アルゴリズムを学ぶ

2020年10月24日

Studio-Sumicco

高島　智俊

アルゴリズムとは

アルゴリズムとは問題解決のための方法や手順のことです。
ITの世界では効率的にデータを扱うための技法として使われることが多いですが、決してITに限った言葉ではなく、
何か目的があり、そこに至るまでの道のり、プロセス、
それらを考える思考、すべてがアルゴリズムといえるでしょう。

例題１

あなたは今、コンビニのレジで、税込108円のおにぎりのお会計をするところです。
お財布には、100円玉2枚と、１０円玉2枚、1円玉3枚が入っています。どのように支払うのが、お釣りの小銭の枚数が最も少なくなるでしょうか？

例題２

5～10までの和を求めよ。

5+6+7+8+9+10＝45

連番の和はガウスの法則を使え。

（最小値+最大値）×和の個数÷２

(5+10)*3＝45

データを使ったアルゴリズム

const list = [4,2,5,1,0,3];

例えば繰り返し処理を使い、この配列を昇順にソートするには
どうすればいいでしょうか。

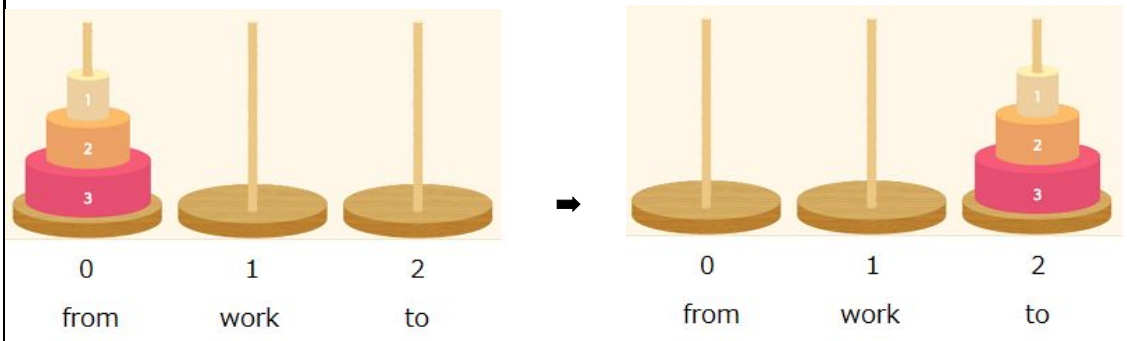
初歩的なソート

```
const list = [4,2,5,1,0,3];
//フォーカスするインデックス
for(let i=0; i<list.length; i++){
    //初期値(i+1)なのでフォーカスするインデックスの次の要素から判定
   for(let j=i+1; j<list.length; j++){
        //フォーカスする要素より値が小さかった場合、値を入れ替える
        if(list[i] > list[j]){
            let kari = list[i];
            list[i] = list[j];
            list[j] = kari;
        }
   }
}
```

他にもバブルソート、選択ソート、ヒープソート、クイックソートなど
様々なソート法があります。
ソート系でも探索系でも、何か一つ、いつでも抜ける刀として、
自分なりのアルゴリズムを持っていることは強みになるでしょう。

ハノイの塔

ハノイの塔とは、円盤を移動させるパズルです。
3本の杭があり、fromにある円盤を状態を保ったままtoに移動できればゴールです。



ルール
1度につき1枚の円盤しか動かせない。
小さい円盤の上にそれより大きな円盤は置けない。

円盤の枚数が1枚、2枚、3枚となったときの円盤の動きと
その最小回数を考えてみよう。

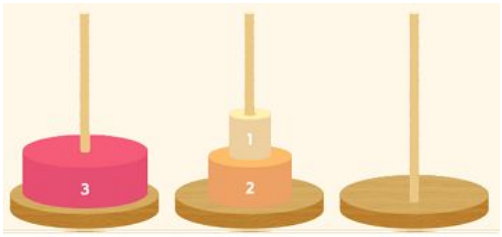
円盤の数とその動きの推移と法則

円盤１枚のとき：１回；
円盤２枚のとき：３回；
円盤３枚のとき：７回；

ハノイの数式は、２　の　円盤の数乗　－　１；
とされていて、すなわち、
円盤マイナス１枚の回数　＊　２　＋　１；

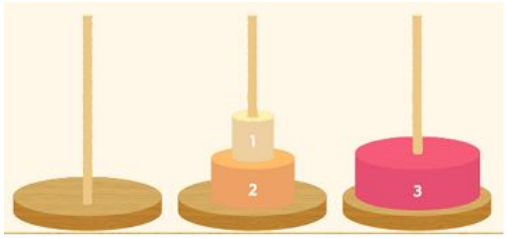
なぜそうなるのか！

なぜなら結局のところ、一番底辺の円盤を動かすには、
円盤マイナス１のときのゴールを真ん中のworkにつくり、
そして初めて底辺がtoに移動でき、さらにworkにつくった回数
を介してtoに移動するからです。

