知能情報処理　探索(3)

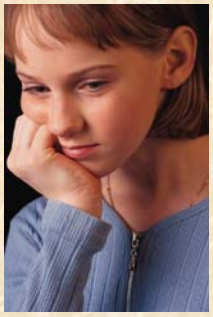
先を読んで知的な行動を選択するエージェント

# 知識に基づく探索
## ― ヒューリスティック探索 ―
(Heuristic Search)

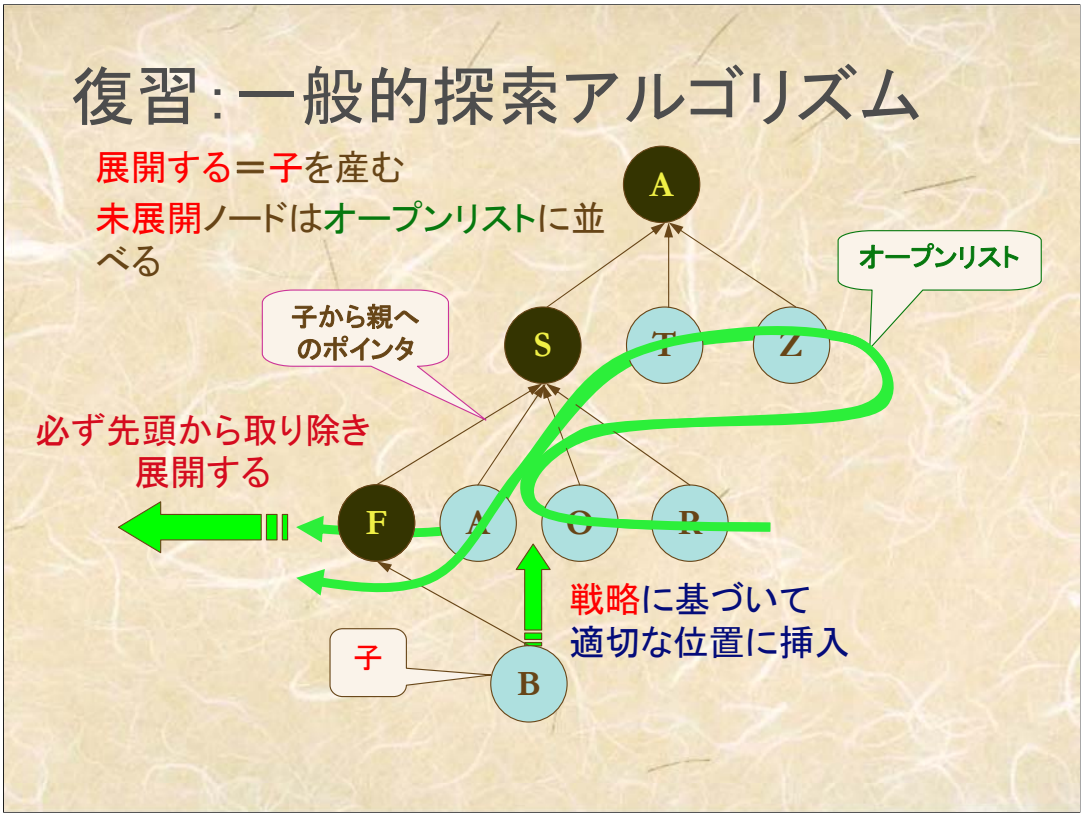
- 最良優先探索
- 均一コスト探索
- 欲張り最良優先探索
- A* 探索

（最良優先探索の具体的な例）

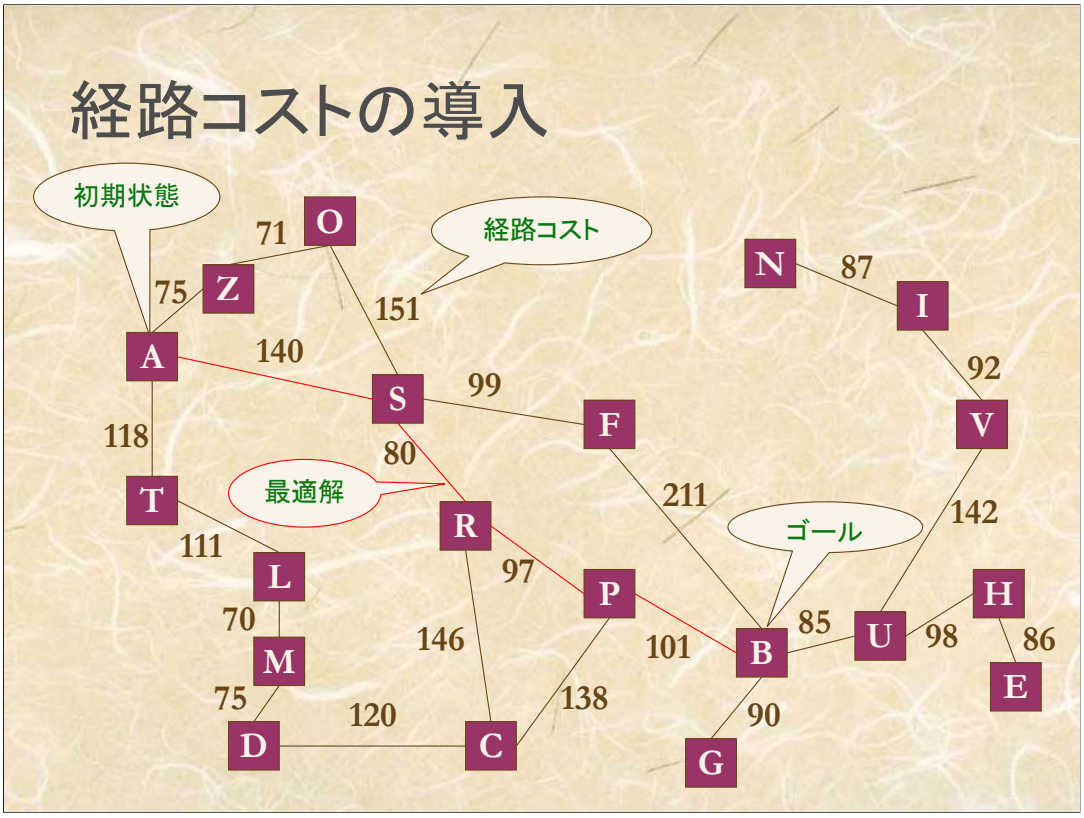
- ヒューリスティック関数について



今回は，**知識を用いる探索**として，**最良優先探索**という一般的な考え方のアルゴリズムを学び，その具体例として，**均一コスト探索**，**欲張り最良優先探索**，**A* 探索**の3つを学ぶ．これらのアルゴリズムにおいては，特定の問題分野に関する「知識」を表現する**ヒューリスティック関数**というものが重要な役割を果たす．


復習：一般的探索アルゴリズム
展開する＝子を産む
未展開ノードはオープンリストに並べる
A
オープンリスト
子から親へのポインタ
S
T
Z
必ず先頭から取り除き展開する
F
A
C
R
戦略に基づいて適切な位置に挿入
子
B

これは前回までの復習.




前回は，暗黙に各**オペレータのコスト**は1としていて，あまり気にかけなかったが，今回ではこの図のようにいろいろなコストがある場合を考えよう．初期状態からゴールまでの経路コストが最小の解が**最適解**である．

なお，この図は，S.ラッセル著：エージェントアプローチ人工知能，共立出版(1997)の図3.3を改変したものである．原著ではこれは単純化されたルーマニアの地図を表しており，AからZまではAradやZerindなど実在の都市名の頭文字となっている．これほど多数の都市の頭文字が重複していない点がすばらしい．


