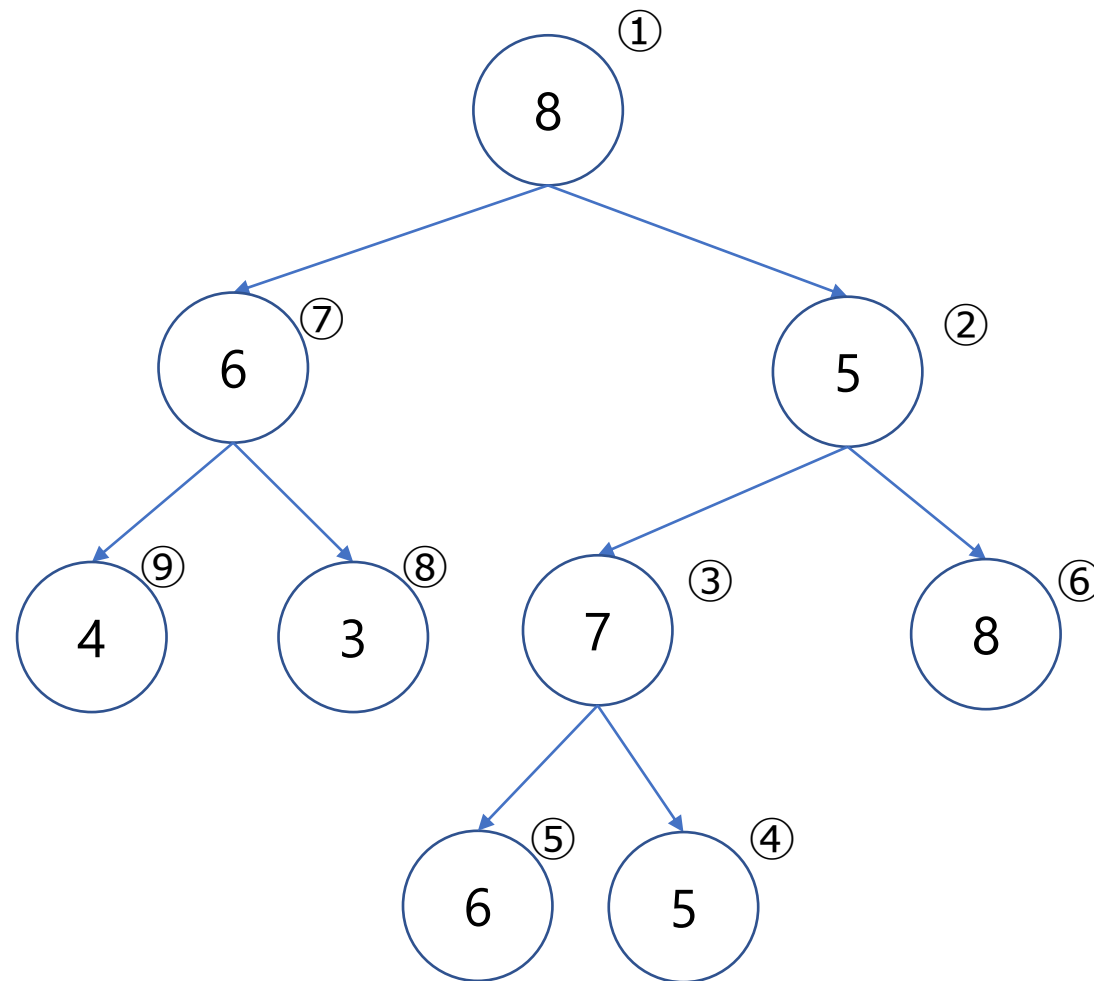
---

- 今までの探索では、外部からの知識がない。
- なにかの知識で見当をつけることが出来れば、もっと効率的に探索を行える
- 先入観や経験則的な情報（ヒューリスティック）を基に、それを中心にして探索する手法を**ヒューリスティック探索**という
- 解に近づいているかどうかを判定する**ヒューリスティック関数 $h(x)$** が別途必要



