データ構造とアルゴリズム

04 再帰アルゴリズム

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

計算複雑度の続き

- 計算複雑度とランダムアクセスの仮定

再帰アルゴリズム

- 再帰アルゴリズムの定義
- ユークリッドの互除法
- ハノイの塔
- 演習問題
- 文献紹介

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

ランダムアクセスの仮定

- ある程度複雑なアルゴリズムにおいては，多くの場合，数値の列や集合を扱う．列（sequence）や集合の参照や代入においては，現在一般に用いられている計算機を意識し，ランダムアクセスを仮定する．すなわち
  - 列 $(x_1, x_2, \ldots, x_n)$ の要素 x_i の参照や代入は，添字 i が指定されているならば，1 回の基本演算とみなす．
  - 列の追加や削除も同様である．
  - 添字が指定されていないものの参照や代入は，1 回の基本演算でできるとは限らないとする．例えば，$(x_1, x_2, \ldots, x_n)$ のうちその値が y のものを参照する，という場合には，何らかの方法で $x_i = y$ を見たす i を探した後に参照する．
  - 要素の順番を考慮しない集合，すなわち普通の集合は，アルゴリズム上（あるいは解析上）の不都合が生じない限り列として計算複雑度を解析する．
  - 離散的な関数も，基本的には，列で表現されているとして計算複雑度を解析する．
  - ……
- 以上の仮定のもとで，Gale-Shapley アルゴリズムの時間複雑度を見積もってみる．

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

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 $N_{\mathsf{W}}(m)$ 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 が空集合でない限り，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 $N_{\mathsf{W}}(m)$ 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 $R_{\mathsf{M}}(m,w) = N_{\mathsf{W}}(m)$ 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 $R_{\mathsf{W}}(w,m) < R_{\mathsf{W}}(w,s(w))$ 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 $R_{\mathsf{W}}(w,m) > R_{\mathsf{W}}(w,s(w))$ 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．

Gale-Shapley アルゴリズムの時間複雑度の単純な見積り

- Gale-Shapley アルゴリズムの時間複雑度を単純に見積もると以下の通りである．
  - Step 1 は $\mathrm{O}(n)$ である．
  - Step 2 は $\mathrm{O}(n)$ である．
  - Step 3 は $\mathrm{O}(n)$ である．
  - Step 4 は Step 4-1 から Step 4-5 を $\mathrm{O}(n^2)$ 回繰り返す．
    - Step 4-1 は $\mathrm{O}(1)$ である．
    - Step 4-2 は $\mathrm{O}(1)$ である．
    - Step 4-3 は $\mathrm{O}(n)$ である．
    - Step 4-4 は $\mathrm{O}(1)$ である．
    - Step 4-5 は Step 4-5-1 と Step 4-5-2 を $\mathrm{O}(1)$ 回繰り返し，Step 4-5-1 は $\mathrm{O}(1)$ であり，Step 4-5-1 は $\mathrm{O}(1)$ である．
  - Step 5 は $\mathrm{O}(n)$ である．
- すなわち時間複雑度は全体で $\mathrm{O}(n^3)$ である．
  - ここで，時間複雑度の意味で最も負荷が高いのは Step 4-3 である．

Gale-Shapley アルゴリズムの時間複雑度の見直し

- Gale-Shapley アルゴリズムに前処理として以下の Step 0 を加える．
  Step 0 男性 $m \in M$ とランキング $r \in \{1, \ldots, n\}$ を引数として，そのランキングの女性 $w \in W$ を返す関数（あるいは表）$f : M \times \{1, \ldots, n\} \to W$ を作成する．
- この Step 0 を加えたものとして，Gale-Shapley アルゴリズムの時間複雑度を再び見積もると以下の通りである．
  - Step 0 は $O(n^2)$ である．Step 1 は $O(n)$ である．
  - ……
  - Step 4 は Step 4-1 から Step 4-5 を $O(n^2)$ 回繰り返す．
    - Step 4-1 は $O(1)$ である．
    - ……
    - Step 4-3 は $O(1)$ である．
    - ……
- すなわち時間複雑度は全体で $O(n^2)$ である．
  - なお，空間複雑度も $O(n^2)$ である．
- このように，アルゴリズムの計算複雑度の解析においては，アルゴリズムの本質を変えない範囲で出来る限りの工夫は行う．

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

再帰アルゴリズム

定義 (再帰表現と再帰アルゴリズム)

アルゴリズム（あるいは関数）の内部で自分自身を用いる（呼び出す）表現方法を再帰（recursion）表現という．再帰表現されたアルゴリズムを再帰アルゴリズム（recursive algorithm）という．

