

アルゴリズムとデータ構造 (序論)



No.1b 2019年4月8日

高橋信行

情報科学 (Computing Science)

- 対象によって成り立っている学問ではない.
- 方法論を研究する学問
 - すべてを対象とし, 対象を形式化し, 形式化されたモデルを研究
 - メタレベルの研究が可能.
- 極めて複雑で大きなシステムを理解するための方法論を提供
 - 自然言語, 遺伝子情報, 脳などは従来の科学の方法論では歯が立たない分野
- 研究の基調となるのは、いわゆるコンピュータ言語
 - 様々なものの本質を見通す道具を手に入れることができるようになる。

計算とは

- 情報を表現し、それを変換していく過程である.
- 情報をどう表現するか、それをどう変換していくか、そういう計算の本質を探究するのが情報科学である.
- 知能とは何かを探ることにもなる.

コアカリキュラム： 専門基準

- ① データ構造とアルゴリズム
- ② コンピュータ アーキテクチャ
- ③ プログラミング言語論
- ④ オペレーティングシステム
- ⑤ コンパイラ
- ⑥ データベース(基礎的な部分)
- ⑦ 情報ネットワーク(基礎的な部分)

アルゴリズムとデータ構造

- 良いプログラムを作るために、必ず学ばねばならない基礎知識
- 優れたアルゴリズムの動作原理の理解とプログラムとしての実現の技術を学ぶ.
 - コンピュータの世界は、日進月歩
 - 物事の本質を理解しておけば、表面的な変化があっても正しい対処
- プログラムの実行に要する計算手間を客観的に評価
 - 効率の良さを評価するため

アルゴリズムとは

■ 問題解決の処方箋

- 処方箋: 医師が薬の種類, 量、服用法を書いた書類

■ 問題を一般的に解く手順

- 任意のデータ例を入力すると, 有限ステップ数の計算によって問題の解を出力して停止するプログラムのこと

■ (計算) 手続き

- 必ずしも有限ステップで停止しない場合を含めるとき

■ 正確に説明できる手順

- 何でも, 標準的なプログラミング言語で書ける.

アルゴリズムが存在する問題とは

- チューリングマシンによって計算できる問題である.
- コンピュータ
 - 万能チューリング機械である.

ユークリッドの互除法:

二つの正整数の最大公約数を求めるアルゴリズム

アルゴリズムの手順:

二つの正整数を m , n とする.

[ステップ1] m を n で割って, 余りを r とする.

[ステップ2] $r=0$ であれば, アルゴリズムは終了
(このとき, n が最大公約数である)

[ステップ3] $m \leftarrow n$ とする。次に $n \leftarrow r$ として、ステップ1に戻る.

ユークリッドの互除法のC プログラム表現

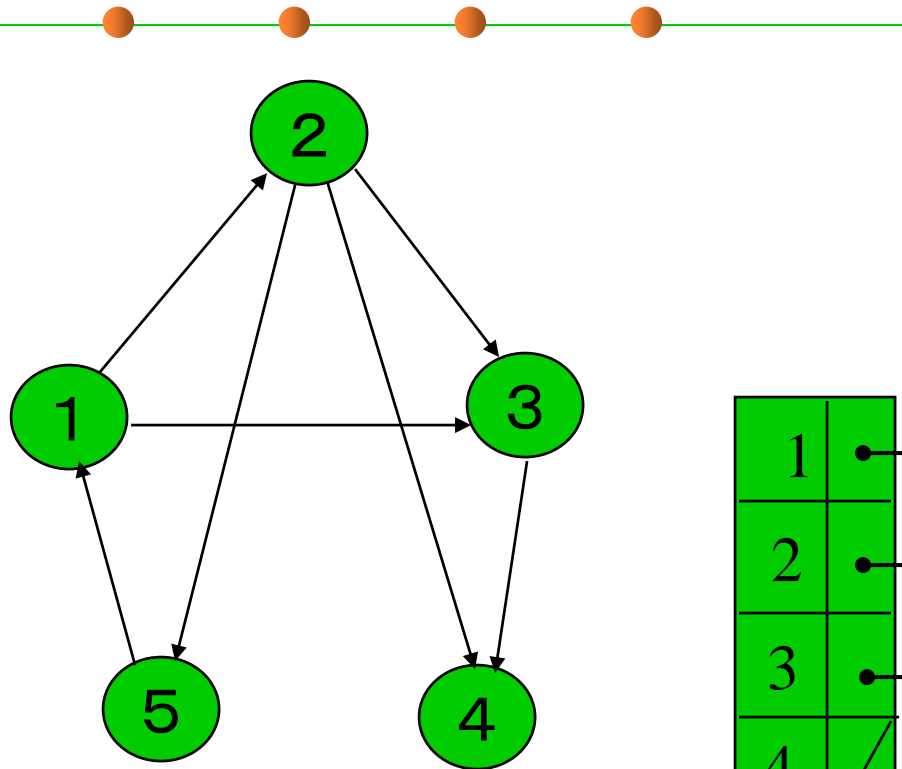
```
int gcd (int m, n){  
    int r ;  
    do {  
        r = m % n ;  
        m = n ;  
        n = r ;  
    } while( r != 0 );  
    gcd = m;  
}
```

