データ構造とアルゴリズム

02 アルゴリズムの記述，アルゴリズムの正当性

宮本 裕一郎
miyamoto あっと sophia.ac.jp

上智大学 理工学部 情報理工学科

目次

再び，アルゴリズムとは？

【再掲】アルゴリズム 【algorithm; algorism】

（アラビアの数学者アル＝フワリズミーの名に因む）

1. アラビア記数法
2. 問題を解決する定型的な手法・技法.
   コンピューターなどで，演算手続きを支持する規則.
   算法.

「広辞苑（第六版）」より

- そもそも「問題」とは？「解決する」とは？

そもそも問題とは？

- アルゴリズムやデータ構造の講義で扱う「問題」は，正確には計算問題である．

定義 (計算問題)

与えられた入力文字列に対して，出力文字列を決める関係を計算問題（computation problem）という．

例

- 最大値を見つける問題
  入力　数値の集合 $\{x_1, x_2, \ldots, x_n\}$
  出力　最大値 $\max\{x_1, x_2, \ldots, x_n\}$
- 数値を小さい順に並べ替える問題
  入力　数値の列 $(x_1, x_2, \ldots, x_n)$
  出力　小さい順に並べ替えられた数値列

最大値を見つける問題

問題 (最大値発見)

入力　数値の集合 $\{x_1, x_2, \ldots, x_n\}$

出力　最大値 $\max\{x_1, x_2, \ldots, x_n\}$

例

- $\{2,3\}$ が入力として与えられたら 3 を出力する.
- $\{5,4\}$ が入力として与えられたら 5 を出力する.
- $\{3,4,2\}$ が入力として与えられたら 4 を出力する.
- $\{7\}$ が入力として与えられたら 7 を出力する.
- $\{3,1,4,1,5,9,2,6,5\}$ が入力として与えられたら 9 を出力する.
- ……

数値を小さい順に並べ替える問題

問題 (昇順並べ替え)

入力 数値の列 $X = (x_1, x_2, \ldots, x_n)$

出力 数値列 $Y = (y_1, y_2, \ldots, y_n)$，ただし $y_1 \leq y_2 \leq \cdots \leq y_n$ かつ X から Y への全単射が存在

例

- $(2,3)$ が入力として与えられたら $(2,3)$ を出力する.
- $(5,4)$ が入力として与えられたら $(4,5)$ を出力する.
- $(3,4,2)$ が入力として与えられたら $(2,3,4)$ を出力する.
- (7) が入力として与えられたら (7) を出力する.
- $(3,1,4,1,5,9,2,6,5)$ が入力として与えられたら $(1,1,2,3,4,5,5,6,9)$ を出力する.
- ……

つまり，アルゴリズムとは

- 計算問題を解決することは，与えられた入力に対して定められた出力を出すことである．
- 与えられた入力に対して，定められた出力を出すための手続きをアルゴリズムという．
- 正確な定義は以下の通りである．

定義 (（狭義の）アルゴリズム)

Turing 機械（Turing machine）で実行可能な手続きをアルゴリズムという [1] ．

- 次ページに「最大値を出力するアルゴリズム」を例として記述してみる．

[1] 他にもラムダ計算による定義などもある．詳しくはチャーチ，チューリングの提唱などのキーワードで調べると良い．
または「情報理工学 III（計算と情報の理論）」を受講いただきたい．

最大値を見つけるアルゴリズムの直感的記述

最大値を出力するアルゴリズム

- x_1 から x_n まで順番に見ていく.
- 途中ではそれまでに見た値のうち最大のものだけを覚える.
- 最後に覚えている値（すなわち最大の値）を出力する.

