第 N 章アルゴリズム (N ∈ ℕ)

63 期　中尾(Aquila 所属、プロジェクトリーダー、機械設計、アルゴリズム担当)

0. O 記法(詳しく知りたい方向け)

まず本文に当たり、O 記法(O-notation)を定義しておく。

$O(g(n))$={f(n):ある正の定数 c,n0 が存在して、全ての n≥n0 に対して cg(n)≥f(n)　≥0 を満たす}

これは定数係数の範囲内で関数の上限を与えるのに用いられる。

(ex) $12n^4 = O(n^4)$

また、下の 5 式は任意の f(n) と g(n) について成立する。

$f(n) = O(f(n))$

$c \cdot O(f(n)) = O(f(n))$

$O(O(f(n))) = O(f(n))$

$O(f(n))O(g(n)) = O(f(n)g(n))$

$O(f(n)g(n)) = f(n)O(g(n))$

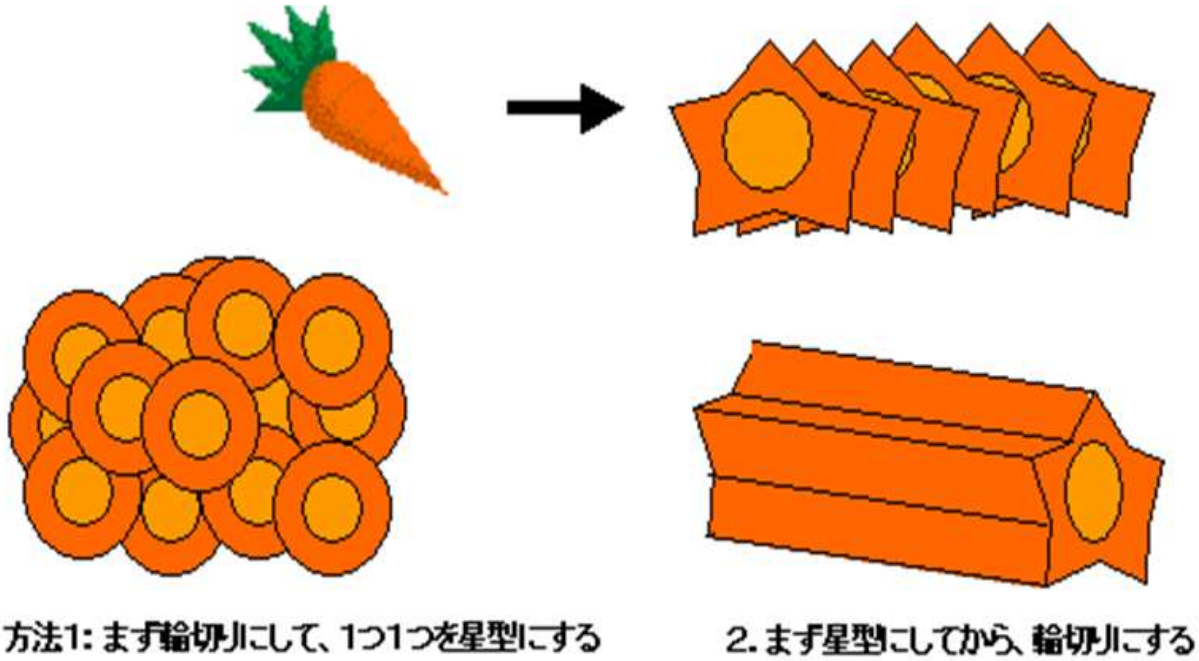
とだらだら定義したが、要は

$O(n)$のアルゴリズムは「n に比例した」回数の計算ステップ
$O(n^2)$のアルゴリズムは「n^2に比例した」回数の計算ステップ
$O(n^3)$のアルゴリズムは「n^3に比例した」回数の計算ステップ
が必要だよということです。

❗ちなみに、$n^2=O(n^3)$は主旨からずれますが間違いではないです。

第 N 章アルゴリズム (N ∈ ℕ)

1.アルゴリズムとは?(ここからは分かりやすいはずです)



(去年のプレゼンで使用　　国立情報学研究所のサイトより)