アルゴリズム＋データ構造＝プログラム

- 問題の解決：
 - プログラムとして実現するのに向いている解法
- データ構造
 - 計算中に用いるデータの構成法
- アルゴリズムの理解にはデータ構造の理解が必須

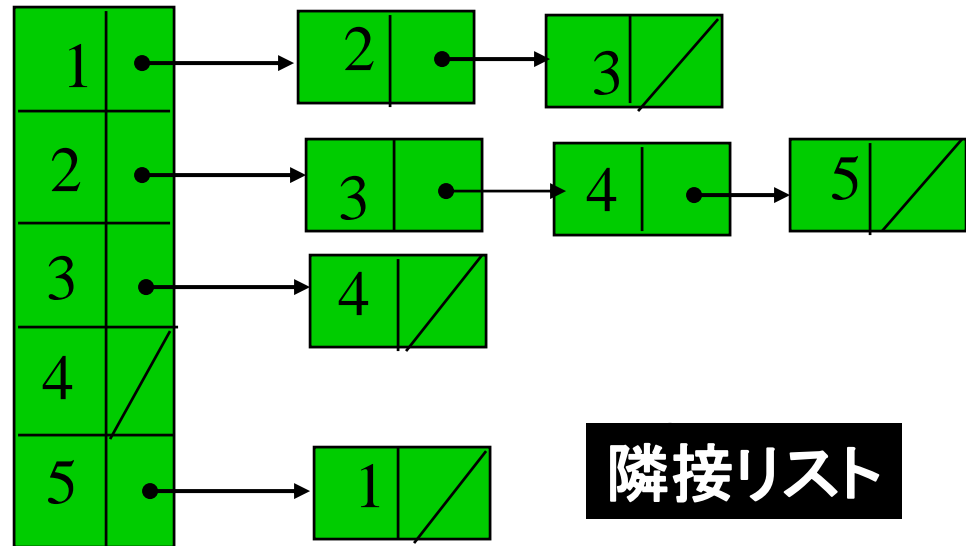
データの表現例

有向グラフ



	1	2	3	4	5
1	0	1	1	0	0
2	0	0	1	1	1
3	0	0	0	1	0
4	0	0	0	0	0
5	1	0	0	0	0

隣接行列



隣接リスト

データ構造の必要性

- 問題に応じて適切なデータ構造の表現によるモデル化が必要
- この物理構造をどのように設計するかによって問題の処理効率が大きく左右される

アルゴリズム(プログラム)の性能

- プログラムの正当性
 - プログラム全体が正しいこと
- アルゴリズムの計算量
 - 処理が速いこと
- データ構造の領域量
 - 作業に要する資源(記憶領域など)が少なくて
すむこと

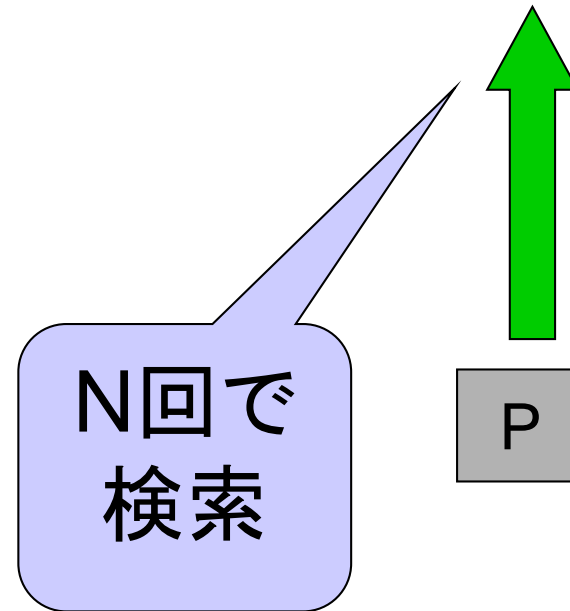
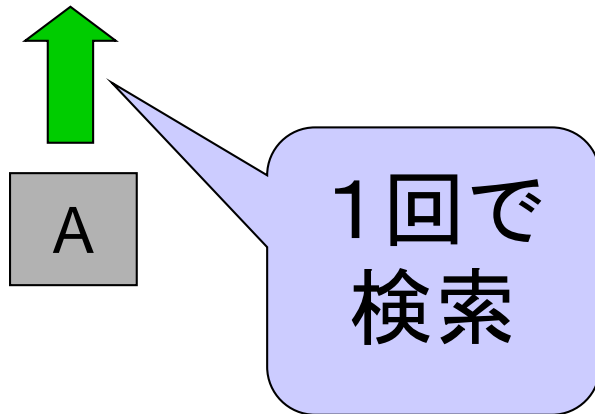
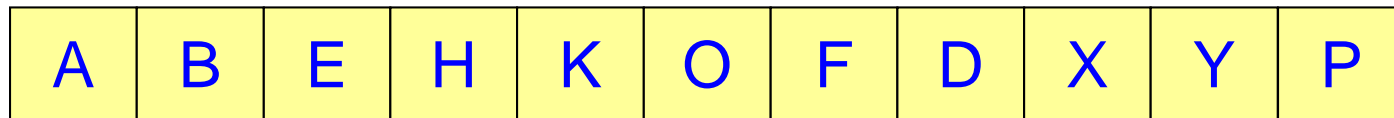
ソフトウェア工学