# 勾配降下法

- コスト $h(x)$ が小さい子節点を優先して調べていく.
- 子を調べている途中で別の枝に移ったりはしない

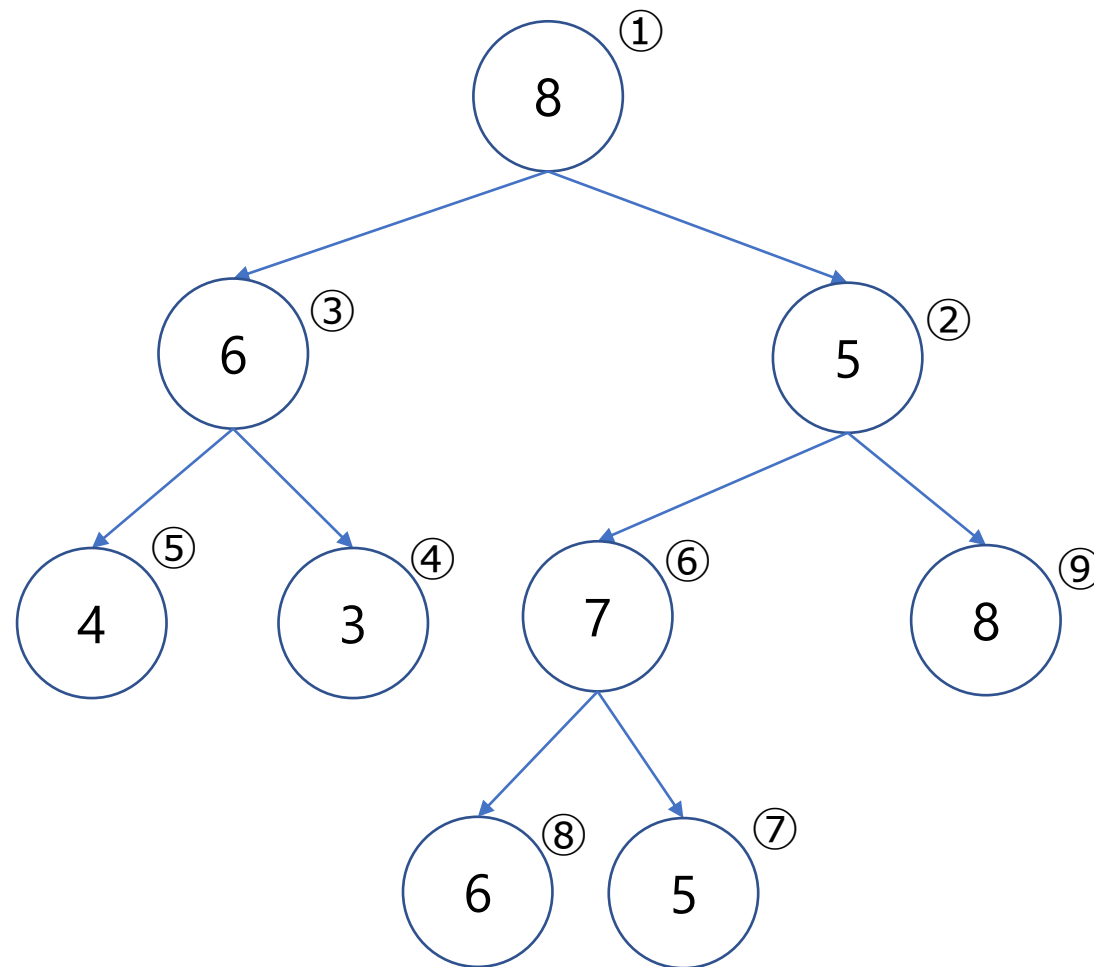


ノード内数字は $h(x)$ のコスト評価, ノード右上は探索の順番

# 最良優先探索

- 勾配降下法で，大局的に見て探索する方法
- 全体から一番良い節の子を見ていく
- ただし，解にたどり着くのが最短とは限らない

数字は $h(x)$ でのコスト評価