「アルゴリズム」とは何か物事を行うときのやり方の事です。

上の画像の例を使って説明します。

N 個の星形の人参の輪切りを作る時の事を考えます。

方法 1:まず輪切りにして 1 つ 1 つを星形にする。

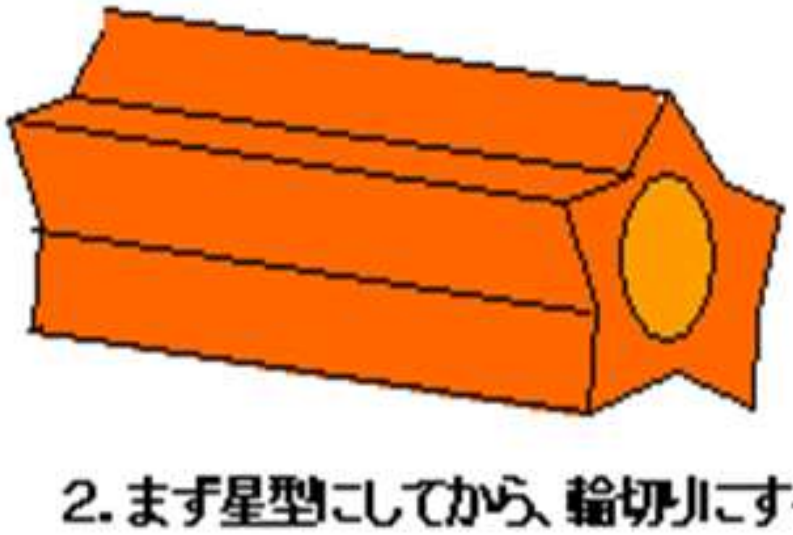
N 個輪切りを作るので、まずは N+1 回のステップ(ニンジンの切断)を行います。

次に、輪切りを型抜きで星形にします。

これを N 個の輪切りニンジンに適応します。これには N 回のステップが必要です。

これらを合わせて 2N+1 回のステップが必要と分かります。

第 N 章アルゴリズム (N ∈ ℕ)



方法 2:まず星形にしてから輪切りにする。
さて、どうやって大きなニンジンを星形にするかは置いておいて、
まずは星形にします。
これには 1 回のステップが必要です。
次に、輪切りにします。
これには N+1 回のステップが必要となります。
これらを合計して、方法 2 では N+2 回のステップが必要と分かります。

まとめ：この 2 つの方法を比較して、方法 2 の方が楽だと分かります。

このように効率よく物事を行う方法を考えることがアルゴリズムの醍醐味と言えましょう。
ちなみに、方法 1,方法 2 共に、$O(n)$ と表現されます。

第 N 章アルゴリズム (N ∈ ℕ)

2．ソート(整列)アルゴリズム

2-1.挿入ソート(writer::63 期黒田)

整列してある配列に追加要素を適切な場所に挿入すること。

平均計算時間・最悪計算時間がともに $O(n^2)$と遅いが、アルゴリズムが単純で実装が容易なため、しばしば用いられる。安定な内部ソート。**基本挿入法**ともいう。

とりあえず、８４３７６５２１の数列を並び替えることを考えます！！！

8	4	3	7	6	5	2	1	（初期データ）
4	8	**3**	7	6	5	2	1	（1 回目のループ終了時）
3	4	8	**7**	6	5	2	1	（2 回目のループ終了時）
3	4	7	8	**6**	5	2	1	（3 回目のループ終了時）
3	4	6	7	8	**5**	2	1	（4 回目のループ終了時）
3	4	5	6	7	8	**2**	1	（5 回目のループ終了時）
2	3	4	5	6	7	8	**1**	（6 回目のループ終了時）
1	2	3	4	5	6	7	8	（7 回目のループ終了時。ソート完了）

アルゴリズム～

まず 0 番目と 1 番目の要素を比較し、順番が逆であれば入れ換える。次に、2 番目の要素が 1 番目までの要素より小さい場合、正しい順に並ぶように「挿入」する（配列の場合、前の要素を後ろに一つずつずらす)。この操作で、2 番目までのデータが整列済みとなる（ただし、さらにデータが挿入される可能性があるので確定ではない)。このあと、3 番目以降の要素について、整列済みデータとの比較と適切な位置への挿入を繰り返す。