例

- $\{2,3\}$ が入力として与えられたら 3 を出力する.
- $\{5,4\}$ が入力として与えられたら 5 を出力する.
- $\{3,4,2\}$ が入力として与えられたら 4 を出力する.
- $\{7\}$ が入力として与えられたら 7 を出力する.
- $\{3,1,4,1,5,9,2,6,5\}$ が入力として与えられたら 9 を出力する.
- ……

- この記述で十分であろうか？

アルゴリズムと調理

- アルゴリズムは料理を作ることに似ている．実際，アルゴリズムのことをレシピ（recipe）とよぶこともある．
- 以下に，チキンカレー（4 人分）のレシピを示す．

材料

鶏もも肉	200 g
玉葱	320 g
馬鈴薯	160 g
人参	120 g
カレー粉	80 g
塩コショウ	少々
鶏がらスープの素	小さじ 1
ガラヌマサラ	少々
ナツメグ	少々
バター	大さじ 1
オリーブオイル	大さじ 1

手順

1. あらかじめ材料を切っておく．馬鈴薯は切ってから水にさらしておく．
2. バターで，玉葱をしんなりするまでよく炒める．炒め終わったら別の容器に移す．
3. オリーブオイルをしき，鶏肉を入れ，塩コショウを少々入れた後，全体に火が通るまで中火で炒める．
4. 馬鈴薯と人参を入れ，炒める．火が通ってきたら，玉葱も合わせる．
5. 水 750 cc で希釈した鶏がらスープを入れ，沸騰したら弱火で 30 分ほど煮込む．この間に，別のフライパンでカレー粉を乾煎りしておく．
6. 火を切り，乾煎りしたカレー粉と香辛料を入れよく混ぜる．カレーの濃さを確認し，好みの濃さに調整をした後，弱火で 10 分ほど煮込む．時折かき混ぜ，焦げ付かないようにする．

もう1つのカレーのレシピ

- 前ページのレシピとは別に，以下の手順も考えられる．

材料

鶏もも肉	200 g
玉葱	320 g
馬鈴薯	160 g
人参	120 g
カレー粉	80 g
塩コショウ	少々
鶏がらスープの素	小さじ 1
ガラヌマサラ	少々
ナツメグ	少々
バター	大さじ 1
オリーブオイル	大さじ 1

手順

1. あらかじめ呼んでおいたシェフに「チキンカレー（4 人分）を作れ」と命令する．

- 前ページのレシピとの違いは「使える道具」（あるいは調理手段）である．
- アルゴリズム（レシピ）を記述する際には，入力（材料），出力（完成品），アルゴリズム（手順）に加えて「基本演算（使える道具）」を明らかにする必要がある．

アルゴリズムの記述

アルゴリズムの記述

アルゴリズムの記述は

- 入力,
- 出力,
- （使うことが許される）基本演算,
- アルゴリズム（手順）,

からなる．そして，可能な限り数式を利用する
（∵ 数式は最もユニバーサルな言語）.

- 一般に，基本演算は当該分野における共通認識として暗に仮定されていることが多い.

使うことが許される基本演算

- この講義では，現在我々が「普通に」使えるコンピューターおよびプログラミング言語で基本とされている演算を基本演算とする．
  - 変数値の参照，代入
  - 変数値の比較
  - （減算を含む）加算，乗算，除算（あるいは商と余りの計算）などの四則演算
  - 集合の和，差，共通部分などの集合演算
  - 平方根，三角関数，対数関数などの基本的な数学関数の計算
- ただし，文脈によっては，上記にない演算を基本演算とする場合も，上記の演算の利用を仮定しない場合もある．

最大値を見つけるアルゴリズムのより厳密な記述

- 最大値を出力するアルゴリズムを Maximum として関数っぽく以下に記述する.
- アルゴリズム Maximum では，変数の参照，変数への代入，実数の大小比較を基本演算と仮定している.
- なお，変数 y が「それまでに見た最大値」に対応する.

Maximum$(\{x_1, x_2, \ldots, x_n\})$:

Step 1 y に x_1 を代入する.

Step 2 それぞれの $i \in \{2, 3, \ldots, n\}$ に関して以下を行う.

　　Step 2-1 もし $y < x_i$ ならば，y に x_i を代入する.

Step 3 y を出力して終了する.