今回の基本的な考え方は，**最良優先探索**(best-first search)というものである．

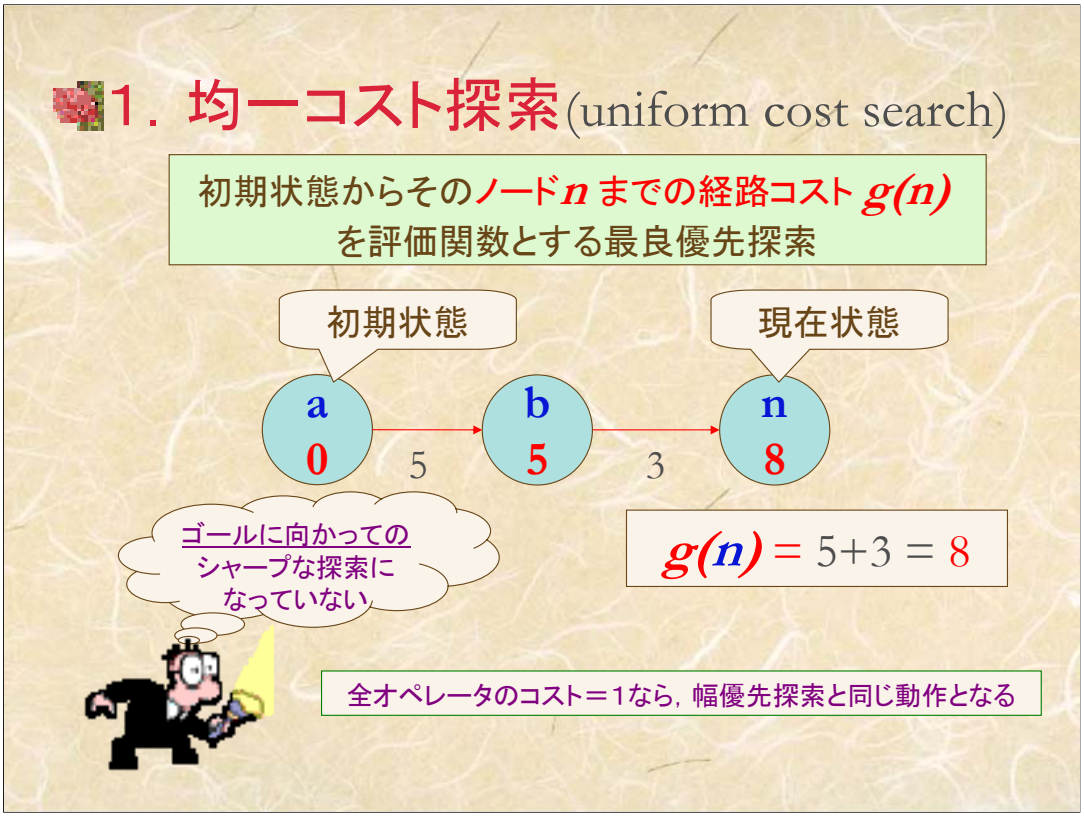
任意のノード *n* を数値 *f(n)* に対応させる何らかの関数を**評価関数**として定義しておく．コストは正の数とする．0や負のコストは考えない．また，ノード数が無限個あるときに，n番目のノードのコストが1/n などと，限りなく0に近い正のコストは考えないことにする．したがって，簡単のため，すべてのコストは正の整数であると考えておけば間違いない．

このとき，最良優先探索アルゴリズムは，未展開の各ノード*n* の評価値 *f(n)* を計算し，その小さい順にオープンリストに並べる．そして，これまでのアルゴリズムと同様に，その先頭ノードを優先的に選んで展開する(子を産む)．つまり，評価値が小さいという意味で現時点でベストに見えるノードを優先的に展開するのである．

評価関数の決め方によって，少なくとも，つぎの3つのバリエーションがある．

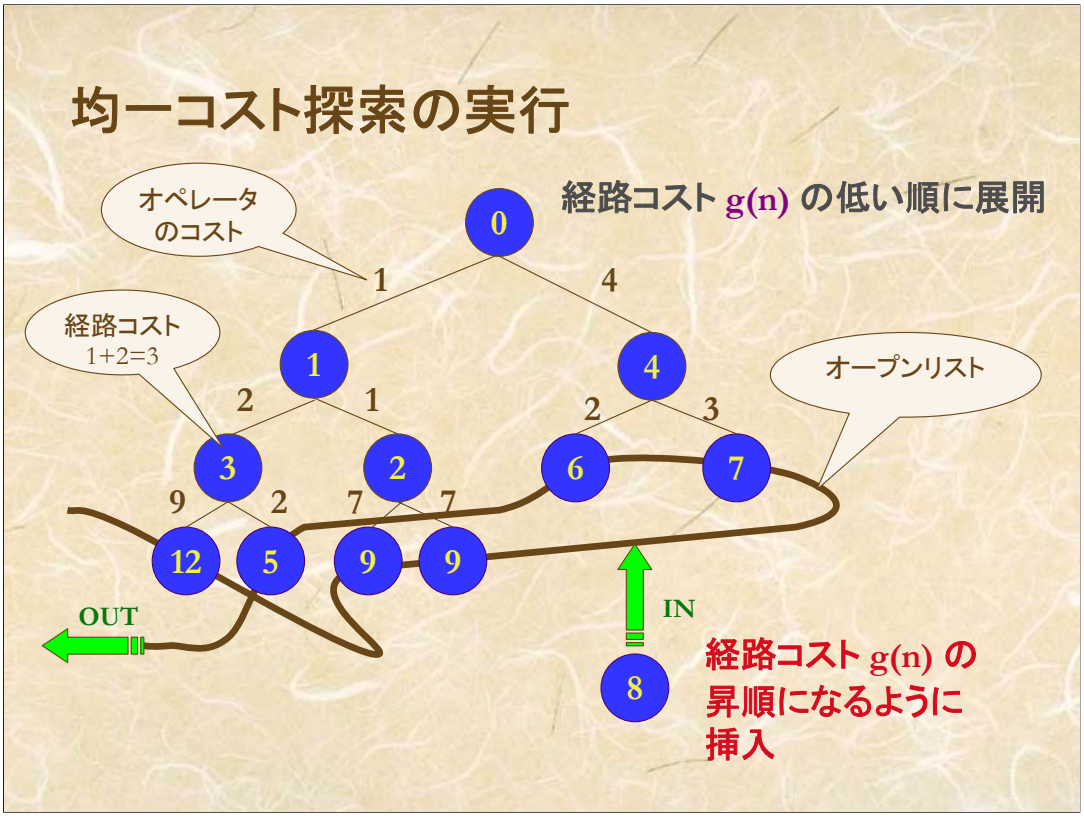
1. 均一コスト探索
2. 欲張り最良優先探索
3. A*探索

次のスライドから，これらを順に見ていく．



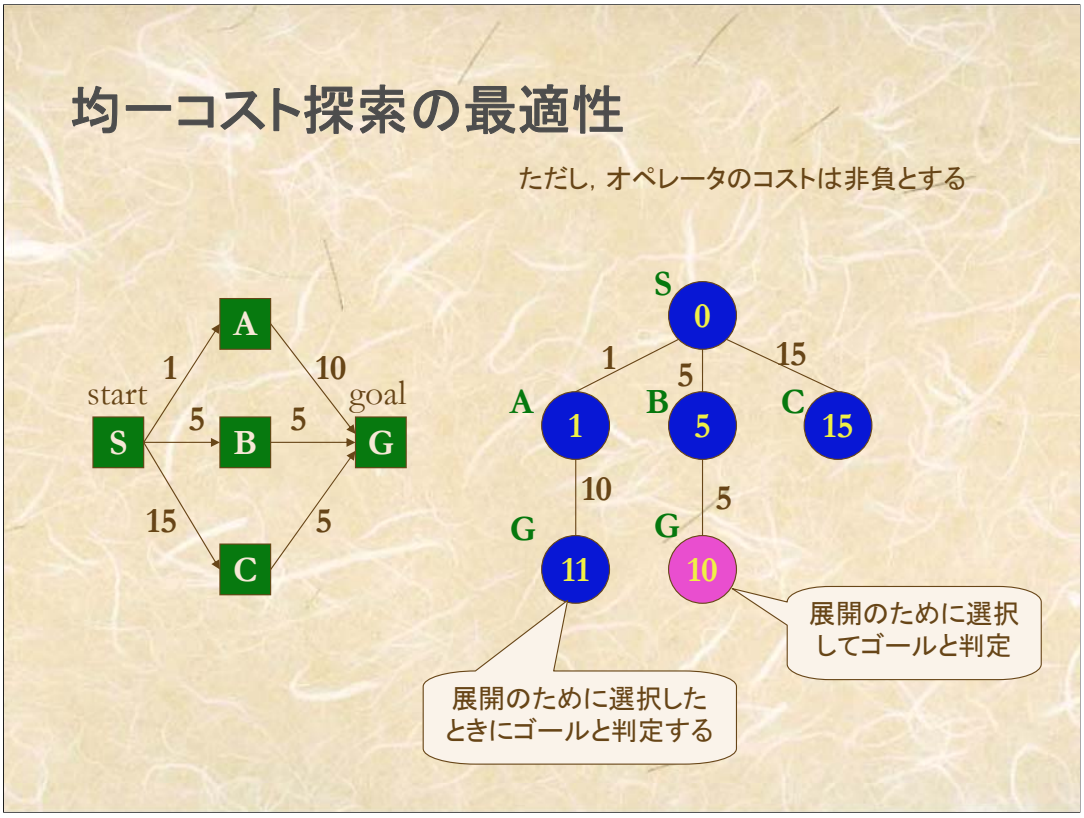

1つめのアルゴリズム「**均一コスト探索**」(uniform cost search)は，初期状態からそのノード***n*** までの経路コスト ***g(n)*** を評価関数とする最良優先探索である．特別な場合として，全オペレータのコスト＝1なら，経路コスト＝ノードの深さとなるので，それを浅い順に展開する幅優先探索と同じ動作となる．