この場合、８個の数字があるので、8－1 回の操作が必要となります。つまり、ｎ個のものを並び替えるとすると、n-1 回の操作が必要となります。

第 N 章アルゴリズム (N ∈ ℕ)

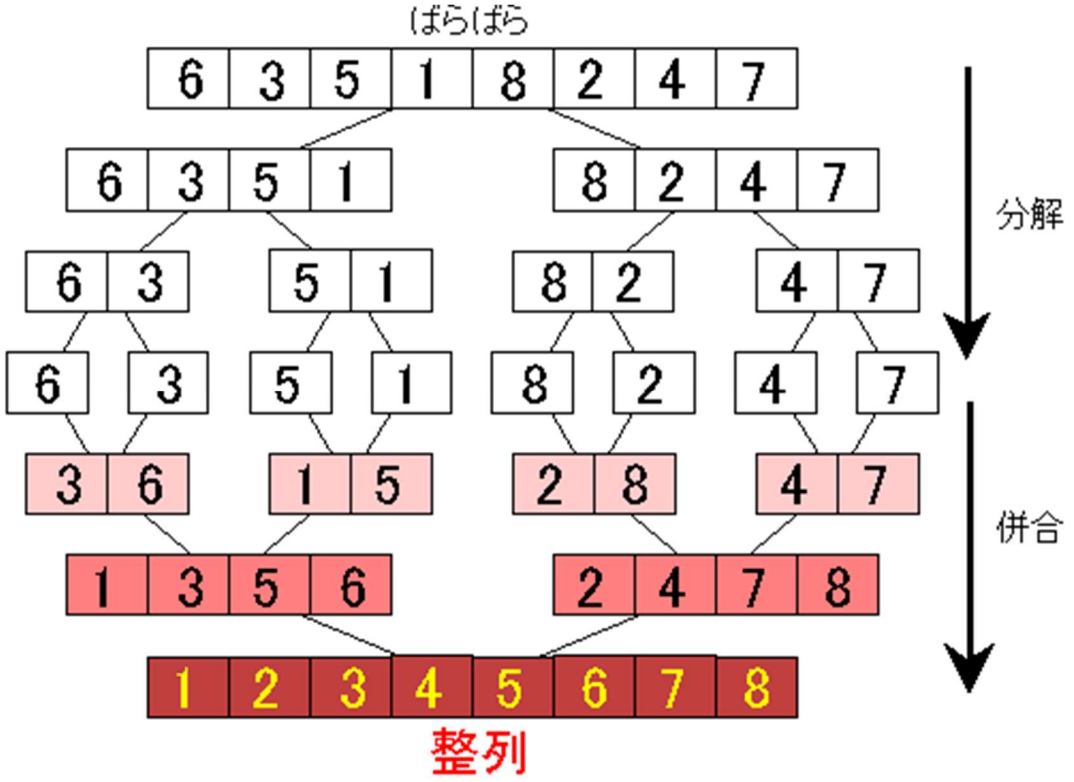
2-2.マージソート

マージソートは分割統治法(divide-and-conquer)パラダイムに基づいたアルゴリズムです。

多くの場合は安定性があり、平均計算時間は $O(n \log n)$ です。(n は数列の要素数)

分割統治法とは、
分割：問題を幾つかのより小さな同じ問題(部分問題)に分割する
統治：部分問題を再帰的に解いていく。
　　ただし、問題のサイズが十分小さい場合は直接的な方法で解く。
結合：部分問題の解を組み合わせて元の問題の解を得る。
の３つの段階から構成されます。

具体的に行きましょう。
今回は数列 A を整列させる事を考えています。



(鹿児島大学のサイトより)

第 N 章アルゴリズム (N ∈ ℕ)

それでは、マージソートを見ていきましょう。

①分割:～問題の分割～

最初の問題は要素数 8 の数列 A を並び替える問題です。

6	3	5	1	8	2	4	7

←数列 A

この問題を半分に分けて、それを問題 a,問題 b とします。

->問題 a：数列 A[0]~A[3]を並び替える問題

6	3	5	1

←数列 A[0]~A[3](A の 0 番目から 3 番目までの数列)