例

- $\{2, 3\}$ が入力として与えられたら 3 を出力する.
- $\{3, 1, 4, 1, 5, 9, 2, 6, 5\}$ が入力として与えられたら 9 を出力する.

最大値を出力するアルゴリズムに関する補足

maximum$(\{x_1, x_2, \ldots, x_n\})$:【再掲】

Step 1 y に x_1 を代入する.

Step 2 それぞれの $i \in \{2, 3, \ldots, n\}$ に関して以下を行う.

　　Step 2-1 もし $y < x_i$ ならば，y に x_i を代入する.

Step 3 y を出力して終了する.

- 一般に，数値の最大値だけでなく，全順序が定義された集合に対しては同様のアルゴリズムが考えられる.
- 最小値を出力するアルゴリズムも同様である.

問題の記述とアルゴリズムの記述に関するまとめ

- 本講義では，関数っぽくアルゴリズムを記述する．その中身は箇条書きで記述する．
- なお，アルゴリズムの記述方法として他にも
  - プログラミング言語による記述，
  - 擬似言語による記述，
  - フローチャートによる記述，

  などが一般に知られている．
- 本講義では，しばしば，問題とアルゴリズムを分離して記述する．

最大公約数とユークリッドの互除法

- ユークリッドの互除法（Euclid's Algorithm）は，最大公約数（Greatest Common Divisor）を見つけるアルゴリズムである．
- 一説によると最古のアルゴリズム（紀元前 300 年頃？）である．

問題 (最大公約数)

入力 $x, y \in \mathbb{N}$

出力 x と y の最大公約数

最大公約数の計算例

入力 自然数 $10807,\ 10403$

計算
- $10807 \div 10403 = 1 \cdots 404$
- $10403 \div 404 = 25 \cdots 303$
- $404 \div 303 = 1 \cdots 101$
- $303 \div 101 = 3 \cdots 0$

出力 最大公約数 101

ユークリッドの互除法

- ユークリッドの互除法を `Euclid` として関数っぽく以下に記す.
- `Euclid` では，変数の参照，変数への代入，整数の大小比較，剰余，最大値，最小値を基本演算と仮定している.

`Euclid`(x, y):

Step 0 x に $\max\{x, y\}$ を，y に $\min\{x, y\}$ を代入する.

Step 1 $y > 0$ の間，以下を繰り返す.

　　Step 1-1 x に y を，y に x を y で割った余りを代入する.

Step 2 x を出力して終了する.

平方根の見積もり

- ここでは，変数の参照，変数への代入，実数の大小比較，実数に対する四則演算（和，差，積，商）は使えるけれど，平方根の計算は直接使えないとして以下の問題に対するアルゴリズムを考える．

問題 (平方根の見積もり)

入力　$x \in \mathbb{N}$，精度 $p \in \mathbb{R}_{>0}$

出力　$a \leq \sqrt{x} \leq b$ かつ $b - a \leq p$ である，$a, b \in \mathbb{R}_{\geq 0}$

- ここで a は $\sqrt{x}$ の下界，b は $\sqrt{x}$ の上界という気持ちである．
- このような問題に対する単純なアルゴリズムとして 2 分探索（binary search）が知られている．

2 分探索の直感的理解

- 例えば，$\sqrt{2}$ を $1/1000$ くらいの誤差で知りたいとする．
- $0 < \sqrt{2} < 2$ は明らかだと思うので，$\sqrt{2}$ の「下界」を 0,「上界」を 2 とする．
- 「上界」と「下界」の差が $1/1000$ 未満になるまで以下を繰り返す．
  - 「上界」と「下界」の「中間」を m とする．
  - もし $m^2 > 2$ ならば，「下界」$< \sqrt{2} <$「中間」であるはずなので,「中間」を新たな「上界」とする．
  - もし $m^2 \leq 2$ ならば，「中間」$\leq \sqrt{2} <$「上界」であるはずなので,「中間」を新たな「下界」とする．
  - いずれの場合にしろ，新たな「上界」と「下界」の差は，元の「上界」と「下界」の差の半分になっている．
- $\sqrt{2}$ の「下界」と「上界」を出力して終了する．

平方根の見積もりのための 2 分探索

- 平方根の見積もりを SquareRootBinarySearch として関数っぽく以下に記す.