ノード内数字は $h(x)$ のコスト評価，ノード右上は探索の順番

# ヒューリスティック関数の作り方

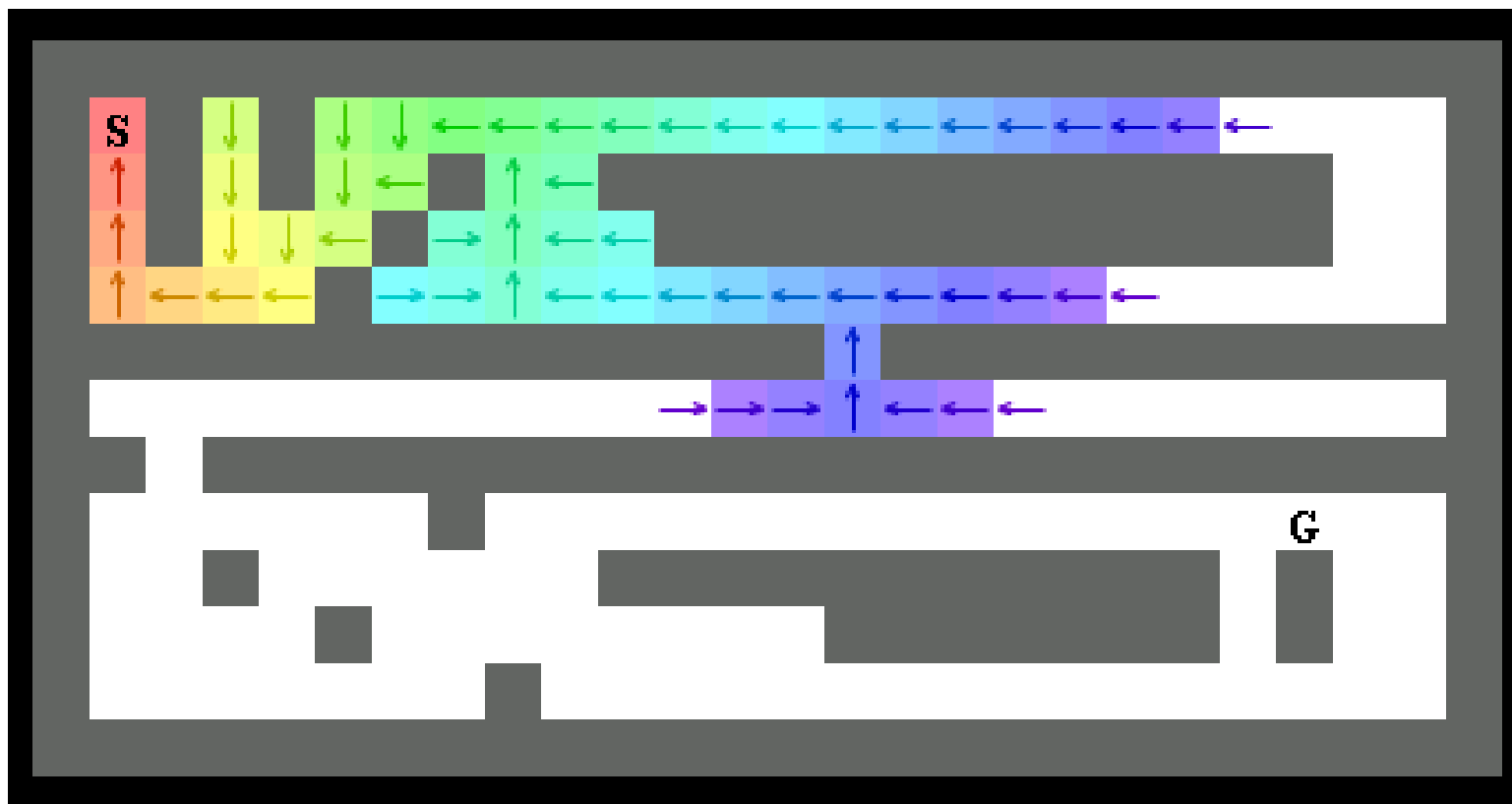
1	2	3
4	5	6
7	8	

- 例えば,  $3 \times 3$  のマスで 1 ~ 8 の駒を順に並べるようなパズルであれば, 一つ動かした場合にどれだけ正解の図面と合致しているかを評価する
- 静的評価 : 次の動き以上の先読みを行わない評価.
- 動的評価 : 数手先を先読みした上での評価.