- 対象：大きいプログラム
- 解くべき問題を理解し規定すること
- 問題の複雑さを理解すること
- より小さい部分に分割していくこと
 - 基本アルゴリズムの使用

計算量の評価

- 問題を解くのに必要な演算の個数
- 問題のサイズ(N)の関数
 - 効率をサイズ(N)と必要な演算の個数の間の関係として表現
- 計算量を解析
 - N の変化で, 必要な計算時間にどのように影響するかをよく見通すことができる.

線形探索



線形探索

- 異なるN個あるデータから任意の1つを探す
- N回の予測

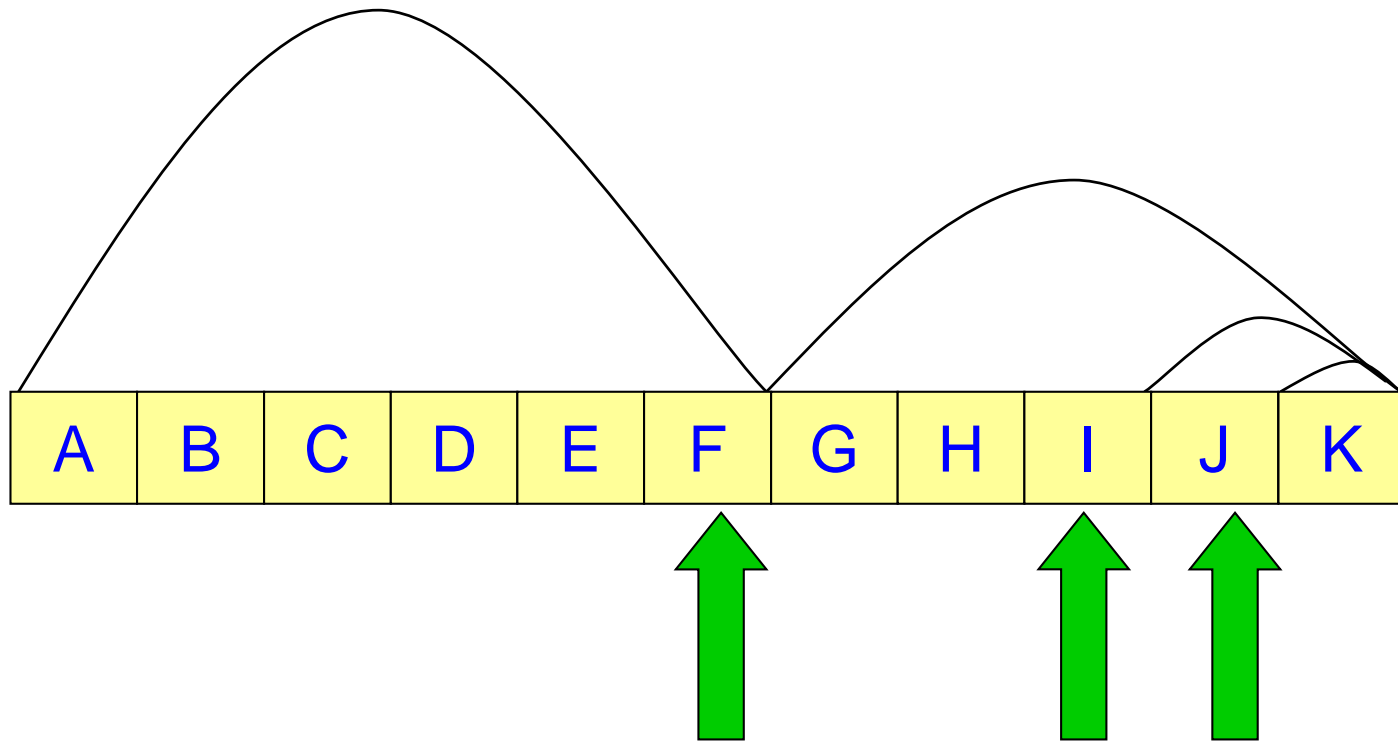
$$1 + 2 + \dots + N = \frac{N(N+1)}{2}$$

