自己紹介

山﨑　秀記

1949年6月14日生まれ
放送大学客員教授

yamasaki.hideki@ouj.ac.jp
（質問・相談はメールで）

二日間、よろしくお願いします。

授業概要

　アルゴリズムは、問題解決のための手順・手続きを意味します。この授業では、実際に手を動かしながら、問題の解決手順を設計する際の基本的な概念や技法、課題を幅広く直感的に理解することを目指します。　アルゴリズムの説明は、実現の詳細に立ち入ることなく、できるだけ簡潔に日本語で記述します。したがって、特定のプログラミング言語の理解・習得を目指すものではありません。

受講者へのメッセージ

　この授業では数学的な内容を含みますが、文系高卒程度の知識を前提に講義します。ただし、学習を表面的な理解に留めずより深く自分のものとするために、授業中に提示する課題、パズル等に積極的に取り組み、コンピュータによる問題解決の手順について考えてみて下さい。
　プログラミングの知識は、理解には役立ちますが、必須ではありません。

レポート課題

興味を持ったパズルの答、この授業を通して学んだこと・考えたこと、授業への意見・感想等をレポートとして提出してください。

期日：授業終了後１週間以内
提出場所：多摩学習センター事務室に
　　窓口提出・郵送・メール添付（tama-sc@ouj.ac.jp）
　　山崎へ直接　メール添付（yamasaki.hideki@ouj.ac.jp）
形式：A4、用紙は何でもかまいません。メール添付はWORDあるいはPDFで。
表紙：タイトル：アルゴリズムの考え方レポート、学籍番号、氏名、提出日

１．講義概要

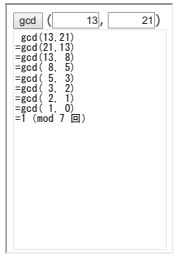
アルゴリズム：問題解決の手順

本講義では、アルゴリズムに関する様々なアイディアを紹介することを通して、コンピュータによる問題解決の基本的な考え方を身に付けることを目的としている。
今回は整列法を例に、アルゴリズムを比較・評価する際の尺度について見てゆく。

最古（？）のアルゴリズム：**ユークリッドの互除法**
正整数 m, n の最大公約数 $\gcd(m, n)$ を、

$$\gcd(m,n) = \begin{cases} \gcd(n, m \bmod n) \\ n \qquad\qquad\qquad m \bmod n = 0 \text{のとき} \end{cases}$$

を利用して求める。（$m \bmod n$ はmをnで割った余り）



パズル．ユークリッドの互除法で一番手間（ mod の計算回数）がかかる数値の組み合わせは、どのような場合か
ヒント．計算過程で常に $m \bmod n = m - n$ になる場合

整列法の実験
1. 10枚の紙に異なる2桁の数を書き、適当に並べる
2. 数の大きさの順に並べ変える
3. その時の方策を言葉にしてみよう

参考：ダンスで覚えるソートアルゴリズム

２．アルゴリズムの評価基準

アルゴリズムの評価基準
1. **計算時間**：出力を得るまでの時間
2. **記憶容量**：計算に必要なメモリー量
3. **分かり易さ**

整列アルゴリズムの例

評価基準	時間	記憶容量	分かり易さ	備考
選択ソート	×	○	◎	
クィックソート	◎	○	×	最速
挿入ソート	△	○	○	ほぼ整列データには速い
マージソート	○	△	○	外部記憶中の大量データに適用可能
基数ソート	○	△	△	データに制限有り

注．計算時間や記憶容量は入力のサイズに依存する

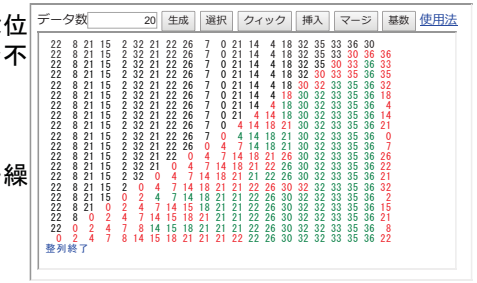
３．整列（ソート）法の色々

計算時間は比較回数（**赤色のデータ数**）に比例
選択ソート
「未整列部（赤）の最大値を右端と交換（青）」を繰り返す。
　比較回数：$n(n+1)/2$
クイックソート
「未整列部[赤]を右端の値を基準に、以下[赤]と以上[赤]に分割」を繰り返す
　比較回数：平均$n \log_2 n$に比例、最悪$n(n+1)/2$

パズル：計算時間（単位ミリ秒）を計測・予測せよ。

データ数	10000	20000	30000	1000000（予測）
選択ソート				
データ数	1000000	2000000	3000000	100000000（予測）
クイックソート				

挿入ソート
「仮整列部(赤と緑)の直前データを適切な位置に挿入する」を繰り返す。(緑は挿入で不動だった部分、右端の赤は番兵)
　比較回数：平均$n^2/4$
マージソート
「隣接する仮整列部([赤])を併合する」を繰り返す
　比較回数：$n \log_2 n$



パズル：計算時間(単位ミリ秒)を計測・予測せよ。

データ数	10000	20000	30000	1000000(予測)
挿入ソート				
データ数	1000000	2000000	3000000	100000000(予測)
マージソート				

パズル：次の方法で整列できる。確かめよ
1. 1桁目の値で分類し、順に並べる
2. 1.の順に、2桁目で分類し順に並べる
3. その繰り返し

バケツソート
各値の累積頻度を求め、整列後の位置を計算
値の範囲が小さいときに有効
計算時間：(データ数＋値の個数)に比例
基数ソート
桁数・文字列長が一定のデータに有効
下の桁から順にバケツソートを繰り返す
計算時間：(データ数＋値の個数)×桁数⇒桁数一定ならデータ数nに比例

パズル：計算時間(単位ミリ秒)を計測・予測せよ。

データ数	1000000	2000000	3000000	100000000(予測)
基数ソート				

4．オーダーの概念

アルゴリズムを実装したプログラムの**実際の計算時間は**
1. 使用するプログラミング言語によって異なる
2. 実行するコンピュータ(さらに実行時の環境)によっても異なる
3. 入力データのサイズによって異なる
4. 指標となる**適切な演算の実行回数に比例**する

計算時間の見積もり手順
　適切な演算を選ぶ
⇒ その実行回数を入力サイズの関数として表す
⇒ 小サイズのサンプル入力に対する計算時間を測る
⇒ 実際の入力(サイズ)での計算時間を見積もる

表 6.1　関数値とサイズ 10 の計算時間を単位（ミリ秒）とした計算時間

n	1	2	4	8	10	100	10^3	10^4
					1 ミリ秒	10 ミリ秒	0.1 秒	1 秒
n^2	1	4	16	64	100	10^4	10^6	10^8
					1 ミリ秒	0.1 秒	10 秒	16 分 40 秒
$\log_2 n$	0	1	2	3	3.322	6.644	9.966	13.288
					1 ミリ秒	2 ミリ秒	3 ミリ秒	4 ミリ秒
$n \log_2 n$	0	2	8	24	33.22	664.4	9966	132877
					1 ミリ秒	20 ミリ秒	0.3 秒	4 秒
2^n	2	4	16	256	1024	1.268×10^{30}	1.072×10^{301}	1.995×10^{3010}
					1 ミリ秒	3.9×10^{16} 年	—	—

計算時間の増え方や基準サイズの計算時間の何倍になるかを問題にする立場
　例えばサイズnの入力に対する演算回数が$2n^2+5n$であるとき
　　nがある程度大きくなると、$5n$の項はn^2に比べて無視できるし
　　n^2の係数2も無関係
　このとき、 $2n^2+5n$のオーダーはn^2であるという。
このように、(1変数)関数がいくつかの項の和で表されているときに、(変数の増加に伴う)増え方のもっとも大きな項を取り出し、その定数係数を無視したものを、その関数の**オーダー**という。

パズル．次の関数のオーダーを答よ。
（１）$3n^2+5$　　（２）$n \log_2 n+3n$　　（３）2^n+20n^3

パズル．表の空欄を埋めよ

関数	関数値				関数値					
					入力サイズ10で1ﾐﾘ秒としたときの計算時間					
n	1	2	4	8	10	20	30	100	1000	10000
					1ﾐﾘ秒	2ﾐﾘ秒	ﾐﾘ秒	10ﾐﾘ秒	秒	1秒
n^2	1	4	16	64	100		900		1000000	
					1ﾐﾘ秒	4ﾐﾘ秒	ﾐﾘ秒	0.1秒	秒	16分40秒
$\log_2 n$	0	1	2	3	3.322	4.322	4.907	6.644	9.966	13.288
					1ﾐﾘ秒	1.30ﾐﾘ秒	1.48ﾐﾘ秒	2ﾐﾘ秒	ﾐﾘ秒	4ﾐﾘ秒
$n \log_2 n$	0	2	8	24	33.22		147.2	664.4	9966	132877
					1ﾐﾘ秒	2.60ﾐﾘ秒	4.44ﾐﾘ秒	ﾐﾘ秒	0.3秒	4秒
2^n	2	4	16	256	1024	1048576	10億7374万	1.268×10^{30}	1.072×10^{301}	1.995×10^{3010}
					1ﾐﾘ秒	秒	17分49秒	3.9×10^{16}年	・・・	・・・

５．整列アルゴリズムの計算量

nはデータ数

	時間	記憶容量	備考
選択ソート	n^2	n	
クィックソート	$n \log n$	n	最速
挿入ソート	n^2	n	ほぼ整列データには速い
マージソート	$n \log n$	$2n$または1	外部記憶中のデータに適用する場合の記憶容量は定数(1)
基数ソート	nk	$n+m$	kはデータの桁数。mはバケツの個数

６．ベキ乗計算アルゴリズム

y^xの素朴な計算法 $y^2, y^3, \ldots, y^x$とyを順に掛ける
$x-1$回の掛算⇒オーダーxの計算時間⇒$10^{x\text{の桁数}}$の計算時間
桁数が1増えると10倍の時間（入力xのサイズはxの桁数）