- **最大値を出力する再帰アルゴリズムを** `MaxRecursive` **として関数っぽく以下に記す．**
- **再帰アルゴリズム** `MaxRecursive` **では，変数の参照，値の大小比較，アルゴリズムの呼び出しを基本演算と仮定している．**
- `MaxRecursive` **の時間複雑度は** $\mathrm{O}(n)$ **である．**
- **既出の** `Maximum` **と** `MaxRecursive` **は本質的には同じである．**

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

Step 1 **もし** $n = 1$ **ならば，** x_1 **を出力して終了する．**

Step 2 **もし** $n > 1$ **ならば，** `MaxRecursive`$(\{x_1, \ldots, x_{n-1}\})$ **の出力結果と** x_n **の大きい方を出力して終了する．**

再帰アルゴリズムのまとめ

- 再帰で表現する方が簡潔に記述可能な場合がある．
- 多くのプログラミング言語では再帰の表現をそのまま使える．
  - ただし「深すぎる」再帰は stack overflow となるので注意する [1]．
- 再帰で表現をすると，アルゴリズムの正当性や計算複雑度が（人間にとって）わかりやすい場合がある．なお，
  - 繰り返しで表現できるものは必ず再帰で表現できる，
  - 再帰で表現できるものは必ず繰り返しで表現できる，

  ということが知られている．

[1] stack overflow の意味に関しては，3 回くらい後の講義で説明する．

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

ユークリッドの互除法の再帰表現

- 以下では簡単のため，最大公約数問題の入力 x, y に関しては，$x \geq y$ を仮定する．
- ユークリッドの互除法 Euclid を以下に再掲する．

Euclid(x, y):【再掲】

Step 1　$y > 0$ である限り，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．

- ユークリッドの互除法を再帰表現したものを EuclidRecursive として関数っぽく以下に記す．
- EuclidRecursive では，整数の大小比較，剰余，アルゴリズムの呼び出しを基本演算と仮定している．

EuclidRecursive(x, y):

Step 1　もし $y = 0$ ならば，x を出力して終了する．
Step 2　もし $y > 0$ ならば，EuclidRecursive$(y,\ x \bmod y)$ の出力結果を出力して終了する．

ユークリッドの互除法が必ず正答を出力することの証明

- 以下の定理が古くから知られている．
- `EuclidRecursive` はこの定理をそのまま実行しているに等しい．

定理

$x, y \in \mathbb{N},\ x \geq y \quad \Longrightarrow \quad GCD(x,\ y) = GCD(y,\ x \bmod y)$

証明.

- x を y で割った商を q とすると，$x \bmod y$ は $x - qy$ と表せる．
- x も y も割り切れる数は $x - qy$ も割り切れる．
  $\therefore \mathrm{GCD}(x, y) \leq \mathrm{GCD}(y, x - qy)$
- $x - qy$ も y も割り切れる数は x も y も割り切れる．
  $\therefore \mathrm{GCD}(x, y) \geq \mathrm{GCD}(y, x - qy)$

□

ユークリッドの互除法の時間複雑度

- **割り算の余りに関して，以下の補題が古くから知られている．**

Lemma (**割り算の余りの桁数**)

$x \geq y \Longrightarrow (x \bmod y) < x/2$

証明.

- $y \leq x/2 \Longrightarrow (x \bmod y) < y \leq x/2$
- $y > x/2 \Longrightarrow (x \bmod y) = x - y < x/2$

□

- **入力の自然数 x は（そしてもちろん y も）n bit で表現されているとする．**
- **この補題より，再帰版ユークリッドの互除法 `EuclidRecursive` では，$O(n)$ 回の基本演算を行っていることがわかる．**

ユークリッドの互除法の時間複雑度の厳密な見積り

- ユークリッドの互除法の時間複雑度を論じるような場面では，割り算の余り（剰余）の計算を基本演算とするのは適切ではないかもしれない．
- より基本的な（と思われる）演算である，参照，代入，数値の大小比較，加減算，「2 で割った商と余り」を基本演算とすると，n bit の整数の剰余演算は $\mathrm{O}(n^2)$ の時間複雑度でできる．（演習問題）
- この場合，ユークリッドの互除法の時間複雑度は $\mathrm{O}(n^3)$ と見積もれる．