SquareRootBinarySearch(x, p):

Step 1 下界 a に 0 を，上界 b に x を代入する.

Step 2 $b - a > p$ である限り，以下を繰り返す.

　Step 2-1 m に $\frac{a+b}{2}$ を代入する.

　Step 2-2 もし $m^2 > x$ ならば，b に m を代入する.

　Step 2-3 もし $m^2 \leq x$ ならば，a に m を代入する.

Step 2 a, b を出力して終了する.

例

- $x = 2,\ p = 0.1$ が入力として与えられたら，$a = 1.375,\ b = 1.4375$ を出力する.

平方根の見積もりのための 2 分探索の補足

- SquareRootBinarySearch では，$\sqrt{x}$ が存在する範囲 $[a, b]$ が，繰り返しごとに半分になる．
- SquareRootBinarySearch は 2 分法とよばれることもある．
- SquareRootBinarySearch のような計算は，一般に「アルゴリズムとデータ構造」というような名前の講義では扱われず,「数値計算法」というような名前の講義で扱われる．
- 一般に，2 分探索は平方根の見積だけでなく「目的との順序関係が簡単にわかり」かつ「要素が単調に（順番に）並んでいる」集合に対して適用可能である．

安定マッチング問題の記述

入力 男性の集合 M，女性の集合 W, (ただし $|M| = |W|$)
異性に対する完全なるランキング，すなわち
男性 $m \in M$ の女性 $w \in W$ に対する順位 $R_{\mathrm{M}} \colon M \times W \to \mathbb{N}$ と
女性 $w \in W$ の男性 $m \in M$ に対する順位 $R_{\mathrm{W}} \colon W \times M \to \mathbb{N}$
（ただしここでは，順位の値が小さいほど好ましいとする.）

出力 安定マッチング $S \subset M \times W$，すなわち

- 完全マッチングであり,
  - どの $m \in M$ もどの $w \in W$ も S に高々1 回しか現れず,
  - かつ，$|S| = |M| = |W|$
- かつ，ブロッキングペア
  - ブロッキングペア $(m, w) \in M \times W$ とは，マッチング S における m, w のそれぞれのパートナー w', m'，すなわち $(m, w'), (m', w) \in S$ を満たすもの，に対して $R_{\mathrm{M}}(m, w') > R_{\mathrm{M}}(m, w)$ かつ $R_{\mathrm{W}}(w, m') > R_{\mathrm{W}}(w, m)$ を満たすもの

を含まないもの

Gale-Shapley アルゴリズムの記述

- Gale-Shapley アルゴリズムを `GaleShapley` として関数っぽく以下に記す．
- アルゴリズム `GaleShapley` では，変数の参照，変数への代入，集合の要素の発見，集合からの要素の取り出し，集合への要素の追加，整数の加算，整数の大小比較を基本演算と仮定している．

`GaleShapley`$(M, W, R_{\mathrm{M}}, R_{\mathrm{W}})$:

Step 1 フリーな男性の集合 F に M を代入する．

Step 2 それぞれの男性 $m \in M$ に関して，次に告白する順位 $N_{\mathrm{W}}(m)$ に 0 を代入する．

Step 3 それぞれの女性 $w \in W$ に関して，お相手 $s(w)$ に空要素を代入する．

Step 4 フリーな男性の集合 F が空集合でない間，以下を繰り返す．

- Step 4-1 フリーな男性の集合 F から要素（男性）を 1 つ取り出し，それを m とする．
- Step 4-2 男性 m が次に告白する順位 $N_{\mathrm{W}}(m)$ を 1 増やす．
- Step 4-3 その男性 m が次に告白する相手 w，すなわち $R_{\mathrm{M}}(m,w) = N_{\mathrm{W}}(m)$ を満たす w を見つける．
- Step 4-4 もし女性 w のお相手 $s(w)$ が空要素ならば，$s(w)$ に m を代入する．
- Step 4-5 もし女性 w のお相手 $s(w)$ が空要素でないならば以下を行う．
  - Step 4-5-1 もし $R_{\mathrm{W}}(w,m) < R_{\mathrm{W}}(w,s(w))$ ならば，フリーな男性の集合 F に $s(w)$ を加え，女性 w のお相手 $s(w)$ に m を代入する．
  - Step 4-5-2 もし $R_{\mathrm{W}}(w,m) > R_{\mathrm{W}}(w,s(w))$ ならば，フリーな男性の集合 F に m を加える．

Step 5 マッチング $\{(s(w),w) \mid w \in W\}$ を出力して終了する．

演習問題: 総当り戦スケジュール表作成問題の記述

問題 偶数チームからなる総当り戦スケジュール表の作成問題を，すなわち入力と出力の関係を，数式を用いて記述してみよう．

解答例 **入力** チームの集合 $T = \{t_1, t_2, \ldots, t_n\}$ （ただし n は偶数）
出力 チーム $t \in T$ の $d \in \{1, 2, \ldots, n-1\}$ 日目の対戦相手 $p(t,d) \in T$，ただし，

- $\forall t \in T,\ d \neq d' \rightarrow p(t,d) \neq p(t,d')$，かつ，
- $\forall t \in T,\ \forall d \in \{1, 2, \ldots, n-1\},\ p(p(t,d),d) = t$，かつ，
- $\forall t \in T,\ \forall d \in \{1, 2, \ldots, n-1\},\ p(t,d) \neq t$

演習問題: circle method の記述

問題　偶数チームからなる総当り戦スケジュール表作成問題に対する circle method を，程よく数式を用いて記述してみよう．

解答例
- Circle method を `CircleMethod` として関数っぽく以下に記す．
- `CircleMethod` では，変数の参照，変数への代入を基本演算と仮定している．
- d は日付に対応する変数，c_i は i 番目の場所にあるチーム名に対応する変数である．c_n は Circle Method において中心に置かれたチームに対応している．c'_i は c_i を 1 つずつずらすための変数である．

`CircleMethod`$(t_1, t_2, \ldots, t_n)$:

Step 1　それぞれの $i \in \{1, 2, \ldots, n\}$ に関して，c_{i-1} に t_i を代入する．

Step 2　それぞれの $d \in (1, 2, \ldots, n-1)$ に関して，以下を繰り返す．

- Step 2-1　それぞれの $i \in \{0, 1, \ldots, n-1\}$ に関して，$p(c_i, d)$ に c_{n-i-1} を代入する．
- Step 2-2　それぞれの $i \in \{1, \ldots, n-2\}$ に関して，c'_i に c_{i-1} を代入する．
- Step 2-3　c'_0 に c_{n-2} を代入する．
- Step 2-4　それぞれの $i \in \{0, \ldots, n-2\}$ に関して，c_i に c'_i を代入する．

Step 3　それぞれの $t \in T$, $d \in \{1, 2, \ldots, n-1\}$ に関して，$p(t, d)$ を出力する．

演習問題: アル・フワーリズミーの乗算方法の記述

問題　アル・フワーリズミーの乗算方法を，程よく数式を用いて記述してみよう.（ただし，入力は自然数に限定する.）

解答例

- アル・フワーリズミーの乗算方法を `AlKhwarizmiProd` として関数っぽく以下に記す.
- `AlKhwarizmiProd` では，変数の参照，変数への代入，加算，割り算（商と余りの計算）を基本演算と仮定している.
- z は積を格納する変数である.

`AlKhwarizmiProd`(x, y):

Step 1　z に 0 を代入する.

Step 2　$x \geq 1$ の間，以下を繰り返す.