Binary法 xの2進表示と、$y^1, y^2, y^4, y^8, \ldots$の値を利用する
例．22は2進5桁の10110なので$y^{22} = y^{16+4+2} = y^{16} y^4 y^2$。
y^2, y^4, y^8, y^{16}の計算（2乗の掛算4回）と$y^{16} y^4 y^2$の計算（掛算2回）で掛算6回
xが2進k桁の解き $\begin{cases} y^2, y^4, \ldots, y^{2^{k-1}} \text{の系列の計算に} k-1\text{回} \\ \text{組み合わせて} y^x \text{を計算するのに(最大)} k-1\text{回} \end{cases}$ 計$2k-2$回
オーダーはxの2進桁数（$\log_2 x$）：素朴法に比べ圧倒的に速い（オーダーの表）

パズル．MOD電卓（右上）でy^xの計算時間に関する次の問に答えよ（yの値は任意）

1. $x = 10^6, 10^7, 10^8$に対し素朴法【y^x】とBinary法【速y^x】の計算時間を調べよ。
2. $x = 10^{15}$に対する【y^x】の計算にはおおよそ何日かかるか。
   ヒント：$x = 10^8$乗に1秒かかれば$x = 10^{15}$乗には10^7秒$/60/60/24 \simeq 116$日。

１．辞書クラス

辞書クラスの様々な実現法を通し、アルゴリズムの応用のされかたを見てゆこう。

クラス：型（タイプ）
　メソッドとプロパティを持つ
オブジェクト：その型のデータ
　プロパティの値を定めたもの

住所録　【番号】がキー項目

項目名	【番号】	【氏名】	【住所】
レコード	001	青森健太	青森県青森市…
	020	宮城遼	宮城県仙台市…
	017	長野愛子	長野県長野市…
	…	…	…

レコードクラス
　プロパティ：キー(値はレコード毎に一意で重複しない)、項目リスト
　メソッド：get(項目名)、set(項目名)

辞書クラス
　プロパティ：キー項目を持つレコードの集合
　メソッド：探索(キー)、追加（レコード）、削除（キー）

辞書の基本メソッド（レコードの探索・追加・削除）を効率よく行うために、様々な編成法が考案されている。次表は、これから学ぶ基本的な編成法の計算時間(ステップ数)である。これを見ると直接編成が一番効率が良いが、後で見るように、特別な条件のもとでしか成立しない。

（ ）内はレコード数$N = 1,000,000$の時の平均ステップ数

編成法	探索	追加	削除	備考
順編成	$N/2$ (500, 000)	$N/2+1$ (500, 001)	$N/2+1$ (500, 001)	1は定数ステップの意味 500, 001の1にあまり意味はない
整列順編成	$\log_2 N$ (20)	$\log_2 N + N/2$ (500, 020)	$\log_2 N + N/2$ (500, 020)	
直接編成	1	1	1	衝突が生じない場合
索引順編成	$1 + N/M/2$ (501)	$1 + N/M/2$ (501)	$1 + N/M/2$ (501)	ブロック数$M = 1,000$とした。
2分探索木	$\log_2 N$ (20)	$\log_2 N+1$ (20+1)	$\log_2 N+1$ (20+1)	2分探索と1ステップの追加削除が可能なデータ構造

２．順編成

レコードを単に1列にならべて管理

順探索：頭から順に探す
追加：無ければ列の最後に追加
削除：あれば削除し、最終レコードをそこに移す

注．デモで、操作のステップ数は赤字の個数に等しい

パズル．レコード数Nの順編成辞書について
探索、追加、削除にかかる(平均あるいは最悪の)ステップ数を示し、理由を説明せよ。

３．整列順編成

レコードをキーで整列して管理

2分探索：探索範囲の中央値(赤)と比べて、「小さければ前半を、大きければ後半を探索範囲(青)とする」を繰り返す
追加：　無ければ挿入する(大きなレコードを後にずらす)
削除：　あれば削除し、後のレコードを順に詰める

注．デモで、操作のステップ数は赤字の個数に等しい

パズル．レコード数Nの整列順編成辞書について
探索、追加、削除にかかる(平均あるいは最悪の)ステップ数を示し、理由を説明せよ。

４．直接編成

キー値からレコード(の格納)番地を直接計算するハッシュ関数を利用
　　キーの値 ⇒【ハッシュ関数】⇒ レコードの格納位置
ハッシュ関数があれば、探索、追加、削除は、１(定数)ステップ
例．学生レコードの辞書
　2016年入学生(300名以下)の学籍番号が2016001～2016300ならば、ハッシュ関数を$h(k) = k - 2016000$とし、レコードを1～300の番地(位置)に置く。

$$\text{番地} = h(\text{学籍番号}) = \text{学籍番号} - 2016000$$

番地	学籍番号	氏名	住所
1	2016001	青森健太	青森県青森市…
2	2016002	宮城遼	宮城県仙台市…
3			
4	2016004	長野愛子	長野県長野市…
…	…	…	…

適当なハッシュ関数があるとは限らない
通常、キー値の候補数はレコード(格納番地)数に比べ通常莫大
例．10文字の英数字$(26 + 10 = 36$通り)からなるユーザIDをキーとする
　　キーの候補数は$36^{10} \simeq 3656$兆で、ユーザ数が3656人ならその1兆倍

通常、ハッシュ関数は、異なるキー(の候補値)に対し、同じ値を割り当てる
⇒存在するレコード(のキー値)にハッシュ関数が同じ値を割り当てる（衝突）
⇒直接編成法は使えない⇒解決策(の1つ)が、索引順編成

コラム．ハッシュという言葉の意味

ハッシュ(hash)は、本来(肉を)切り刻むという意味で、ハヤシライスのハヤシ(hashed beef)の語源でもある。ハッシュ関数は、衝突をさけるため、単純な関数ではなく、もとのキー値を分解した(切り刻んだ)各部分が関数値に影響するように作る必要があることから付けられた名前である。

５．索引順編成

直接編成での衝突の解決策
レコードの格納領域は、ハッシュ値ごとのブロックに分けられ(索引部)、ブロック内でレコードは順編成

ブロックの探索：直接探索
ブロック内の探索、追加、削除：順編成の探索、追加、削除

右上のデモでは簡単のため、5で割った余りをハッシュ値にしている。

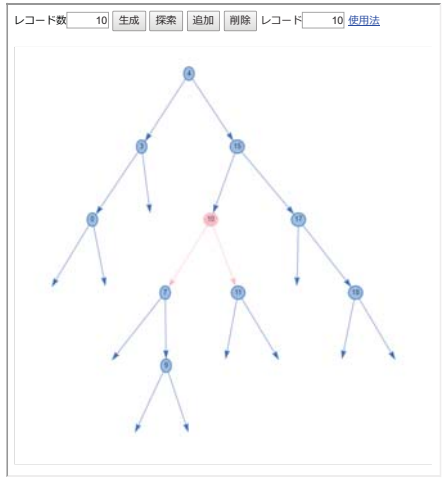
パズル．レコード数N、ブロック数M の索引順編成辞書について

探索、追加、削除にかかる(平均あるいは最悪の)ステップ数を示し、理由を説明せよ。

６．２分探索木

2分探索と、探索後に定数ステップでの追加・削除が可能なデータ構造
各節点は左子と右子を持ち
　左子の値＜節点の値＜右子の値

探索：木の根(最上節点)から始めて
　節点の値より小⇒左部分木を探す
　節点の値より大⇒右部分木を探す
を繰り返す
追加：探索で場所を見つけそこに追加
削除：削除節点nを探索。nに
　左子がない⇒右子をnの位置に移動
　左子がある⇒
　　左部分木中の最大値をnに代入し
　　その最大値節点を削除(左子を移動)



パズル．レコード数N の２文探索木について
探索、追加、削除にかかる平均のステップ数を示し、理由を説明せよ。

２分探索木は、
　平均的には効率が良いが
　レコードの挿入・削除の順番によっては木のバランスが崩れ
　最悪の場合、探索・追加・削除にオーダー N の時間がかかるため
木の平衡を保ち、常にオーダー $\log N$ にするための様々な工夫がなされている。

アルゴリズムの設計指針

アルゴリズムを設計する際の様々な指針について紹介する。

強欲アルゴリズム
　単一の戦略で解く

分割統治法
　問題を小さく分割してアタック

再帰法
　小サイズの問題の解を仮定して解く（数学的帰納法）

動的計画法
　部分問題の解を保存して利用

試行錯誤法
　しらみつぶしに調べる

１．強欲アルゴリズム

単一の戦略で、局面ごとの最適手(解)が選択できるような問題に適用

例．コインの支払い問題：x円を支払う最少枚数の硬貨を求めよ
100円、50円、10円、5円、1円の各硬貨が（十分な枚数）ある。
戦略：高い硬貨から順に、その硬貨で支払えるだけ支払う（$\div$は切り捨て、modは割った余り）
①100円硬貨で$(x \div 100)$枚支払い残額をxとする（$x \leftarrow x \mod 100$）
② 50円硬貨で$(x \div 50)$枚支払い残額をxとする（$x \leftarrow x \mod 50$）
③ 10円硬貨で$(x \div 10)$枚支払い残額をxとする（$x \leftarrow x \mod 10$）
④ 5円硬貨で$(x \div 5)$枚支払い残額をxとする（$x \leftarrow x \mod 5$）
⑤ 1円硬貨でx枚支払う

40円硬貨があると、強欲アルゴリズムは適用できない。
パズル：50円、40円、10円硬貨で80円支払う最少枚数は？
1. 強欲アルゴリズムが与える解をもとめよ
2. 最少枚数解を求めよ

２．分割統治法

問題を
・互に独立な部分問題に分割し
・それらの解を組み合わせて
もとの問題を解く

全体問題を解く：主プログラム、関数 main
部分問題を解く：サブルーチン、手続き、関数

ライブラリ：共通して使われる部分問題アルゴリズム集
　整列アルゴリズム、辞書クラス

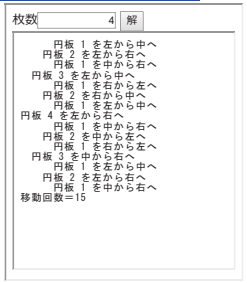
３．再帰法

分割統治法で、部分問題に、元の問題と同じでサイズの小さな問題が現れるとき
問題の解が(分からなくても)あると仮定して解ける（数学的帰納法、再帰呼出し）

例．ハノイの塔：右図のように置かれたn枚の円板を左端の杭から右端の杭に移す。その際、
　①移動は杭から杭へ一枚ずつ
　②円板をより小さな円板の上には置けない
解．円盤が0枚$(n = 0)$なら何もしなくてよい
$n-1$枚の円板が移せると仮定して
　①上の$n-1$枚の円板を左端から中央に移す
　②最大(一番下)の円板を左端から右端に移す
　③中央の$n-1$枚の円板を右端に移す