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

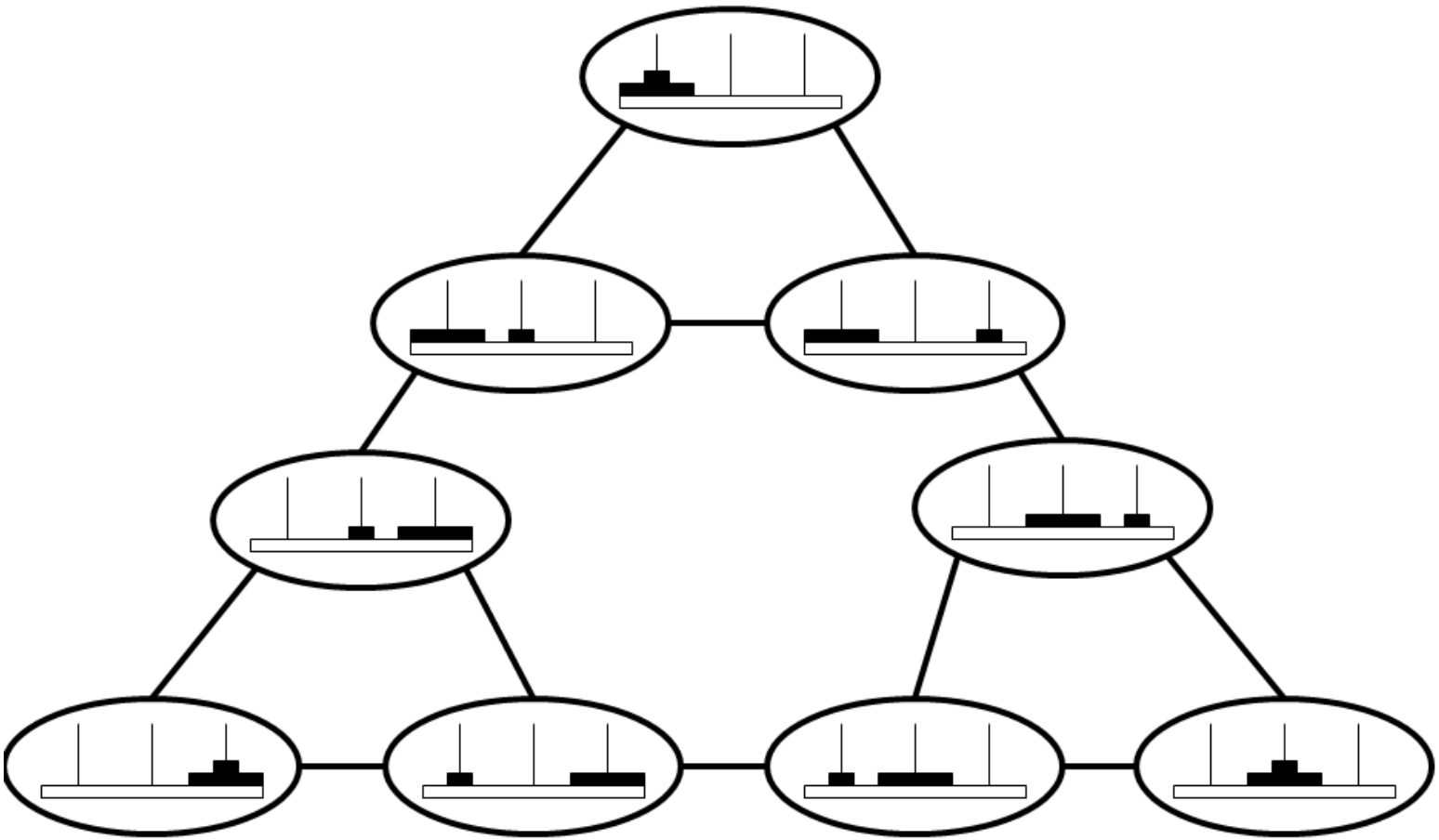
再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

ハノイの塔（the tower of Hanoi）

天地創造の時，神は 64 枚の純金の円盤を 3 本のダイヤモンドの柱のうちの 1 本に立てた．円盤は大きいものが下にあり，上に行くに従って小さくなっていた．ブラーマ寺院の僧侶たちは，定めによって，昼夜を問わず，円盤を別の柱に移し変えている．僧侶たちは 1 度に 1 枚の円盤を動かすことができる．そして，動かした円盤の下に，それよりも小さいものがあってはならない．彼らがその仕事を終えたとき，寺院も僧侶も砕け散って塵となり，またそのとき世界も終わりを告げるという・・・