# ダイクストラ法

- 水を垂らすと全方位に均等に広がるように、一つずつ広げていきはじめにたどり着いた経路を最短とする

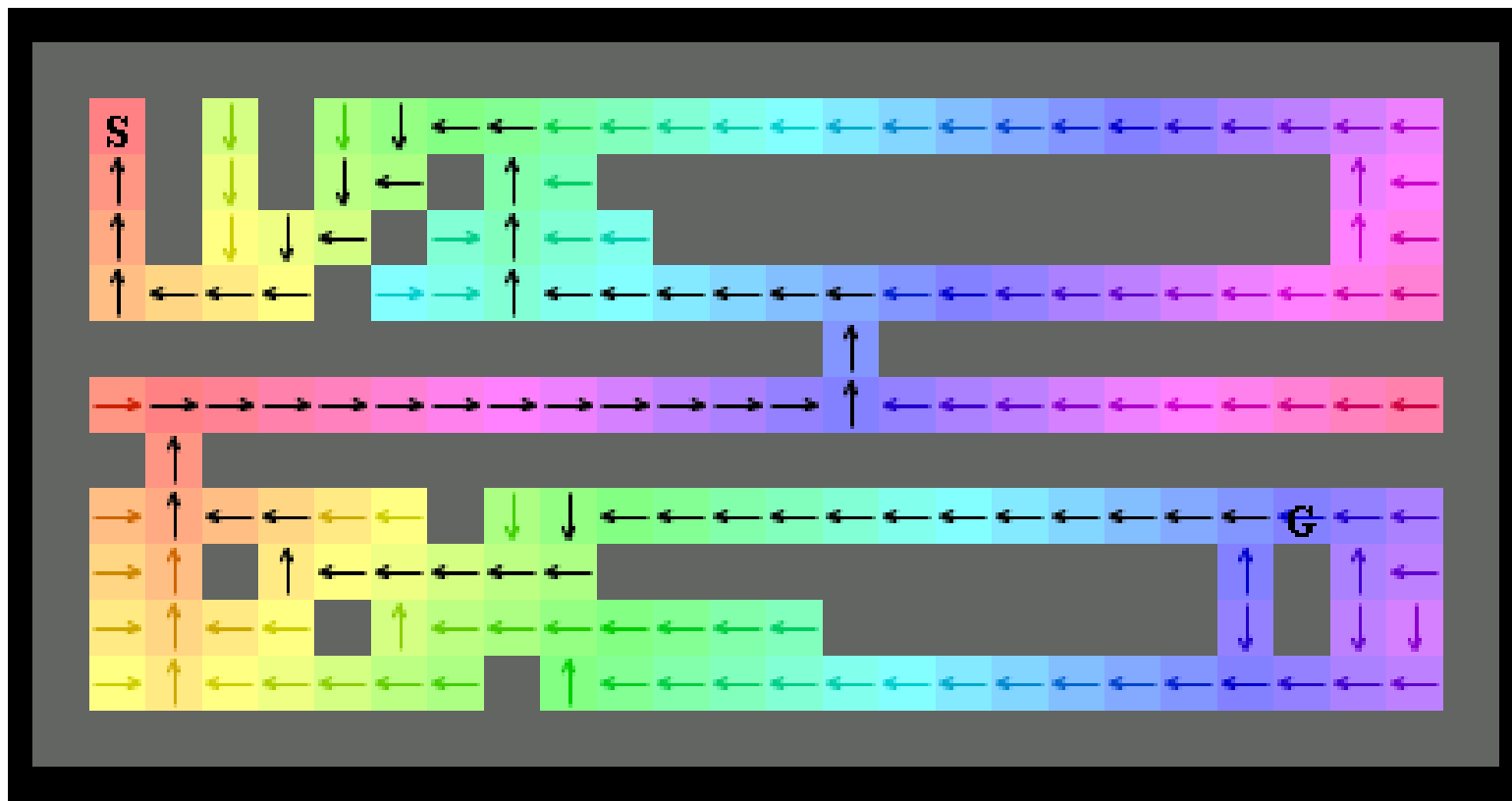






# ダイクストラ法

- 水を垂らすと全方位に均等に広がるように、一つずつ広げていきはじめにたどり着いた経路を最短とする



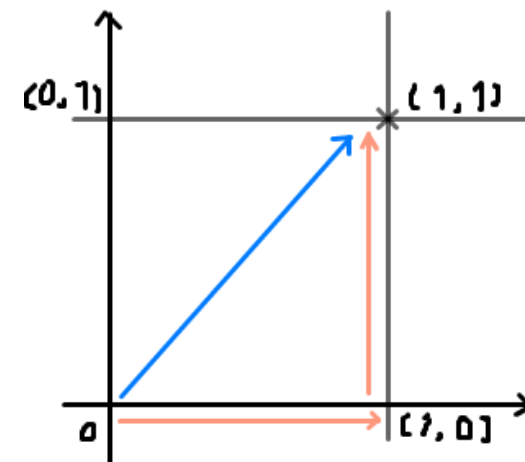