（ウィキペディアより）

パズル．$n = 4$のハノイの塔を手を動かして解いてみよう。



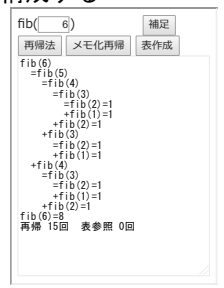
整列アルゴリズムの再考
基数法以外を再帰的観点から、
問題分割と解の統合の手法を整理
（桃色は手間をかけている部分）
計算時間は、分割のサイズによる
　1つと残り⇒n^2、均等⇒$n \log_2 n$
再帰アルゴリズムでは、
　部分問題のサイズ均等化⇒効率化

アルゴリズム	問題分割	解の統合	時間オーダー
選択ソート	最大値と残り	不要	n^2
クイックソート	基準値以下と以上	不要	平均$n \log_2 n$
挿入ソート	１個と残り	挿入（併合）	n^2
マージソート	半分ずつ	併合	$n \log_2 n$

注．挿入・マージソートの再帰版は、デモの最終ステップを考えると分かりやすい

動的計画法（ダイナミックプログラミング）

再帰法で同じ部分解を繰り返し必要とする場合に、部分解の表を作り再計算をさける
・メモ化再帰：再帰呼び出しで一度計算した部分解を表に保持する
・表作成：必要になりそうな部分解を（小さい方から順に）すべて構成する



例．フィボナッチ数列　$1, 1, 2, 3, 5, 8, 13, 21, \ldots$

$$f_{ib}(n) = \begin{cases} 1 & n < 3 \\ f_{ib}(n-1) + f_{ib}(n-2) & \text{それ以外} \end{cases}$$

表作成：$f_{ib}(1), f_{ib}(2), \ldots, f_{ib}(n)$の順に求める

パズル．［再帰法］計算の再帰回数と$f_{ib}(n)$の値との関係を求めよ。
（［再帰法］で大きなnの$f_{ib}(n)$を計算してはいけない！）

コラム．

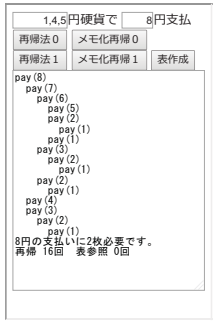
1. $f_{ib}(n) = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^n = \left[\frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^n\right] = \left[\frac{(1.618\cdots)^n}{2.236\cdots}\right]$
   （[]は小数点以下四捨五入）という式でも計算できる。ただし計算誤差が生じ$f_{ib}(76)$の［補足］は、［表作成］や［メモ化再帰］の正しい値からずれる。
2. ユークリッドの互除法は高速：最も手間がかかるのは次の場合で、 mod がn回必要になる
   $\gcd(f_{ib}(n), f_{ib}(n+1))$
   $= \gcd(f_{ib}(n+1), f_{ib}(n))$
   $= \gcd(f_{ib}(n), f_{ib}(n-1))$
   $\cdots$
   $= \gcd(f_{ib}(3), f_{ib}(2))$　$(= \gcd(2,1)\ )$
   $= \gcd(1,0) = 1$
   つまり、互除法による$\gcd(j,k)$の最悪計算時間は $\log_{1.618}(\min(j,k))$ に比例する

例．　一般的なコインの支払い問題
$p_{ay}(x)$：x円の最少支払枚数、支払えない場合は無限大
再帰式０：最少支払いを分割した各々も最少支払なので

$$p_{ay}(x) = \begin{cases} 1 & x\text{円硬貨がある} \\ \min\limits_{0<y<x}\left(p_{ay}(y) + p_{ay}(x-y)\right) & \text{それ以外} \end{cases}$$

再帰式１：最少支払から硬貨を1枚除いたものも最少支払なので

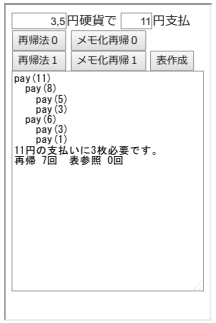
$$p_{ay}(x) = \begin{cases} 1 & x\text{円硬貨がある} \\ \min\limits_{y\text{円硬貨}}\left(p_{ay}(x-y) + 1\right) & \text{それ以外} \end{cases}$$



表作成法における工夫：「使える硬貨の種類を1つずつ増やす」
　すべてのxに対し、$g(x) \leftarrow \infty$
　各y円硬貨について
　　$p_{ay}(x) \leftarrow \min(p_{ay}(x), p_{ay}(x-y)+1)$
　に基づき、$p_{ay}(x)$の表を計算しなおす

メモ化再帰と表作成法

パズル. 3,5円硬貨で11円支払う場合をアプリで求め、その計算過程を調べてみよう。



(メモ化)再帰法：
　部分問題への分割ができれば、直接的な解法の記述は不必要
　再帰呼び出しには、それなりの負荷がかかる。
　再帰プログラムを自動的にメモ化して実行するシステムもある。
表作成法：
　直接的な解法の記述が必要。
　一般に高速
　不要な値を計算する可能性がある。

遅延評価
　式の評価を、実際に必要になるまで遅らせる手法や実装。

試行錯誤法

解の**各候補** x
　xが解ならば x を出力(して終了)

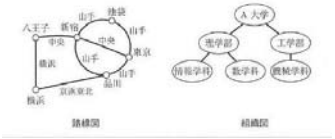
候補を、途中解を伸ばしていって構成する（迷路等）⇒再帰法
　Solve(x)
　　x が解なら出力(して終了)
　　x からの**各伸長候補** w
　　Solve(xw)
と定義して Solve(空) とする。
再帰呼び出しの実現法により、この手法が行うのは
　途中解を伸ばせるだけ伸ばして失敗したら戻る、解空間の深さ優先探索

伸長候補の順位付け：評価関数
・伸長しても解が得られないことがあらかじめ分かれば、候補に入れない
・解が得られそうな候補から順に調べる

グラフとは

多くの問題が定式化(表現)可能

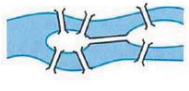
頂点（○や□で表す）を辺で結んだもの。
　有向グラフ：有向辺（→）
　無向グラフ：無向辺（—）
必要に応じて、頂点や辺にラベル

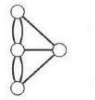


地下鉄路線図　放送大学組織図　ディズニーが描いた戦略図

グラフ理論の始まり

ケーニヒスベルグの橋の問題
７つの橋を一度ずつ通る散歩道は？





中央大学 山本慎教授

オイラーは、グラフの一筆書きの問題に翻訳し、不可能であることを示した。
路：つながった辺の列
閉路：出発点に戻る路
オイラー路：すべての辺を1度ずつ通る路(一筆書き)
オイラー閉路：出発点に戻るオイラー路

ウィキペディア

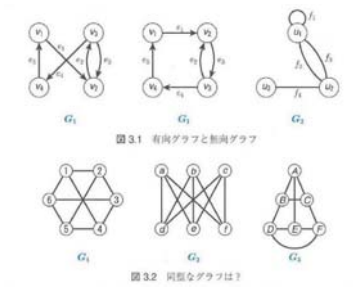
パズル：ケーニヒスベルグの7つの橋を一度ずつ通る路はあるか。

同型なグラフ

同じグラフ：頂点の位置を適当に動かすと重なる（頂点の位置は無視）

同型なグラフ：頂点の名前の付け替えで同じグラフと見なせる

パズル：図３．２ のG_1とG_2が同型であり、G_3がそれらと同形でないことを示せ。



頂点探索アルゴリズム

定められた始点から到達可能な全頂点を
一定の規則にしたがって訪問し処理する

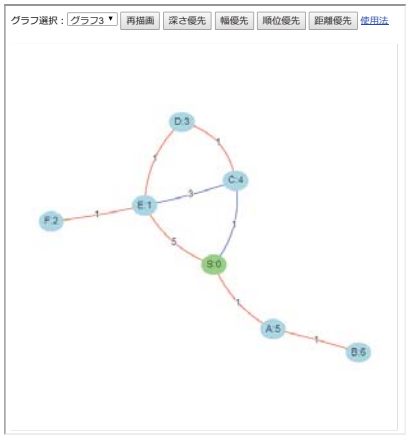
頂点探索アルゴリズムの一般形
縁←{(*始点*, *始点*)}
*縁*が空でない間
　辺(u, v)を*縁*から取り出す
　vが未訪問なら
　　vを訪問し処理を行う
　　vの各辺(v, w)を*縁*に加える

縁からの辺取り出し方による分類
　深さ優先探索：後に追加された辺を先に（LIFO）
　幅優先探索　：先に追加された辺を先に（FIFO）
　最良優先探索：順位の高い辺を先に

深さ優先探索

始点から到達可能な全頂点を深さ優先で訪問し処理する
縁をスタック（プッシュとポップ）で実現
　プッシュ：列の後尾に追加
　ポップ：列の後尾から取出し

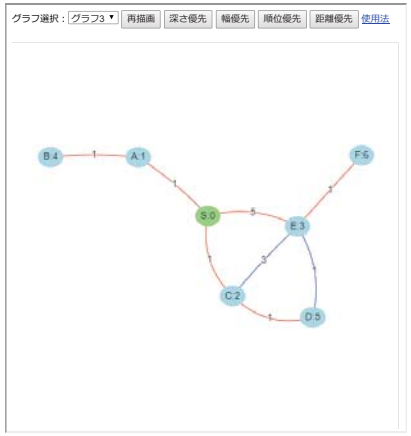
深さ優先探索アルゴリズム
縁←[(*始点*, *始点*)]
*縁*が空でない間
　辺(u, v)を*縁*から**ポップ**
　vが未訪問なら
　　vを訪問し処理を行う
　　vの各辺(v, w)を*縁*にプッシュ