->問題 b：数列 A[4]~A[7]を並び替える問題

8	2	4	7

←数列 A[4]~A[7](A の 4 番目から 7 番目までの数列)

つまり、ある数列を並び替える問題を
元の数列の半分の長さの数列を並び替える問題に分けるということです。

でも、どんどん分けていくと、
要素が 1 つしかない数列を並び替える問題となり、分けられなくなります。

この状態が「問題のサイズが十分小さい」状態に当たります。

第 N 章アルゴリズム (N ∈ ℕ)

②統治&結合(写真の併合に当たる)：～問題の解決～

“マージ”と呼ばれる手続きを行います。

この手続きは 2 つのソート済みの数列をくっつけて、

より大きなソート済み数列をつくる手続きです。

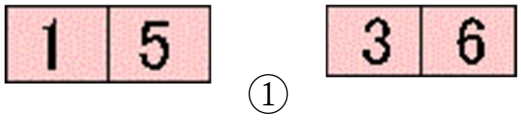
マージ

与えられた 2 つの数列の先頭の数字を比べて、小さい方を新しい数列の後ろに

加えて、加えられた要素を元の数列から取り除く。

この作業を与えられた数列が共に空になるまで続ける。

数列{1,5}と数列{3,6}を題材にして考えてみます。



①　新しい数列{(最初は何もない)}

先頭の数字 1 と 5 を比べて、1 の方が小さいので新しい数列に 1 を加える。

-> 新しい数列{1}

-> 数列{1,5} は 数列{5}に。(1 が取り除かれた。)

②{5}と{3,6}の先頭の数字を比べて 3 の方が小さいので、3 を加える。

-> 新しい数列{1,3}

-> 数列{3,6}は数列{6}に。

③これを繰り返す。

-> 新しい数列{1,3,5,6}

これによって分割された数列が整列しながら、元の長さに結合されて行きます。

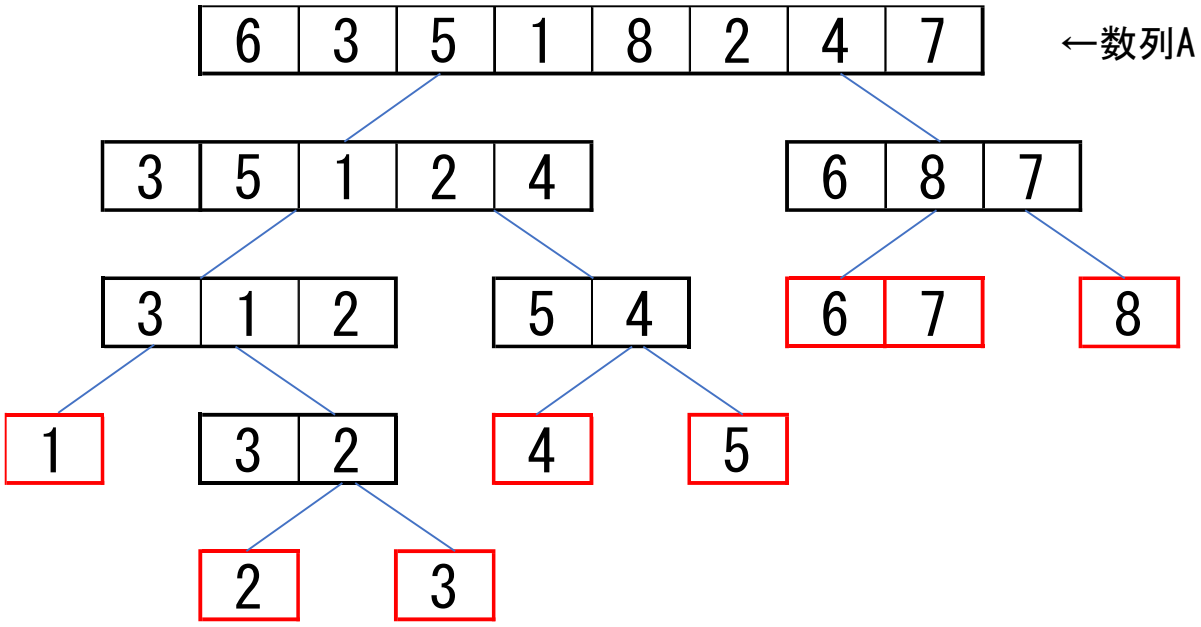
第 N 章アルゴリズム (N ∈ ℕ)