# A\*アルゴリズム

- ダイクストラ法にてゴールまでの距離を考慮した手法
- 各ノードのスコア $f^*$ を調べ,  $f^*$ が小さいノードから探索

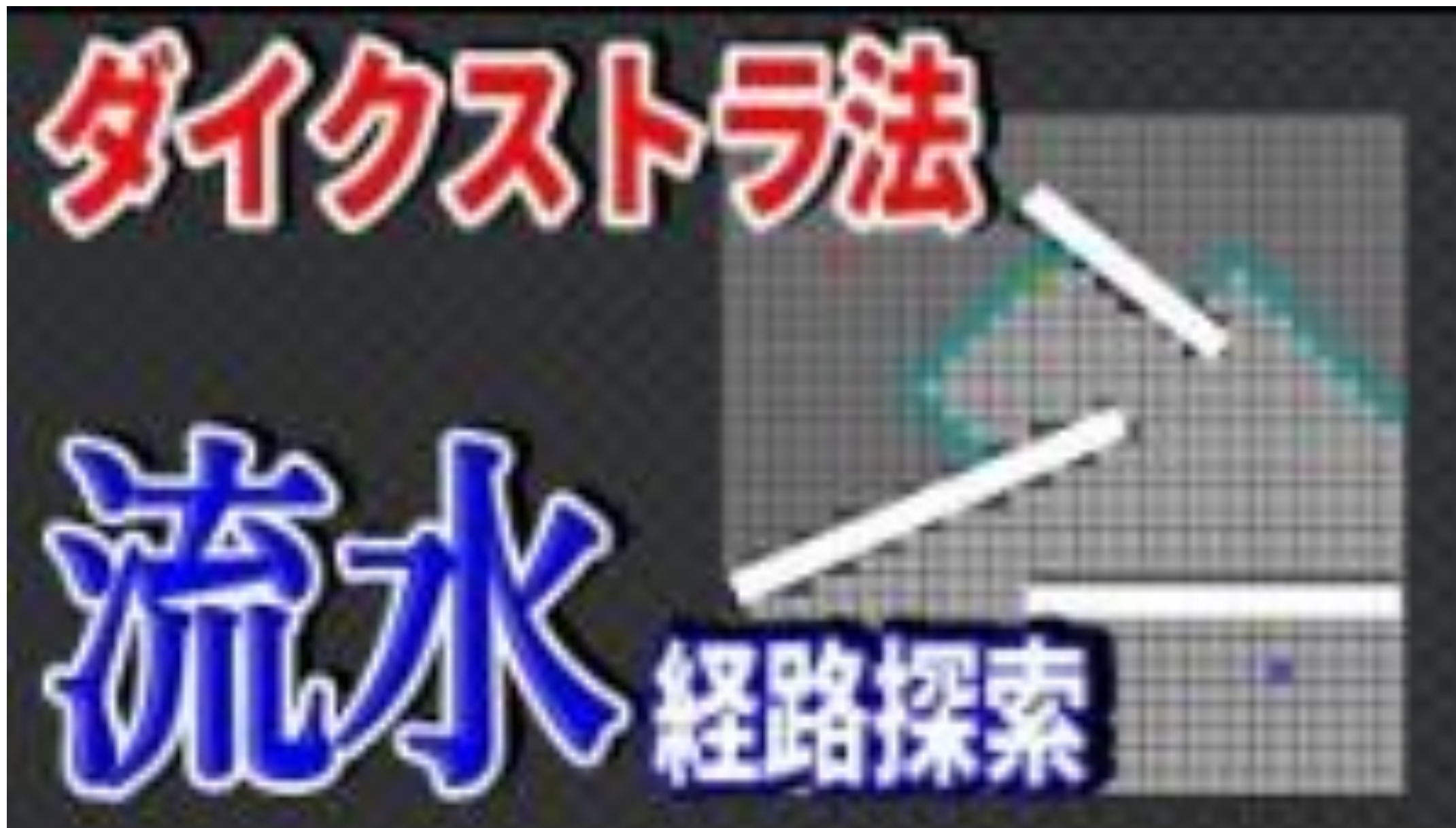
$$f^* = g^*(x) + h^*(x)$$

ここで $g^*(x)$ はスタートからの距離であり,  $h^*(x)$ はゴールへの近さを示す.  $h^*(x)$ では, 推定値として直線距離やマンハッタン距離を用いる