幅優先探索

始点から到達可能な全頂点を幅優先
（近いもの順）で訪問し処理する
縁をキュー（プッシュとシフト）で実現
　プッシュ：列の後尾に追加
　シフト：列の先頭から取出し

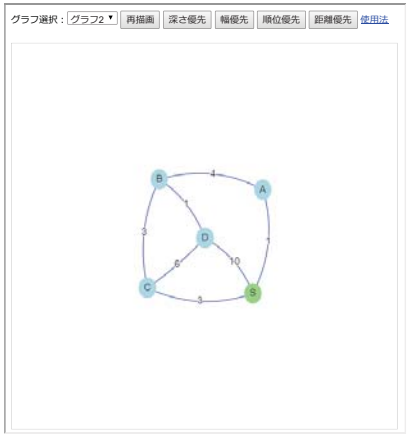
幅優先探索アルゴリズム
縁←[(*始点*, *始点*)]
*縁*が空でない間
　辺(u, v)を*縁*から**シフト**
　vが未訪問なら
　　vを訪問し処理を行う
　　vの各辺(v, w)を*縁*にプッシュ



最良（順位）優先探索

始点から到達可能な全頂点を**優先順**に訪問し処理する
縁を順位キュー（プッシュと最良取出し）で実現
　プッシュ：列に追加
　最良取出し：列中の最良要素を取出し

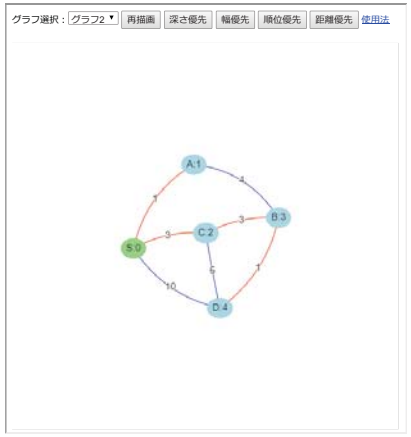
最良優先探索アルゴリズム
縁←{(*始点*, *始点*, 順位1)}
*縁*が空でない間
　辺$(u, v,$ 順位$)$を*縁*から**最良取出し**
　vが未訪問なら
　　vを訪問し処理を行う
　　vの各辺$(v, w,$ 順位$)$を*縁*にプッシュ



辺長優先探索：プラムのアルゴリズム

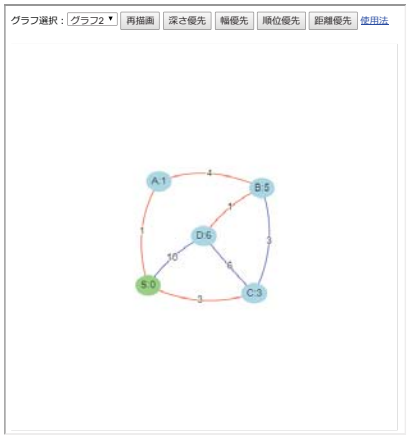
始点から到達可能な全頂点に至る重み最小の辺集合
（最小全域木）を求める
プラムのアルゴリズム
縁←{(*始点*, *始点*, 0)}；*木*←空
*縁*が空でない間
　辺$(u, v,$ 長さ$d)$を*縁*から**dで最良取出し**
　vが未訪問なら
　　vを訪問済みにし辺(u, v)を*木*に追加
　　vの各辺$(v, w,$ 長さ$)$を*縁*にプッシュ

縁に対する操作（最良取出し、プッシュ）を効率的に
実現するデータ構造
　2分探索木
　ヒープ



最短距離探索：ダイクストラのアルゴリズム

始点からの最短距離・経路を求める



ダイクストラのアルゴリズム
縁←{(*始点*, *始点*, 距離0)}；*路*←空
*縁*が空でない間
　辺(u, v, d)を*縁*から**距離d優先取出し**
　vが未訪問なら
　　vの距離をd(訪問済み)とし
　　(u, v)を*路*に追加
　　vの各辺(v, w, l)に対し
　　　*縁*に$(v, w, d + l)$をプッシュ

縁に対する操作（最良取出し、プッシュ）を効率的に実現するデータ構造
　2分探索木
　ヒープ

前処理とは

あらかじめデータに前処理を施すことによって、処理本体(時間)を効率化する
注意．処理本体が重くなければ、前処理の手間が負担になることがある

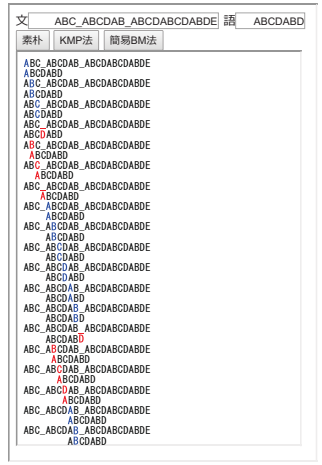
yamasa 例．多項式の反復計算：ホーナー法
$f(x)=5x^3+2x^2+6x+7$の計算：掛け算$3+2+1=6$回、足し算3回
$f(x)=((5x+2)x+6)x+7$の計算：掛け算3回、足し算3回
$f(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$の計算
　掛け算$\sum_{k=0}^{n}k=\frac{n(n+1)}{2}$回、足し算$n$回
$f(x)=(\cdots((a_nx+a_{n-1})x+a_{n-2})x+\cdots+a_1)x+a_0$の計算
　掛け算n回、足し算n回
　　$y\leftarrow a_n$
　　$i=n-1,n-2,\cdots,0$ に対し　$y\leftarrow y\times x+a_i$

テキスト探索

指定された文(テキスト)中から、指定された語の位置を求める。
インターネット検索の基本機能

素朴な方法（文や語は第0文字から始まる）
　$i\leftarrow 0;\ j\leftarrow 0;$
①語の第j文字を文の第i文字と照合する
　　一致すればi,jを共に1増やす
　　　語の最後にきたら成功。$i-j$を返して終了
　　しなければ再照合：$i\leftarrow i-j+1,\ j\leftarrow 0;$とし①へ

パズル．最悪の振舞をする次の例における文字の比較回数求めよ
文：$aaaaaaaaaaaaaaaaaaaaaaaab=a^{24}b$
語：$aaaab=a^4b$



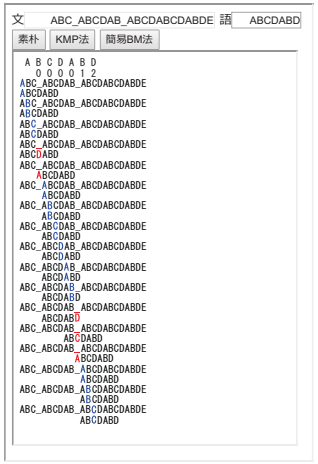
参考

KMP法：Knuth, Morris, Pratt

素朴な方法を改善

KMP法（文や語は第0文字から始まる）
　$i\leftarrow 0;\ j\leftarrow 0;$
①語の第j文字を文の第i文字と比較する
　　一致すればi,jを共に1増やす
　　　語の最後にきたら成功。$i-j$を返して終了
　　しなければ再照合：($j=0$ならiを1増やし)
　　　$j\leftarrow f(j);$とし①へ

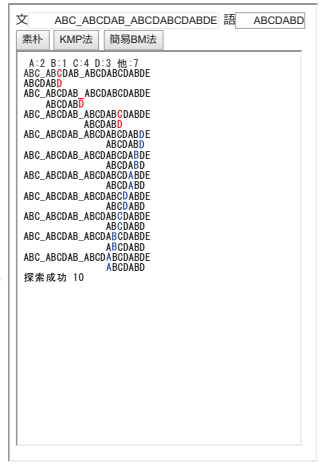
アイディア：文の第i文字と語の第j文字で不一致が起きた
⇒文の第$(i-j)$~$(i-1)$文字＝語の第0~$(j-1)$文字
⇒その部分の再照合結果は、語の情報だけから分かる
⇒語中の照合再開位置$f(j)$を前もって(文と無関係に)計算できる（筈！）



BM法：Boyer, Moorによる

簡易BM法（語の最終文字位置＝長さ-1をnとする）
　$i\leftarrow n;\ j\leftarrow n;$
①語の第j文字を文の第i文字と比較する
　　一致すればi,jを共に1減らす
　　　語の先頭にきたら成功。iを返して終了
　　しなければ再照合：
　　　$i\leftarrow$パターン右端$+\,g($文の第i文字$);$
　　　$j\leftarrow n;$とし①へ

関数$g(c)$の値は語中の文字cの最右出現位置から求めることができる
本来のBM法は、再照合位置を求める二つの表(関数)を利用
- $f($文字$)=$語中の文字の最小出現位置
- KMP法のアイディア



正規（正則）表現とは

正規表現は、文字列パターンの記述であり文字列wは正規表現αに合致するという。
正規表現による検索・置換機能が実装されているプログラミング言語も多い
∵）正規表現と後述の有限オートマトン（有限状態機械）は識別能力が同等

基本の表現

記法	説明
文字列	文字列や空列はそのまま文字列（空列）と合致。
連接	正規表現を続けて書けば、それらを続けた文字列と合致
|	和(または)。どちらかのパターンと合致すれば合致。
*	直前の文字(文字列)の0回以上の繰り返しと合致

例. (0|1)*01 は
0 または[|] 1 を 0回以上つなげた[*] 後に 01 をつなげた文字列
即ち、01 で終わる 0,1 の文字列を表す。

さらに次節で述べるような多くの便宜的記法がある。

正規表現の記法

繰返し

記法	説明	使用例
+	1回以上の繰り返し	12+ → 12、122、・・・ と合致 12+=122*
?	0,1回の繰り返し	12? → 1、12 と合致 12?=1|12
{n}	n回繰り返し	(ab){3} → abababに合致
{n, }	n回以上繰り返し	a{3,} →aaa、aaaa、…に合致
{m,n}	m〜n回繰り返し	a{3,5} →aaa、aaaa、aaaaaに合致

正規表現の記法

文字の集合：1文字指定する

記法	説明	使用例
.	改行文字を除く任意の1文字	windows.. windowsの後に2文字続く文字列と合致
[ ][ ]	内に並べた文字のどれか	[ab]＝a|b
		[a-z]＝a|b|・・・|z 小文字アルファベット全て
-	文字の区間。[ ] 内で用いる	[0-9]＝0|1|・・・|9 全角の数字
		[0-9A-Z] 大文字のアルファベットか数字
^	否定。[ ] 内で用いる	[^a-z] 小文字のアルファベット以外全て