2-3.クイックソート(63 期松下(Aquila 所属、機体製作、デバッグ))

クイックソートもマージソートと同じく分割統治法を用いますが、

要素の分け方が違います。数列を中身の数字によって分けるのです。

例えば下の図のようなものがあります。



上図は「軸要素」というしきい値を決定しそれによって 2 つに分けています。

今回の「軸要素」は[数列の左から 2 つの内大きい方]です。

具体的に説明すると

① 数列 A の左側にある | 6 | 3 | から大きい方である 6 を基準に数列 A を以下の二つに分けます。

-> 6 より小さいもの | 3 | 5 | 1 | 2 | 4 | ←数列 B

-> 6 以上のもの | 6 | 8 | 7 | ←数列 C

② ①と同じように基準をきめて二つに分けていきます。

以上を続けると上図のように並び替えられます。

2-4.中央地ソート(松下)

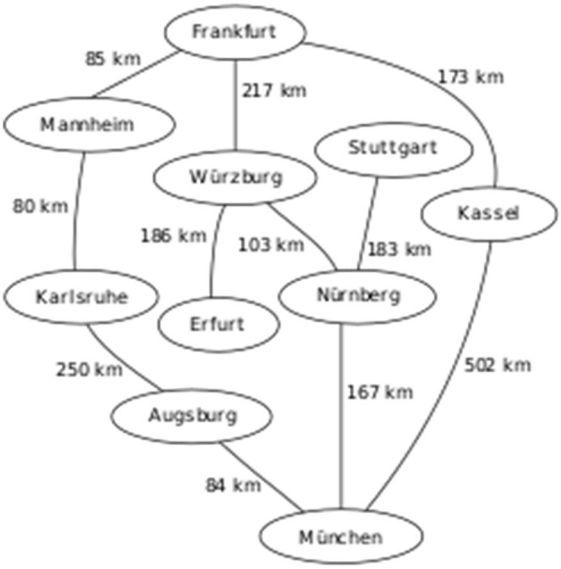
中央地ソートはクイックソートの一種で「軸要素」が中央地になっています。

第 N 章アルゴリズム (N ∈ ℕ)

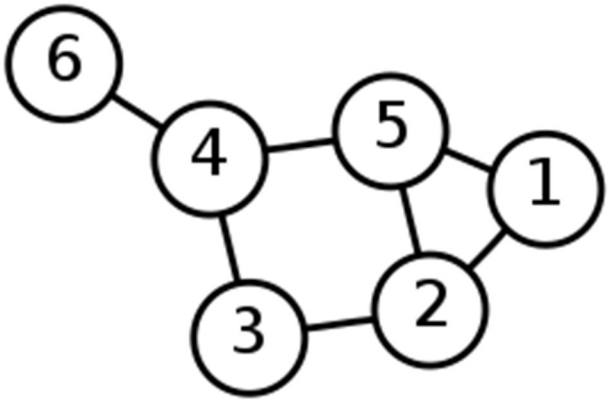
3.探索アルゴリズム

3-0.何を探索するのか

これらのアルゴリズムが探索するのはグラフです。
迷路や人間関係、物事の論理関係、論理式をグラフで表現して探索します。



グラフとは…頂点と辺で構成される下図のような物です。



↑ドイツの都市間の接続を表現した物

第 N 章アルゴリズム (N ∈ ℕ)

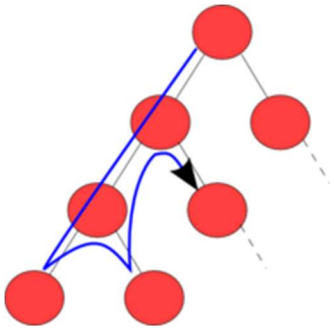
3-1.深さ優先探索/幅優先探索

深さ優先探索はスタック(stack)、幅優先探索はキュー(queue)を用いて探索します。時間計算量は $O(|E|)$ です。(E:辺の数)