このアルゴリズムは，ゴールがどのあたりにあるのかということをまったく気にせず，とにかく，これまで進んできた経路のコストが最も小さいものを展開するので，「広く浅く」という感じの探索になりがちで，ゴール目指してシャープに突き進んでいくような動作は望めない．




このスライドは均一コスト探索の実行のようすをアニメーションで表現している．ノードを表す円の中の数値はg(n)の値である．未展開のノードのうち，その数値が最も小さいものが選ばれ，展開される．

その途中で，オープンリストのようすも確認しておこう．




均一コスト探索は最適性がある．つまり，最初に見つける解は，必ず最適解である．

そのデモとして簡単な例を見てみよう．

この例では，最適解は S→B→G で，そのコストは10である．

アルゴリズムでは，途中でコスト11の解 S→A→G が生成されるが，このノードはまだ展開されないので，アルゴリズムはまだこれがゴールだとは認識しない．

そのうち，S→B→G が生成され，これがさらに展開される段になって，これがゴールであることに気付き，それを解とする．したがって，最初に見つけた解が最適解になっている．

一般に，このアルゴリズムに最適性があるのは自明であろう．ゴールの判定は，ノードの展開の直前に行われることがポイントである．経路コストが最小のものから展開するので，最適解のゴールノードより早くに経路コストがもっと大きな他のゴールノードが展開されることはない．


均一コスト探索の理論的性質をまとめるとこのスライドのようになる.


2つめのアルゴリズム「**欲張り最良優先探索**」(greedy best-first search)は，そのノード***n***からゴールまでの予想コスト ***h(n)*** を評価関数とする最良優先探索である．ただし，$h(n) \geqq 0$ とする．$h(n)$ は，ノードn からゴールまでの最短経路コストの見積りであり，**ヒューリスティック関数**と呼ばれている．これは，あくまでもこの問題分野独自に関する知識や経験からの「予想」とか「見積もり」であって，その情報が正確である保証はないことに注意しよう．しかし，全くのでたらめではないことは期待したい．

このアルゴリズムは，$h(n)$ の値の小さい順に未展開ノードを展開する．したがって，オープンリストは$h(n)$ の値の小さい順に並べておく．

先ほどの均一コスト探索とは対照的に，このアルゴリズムはこれまでたどってきた経路のコストがどうだったかという過去は一切問わない．とにかく，今後の予想コスト$h(n)$ のみに着目し，それが現時点で最小の未展開ノードが最も有望である(ベストに見える)として展開する．

古い教科書などでは，この欲張り最良優先探索のことを単に最良優先探索と呼んでいることもあるので注意しよう．

## ヒューリスティック(heuristic)とは？

- 語源： アルキメデスが風呂で浮力の法則を発見したときに叫んだ”Heurika !”（ユーリカ！発見した！）
- 経験から発見した知識のこと

- 最悪ケースの性能は必ずしも上げないが，平均的または典型的にはうまくいく手法

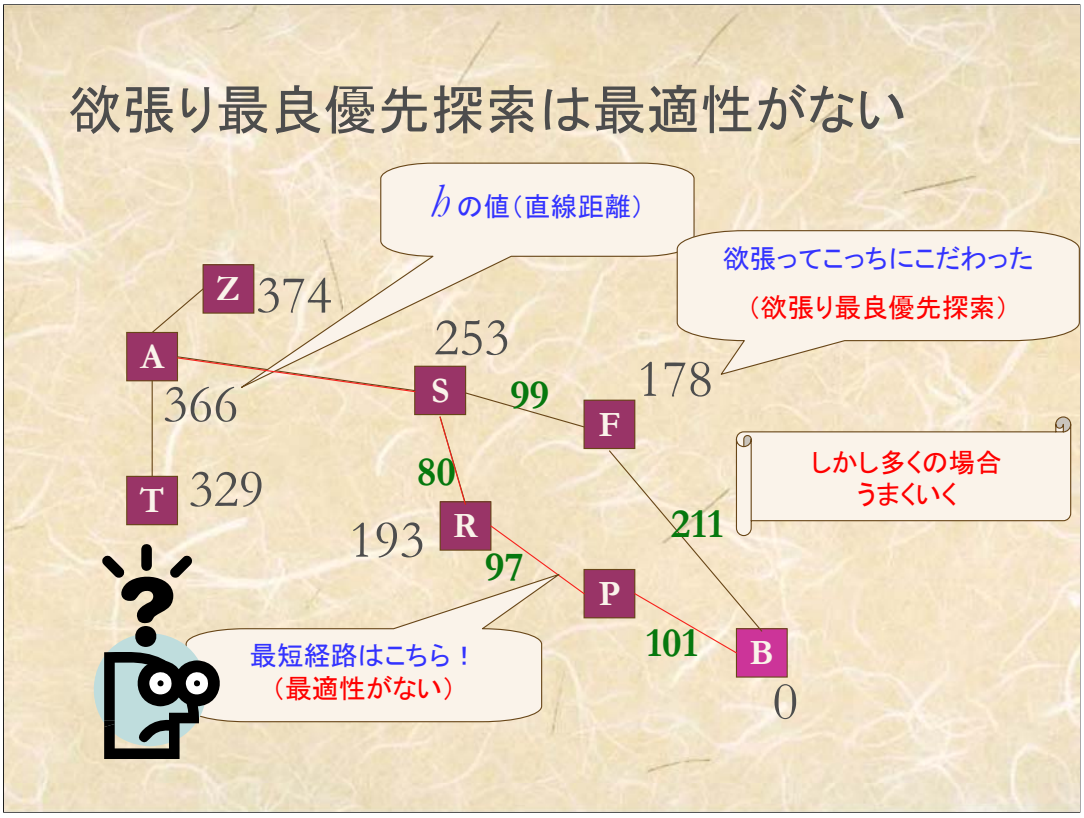
ここで出てきた「**ヒューリスティック**」という言葉は人工知能（AI)技術にとって，非常に重要な用語である．

これは，アルキメデスが風呂で浮力の法則を発見したときに叫んだ”Heurika !”（ユーリカ！発見した！）という言葉が語源で，AIでは，経験から発見した知識のことを指している．

しかし，有り得る可能性のすべてを経験したわけではないので，これは数学的に完全に正しい知識であるという保証はない．しかし，アルゴリズムの性能については，最悪ケースの性能は必ずしも上げないが，平均的あるいは典型的にはうまくいく手法であることが多い．

AIにはこのようなアルゴリズムが多いのである．理論家が（最悪のケースで）指数関数的な計算量になるので実用的でないとする難しい問題でも，AI研究者はあきらめず，ヒューリスティクスを利用して，たいていの場合には高速に解を見つける技術の開発を目指すのである．


都市のナビゲーションの例では，h(n) を「nからゴールまでの直線距離」と定義すればよい．正しい値はもちろん「nからゴールまでの最短の道のり」なのだが，2点の緯度と経度から三平方の定理で直線距離を算出すれば，ゴールまでの道のりの良い見積もりとなるだろう．




欲張り最良優先探索は最適性がないので，最初に見つけた解が最適解とは限らない．この例（アニメーション）がその証拠である．最初に最適解でない方の解を見つけてしまう．


欲張り最良優先探索には完全性もない．すなわち，解があるにも関わらず，解を発見できない場合がある．このスライドの例では，不適切なヒューリスティック関数に引きずられて，際限なく変な方向を探してしまう．


欲張り最良優先探索の性質をまとめるとこのようになる．この結果を見ると全く良いところがないように見えてしまうが，実際には必ずしもそうではなく，「欲張り」であることが功を奏して，効率良く解を発見できることがある．その解は最適解である保証はないが，実用上，十分役に立つ場合がある．