合致する文字列は、通常はパターンに適合する最長の文字列(最長合致)である
が、"?"を付加すれば最短合致となる。
12+ →1222222222 に対し10桁目の1222222222までと合致
12+? →1222222222 に対し2桁目の12までと合致

正規表現の記法

特殊な意味を持つ表現

記法	説明	使用例
^	行頭	^abc →行頭にある abc と合致
$	行末	abc$ →行末にある abc と合致

記法	説明	記法	説明
¥w	アルファベット、数字又は下線	¥W	アルファベット、数字、下線以外
¥d	数字	¥D	数字以外。[^0-9] と同じ
¥s	空白文字(スペース、タブ、改行)	¥S	空白文字以外
¥b	単語の区切り	¥B	単語の区切り以外
¥n	改行		
¥r	リターン(復帰)		
¥t	タブ		

正規表現による英文分析

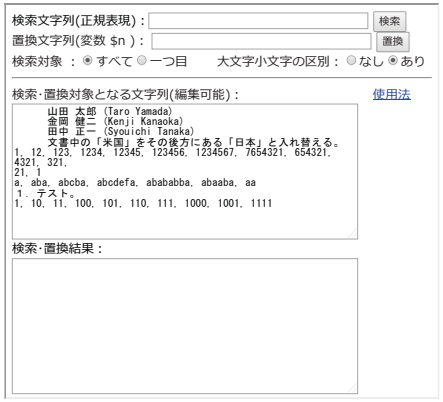
正規表現による検索では grep が有名だが、ここでは英文の分析に利用する例を紹介しよう。これによって、文書・文献の特徴を数量的に抽出し、著者の異同や時代の推定、思想と文体との関係等を分析することも可能になる。その意味で、正規表現検索は文体の処理・解析のための重要なツールになっている。

実験. 秋田大学のSCORP検索(http://www.mis.med.akita-u.ac.jp/%7Etaka/FreeSCORP/)
1. Search Word欄に色々な正規表現を入れて試してみよう。
2. 適当な動詞(過去、過去分詞、三単元を含む)の出現数を調べてみよう。

パズル. 正規表現 ¥s(for|in|on|to|with)¥s[a-z]+ing¥s(for|in|of|on|to|with)¥s は何を意味するか考えよ。

正規表現による検索と置換

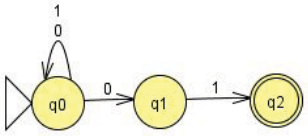
$n：正規表現の$n$番目の括弧対で括られた部分に合致した文字列
右のフレームで検索文字列や置換文字列を指定して[検索]、[置換]を押すと結果が下の欄に表示される（Java Scriptで実装）



パズル. 1〜5の正規表現を求め検索・置換せよ。
1. 3桁の数字
2. 3000未満の非負整数
3. aで始まりaで終わる6文字以下の英小文字列
4. 全角数字０～９で始まり句点。で終わる行
5. 英字の姓と名を入れ替よ

有限オートマトン（FA）

Σ上の有限オートマトン（FA）M
状態（○）と遷移（ラベル付き矢印 $\xrightarrow{a}$, $a \in \Sigma^?$）
入力a（εの場合は入力なし）による状態遷移
開始状態(▷○、1つ)と、受理状態(◎、複数可)
開始状態から受理状態への遷移列（路）の入力（ラベル）列を受理
受理入力列の集合を認識(受理)。



時点表示：(状態, 未読文字列)

$(q_0,1001) \vdash (q_0,001) \vdash (q_0,01) \vdash (q_1,1) \vdash (q_2,\varepsilon) \Rightarrow (q_0,1001) \overset{*}{\vdash} (q_2,\varepsilon)$：受理

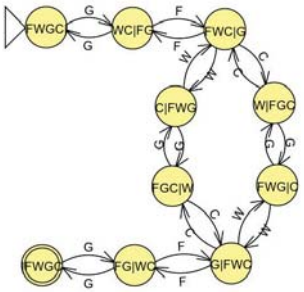
$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{0} q_1 \xrightarrow{1} q_2$を$q_0 \xrightarrow{1001} q_2$と書けば $(q,w) \overset{*}{\vdash} (p,\varepsilon) \Leftrightarrow q \xrightarrow{w} p$。

FAMが認識する言語：$L(M) = \{w \mid (\triangleright\bigcirc, w) \overset{*}{\vdash} (◎, \varepsilon)\} = \{w \mid \triangleright\bigcirc \xrightarrow{w} ◎\}$

パズル. 右上図のFAについて
①1001に対する可能な動作をすべて調べよ
②どのような文字列を受理するか

農夫パズル

オオカミ(W)とヤギ(G)を連れキャベツの籠(C)を持った農夫(F)が小舟で川を渡りたい。
1. 小舟には農夫の他一匹(一籠)しか乗らない
2. 農夫がいないと、オオカミはヤギを食べ
ヤギはキャベツを食べる
無事川を渡るにはどうすればよいか。



状態：両岸にいる(ある)ものの配置
遷移：農夫が小舟で運ぶもの W, G, C。なければ F
開始状態：▷(FWGC)　受理状態：((FWGC))

パズル. 農法パズルの解を２つ示せ。

3で割り切れる2進数

3で割り切れる2進数を受理する有限オートマトンを考えてみよう。
基本的なアイディアは
　開始状態から$0,1$の列wで状態q_kに到達⇔2進数wの値を3で割った余りがk
となるようにすることである。
2進数wの値を$[w]_2$で表すと、$[w0]_2 = 2[w]_2$, $[w1]_2 = 2[w]_2 + 1$である。

パズル．上のアイディアに沿って、有限オートマトンを描け。
　答　JFLAPファイル

正規集合⇒有限オートマトン

次の手順で有限オートマトンを構成する。
- 与えられた文字列を受理する有限オートマトンを作る
- 2つの有限オートマトンから、それらが認識する文字列集合の連接を認識する有限オートマトンを作る
- 2つの有限オートマトンから、それらが認識する文字列集合の和集合を認識する有限オートマトンを作る
- 有限オートマトンから、それが認識する文字列集合の閉包を認識する有限オートマトンを作る

パズル：(a∪b*c)*cを認識する有限オートマトンを示せ。
ヒント：有限オートマトンは開始状態と受理状態がそれぞれ1つで、開始状態への遷移と、受理状態からの遷移はないものを考えよう。

有限オートマトンと正規集合

定理．次の条件は同値⇒正規集合
1. 有限オートマトンで受理される
2. 決定性有限オートマトンで受理される
3. 正規表現で表される

定理．A,BがΣ上の正規集合ならば、A∪B, A∩B, Σ*-Aも正規集合である。
略証．A∪Bも正規集合なのは明らか。Aを認識する決定性有限オートマトンの受理状態と非受理状態を入れ替えた決定性有限オートマトンはΣ*-Aを認識する（Σ*はΣ上の文字列の全体を表す）。

パズル．文字列の集合Aに対し、A^RをAの文字列を逆順にした文字列の集合とする。Aが正規集合ならA^Rも正規集合であることを示せ。

決定性有限オートマトン（DFA）

辺のラベル（文字）の集合をΣとする。
有限オートマトンは次の条件を満たすとき決定性であるという。
　ε遷移を持たず、すべての状態とラベル($\in \Sigma$)の組に対して、遷移先の状態が1つ

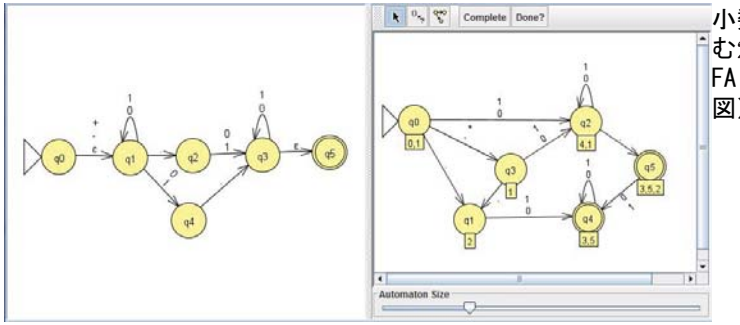
DFA(Deterministic Finite Automaton)では、
　開始状態から与えられた入力(ラベル列)を持つ路は唯一つ(ある)
　入力は、到達先の状態が受理状態なら受理、受理状態でなければ非受理
FAでは、
　開始状態から与えられた入力(ラベル列)を持つ路は複数(または0個ある)
　入力は、到達可能な状態の中に受理状態があれば受理、なければ非受理

注．状態とラベル($\in \Sigma$)の組に対し複数の遷移を許すものをNFA(非決定性有限オートマトン, Nondeterministic Finite Automaton)、さらにεラベルの遷移を許すものをε-NFAと区別することもある。この授業でのFAはε-NFA。

FA→DFAの変換

定理．FAMで認識可能⇒DFAM_Dで認識可能
略証．M_Dを次のように作る。

- M_Dの入力wで到達する状態：＝Mのwで到達可能な状態の集合
- M_Dの開始状態：＝Mの開始状態からεで到達可能な状態の集合
- M_Dの受理状態：＝Mの受理状態を含む状態

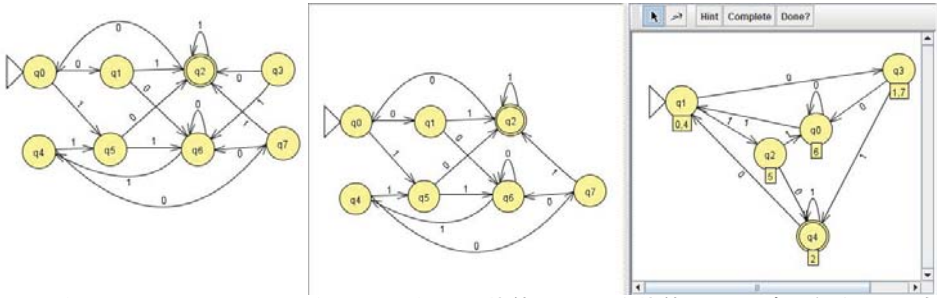


小数点（符号）を含む2進数を認識するFA（左図）とDFA（右図）

DFAの最小化

DFAの状態p,qに対し$\textcircled{p}\xrightarrow{w}\textcircled{\scriptsize◎} \Leftrightarrow \textcircled{q}\xrightarrow{w}\textcircled{\scriptsize◎}$、即ちそれぞれを開始状態とみて同じ言語を認識する、ときに同値であるとする。