- Step 2-1　もし x が奇数ならば，z に y を加える.
- Step 2-2　x に $\lfloor x/2 \rfloor$ を代入する.
- Step 2-3　y を 2 倍にする.

Step 3　z を出力して終了する.

例

12	34	(←削除)
6	68	(←削除)
3	136	
1	272	
	408	(←答)

演習問題: 単純な素数列挙アルゴリズムの記述

- 以下に素数を列挙する方法をアバウトに説明する.
  - 素数とは，2 以上の整数のうち，1 と自分だけが約数であるような整数である.
  - 整数 x が素数であるか否かは，2 以上 x 未満の整数で割り切れるか否かでわかる．1 つでも割り切れる整数があったならば素数でない.
  - 与えられた整数 n 以下の素数を列挙するには，2 から n までの整数それぞれに関して素数か否か判定し，素数だけ出力すればよい.
- この方針に基づいて，n 以下の素数を列挙するアルゴリズムを箇条書きで記述してみよう！

単純な素数列挙アルゴリズムの記述の解答例その 1: 素数判定

- 単純な素数列挙アルゴリズムの記述の解答例を段階的に記す.
- まず素数判定問題を以下に定義する.

問題 (素数判定)

入力　2 以上の整数 x

出力　x が素数ならば `True`，素数でないならば `False`

- 次に，素数判定問題に対する単純なアルゴリズムを `PrimeSimple` として関数っぽく以下に記す.
- `PrimeSimple` では，変数の参照，変数への代入，割り算（余りの計算）を基本演算と仮定している.

`PrimeSimple`(x):

Step 1　それぞれの $y \in \{2, \ldots, x-1\}$ に関して，以下を繰り返す.

　　Step 1-1　もし $x \mod y = 0$ ならば，`False` を出力し終了する.

Step 2　`True` を出力し終了する.

単純な素数列挙アルゴリズムの記述の解答例その 2: 素数列挙

- 素数列挙問題を改めて以下に定義する.

問題 (素数列挙)

入力　2 以上の整数 n

出力　n 以下の素数全て

- 単純な素数判定に基づいて素数を列挙するアルゴリズムを `EnumeratePrime` として関数っぽく以下に記す.
- `EnumeratePrime` では，変数の参照，変数への代入を基本演算と仮定し，`PrimeSimple` の利用も仮定している.
- P は素数の集合を格納する変数である.

`EnumeratePrime`(n):

Step 1　P を空集合とする.

Step 2　それぞれの $x \in \{2, \ldots, n\}$ に関して，以下を繰り返す.

　　Step 2-1　もし `PrimeSimple`(x) の出力が `True` ならば，P に x を加える.

Step 3　P を出力して終了する.

演習問題: 平方根を利用した素数列挙アルゴリズムの記述

- 実は，整数 x が素数であるか否かを判定するには，2 以上 x 未満の全ての整数で割ってみる必要はない.
- 最悪の場合でも $\sqrt{x}$ 以下の整数で割って確かめれば十分である.
- この事実に基づいて少しだけ計算の手間を省いた，n 以下の素数を列挙するアルゴリズムを箇条書きで記述してみよう！

平方根を利用した素数列挙アルゴリズムの記述の解答例その 1: 素数判定

- 平方根を利用した素数判定アルゴリズムを PrimeSqrt として関数っぽく以下に記す.
- PrimeSqrt では，これまでの基本演算に加えて，平方根の計算（正確には平方根以下の最大の整数の計算）を基本演算として仮定している．[2]

PrimeSqrt(x):

Step 1 それぞれの $y \in \{2, \ldots, \lfloor \sqrt{x} \rfloor\}$ に関して，以下を繰り返す.

　　Step 1-1 もし $x \mod y = 0$ ならば，False を出力し終了する.

Step 2 True を出力し終了する.

[2]平方根の計算を基本演算としない場合には，例えば，平方根の見積もりに対する SquareRootBinarySearch を用いれば良い.