3つめのアルゴリズム「**A*探索**」(A* search：A*は「エイスター」と読む)は，今回の授業の山場で，かつ，探索アルゴリズムの最も重要なものの1つである．このアルゴリズムは，***n*** *を経由する最短経路の見積もりコスト*を ***f (n)***=*g(n)+h(n)* で定義し，それを評価関数とする最良優先探索である．

*g(n)* は「ここまでのコスト」という過去の実績を表し，*h(n)* は「ここからの予想コスト」という将来の見積もりを表している．その和は，初期状態から *n* を経由してゴールまでの最短経路のコストの見積もりになっている．この値の小さい順に未展開ノードを展開するのがこのアルゴリズムである．


すべてのノード n について，

予想最小コスト *h(n)* ≦ 実際の最小コスト *h*(n)*

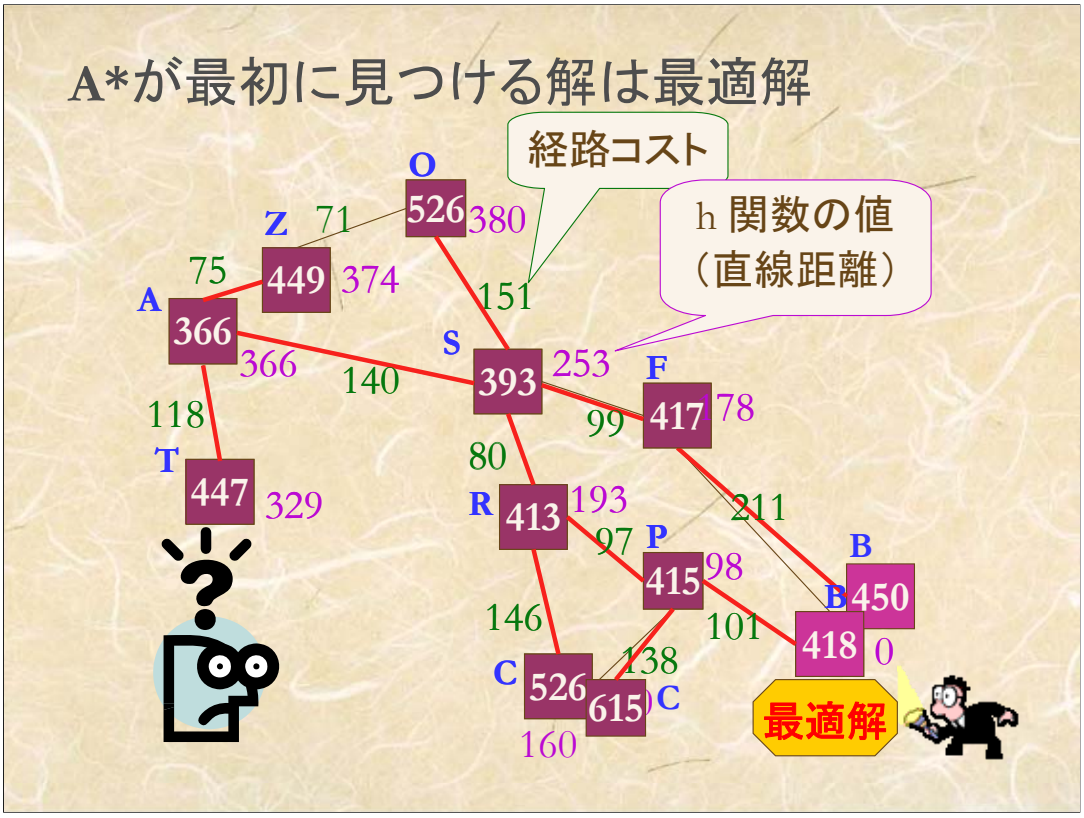
が成り立つとき，そのヒューリスティック関数 *h(n)* は**許容的**(admissible)であるという．許容的なヒューリスティックは，実際よりも小さめにコストを予想するので，**楽観的**なヒューリスティックともいう．

ナビゲーションの例では，*h(n)* を「ゴールまでの直線距離」とすれば，それは許容的なヒューリスティック関数である．


**A*** 探索は完全である. すなわち, 解があれば必ず発見する. その理由は均一コスト探索と同様である. 各オペレータ(辺)のコストとヒューリスティック関数の値が正の数なので, 欲張り最良優先探索が完全でないことを示した例のように, 目標状態へ至る経路を無視して限りなく探索木の深いところへ進んでいくと, その経路の先端にあるノードの評価値は限りなく増大するので, いずれは解に至る経路の先端にあるOPENノードの方が評価値が小さくなり, それが展開のために選ばれて, 解へ一歩接近するからである.

また, **A*** 探索は, ヒューリスティック関数 *h(n)* が許容的であれば, 最適性がある. その直観的な説明は後述する.

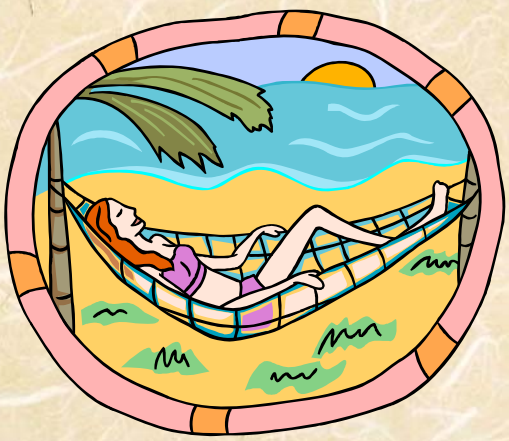
計算量はヒューリスティックの精度に依存することも, 後のスライドでわかる.