定理．DFAの状態を、開始状態から到達不能な状態を除いて同値類に分割して得られるDFAは、同じ言語を受理する状態数最小のDFAである。



DFA（左図）とその最小化（右図）：左図の状態q3は開始状態から到達不能なので除かれる(中央図)。

乱数アルゴリズム

計算における乱数の利用は、従来、数値積分の分野で用いられてきた。
数値積分におけるモンテカルロ法：πの計算

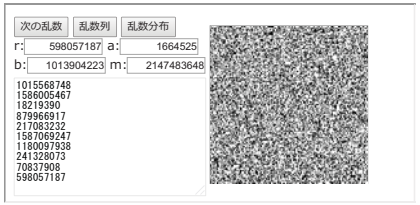
最近、乱数を利用した確率的(乱択)アルゴリズムの理論が発展した
乱択アルゴリズムには2種類ある。ともに、乱数を利用して、正答を高確率で返すが、
１．モンテカルロ法は、常に答を返すが、間違うこともある。
２．ラスベガス法は、答えを返さないこともあるが、誤答を返すことはない。
ラスベガス法で、一定時間後に必ず答を返すようにすれば、モンテカルロ法になる。

乱数の生成法

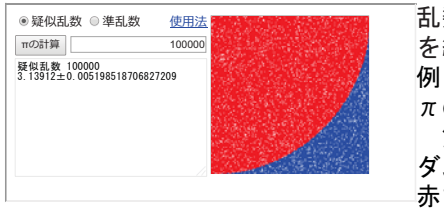
真の乱数：決定的に動作するコンピュータでは生成できない！定義さえ難しい
疑似乱数生成法 線型合同法：定数a, b, mを使って、疑似乱数rを次の式で作る
$r \leftarrow ar_n + b \mod m$
線型合同法の欠点
1. 最大周期がm
2. 下位ビットに規則性がある
3. よって m を十分大きくしなければならない

新しい疑似乱数生成法
Mersenne Twister、 xorshift

パズル．周期 $m = 4096, 65536$ のとき乱数分布はどうなるか。

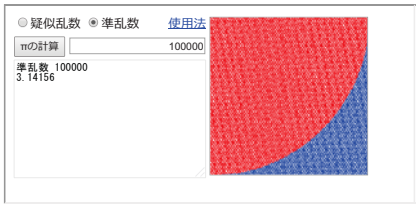


数値積分：k桁目までを求めるモンテカルロ法



乱数を利用して、(多重)定積分を計算する手法を総称してモンテカルロ法という
例：単位円の面積、すなわち円周率を求めるπの計算
　第1象限中の単位正方形上に n 個の点をランダムにばらまき、その点が単位円に含まれれば赤で、含まれなければ青で表示。点が単位円に含まれる確率は4分円の面積を近似するから、それを4倍してπの近似値とする。
±の後の値は標準誤差であり、$\sqrt{n}$に反比例するので、信頼できる精度を1桁あげたければ、点の数nを100倍にする必要があり、計算時間は当然nに比例する。

モンテカルロ法による積分では、コンピュータ(言語)が標準で提供する擬似乱数よりも、無理数の1倍，2倍，3倍，･･･の小数部分を値とする準乱数のほうが精度がよい。



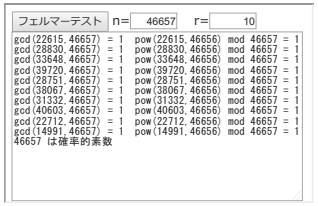
パズル．上のπの計算アプリで、以下の実験を行え。
1. 点の数 n=1,000,000 のときのπの値および計算時間を疑似乱数の場合と準乱数の場合で比較せよ。
2. 疑似乱数でn=2,000,000 4,000,000 のときの精度と計算時間を比較し、n=100,000,000 のときの精度と計算時間を予測せよ。

確率的素数判定法（モンテカルロ法）

フェルマーテスト：以下の非素数判定を繰り返しすり抜けることをもって、確率的素数判定を行う

フェルマーの小定理の対偶

n と互いに素な整数 a が $a^{n-1} \not\equiv 1 \pmod{n}$ を満たせば、n は素数でない。



フェルマーテストのアルゴリズム

以下 1. ～3. を適当な回数(r 回)繰りかえす

1. 2 以上 n 未満の自然数 a をランダムに選ぶ。
2. a と n の公約数が 1 でなければ、「nは合成数」と出力して終了
3. $a^{n-1} \not\equiv 1 \pmod{n}$ ならば、「nは合成数」と出力して終了

「nは確率的素数」と出力して終了

フェルマーテストは、ユークリッドの互除法（公約数の計算）やBinary法（べき乗計算）を用いて高速に計算できる。
フェルマーテストが「合成数」と出力したとき、n は実際に合成数である。
「確率的素数」と出力したとき、n は実際に素数であるとは限らないが、
十分回数繰りかえせば、高い確率で素数である。 しかし・・・

フェルマーテストは完全な素数判定法ではない

カーマイケル数（絶対擬素数）は合成数である
にもかかわらず、自身と互いに素な任意の a に対して $a^{n-1} \not\equiv 1 \pmod{n}$
なので、「確率的素数」と判定されてしまうことがある。

パズル．カーマイケル数 46657 が $r = 10$ のフェルマーテストをすり抜け、「確率的素数」とされる確率を求めよ。
ヒント．カーマイケル数 n が「合成数」と判定されないのは選んだ a との公約数が 1 のときなので、$46657 = 13 \times 37 \times 97$ が $r = 1$ のフェルマーテストをすり抜け、「確率的素数」とされる確率は $\dfrac{12 \times 36 \times 96 - 1}{46657 - 1}$ である。

フェルマーテストを改善した、ミラー–ラビン素数判定法がある。
また最近、決定的多項式時間で動作するAKS素数判定法も開発された。

ラスベガス法

高確率で正答を返し、間違うことはないが、
低確率で、答えが得られない

例．確率的クイックソート：データ数 n、c は適当な定数
・分割基準値をランダムに選んで計算
・$cn \log n$ 時間後に終了していなければ強制的に終了

クイックソートは
　多くの入力に対する計算時間は$n \log n$に比例するが
　最悪の入力に対する計算時間はn^2に比例する。
確率的クイックソートは
　すべての入力に対して計算時間は$n \log n$に比例するが
　答えを返さないことがありうる

パズル．クイックソートのモンテルカルロアルゴリズムを述べよ。

データマイニング

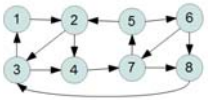
データマイニング：多量のデータ集合から、意味のある情報を抜き出す手法
　多量のデータが電子的に蓄積され
　多量のデータを処理・分析することが可能に
　予備知識なしに理解可能な計算法のいくつかを直感的に紹介

1. グーグルのページランクアルゴリズム
2. 推薦システム
3. マーケットバスケット分析（アソシエーションルール）

グーグルのページランク

グーグルの創設者 Sergey Brin と Lawrence Page による
現在のグーグルの隆盛の基礎をなす技術
ページの重要度をそのページがどれだけ閲覧されるのかで決める
Webページの間のリンク関係に基づいて測る

パズル：1から8の番号のついたWebページの間のリンク関係が右のようなグラフ(ネットワーク)になっている。このとき、どのページが重要、すなわち、よく閲覧されるか。



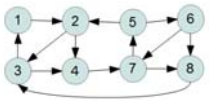
隣接行列⇒遷移確率行列

グラフの隣接行列A：

$$A(i,j)=\begin{cases}1 & \text{頂点}i\text{から頂点}j\text{へ矢印(リンク)}\\ 0 & \text{なければ}\end{cases}$$

隣接行列Aから遷移確率行列Pを求めるとき
A. 頂点のどれか(例えば１)から出発
B. 矢印のどれかを等確率で選んで、遷移(移動)
C. 出る矢印がなければ、その頂点にとどまる
頂点iから出る矢印の本数をkとしたとき

$$P(i,j)=\begin{cases}A(i,j)/k & k>0\ (\text{条件B})\\ 1 & k=0\text{で}i=j\ (\text{条件C})\\ 0 & k=0\text{で}i\neq j\end{cases}$$



使用法

初期状態	1	2	3	4	5	6	7	8	9	10
	1	0	0	0	0	0	0	0		

ページ数≦10
8
パラメータ (0≦・≦1.0)
1.0
準備 更新

隣接行列	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	0	0	0		
2	0	0	1	1	0	0	0	0		
3	1	0	0	1	0	0	0	0		
4	0	0	0	0	0	0	1	0		
5	0	1	0	0	0	1	0	0		
6	0	0	0	0	0	0	1	1		
7	0	0	0	0	1	0	0	1		
8	0	1	0	0	0	0	0	0		
9										
10										

確率遷移行列

0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.500	0.500	0.000	0.000	0.000	0.000	0.000
0.500	0.000	0.000	0.500	0.000	0.000	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	1.000	0.000	0.000
0.000	0.500	0.000	0.000	0.000	0.500	0.000	0.000	0.000
0.000	0.000	0.000	0.000	0.000	0.000	0.500	0.500	0.000
0.000	0.000	0.000	0.000	0.500	0.000	0.000	0.500	0.000
0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

0回目	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1回目	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
2回目	0.000	0.000	0.500	0.500	0.000	0.000	0.000	0.000	0.000
3回目	0.250	0.000	0.000	0.250	0.000	0.000	0.500	0.000	0.000
4回目	0.000	0.250	0.000	0.000	0.250	0.000	0.250	0.250	0.000
5回目	0.000	0.375	0.125	0.125	0.125	0.125	0.000	0.125	0.000
6回目	0.063	0.188	0.188	0.250	0.000	0.063	0.188	0.063	0.000
7回目	0.094	0.125	0.094	0.188	0.094	0.000	0.281	0.125	0.000
8回目	0.047	0.266	0.063	0.109	0.141	0.047	0.188	0.141	0.000
9回目	0.031	0.258	0.133	0.164	0.094	0.070	0.133	0.117	0.000
10回目	0.066	0.195	0.129	0.195	0.066	0.047	0.199	0.102	0.000
11回目	0.064	0.201	0.098	0.162	0.100	0.033	0.219	0.123	0.000