これをNで割ると

$$\text{平均の予測回数} = \frac{N+1}{2}$$

2分探索法

n要素の配列A: 整列されている.



2分探索の計算量

$$N / \underbrace{2 / 2 / 2 / \cdots / 2}_{k\text{個}} = 1$$

ここで、 k は必要な予測回数を表している。
これを簡単化すると

$$\frac{N}{2^k} = 1 \quad \text{または、} N = 2^k \rightarrow k = \log_2 N$$

2分探索の計算量

- 2分探索法で数を探すのに必要なステップ数
 - 最悪の場合: その個数の2を底とする対数
 - 平均の場合: 最悪の場合より1だけ小さい

アルゴリズムの実行時間

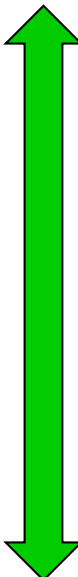
■ $O(f(N))$:

- 関数 $f(N)$ に対して、アルゴリズムの実行時間
- ある定数 C に対して、 N が十分に大きい限り、アルゴリズムの実行に必要な時間が

$$C \times f(N)$$

を超えないことをいう。

計算量の増加率(教科書101ページ)




計算量	Nが2倍になったとき の実行時間の変化	呼び名
$O(1)$	なし	定数
$O(\log N)$	小さい定数	対数
$O(N)$	2倍	線形
$O(N \log N)$	2倍と少し	$n \log n$
$O(N^2)$	4倍	2乗
$O(N^k)$	2^k 倍	多項式
$O(\alpha^N) \quad \alpha > 1$	α に依存するが 非常に大きい	指数

コンピュータ科学の限界

- 実現できれば有用な多くの計算が実現できない理由
- プログラムの実行時間があまりにも長すぎる.
 - 答えを出すのに1世紀もかかりそうなプログラムの場合
- 現在や将来可能な科学技術を駆使しても、解決できない計算不能という問題
- 問題を解決してくれるアルゴリズムが不明
 - その解決方法がわからない.
 - このような問題の多くは人工知能に関するもの.

計算の複雑さ

- 多項式時間： 計算容易
 - $f(n)$ の中の最大の要素が $n, n \log n, n^2, \dots$ のとき
- 指数的： 計算困難
 - $2^n, n!, n^n$ などのとき



関数	n =10	50	100	300

n	10	50	100	300
$n \log n$	33	282	665	2,469
n^2	100	2,500	10,000	90,000
n^3	1,000	125,000	10^6	2.7×10^7
2^n	1,024	$\sim 10^{25}$	$\sim 10^{30}$	$\sim 10^{91}$
$n!$	3.6×10^6	$\sim 10^{64}$	$\sim 10^{160}$	$\sim 10^{622}$
n^n	$\sim 10^{10}$	$\sim 10^{84}$	$\sim 10^{200}$	$\sim 10^{743}$

計算困難

プログラムの実行時間

$$t = 5.49 \times 10^{-3} \times 2^n$$



n	t (近似値)
5	0.17 秒
10	5.62 秒
15	3.00 分
20	1.60 時間
25	2.13 日
30	68.23 日
35	5.98 年
40	191.30 年
45	6120.94 年
50	195,870 年
55	6,267,740 年
60	200,571,000 年
65	6,418,270,000 年
70	205,385,000,000 年

不可能性：解くことのできない問題の存在

- チューリング機械で解けるアルゴリズムが存在しない問題
- チューリング問題の停止問題
 - あるプログラムが無限ループを回るかを判定し、回るか回らないかを有限時間で出力するようなプログラムは作ることはいできない。
- 解くアルゴリズムの無い問題
 - 2つのプログラムがすべての可能なデータに対して全く同じ動作をするかを判定する。

情報を扱う機械としてのコンピュータ

- 人間の脳とコンピュータ
 - 計算 計算の効率化
 - 記憶 データベース
 - 推論 論理
- コンピュータネットワーク
 - コンピュータと通信
 - インターネット