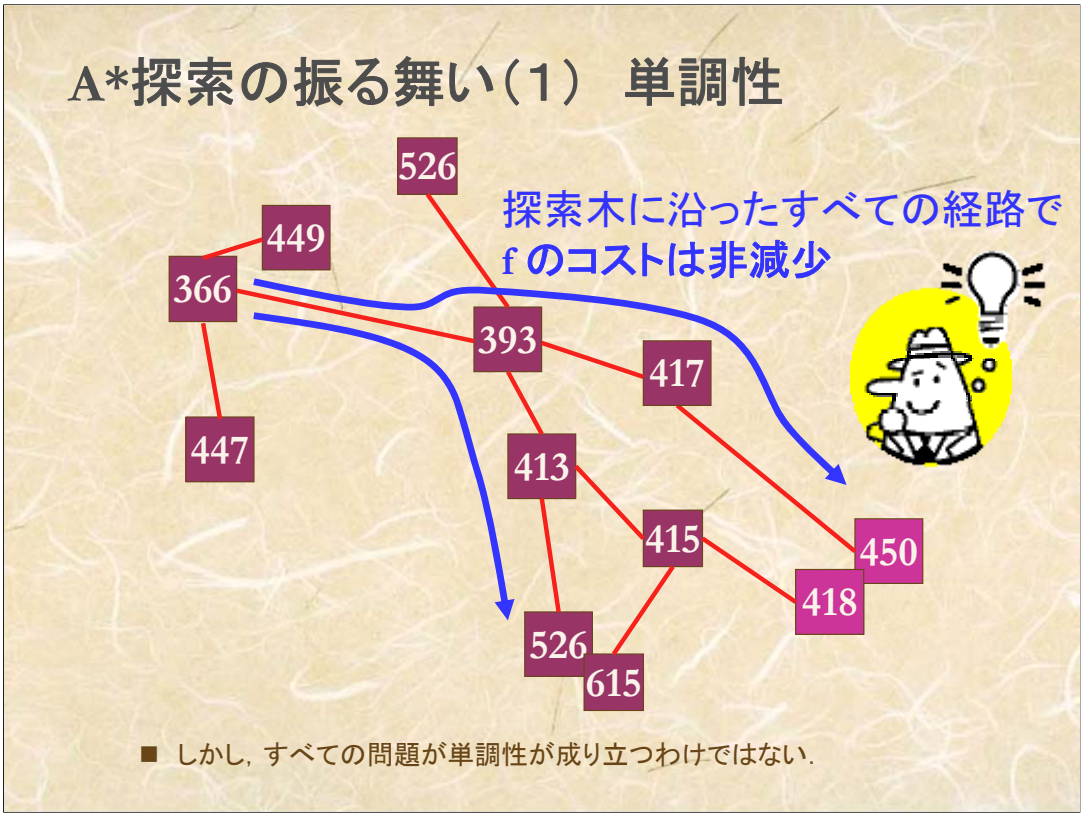



これは，**A***が最初に見つける解が最適解であることのデモである．*h(n)* を「ゴールまでの直線距離」と定義することにより，許容的なものとしてある．

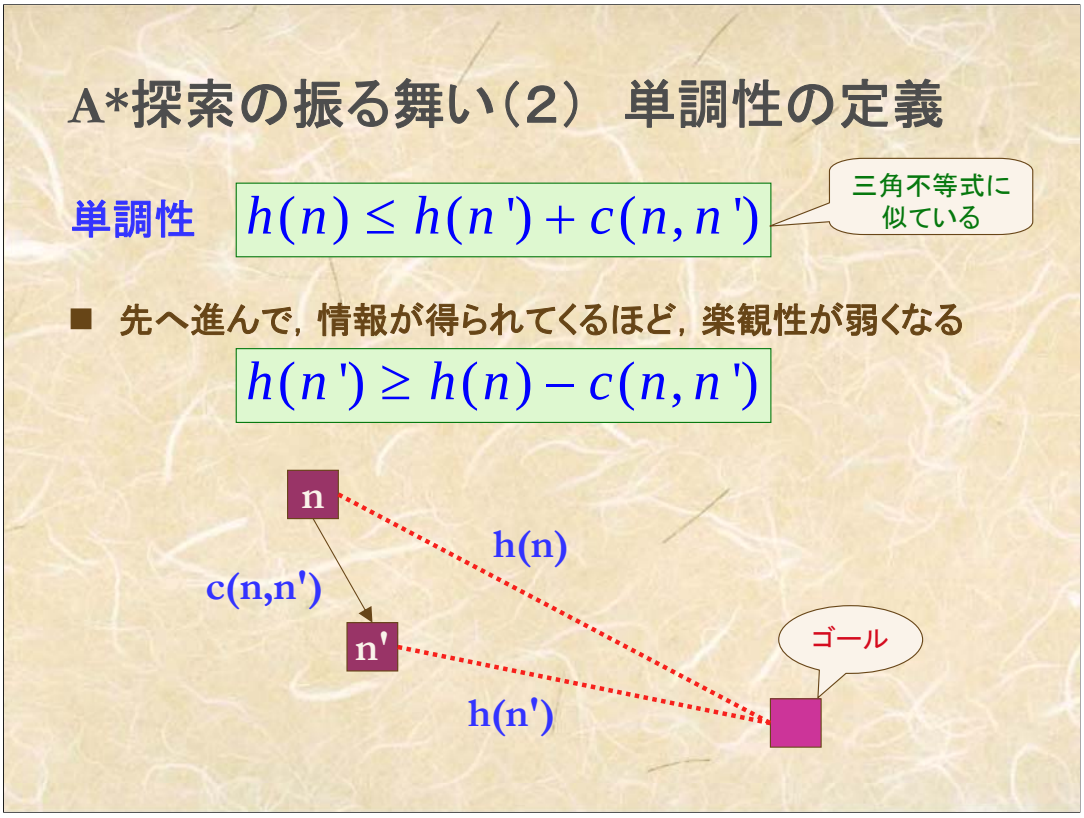
最適解でない方の経路が先に生成されるが，最終的に，先にゴールと判断されるのは最適解の経路の方であることに注意しよう．

# 休憩



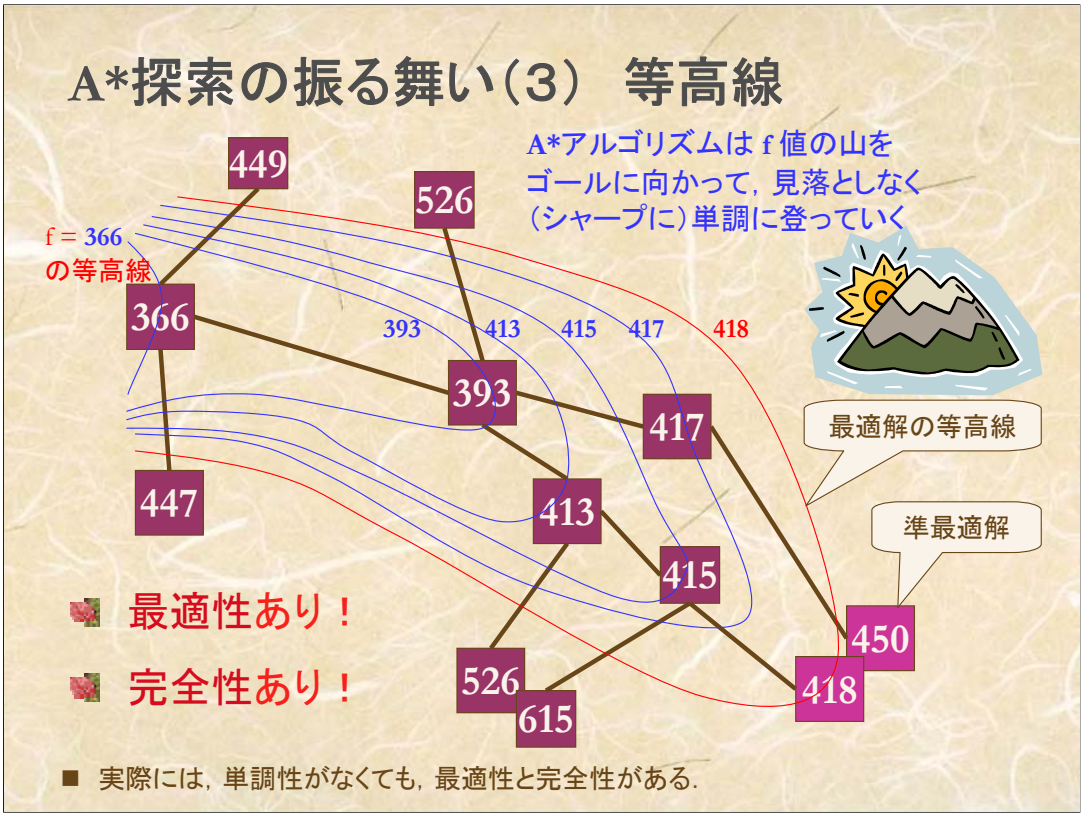



先ほどの例題では，探索木に沿ったすべての経路で f のコストは非減少になっている．これを**単調性**という．すべての問題で単調性が成り立つわけではないが，そうなることが多い．単調性の正式な定義と，その意味合いはつぎのスライドで．




任意の *n*, *n'* についてこのスライドで示した不等式が成立するとき，ヒューリスティック関数 *h(n)* は**単調**であるという．この不等式は，このスライドの図と比較すれば，いわゆる「**三角不等式**」(三角形の1辺の長さは，他の2辺の長さの和を超えない)に似ていることにも留意しよう．

この不等式を2つ目のように変形すればわかるように，これは，先へ進んで，情報が得られてくるほど，楽観性が弱くなるということを表している．つまり，最初は将来についてかなり楽観的だったのが，先へ進んできてみていろいろな情報が見えてくると，現実のきびしさがわかってきて，その楽観性が後退してくるというわけで，我々の人生に通ずるものがある．とほほほほ．




単調性があるときには，A*の振る舞いは *f(n)* の等高線を低い方から高い方へ地道に引いていく手順として理解できる．そうすると，最適性が成り立つ理由を理解できる．すなわち，初期状態ノードから開始するすべての経路において，評価値fは単調に増加しているときに，このアルゴリズムは，fの値の小さいノードの順に確実にOPENノードから選んで展開し，探索木を成長させていく．したがって，最適解より大きなfの値を持つゴールノードを展開する以前に，必ず最適解のゴールノードを展開することになり，そのときにゴール判定の結果それを解とするのである．したがって，最初に見つかる解は必ず最適解になっている．

このように，単調性があれば直観的な説明をしやすいのだが，実際には，単調性がないときにも，A*アルゴリズムはヒューリスティック関数 *h(n)* が許容的であれば最適性がある．