→ 直線距離  
→ マンハッタン距離

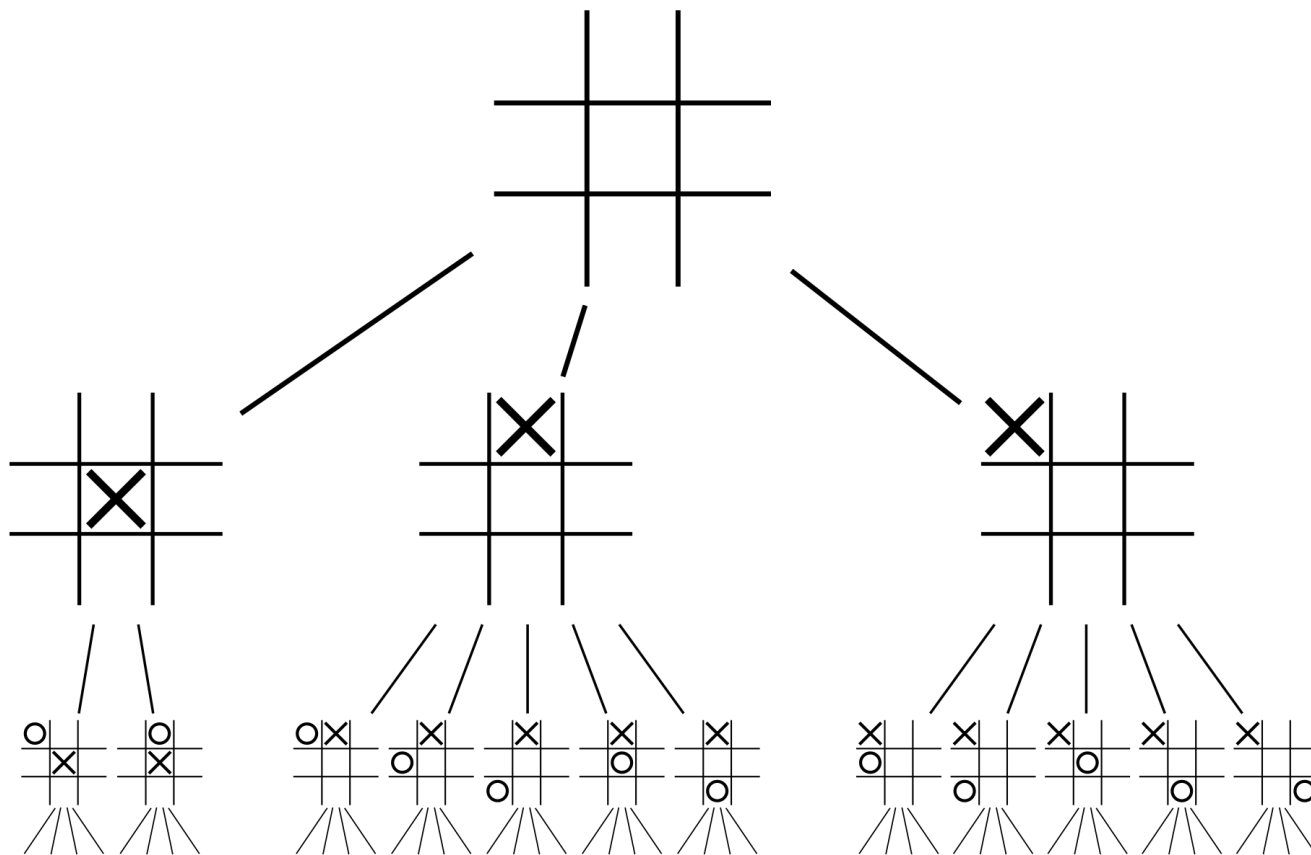


# ダイクストラ法とA\*アルゴリズム



# ゲーム木探索

- 自分の指し手と相手の指し手を交互に表した木を，ゲーム木と呼ぶ
- **完全ゲーム**：チェスや囲碁，将棋など，交互に操作してどちらかが勝つゲーム。運が絡む麻雀やバックギャモンなどは考えない。
- 膨大な盤面をすべて考えることは不可能に近いため，一部を探索して指し手を決める戦略を導入しなければならない。