平方根を利用した素数列挙アルゴリズムの記述の解答例その 2: 素数列挙

- 以下に平方根を利用して素数を列挙するアルゴリズムを EnumeratePrimeSqrt として関数っぽく以下に記す.
- EnumeratePrimeSqrt では，変数の参照，変数への代入を基本演算と仮定し，PrimeSqrt の利用も仮定している.
- 前出の EnumeratePrime と見比べると，PrimeSimple の代わりに PrimeSqrt と使っているだけである.

EnumeratePrimeSqrt(n):

Step 1 P を空集合とする.

Step 2 それぞれの $x \in \{2, \ldots, n\}$ に関して，以下を繰り返す.

　　Step 2-1 もし PrimeSqrt(x) の出力が True ならば，P に x を加える.

Step 3 P を出力して終了する.

演習問題: エラトステネスの篩の記述

- 実は，もっと効率的に n 以下の素数を列挙する方法が古くから知られている．それがエラトステネスの篩である．
- 以下にその方法をアバウトに説明する．
  - 2 から n までの数字が書かれたカードを 1 枚ずつ用意する．
  - 素数が記されたカードを入れる箱を用意する．
  - 以下，カードが残っている限り以下を繰り返す．
    - 残っているカードのうち 1 番小さい数値のカードを素数の箱に入れる．
    - そのカードの倍数が書かれているカードを捨てる．
- この方針に基づいて，n 以下の素数を列挙するアルゴリズムを箇条書きで記述してみよう！

エラトステネスの篩の解答例

- エラトステネスの篩を Eratosthenes として関数っぽく以下に記す．
- Eratosthenes では，変数の参照，変数への代入，加算，乗算を基本演算として仮定している．
- P は素数を入れる箱に対応する変数である．そして q_i はカードに対応する変数であり，素数のとき True，素数でないとき False としている．

Eratosthenes(n):

Step 1 P を空集合とする．

Step 2 それぞれの $i \in \{2,3,\ldots,n\}$ に関して，q_i を True とする．

Step 3 それぞれの $i \in (2,3,\ldots,n)$ に関して，以下を繰り返す．

　Step 3-1 もし q_i が True ならば，以下を行う．

　　Step 3-1-1 P に i を加える．

　　Step 3-1-2 i の倍数であるようなそれぞれの j（ただし n 以下）に関して，q_j を False とする．

Step 4 P を出力し終了する．

演習問題: ユークリッドの互除法の復習[3]

1. 与えられた正整数 x, y の最大公約数を見つけるアルゴリズムであるユークリッドの互除法 `Euclid` を以下に記す.

`Euclid`(x, y):

Step 0 x に $\max\{x, y\}$ を，y に $\min\{x, y\}$ を代入する.
Step 1 $y > 0$ の間，以下を繰り返す.

Step 1-1 [　　]

Step 2 x を出力して終了する.

空欄を埋めよ.

2. 2993 と 2911 の最大公約数は [　　] である．空欄を埋めよ.

[3] 2014 年度期末試験問題より

Gale-Shapley アルゴリズムは必ず終わる

定理

Gale-Shapley アルゴリズムは必ず終わる.

証明.

- 背理法で示す.
- Gale-Shapley アルゴリズムが終わらないと仮定する.
- このとき，全ての女性にプロポーズしたにもかかわらずフリーなままの男性がいることになる.
- これはおかしい．なぜならば,
  - どの女性も一度プロポーズされたなら,
    その後はずっといずれかの男性と婚約している.
  - 少なくとも一人の男性が全ての女性にプロポーズした
    ならば，どの女性もいずれかの男性と婚約しているはずである.
  - 入力の仮定より，男性と女性の数は同じなのでその男性が
    フリーであるはずがない.

□

Gale-Shapley アルゴリズムは安定マッチングを出力する

定理

Gale-Shapley アルゴリズムは必ず安定マッチングを出力する.

証明.