（証明）最適解の経路上にあるOPENノード n の評価値 f(n) は，h(n)が許容的であるため，最適解のコスト C*以内，すなわち，

$$f(n) = g(n) + h(n) \leqq g(n) + h^*(n) = C^*$$

であるのだが，その一方で，最適解ではない経路のゴールノード G の評価値は，最適解でないことの定義より，C*を超える，すなわち

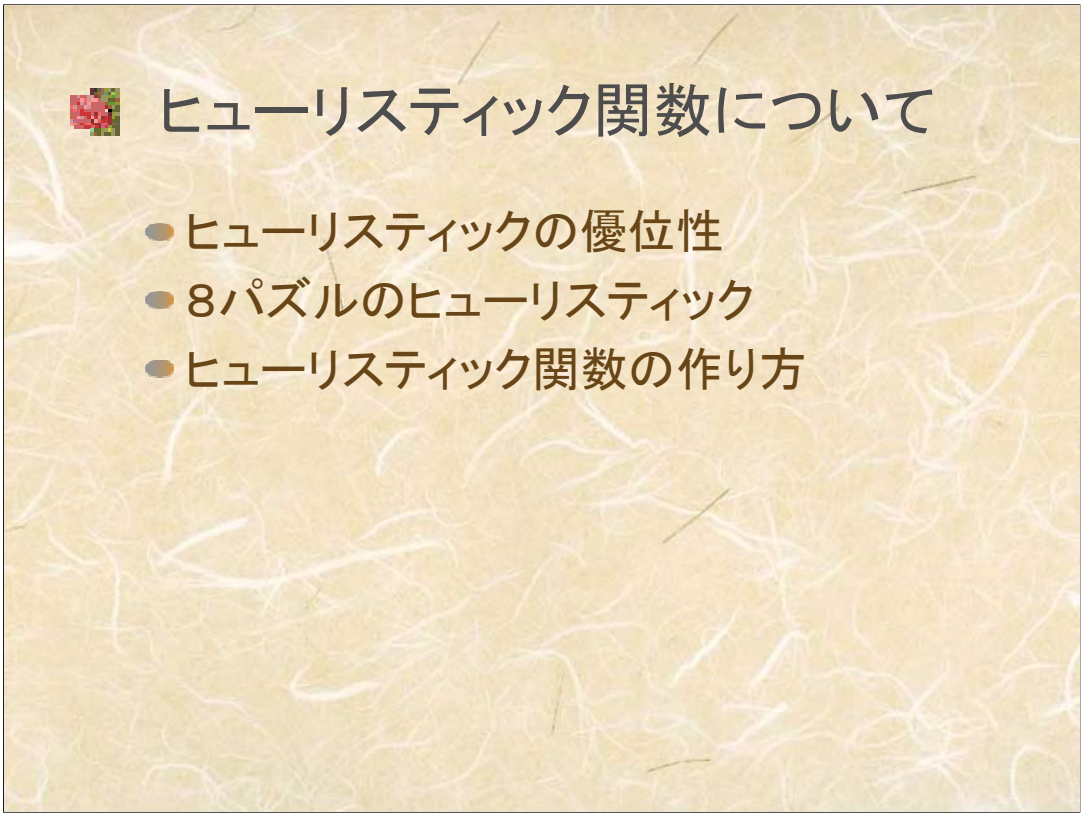
$$f(G) > C^*$$

である．この2つの不等式より，

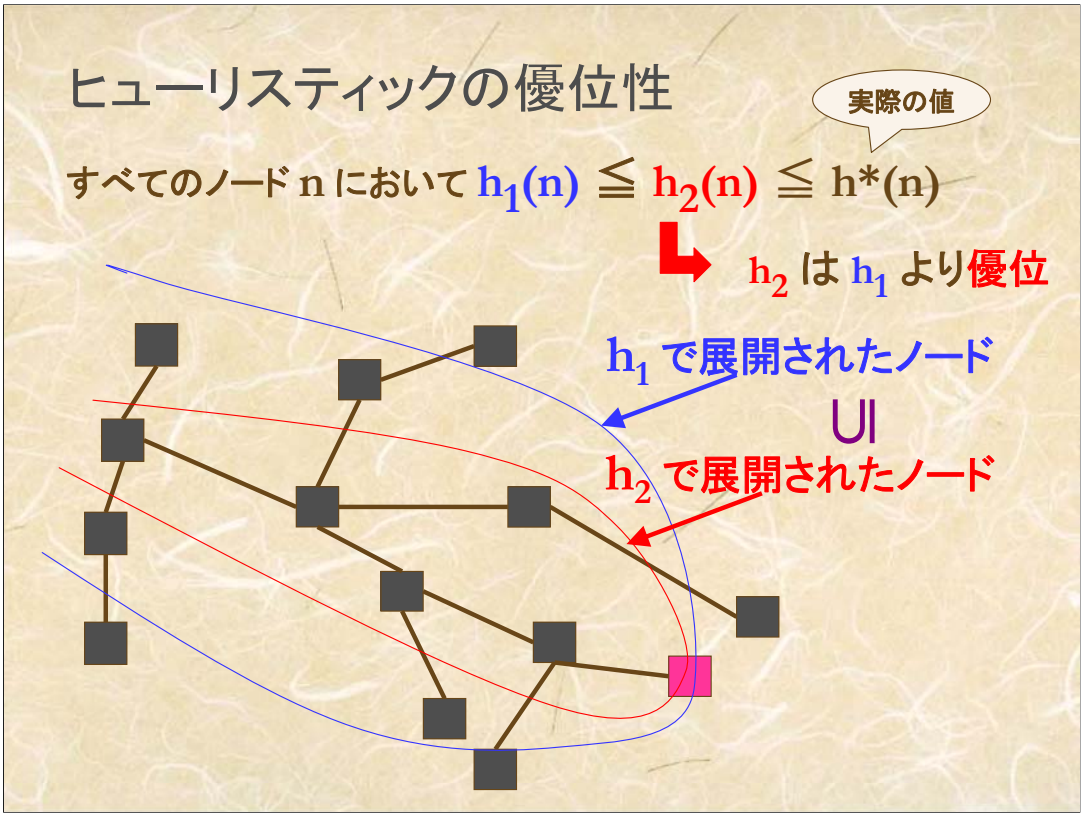
$$f(n) < f(G)$$

なので，必ず，Gより先に n がOPENリストから選ばれて展開される．このことから，最適解が見つからない間は，非最適解のゴールノードGは決してOPENリストから選ばれないことがわかる．したがって，最初に見つかる解は最適解である．


今回学んだ3つのアルゴリズムに対して，4つの評価尺度を検討した結果を表にまとめておく．




ヒューリスティック関数について，この3つの点から補足しておく．




すべてのノード n において

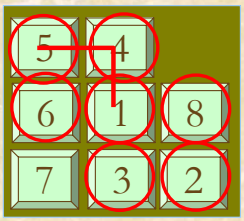
$h_1(n) \leqq h_2(n) \leqq h^*(n)$ =実際の値

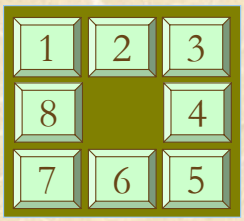
であるとき，$h_2$ は $h_1$ より**優位**であるという．より精度の良い見積もりということである．

このとき，ヒューリスティック関数として優位な $h_2$ を用いたA*アルゴリズムによって展開されるノードは，必ず，優位でない $h_1$ を用いたA*アルゴリズムによっても展開される．つまり，優位なヒューリスティック関数を用いた方が，はっきり効率が良いことが理論的に言えるのである．

## 8パズルのヒューリスティック関数

初期状態




ゴール

- 候補1　$h_1$=ゴールの位置にないタイルの数．上の例では7．
- 候補2　$h_2$=各タイルのゴール状態までのマンハッタン距離の和．上の例では18．

２＋３＋３＋２＋４＋２＋０＋２ ＝18

■どちらも許容的（楽観的）　■$h_2$は$h_1$より優位．

8パズルのヒューリスティック関数として，このスライドで示す2つが良く知られている．

ヒューリスティック関数$h_1$は，単に，「ゴールの位置にないタイルの数」を評価値とするものである．このスライドの初期状態では，「7」以外の7個のタイルがゴールの位置と異なるので，評価値は7となる．

ヒューリスティック関数$h_2$は，「各タイルのゴール状態までのマンハッタン距離の和」を評価値とする．**マンハッタン距離**とは，斜めに進むことができずに，上下左右だけの移動の総距離によって2点間の「距離」と定義するもので，ニューヨークのマンハッタン地区が，我が国の京都や札幌のように，道路が碁盤の目状になっていることから命名されている．スライドの例では，タイル1とゴール状態とのマンハッタン距離は2である．同様に他の7つのタイルについてマンハッタン距離を求め，総和を計算すると18となる．