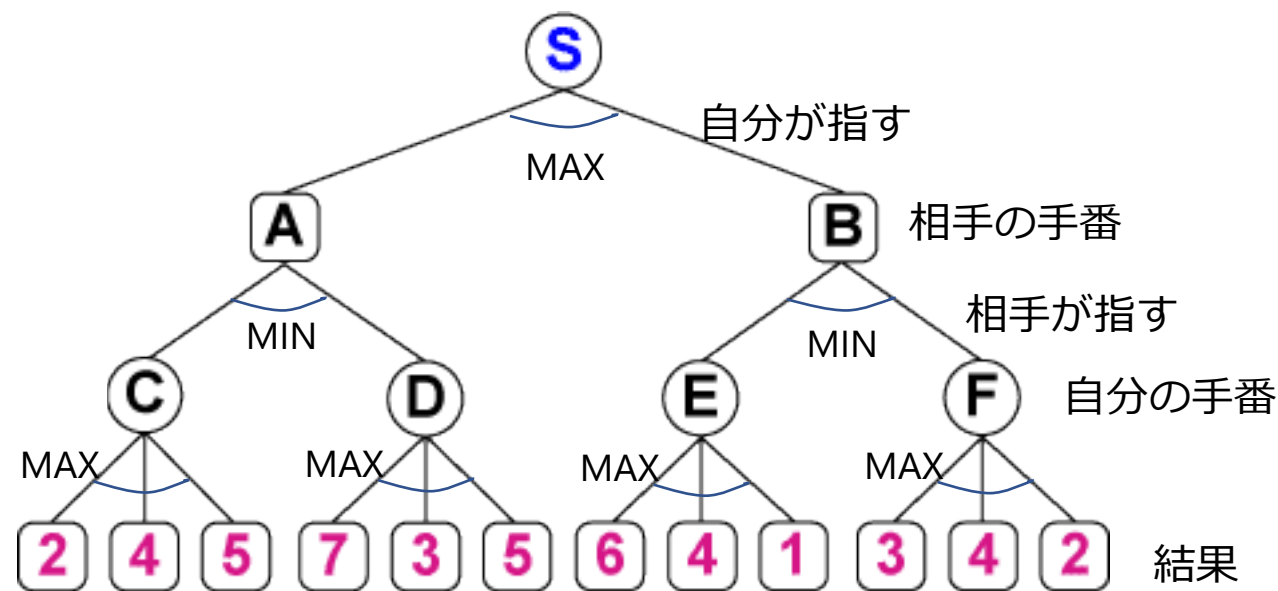
# ミニマックス法

- 想定される損害を最小にする手を選ぶ方法
- 自分は最も有効な手を打ち，相手は自分に最も不利（＝相手が有利）になる手を選ぶ

下から考えて，  
自分の手番では最大にしたいため  
C=5, D=7, E=6, F=4

相手は最も評価値を下げる手を打つため  
A=5, B=4

そのため，自分は最善の左指し手を選択する



数字は自分視点での評価（大きいほど自分有利）を示す



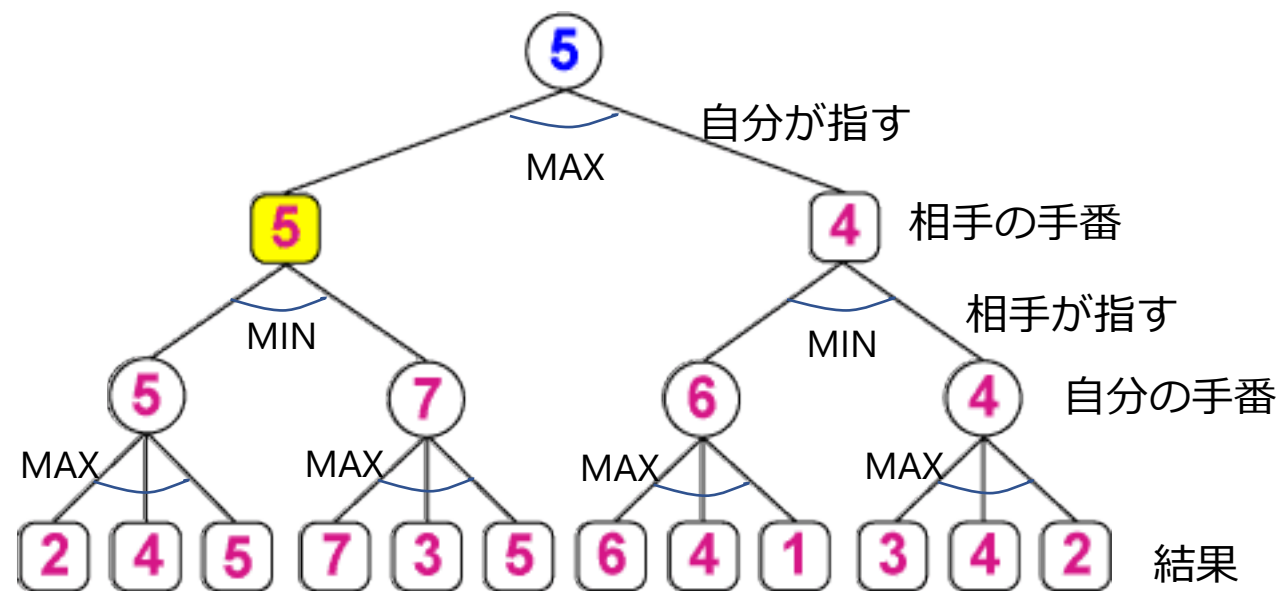
# ミニマックス法

- 想定される損害を最小にする手を選ぶ方法
- 自分は最も有効な手を打ち，相手は自分に最も不利（＝相手が有利）になる手を選ぶ

下から考えて，  
自分の手番では最大にしたいため  
C=5, D=7, E=6, F=4

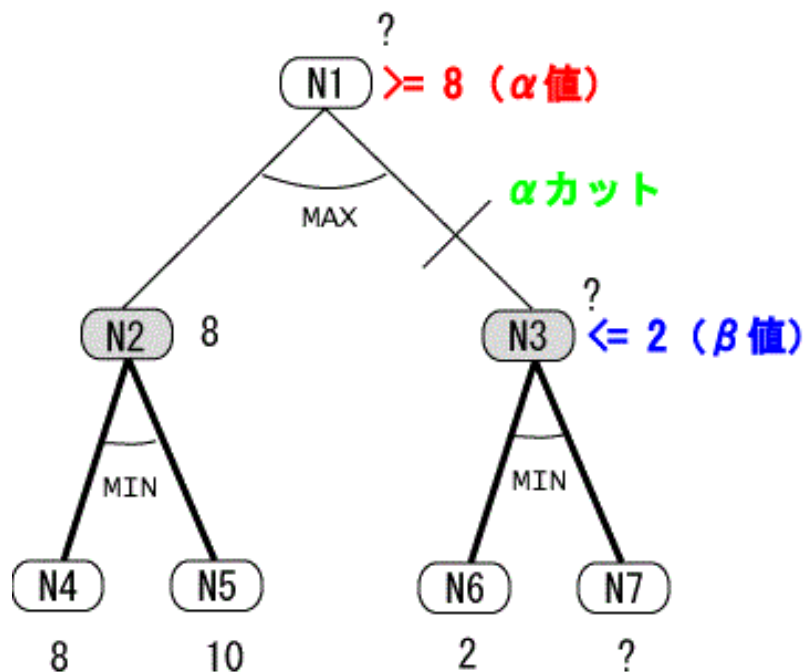
相手は最も評価値を下げる手を打つため  
A=5, B=4