Stack とは:上から入れて、上から出す箱です。(LIFO)
1 個しか出てこないマーブル(甘い奴)を想像して頂けると分かり易いと思います。(Stack の絵)

Queue とは：上から入れて、下から出す箱です。(FIFO)
中にカードを貯められる改札機みたいな感じです。(Queue の絵)

幅優先探索は Queue の中に今探索中のノードにつながるノードを挿入して、
Queue からノードを出してくるというステップを繰り返して探索します。
深さ優先探索は幅優先探索での Queue に当たる部分を stack に入れ替えたものです。ただ、パソコンの処理の並び方は stack と同じなので、
再帰を利用すると簡単に実装できます。



←深さ優先探索のイメージ

第 N 章アルゴリズム (N ∈ ℕ)

3-2.最良優先探索

最良優先探索は幅優先探索を何らかの規則に沿って探索するように拡張した
アルゴリズムです。
何らかの規則に沿って探索させる為に、先ほどの Queue を改造します。

①まず、何らかの規則を表現する評価関数を準備します。
②先ほどの Queue を評価関数の大きい順にソートされた
　　Queue(Priority Queue)になるように改造します。

たったこれだけで最良優先探索に変わります。

また、
Priority Queue は 2 分ヒープを用いて実装されることが多いです。

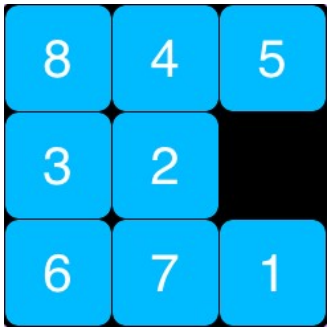
ちなみに、
ダイクストラ法(単一始点最短経路問題、$O((E+V)\log V)$や、
　次に紹介する A*アルゴリズムなどが最良優先探索に含まれます。

第 N 章アルゴリズム (N ∈ ℕ)

3-3.A*探索

A*は上記の通り、一種の最良優先探索です。
評価用の関数は、初期地点から現在の地点までの深さを表す g(n) と、目標地点までの距離と予想される値を返す h(n) という関数を用いて、*F(n)=h(n)+g(n)*と表されます。
ちなみに、この h(n)はヒューリスティック関数と呼ばれます。

マンハッタン距離を用いた 8 パズルの解法の探索などが有名ですね。



←8 パズル(小さい頃に景品で貰った事を覚えています。)

読者のほとんどはお気づきだと思いますが、このアルゴリズムの性能はこの評価関数 h(n)に大きく左右されます。

更に感の良い読者ならお気づきかもしれませんが、このアルゴリズムはダイクストラ法の一般化です。h(n)が無ければ、ダイクストラアルゴリズムと何も変わりません。
つまり、探索路にコストが負の辺が存在すればこのアルゴリズムの完全性は失われてしまいます。
また、ゲームの探索をする時に盤面状態が巨大な場合は、A*は適していません。
そういう時のために IDA*や HPA*といったアルゴリズムが開発されました。

[備考]二人零和有限確定完全情報ゲーム

ふたりぜろわゆうげんかくていかんぜんじょうほうげーむ

これはゲーム理論におけるゲームの分類の一つです。
二人零和有限確定完全情報ゲームはその性質上から先読みが可能となります。
そして、これは人工知能の分野で早くから研究されてきました。
ミニマックス法やα-β法が有名です。
一つ一つ意味を説明していきます。

①二人

ゲームを行うプレイヤーが２人ということです。

②零和

ゲーム上でプレイしている全プレイヤーの利益の合計が常にゼロ、または個々のプレイヤーの差す手の組み合わせに対する利益の合計が常に一定ということです。

③有限

ゲームの各プレイヤーの差す手の組み合わせが有限であるということです。
有限でなければ探索をする事は至難の業になりますよね。

④確定

偶然の要素がゲームに関わらないということです。つまり、不確定ゲームは「運ゲー」と言い換えられますね。ちなみに、双六はさいころを振って進むので不確定ゲームです。

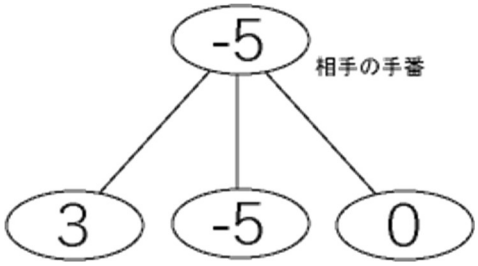
⑤確定情報

これまでに各プレイヤーの行った行動がどのプレイヤーにも公開されているということです。多くのカードゲームはお互いの手がわからないので、
不確定情報ゲームとなります。