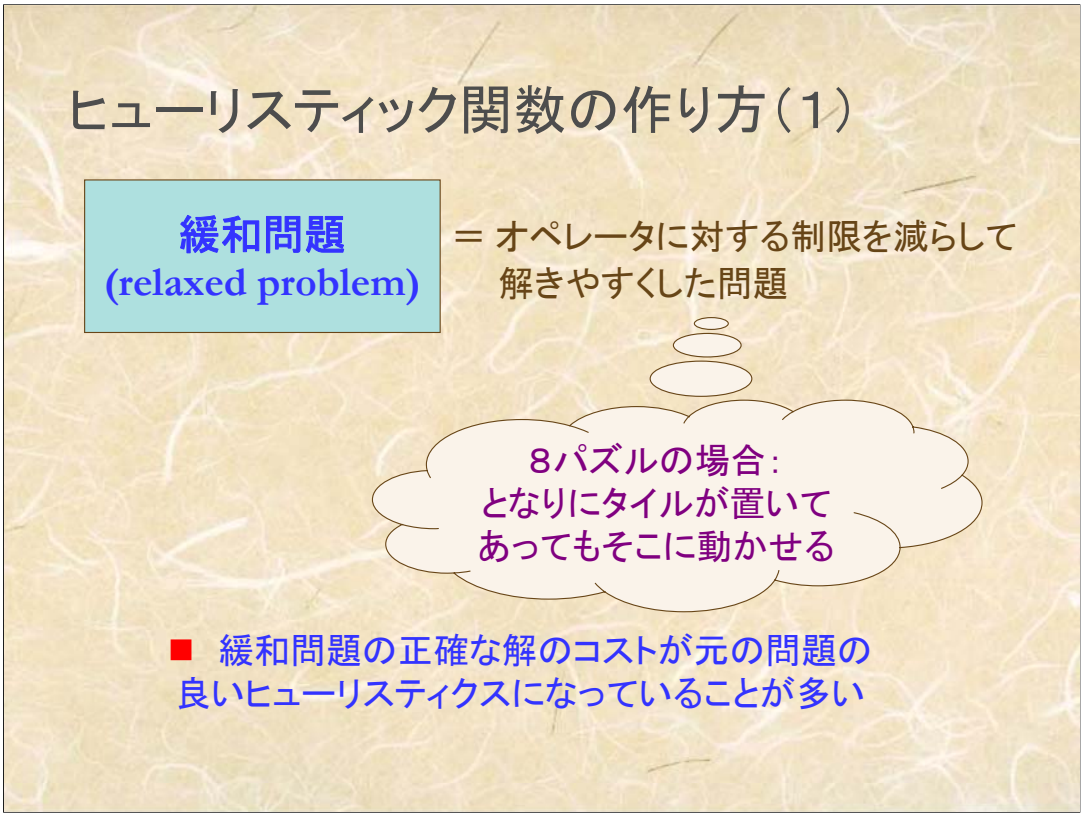
明らかに，どちらのヒューリスティック関数も許容的（楽観的）で，さらに，$h_2$ は $h_1$ より優位である．

# 8パズルの実験結果

| 解の長さ | 展開した平均ノード数 | | |
|---|---|---|---|
| | 反復深化 | A*($h_1$) | A*($h_2$) |
| 2 | 10 | 6 | 6 |
| 4 | 112 | 13 | 12 |
| 6 | 680 | 20 | 18 |
| 8 | 6384 | 39 | 25 |
| 10 | 47127 | 93 | 39 |
| 12 | 364404 | 227 | 73 |
| 14 | 3473941 | 539 | 113 |
| 16 | | 1301 | 211 |
| 18 | | 3056 | 363 |
| 20 | | 7276 | 676 |
| 22 | | 18094 | 1219 |
| 24 | | 39135 | 1641 |

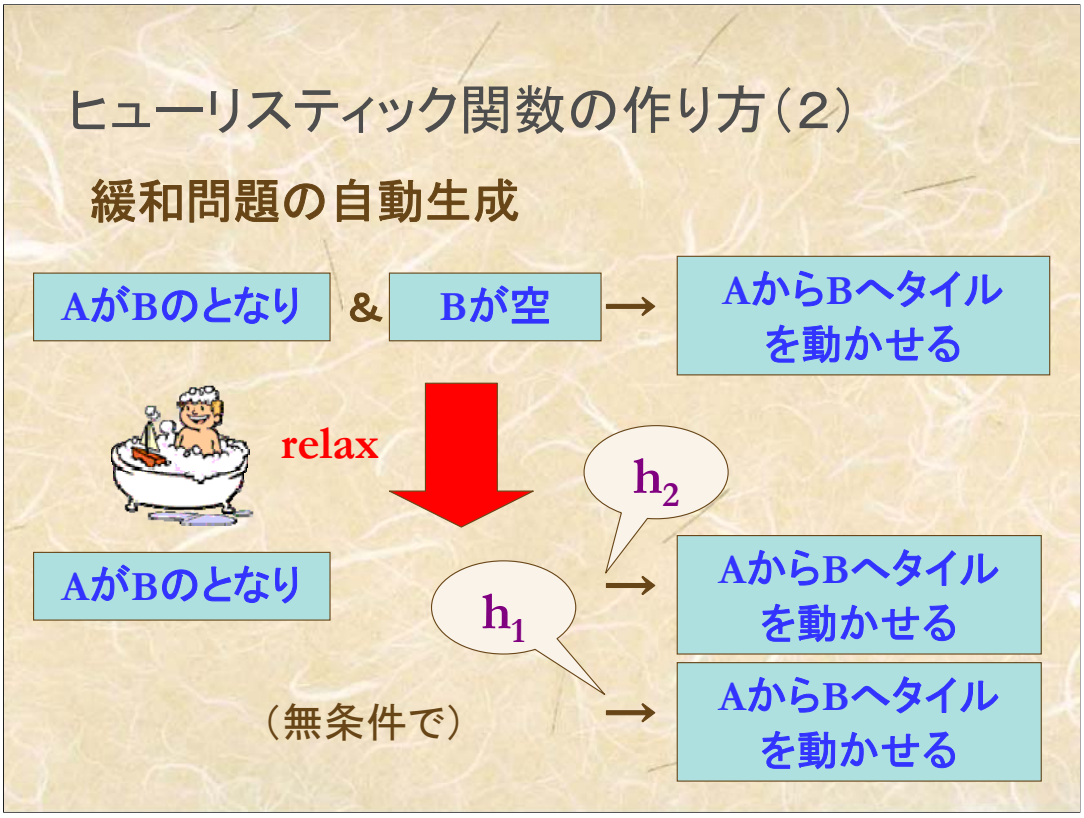
8パズルの実験結果である．初期状態によって最適解の長さが違うので別々にデータを整理してある．

前回学んだ反復深化探索（知識を用いない探索で，前回学んだ範囲ではベストだったもの）よりも，知識 $h_1$ を用いたA*探索が効率が良い．さらに，$h_1$よりも優位なヒューリステッィク $h_2$ を用いるとさらに効率が改善されることがわかる．




ヒューリスティック関数は，経験あるいは「発見」によって見つけるものなのだが，それを見つけるおよそのガイドラインはある．

それは，**弱条件問題**あるいは**緩和問題(relaxed problem)**と呼ばれるものを利用することである．緩和問題とは，オペレータに対する制限を減らして解きやすくした問題のことである．緩和問題の正確な解のコストが元の問題の良いヒューリスティクスになっていることが多い．緩和問題はもとの問題より解を得やすいのだから，「楽観的」な解が得られるからである．




このスライドは，8パズルのヒューリスティック関数を，緩和問題を利用して自動生成する考え方を示している．

まず，タイルを動かすルールを厳密に書くと，つぎのようになる．

**AがBのとなりで，かつ，Bが空ならば，AからBへタイルを動かせる**

このルールの条件部は2つの条件「AがBのとなり」と「Bが空」の論理積になっているので，両方とも真でなければタイルを動かせない．そこで，それを緩和して，もっと緩い条件のもとで動かせるようなルールを考えてみよう．

まず，「AがBのとなり」だけ真であれば動かせるようにするとつぎのようになる．

**AがBのとなりならば，AからBへタイルを動かせる**

こうしてできた緩和問題の最適解は，すでに述べたヒューリスティック関数$h_2$で計算できる．隣りが空いていることを無視してタイルをスライドさせてゴール位置まで動かせるならば，その最短手数がマンハッタン距離となるからである．