一般に各頂点に対し、次の頂点への遷移確率(必ずしもすべて等しいとは限らないが和は１)が与えれらたグラフを**マルコフ連鎖**といい、その行列表現を**遷移確率行列**という。

パズル．確率遷移行列の計算
1. 例示グラフの確率遷移行列を計算せよ
　パラメータ(意味は後述)を１にして、準備ボタンをクリックする
2. ページ(頂点)数を4にして、準備ボタンをクリックし、確率遷移行列が条件のCを満たしていることを確認せよ
　また、どのようなグラフの確率遷移行列を計算しているのか、図示せよ。

定常状態

ある頂点から出発して(確率的な)遷移を繰り返すと、n回目の遷移の後でそれぞれの頂点にいる確率の分布（状態ベクトル）が定義される

$n+1$回目の状態ベクトル$=$ n回目の状態ベクトル×確率遷移行列

ページランクアルゴリズムの基本的な考え方

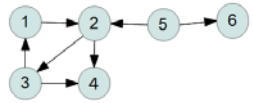
状態ベクトル(確率分布)が定常状態に収束したら、確率はそのページを平均的に訪れる(閲覧する)頻度を表すので、重要度の指標とする

グラフが次の条件を満たすと出発頂点がどこであるかに関わらず状態ベクトルは定常状態に収束する

a. グラフが強連結である(各頂点が互いにつながっている)
b. グラフのサイクル(循環する道)の長さの最大公約数が１である

パズル．定常状態

アプリで準備ボタンをクリックした後、更新ボタンのクリックを繰り返すと、各頂点にいる確率の分布が変化(収束)が見られる。



1. ページ数を3、パラメータを1にして(隣接行列はそのまま！)、条件 b が成り立たないと確率分布が循環することを確認せよ。
2. ページ数を6にすると条件 a が成り立たない。グラフを書いて確認せよ。
3. ページ数を6、パラメータを1とし、異なる定常状態に収束するような初期状態(出発頂点)を求めよ。

パラメータαの導入

パラメータ1の議論

・マルコフ連鎖としてよく知られ、さまざまな場面に応用されてきた
・グラフがWeb上のページリンクを表すという文脈ではおかしな仮定

1. リンクを持たないページ(PDFファイル等)にとどまり続ける
2. 必ずリンクをたどる(各リンクの遷移確率の和が1)

パラメータαの意味

ページでリンク(のどれか)をたどる確率α(＝0.85)
確率1－αで、新たなページを(等確率で)選んでネットサーフィンを始める
遷移確率行列P_αは、総ページ数をNとして

$$P_\alpha(i,j)=\begin{cases}\alpha A(i,j)/k+(1-\alpha)/N & \text{頂点}i\text{から出る矢印の本数}k>0\\ 1/N & k=0\end{cases}$$

・すべてのページの間に矢印(遷移)が付く⇒定常状態に必ず収束
・定常状態への収束の速さはα^nに比例
・計算速度と、定常状態(ページランク)の正確さとのバランス
⇒$\alpha=0.85$（と言われている）

パズル．パラメータα

アプリで収束までの回数および各ページのランク(確率)値を、パラメータα＝１のときとパラメータα=0.85のときで、比較せよ

・グーグルがページのランク付けに利用しているのはリンク構造だけではないし、詳細は公表されていない
・解説したような単純なものでない
・検索サイトがつけるランクをアップさせるための様々な工夫や業者が存在し、検索サイトとの間でいたちごっこ

推薦システム

ショッピングサイトで品物を買う(見る)と、他の品物も勧められる。
あなたが薦められるのは

- **人ベースの推薦**：嗜好(購買行動)が似た客が買った品物
- **物ベースの推薦**：買った(見た)品物と(売れ方が)似ている品物

基本的なアイディア：似た人(物)は、似た動きをする
表 A：本人(0)と人1～人6の物A～Eに対する評価(購買数)、空欄は評価(購買)無し
[人ベース]： 本人と人iの間の(ユークリッド)距離(の2乗)

$$d(i)=\sum_{i,0\text{が共に評価している}X}(A(i,X)-A(0,X))^2$$

本人の物Xに対する予想評価(推薦スコア)

$$=\Big(\sum_{A(i,X)\neq\text{空}}A(i,X)(1+d(i))^{-1}\Big)/\Big(\sum_{A(i,X)\neq\text{空}}(1+d(i))^{-1}\Big)$$

即ち、近い人ほど大きな重み$1/(1+d(i))$を評価$A(i,X)$につけて平均をとる

[物ベース]： 物XとYの間の(ユークリッド)距離(の2乗)

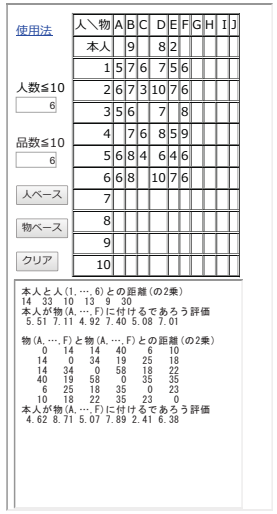
$$d(X,Y)=\sum_{X,Y\text{を共に評価している人}n}(A(n,X)-A(n,Y))^2$$

本人の物Xに対する予想評価(推薦スコア)

$$=\Big(\sum_{A(0,Y)\neq\text{空}}A(0,Y)(1+d(X,Y))^{-1}\Big)/\Big(\sum_{A(0,Y)\neq\text{空}}(1+d(X,Y))^{-1}\Big)$$

即ち、本人が物Yに与えた評価$A(0,Y)$に、物XとYの距離$d(X,Y)$から定まる重み$1/(1+d(X,Y))$をつけた重みつき平均をとる。

実際の推薦システムでは、求めた評価値をそのまま表示するのではなく、評価の高い品物から順に(いくつか)推薦する



推薦法の比較

距離を求める計算：人ベース推薦の方が容易
　人ベース推薦：各人の、本人との距離だけを計算
　物ベース推薦：すべての物の組み合わせについて計算

物ベース推薦の長所
　推薦を行う前に、本人の嗜好がわからなくても、あらかじめ品物の間距離を計算しておける
(人ベース推薦で利用する本人との距離は、本人の嗜好がわかって初めて計算できる)
　各品物に対してそれに近い品物との距離だけ記録しておけばよい

パズル. あらかじめ距離を計算しておけることは、Web上のリアルタイムの応答では決定的な利点である。その理由を考えよ。

マーケットバスケット分析

・スーパーマーケットのレジで集められるPOS(Point Of Sale)データからバスケット(籠)の中身を分析
・どの商品とどの商品が同時に買われやすいかを求める
・伝説的な事例：**週末に紙おむつを買う人はビールも買う**

アソシエーションルール
　次の形の有効なルールを探す
　　$A\to B$：A(を買う人)ならばB(も買う)
AやBは、一般的には複数の商品や条件からなり、冒頭の例は
　　(週末∧紙おむつ)　→　ビール

簡単のため、AやBはそれぞれ単一の商品とする
$\mathcal{U}$をすべての籠の集合
商品Aが入っている籠の集合を$\mathcal{A}$($\mathcal{B}$も同様)
集合$\mathcal{X}$ の要素数を$|\mathcal{X}|$で表す

アソシエーションルール $A \to B$ の有効性を、以下の数値で評価

1. **サポート**$(A \to B) = |\mathcal{A} \cap \mathcal{B}|/|\mathcal{U}|$：ルールを支持する籠が全体に占める比率
   サポート値が低いルールは少数の事例にしか当てはまらず一般性を持たない。
2. **信頼度**$(A \to B) = |\mathcal{A} \cap \mathcal{B}|/|\mathcal{A}|$：$A$を含む籠の中で$B$を含む籠が占める比率
   信頼度が低いルールは、そもそもルールが成り立っていない
3. **改善率**$(A \to B) =$ 信頼度$(A \to B)$/信頼度$(\ \to B) = (|\mathcal{A} \cap \mathcal{B}| \cdot |\mathcal{U}|)/(|\mathcal{A}| \cdot |\mathcal{B}|)$：仮定$A$による信頼度の改善率
   信頼度が高くても改善率が1以下なら、Bが買われる理由がAを買うことにはなく、そもそも無条件に(あるいは他の理由で)Bが買われていることを意味するので、この場合もルールが有効であるとはいえない。

アプリを使って、サポート、信頼度、改善率の意味を理解しよう。

パズル. アソシエーションルールの評価
左のアプリで例示されているデータを使って、以下の問いに答えよ。

1. 単一前後件ルールで最高信頼度のルールはどれか
2. 単一前後件ルールで最高改善度のルールはどれか
3. ルール AB→Cについて、そのサポート、信頼度、改善率の値を求め、値がそうなる理由を説明せよ
4. XY→Zの形で改善率2.5以上のルールを求めよ

使用法

籠数≦10
10

品数≦10
5

ルール評価
AB→C 評価

単一前後件ルール

クリア

籠	A	B	C	D	E	F	G	H	I	J
1	1	1		1						
2	1	1								
3	1	1	1	1						
4	1	1	1							
5	1		1	1	1					
6		1	1							
7		1	1							
8			1	1	1					
9			1							
10			1	1	1					

```
AB→C  サポート=0.20  信頼性=0.50  改善率=0.63
A→B  サポート=0.40  信頼性=0.80  改善率=1.33
A→C  サポート=0.30  信頼性=0.60  改善率=0.75
A→D  サポート=0.30  信頼性=0.60  改善率=1.20
A→E  サポート=0.10  信頼性=0.20  改善率=0.67
B→A  サポート=0.40  信頼性=0.67  改善率=1.33
B→C  サポート=0.40  信頼性=0.67  改善率=0.83
B→D  サポート=0.20  信頼性=0.33  改善率=0.67
B→E  サポート=0.00  信頼性=0.00  改善率=0.00
C→A  サポート=0.30  信頼性=0.38  改善率=0.75
C→B  サポート=0.40  信頼性=0.50  改善率=0.83
C→D  サポート=0.40  信頼性=0.50  改善率=1.00
C→E  サポート=0.30  信頼性=0.38  改善率=1.25
D→A  サポート=0.30  信頼性=0.60  改善率=1.20
D→B  サポート=0.20  信頼性=0.40  改善率=0.67
D→C  サポート=0.40  信頼性=0.80  改善率=1.00
D→E  サポート=0.30  信頼性=0.60  改善率=2.00
E→A  サポート=0.10  信頼性=0.33  改善率=0.67
E→B  サポート=0.00  信頼性=0.00  改善率=0.00
E→C  サポート=0.30  信頼性=1.00  改善率=1.25
E→D  サポート=0.30  信頼性=1.00  改善率=2.00
```