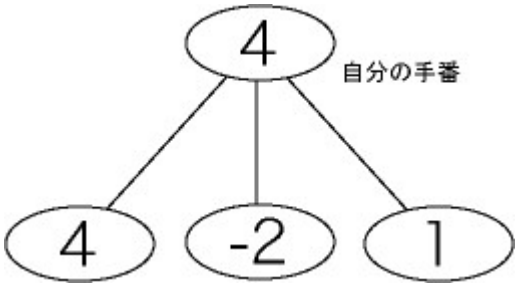
第 N 章アルゴリズム (N ∈ ℕ)

3-4.MINIMAX 法

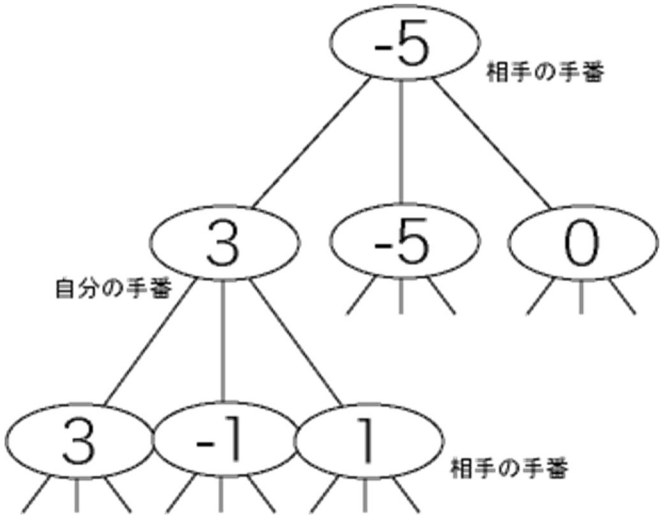
ミニマックス探索は、先ほどの二人零和有限完全確定情報ゲームにおいて、
想定される最大の被害を最小化するアルゴリズムです。



①手を読みたい局面が相手の手番の時
その次の(自分の)面の評価値を最小にする手を選ぶ。
<-(3,-5,0)の最小値は”-5”。



②手を読みたい局面が自分の局面の時
その次の曲面における盤面評価値が最大になる物を選ぶ。
<-(4,-2,1)の最大値は”4”。



③上の①、②の動作を再帰的に繰り返すことでゲームの手番を予想する。
<-こんな感じです。

第 N 章アルゴリズム (N ∈ ℕ)

また、ゲーム木は深くなるにつれて、曲面数が爆発的に増加するので、ある程度以上の手数を読もうとすると、実用的な時間では答えを返せなくなってしまう。

また、このアルゴリズムの応用版として、 α - β 法があります。
これは、MINIMAX 法の弱点である、
評価するのに不必要な局面までしらみつぶしに評価してしまう、という点を改善したもので、読む必要のない局面を刈っています。

第N章アルゴリズム (N ∈ ℕ)

4.終わりに

ここまでパネルを読んでくださった方に感謝いたします。

このパネルを書きあげるにあたって、
いかにして抽象的な概念を誰にでも分かり易く言語化するか、
自分は言語化できるほどの理解ができているのか、
前提知識を見落としていないか等、様々な問題に直面しました。

自分の可能な限りを尽くして、前提知識の無い方でも理解可能なように
書き上げたつもりです。

まだまだ、単一始点最短経路問題など、書き足りないのですが、
ここでお開きにさせていただきます。

「数学を勉強していて、わからないことがあると悩む。
何日も考えたり、本を読んだりしていると、ふと《あ、こういうことか》ってわかる。それはすごく嬉しい体験なんだ。」

By 数学ガール