- 背理法で示す．すなわち，Gale-Shapley アルゴリズムの出力にブロッキングペアが含まれると仮定して矛盾を導く.
- 有葉（α），米太（β），英子（A），美依子（B）がいて，最終的に (α, A), (β, B) のペアができて，α は A よりも B が好きであり，B は β よりも α が好きであるとする.
- このとき α は A よりも先に B にプロポーズしたはずである.
- それにもかかわらず最終的に A と婚約したのは B に断られたからである.
- B が α を断ったのは，α よりも好きな誰かにプロポーズされたからである．あるいはすでにその誰かと婚約していたからである.
- いずれにしろ，最終的に B が β と婚約しているということは，その誰かよりも β の方が好きだからである，あるいはその誰かが β 本人だからである.
- いずれにしても B が α よりも β の方が好きということになり，最初の仮定と矛盾する.

□

演習問題: 安定マッチングの変種

- 男女がそれぞれ n 人である安定マッチング問題を考える.
- ただし，その入力において，男女のそれぞれには k 人の地球人と $n-k$ 人の火星人が含まれているとする．ここで $1 \leq k < n$ とする.
- しかも，どの人の好みにおいても，地球人が火星人よりも好まれているとする.
- この状況において，地球人と火星人のペアを含まない安定マッチングを必ず出力するアルゴリズムは設計できるであろうか？

  ヒント　地球人と火星人のペアを含む安定マッチングとはどのようなものかを考えてみよう．Gale-Shapley アルゴリズムをそのまま適用するとどうなるかな？

安定マッチングの変種の解答例

Gale-Shapley アルゴリズムをそのまま使用すれば良い．なぜならば，

- 地球人と火星人のペアを含む完全マッチングは安定でない．
  - 地球人と火星人をペアとして含む安定マッチングが存在すると仮定して矛盾を導く．
  - 男性集合を $\{m_1, m_2, \ldots, m_n\}$，女性集合を $\{w_1, w_2, \ldots, w_n\}$ とする．そして，$m_1, \ldots, m_k$ と $w_1, \ldots, w_k$ を地球人，残りを火星人とする．
  - 仮に，(m_i, w_j) が安定マッチングに含まれ，$1 \leq i \leq k$ かつ $k < j \leq n$ とする．（男性地球人と女性火星人がペアになっているとする．）このとき，少なくとも 1 人の地球人女性 $w_{j'}$（すなわち $1 \leq j' \leq k$）が存在し，火星人男性とペアになっているはずである．問題設定より，すべての地球人はどの火星人よりも好まれているはずなので $(m_i, w_{j'})$ はブロッキングペアとなり，矛盾となる．
- Gale-Shapley アルゴリズムは，どのような嗜好に対しても必ず安定マッチングを出力する．よって，この安定マッチング問題の変種においても必ず安定マッチングは存在し，それは地球人と火星人を含まない．

さらなる勉強のために

- チキンカレー（4 人分）のレシピは,「艦めし」
  `http://www.mod.go.jp/msdf/kanmeshi/`
  の大湊基地業務隊によるチキンカレー（4 人分）
  `http://www.mod.go.jp/msdf/kanmeshi/menu/cr/006/index.html`
  からの引用である.
- Circle method は，前回に引き続き [宮代隆平, 2007] からの引用である.
- Gale-Shapley アルゴリズムの記述および正当性，そして安定マッチングの変種の演習問題は [Kleinberg and Tardos, 2005] からの引用である.
- Al Khwarizmi と乗算に関しては，引き続き [Dasgupta et al., 2006] からの引用である.

参考文献

[Dasgupta et al., 2006] Dasgupta, S., Papadimitriou, C., and Vazirani, U. (2006).
Algorithms.
McGraw-Hill Science/Engineering/Math.

[Kleinberg and Tardos, 2005] Kleinberg, J. and Tardos, E. (2005).
Algorithm Design.
Addison-Wesley.

[宮代隆平, 2007] 宮代隆平 (2007).
総当りリーグ戦とグラフ理論.
オペレーションズ・リサーチ, 52:547–550.