そのため，自分は最善の左指し手を選択する

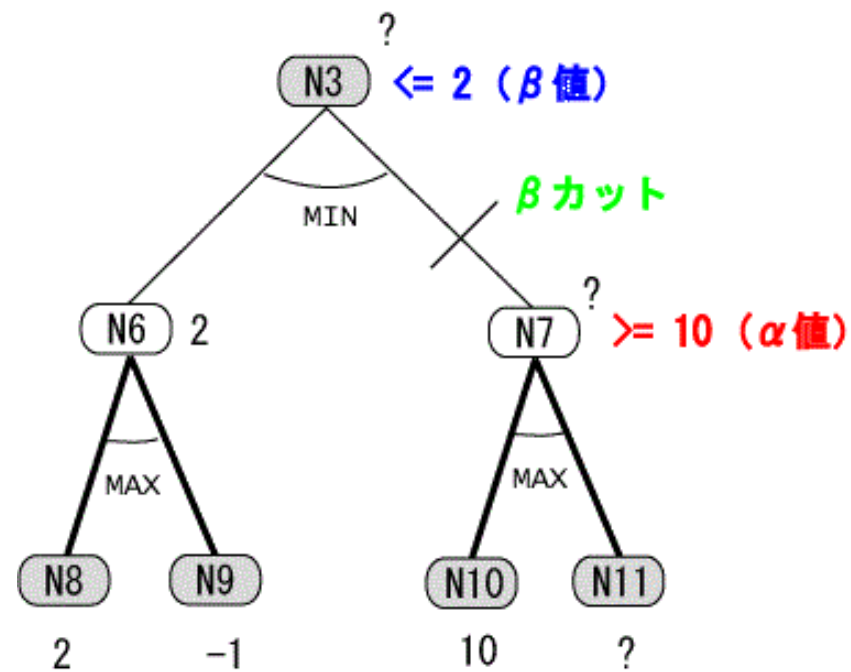


# $\alpha \cdot \beta$ 法

- ミニマックス法は全部を見るため時間がかかる。
- すでに他に良い案があるときは，探索を打ち切る（＝枝刈り）。



N3が2以下とわかった時点で探索を打ち切る



逆に相手のターンでは  
10以上とわかった時点で打ち切る

# ミニマックス法に関する動画（おまけ前まで）

そもそもそれまでの方式では、過去の膨大な戦歴をもとに局面のスコアを人間が決めていたのです。



# まとめ

- 問題の定式化や内部・外部世界について学んだ.
- 問題解決のための探索アルゴリズムやその特徴を学んだ.
- ゲーム木探索の手法について学んだ.

次回は知識の表現方法について説明します.

# レビューシートの提出

- 今日の授業に関するレビューシートを, manabaから提出すること.

後日不明点があれば, 多胡まで.

7号館5階 第9実験室内 第9研究室

tago@net.it-chiba.ac.jp