つぎに，極端な場合として，無条件でタイルを動かせるようにしてみる．

（無条件で）**AからBへタイルを動かせる**

こうしてできた緩和問題の最適解は，ヒューリスティック関数$h_1$で計算できる．ゴール位置にないタイルは1手でゴールに移動できるからである．

つぎに，「Bが空」だけ真であれば動かせるようにするとつぎのようになる．

**Bが空ならば，AからBへタイルを動かせる**

**この緩和問題の最適解のコストを$h_3$**というヒューリスティック関数で表すことにしよう．$h_3(n)$は状態 n を初期状態としたときに，上記のルールでタイルを動かしてゴールの配置とするための最短手数である．この関数の計算方法は読者の演習問題としておこう．

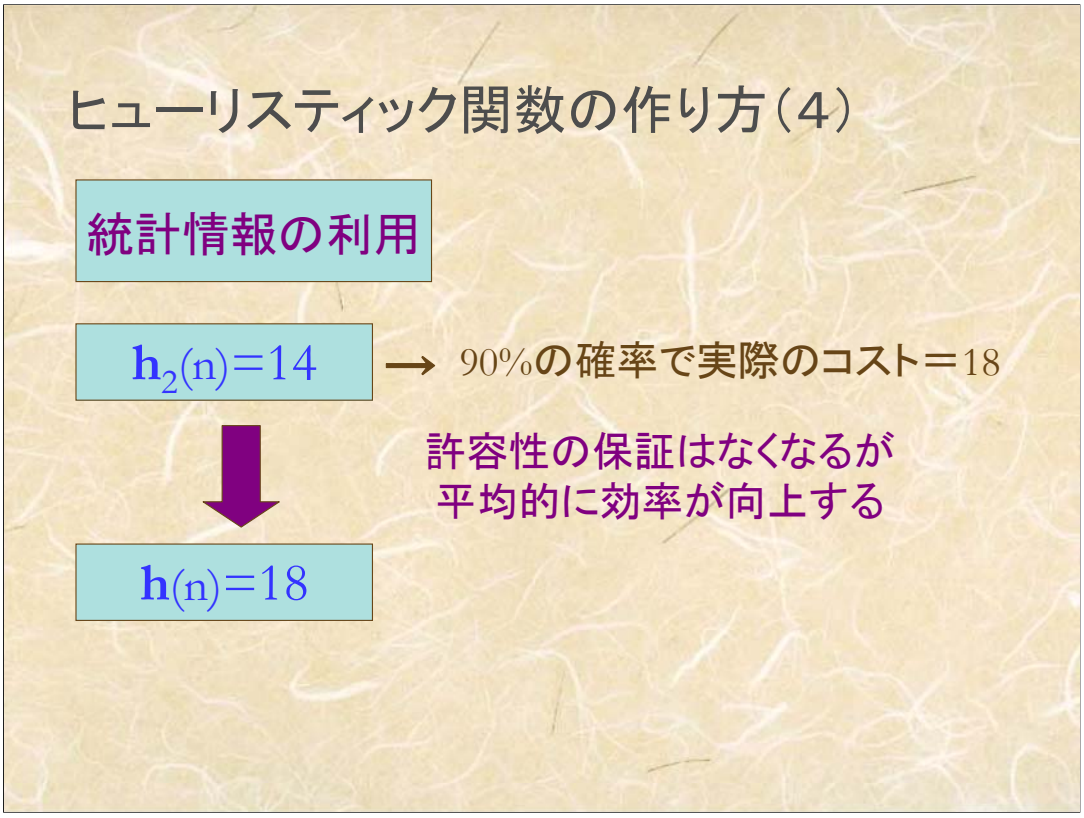
## ヒューリスティック関数の作り方（3）

$h_1, h_2, \ldots, h_m$ という許容的ヒューリスティクスがあり，
どれも他の優位にないとき，どれを選ぶか？

$$\mathbf{h}(n) = \max\left(\mathbf{h}_1(n), \mathbf{h}_2(n), \ldots, \mathbf{h}_m(n)\right)$$

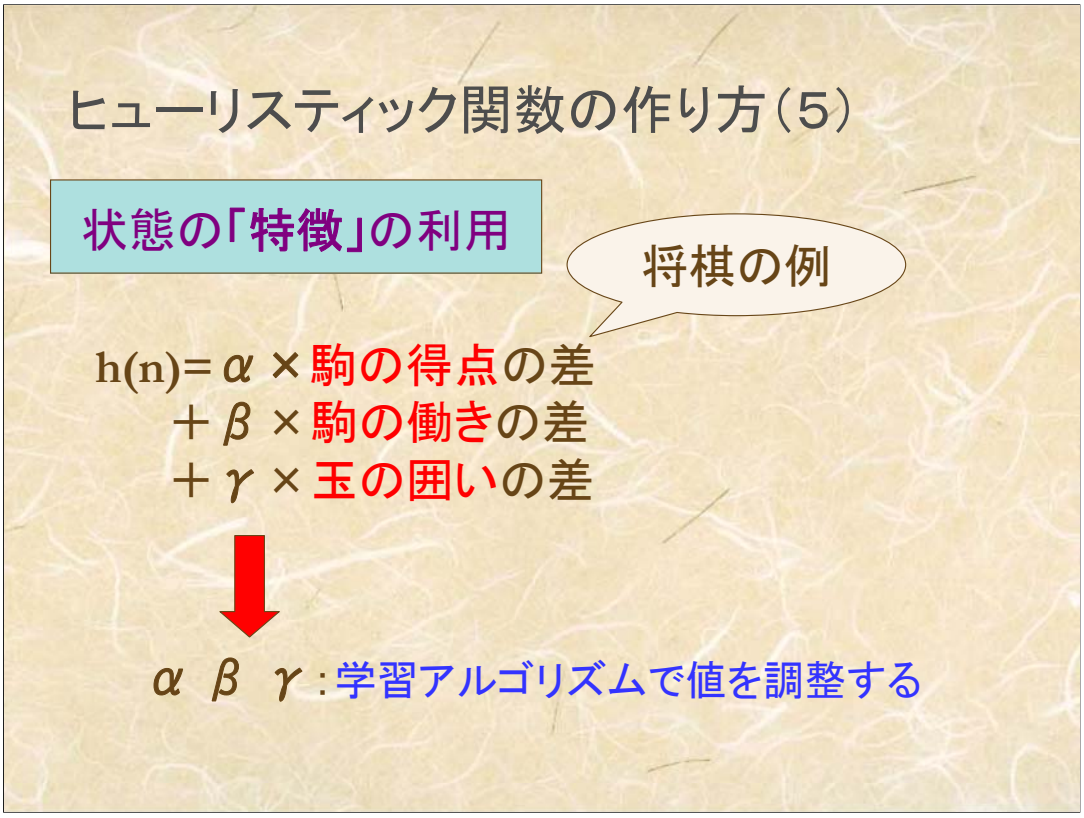
■ hは一つひとつの関数より優位

互いに他よりも優位でないヒューリスティック関数が複数あれば，それらの最大値を求める関数をヒューリスティック関数とすれば，これは一つひとつのどのヒューリスティック関数よりも優位である．この考え方によって，多くのヒューリスティックを合成して，より優れたヒューリスティックを生成することができる．




統計情報を利用して，経験を補強して，より精度の高いヒューリスティック関数を作れば，平均的に効率が向上する．しかし，許容的でなくなるので，完全性と最適性は保証されない．

たとえば，いろいろな実験の結果，$h_2(n)=14$のときには，90%の確率で実際のコストは18だったとしよう．そうすると，次回からは，$h_2(n)=14$のときには $h(n)=18$ とするような新しいヒューリスティック関数を作れば，許容性の保証（したがって，最適性の保証）はなくなるが，平均的に効率が向上することが期待できる．




考えている問題分野における状態の**特徴(feature)**に基づいてヒューリスティック関数をモデル化することもある．特徴とは，状態を構成している膨大な生情報を，問題に合わせて適切に要約した（少量の）情報のことである．状態から特徴を計算することを**特徴抽出**という．複数の特徴をベクトルとして並べたものを**特徴ベクトル**という．

このスライドは将棋の盤面の評価をしている例である．将棋の盤面の情報量は非常に大きいので，それを「駒の得点の差」，「駒の働きの差」，および「玉の囲いの差」という経験的に重要な3つの特徴として（ある適切な方法で）数量的に抽出し，それを $\alpha$，$\beta$，$\gamma$ という実数パラメータで重み付けた和としてヒューリスティック関数を定義している．このようなパラメータの値を事前に決定すること自体が難しい場合には，人工知能ならではの「**学習**」という機能で，実際に将棋の勝負を経験させながら，それを反映させてパラメータの値を動的に調整していく方法もある．