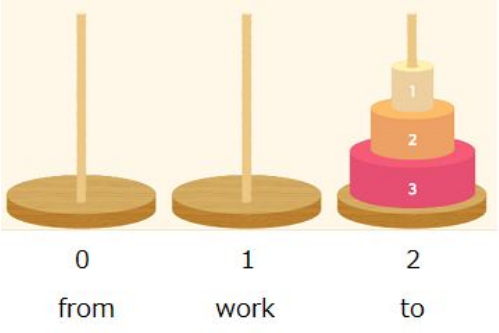


３回

➡



１回



３回

3+1+3＝7；

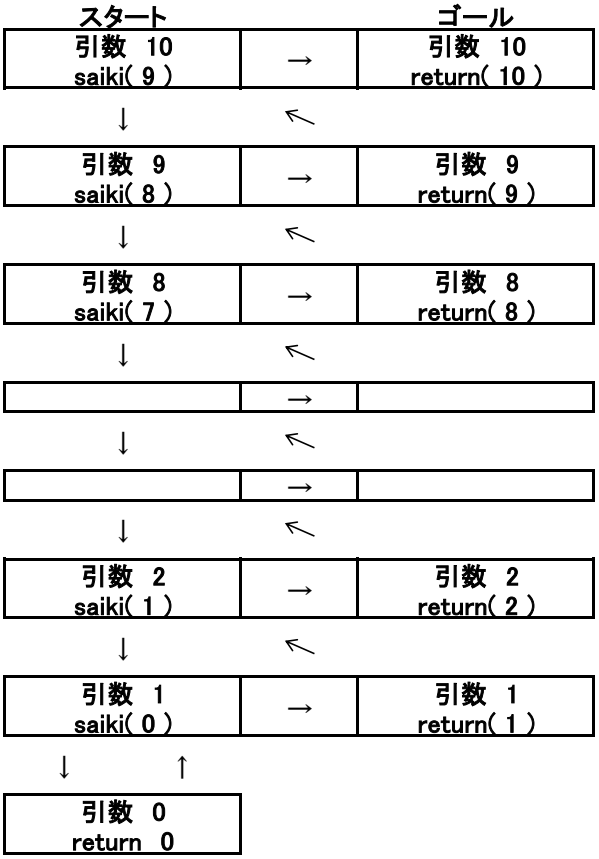
再帰処理とは

定義は様々ですが、基本的には自身の中で自身を呼び出す処理のことです。

例えば繰り返し処理の基礎である0～9の出力を再帰処理でやった場合

```
const saiki = ( n ) => {
  if ( n == 0 ) {
    return 0;
  }
  console.log( saiki( n - 1) );
  return n;
}
saiki( 10 );
```

イメージ図

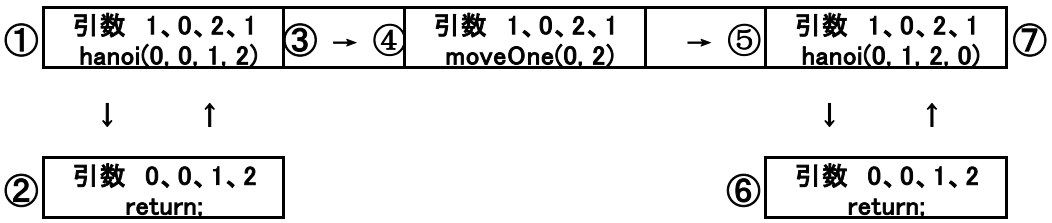


ハノイの再帰処理

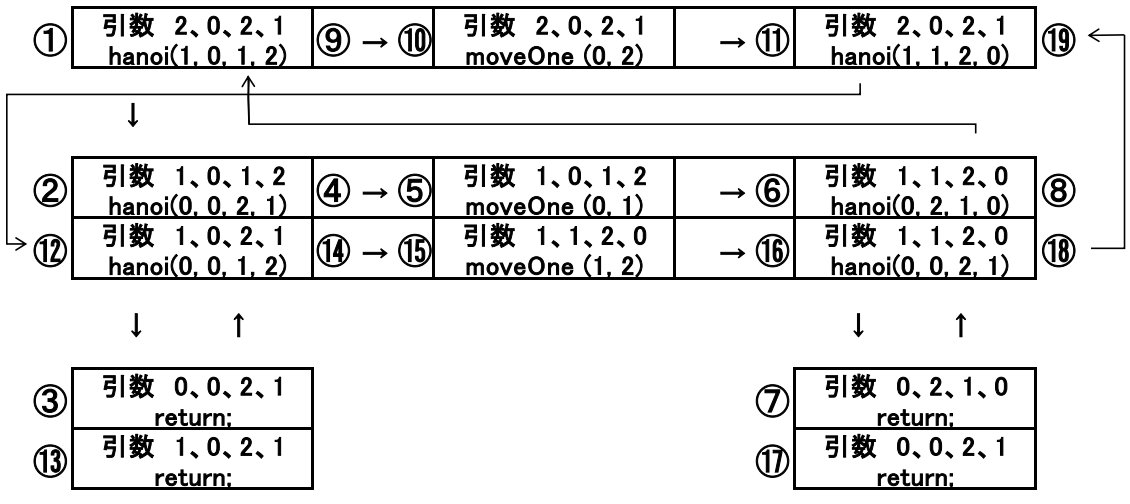
使うメソッドはいたってシンプルでこれだけです。
よくみましょう。

```
hanoi(n, from, to, work) {  //円盤の枚数と杭
    if (n == 0) {  //底打ち
        return;
    }
    hanoi(n-1, from, work, to);  //toをwork入れ替えて呼び出し
    moveOne(from, to);  //fromの１番上にある円盤をtoに移動する実処理
    hanoi(n-1, work, to, from);  //fromをworkに入れ替えて呼び出し
}
```

1枚の場合



2枚の場合



３枚のときはどういう流れになるのかコードを追って考えてみよう。