長所と短所

○対象となるデータの中に潜む有効なルールを探し出す探索的手法
○結果をアソシエーションルールという分かりやすい形で提供
×品数が多くなると、$X \to Y$という形の規則の個数は指数関数的に増大
　⇒すべて調べるのは現実的ではない
×「苗→肥料」のように、当然のルールが多数出る
　⇒真に有効なルールを見出すには、機械だけでは無理
○推薦システムと異なり、個人と紐付けられていない匿名データに対しても有効

手に負えない問題

プログラムが実行可能：計算時間が問題(入力)のサイズ（表現長）の多項式で抑えられる
　素朴べき乗　　：実行可能でない
　高速べき乗、選択法、クイックソート：実行可能

手におえない問題：実行可能なプログラムが存在しない問題

以後、問題をyes/no問題に限定する。

パズル．多項式時間プログラムを実効可能なプログラムとすることの是非を論ぜよ。

非決定性プログラム

非決定性プログラム：非決定性命令を含むプログラム
　非決定性命令：次の動作を有限個の候補の中から選べる
　計算：次の動作をうまく選んだ時に正答yesを正しく計算できればよい
　　⇔可能な計算の中に正答yesを正しく計算するものがあればよい
　　(no と答える場合があってもよい)
　計算時間：正答yesを正しく計算するものの中での最短時間としてよい
但し、正答がnoのときにyesと答える場合があってはいけない。

定理．非決定性プログラムでt(n)時間で計算できる問題は、
決定性プログラムで$a^{t(n)}$時間(aは選択肢数の最大値)で計算できる。
略証：すべての選択肢を試行錯誤の虱潰しで調べればよい。

P、NP

クラスP:決定性多項式時間プログラムで解ける問題
クラスNP：非決定性多項式時間プログラムで解ける問題

P⊆NP。　　P≠NP か否かが有名な未解決問題（ミレニアム問題）

オイラー路の存在判定問題∈P
　∵)オイラーの定理があるから
ハミルトン路の存在判定問題∈NP
- ハミルトン路を非決定的に選択できれば、それがハミルトン路であることの検証は多項式時間で可能
- Pに属すか否かは知られていない

NP問題の別表現
証拠が与えられれば、その答えがYesであることを、問題(入力)サイズの多項式時間内に検証できる問題。

NP完全問題

NP問題pがNP完全：問題pを解くプログラムM_pから、すべてのNP問題qに対して、qを解く多項式時間プログラムM_qを多項式時間で作れる問題

NP完全問題のどれか１つでもP問題であれば、P＝NP
つまり、NP問題の中で、もっとも難しいと考えられる問題

Cookは充足可能性判定問題がNP完全であることを示した(1971)。

充足可能性判定問題
与えられた論理式の値を真にする変数値の割り当てが存在するか否か判定する問題

パズル．P=NPであれば、どのようなことが言えるか。

証明のアイディア

M_p：NP問題Pを解く非決定性多項式p(n)時間プログラム（チューリング機械）
　M_pはサイズnの入力xに対し、N-1=p(n)ステップで問題Pの答P(x)=yes/noを(例えば)最初のセルに書き込んで停止する
M_pの入力xに対する動作列を記述し答がyesかnoかを判定する論理式L(x)をNの多項式時間で作り、L(x)が充足可能⇔P(x)=yes となるようにする

変数	=真 の意味	個数
S_{tq}	時点tのM_pのプログラム命令（内部状態）がq	N×命令（状態）数
T_{tja}	時点tのメモリ（テープセル）jの内容が記号a	N^2×記号数
H_{tk}	時点tのアクセスメモリ番地（ヘッド位置）がk	N^2

注．プログラム（チューリング機械）がアクセスできるメモリ（セル）数≦N

これらの変数に対する真偽の割り当てが、M_pの正しい計算になって(動作規則を満たして)いて出力がyesのときにのみ真となるような論理式をつくればよい。

NP完全問題の例

充足性判定問題
ハミルトン(閉)路問題：与えれたグラフがハミルトン(閉)路を持つか
巡回セールスマン問題：与えれた都市をすべてめぐるコストw以下の路があるか
3彩色問題：与えれたグラフの頂点を3色で塗り分けることができるか

P問題ではなさそうなNP問題
素因数分解：与えられた数の素因数分解を求める
（素数判定は最近P問題であることが示された）

公開鍵暗号

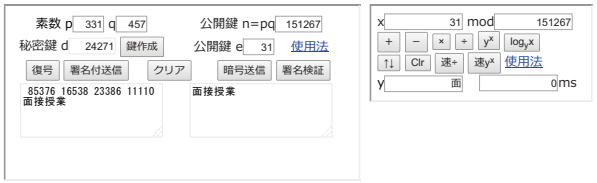
送信者　　通信路　　受信者
平文[暗号化]⇒　暗号　⇒[復号]平文

共通鍵暗号：送信者と受信者で実質的に同じ鍵を用いる
　鍵(と手順)を秘密裡に共有し(ネット上で可能？)
　送られた暗号をその鍵で復号
　例 IBM[1文字前]⇒ HAL ⇒[1文字後]IBM

公開鍵暗号：送信者(暗号化)と受信者(復号)で異なる鍵
　受信者が暗号化の鍵(と手順)を公開
　送られた暗号を秘密にしている鍵で復号
　公開(暗号化)鍵から秘密(復号)鍵が分からないのか？
　　RSA暗号、楕円曲線暗号

パズル．通信路の盗聴者が、公開鍵暗号を破るための方法を考えよ。

RSA公開鍵暗号の作成実験



p, qが素数⇒$x=x^{1+(p-1)(q-1)} \pmod{pq}$
$x=x^{1+(p-1)(q-1)}=x^{1+2(p-1)(q-1)}=\cdots \pmod{pq}$なので、
$de=1 \pmod{(p-1)(q-1)}$なら$(x^d)^e=(x^e)^d=x^{de}=x \pmod{pq}$。
n=pqを法としたd乗とe乗が互いに逆関数⇒暗号化と復号に利用

公開鍵：nとe　　秘密鍵：nとd（とp, q）

RSA暗号の安全性

RSA暗号を破るには、次のような方法が考えられる。

1. 暗号(の値)を、nを法として1乗、2乗、3乗、…して元に戻るか否かで調べる。d乗まで調べた時点で元に戻るから、秘密鍵 d をいつかは知ることができる。
2. nの約数p(q)を探し、(p-1)(q-1)を法とする(1÷e)を計算してdを求める。
3. 推測した平文(の一部)xをmod nでe乗して盗聴した暗号(の一部)yと一致するか見る。

これらはいずれも虱潰しの方法で、素朴ベキ乗計算の実習でみたように、nの値や秘密鍵、素数p、qを十分おおきくとれば、現実的ではない。また、与えられた x, y, n, に対し、$x=y^d \pmod n$ を満たすdを求める効率的な方法は知られていないし、nからp, qを求める効率的な因数分解の方法も知られていない。したがって、安全と考えてよい。

ところで、x^dの計算は復号計算でもあるから、その計算に x を(d-1)回掛ける素朴な方法しかなければ、復号にとんでもない時間がかかってしまうことになり、RSA暗号は機能しない。幸い、d の値がわかっていれば、d 乗を高速に行うBinary法が知られている。

RSA暗号の安全性とP≠NP

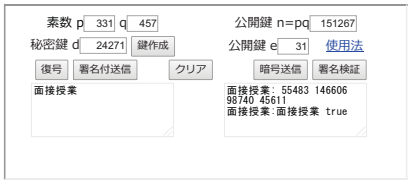
前ページでは、復号は、秘密鍵 dを知っている本人は高速実行できるが、たとえ暗号を入手(盗聴)できたとしても、秘密鍵を知らない第3者が可能な素朴な方法では、現実的な時間内に実行できないことがわかった。

現在のところRSA暗号を現実的な時間内に破る方法は知られていないが、何か巧妙な方法が、まだ見つかっていないだけで、今後見つかる可能性が0とは決まっていない。そのような方法は存在しえず、それゆえRSA暗号は安全であると広く信じられてはいるが、そのことの数学的に厳密な証明は得られていない。

これは、今回紹介したP≠NP問題というクレイ数学研究所から100万ドルの賞金がかけられている有名な未解決問題に、密接に関係する問題である。

電子署名

署名＝文書を秘密鍵で暗号化したもの
文書＋署名の受取人は公開鍵で署名を復号して文書との一致を検証する



公開鍵で復号できる暗号を作れるのは秘密鍵の持ち主だけ
⇒文書の改ざん、署名の偽造(なりすまし)が防げる

電子署名はゼロ知識証明
秘密(鍵)を明かすことなく、秘密(鍵)を知っていることを証明する（納得させる）

マジックプロトコル

コイントス：A,B 両者が信頼できる第三者を介さずに通信線上で行う
コインを投げるAは、Bの答を聞いた後でコインの裏表を変えられない
裏表を当てるBは、Aが投げたコインを盗み見できない
Bが答えた後、その当否を両者(特にB)が確認できる

プロトコル(通信手順)

1. Aは乱数rをBに送り、rのAによる署名の偶奇をBに問う。
2. Bは偶奇を答える。
3. Aはrの署名をBに送り、Bはそれを検証する

パズル. 上のプロトコルがうまく働くことを示せ。

他のマジックプロトコル：いずれも通信線で第三者を介さずに行う
金持ち比べ：互いの資産額を知らせずに、その大小を比べる
カード配り：トランプのカードを配る
電子投票：投票の秘密を保って投・開票を行う