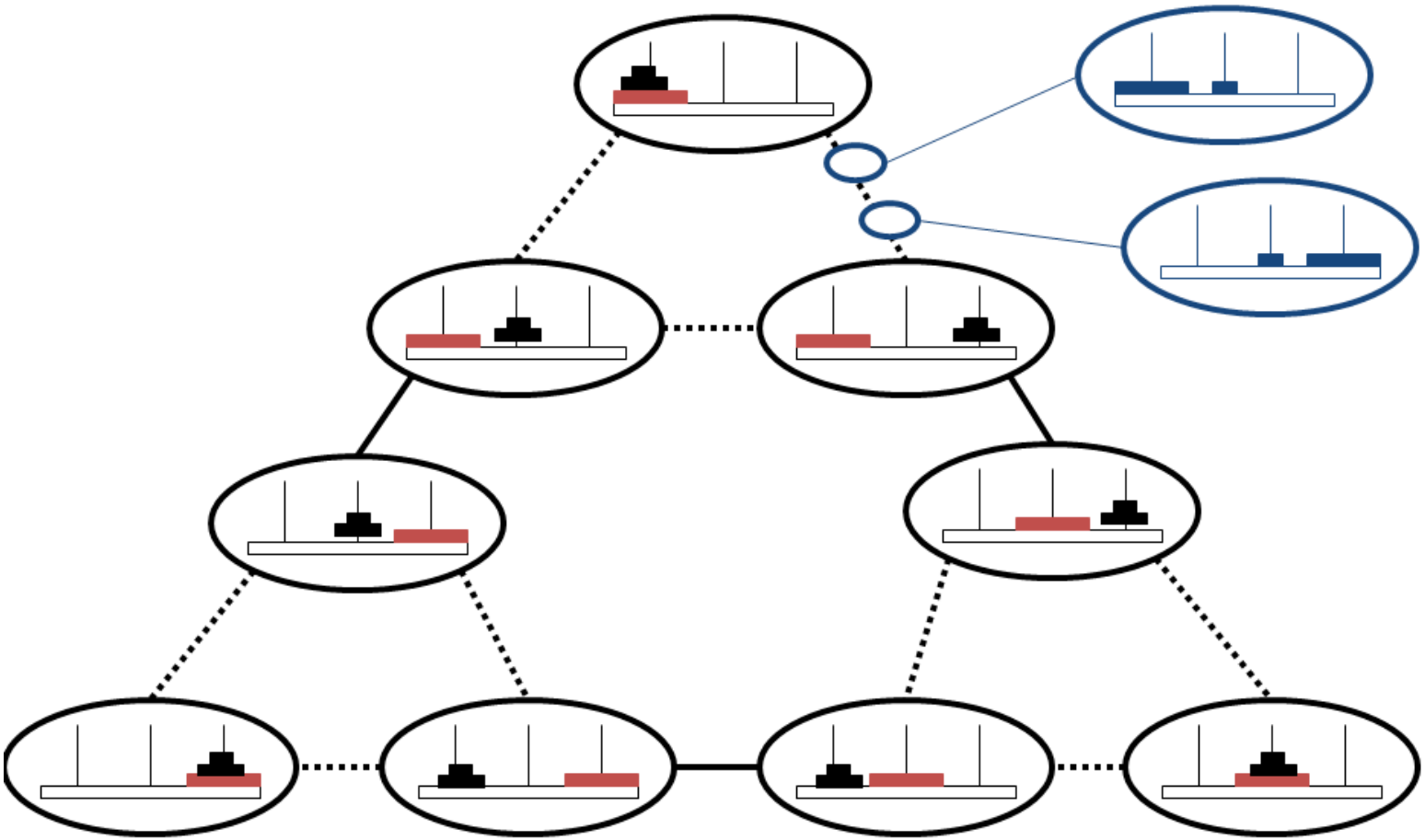
ハノイの塔の状態遷移を図で表現

円盤が 2 枚の場合



ハノイの塔の状態遷移を図で表現

円盤が 3 枚の場合



ハノイの塔を正確に表現

問題 (ハノイの塔)

入力 X, Y, Z の 3 本の柱と n 枚の円盤からなるハノイの塔．
- ただし，円盤は全て柱 X に積まれている．

出力 全ての円盤が柱 Y に積まれているハノイの塔．

- 大きい円盤が小さい円盤の上に来てはいけないと仮定する．
- 円盤 1 枚を，いずれかの柱からいずれかの柱に移動することを基本演算と仮定する．
- 最小の基本演算回数で出力したい．

ハノイの塔の解法を再帰で表現

ハノイの塔の解法を再帰で表現したものを `HanoiRecursive` **として関数っぽく以下に記す．**

`HanoiRecursive`(X, Y, Z, n):

Step 1 **もし** $n = 1$ **ならば，円盤を柱** X **から柱** Y **に移動して終了する．**

Step 2 **もし** $n > 1$ **ならば，以下を実行し終了する．**

- Step 2-1 **小さい方から** $n - 1$ **枚の円盤に対して** `HanoiRecursive`$(X, Z, Y, n-1)$ **を実行する．**
- Step 2-2 1 **番大きい円盤を柱** X **から柱** Y **に移動する．**
- Step 2-3 **小さい方から** $n - 1$ **枚の円盤に対して** `HanoiRecursive`$(Z, Y, X, n-1)$ **を実行する．**

再帰表現を元に時間複雑度を計算

- `HanoiRecursive`(X, Y, Z, n) **の円盤移動（あるいは基本演算）回数を** $T(n)$ **とする．**
- $T(1) = 1$
- $n > 1$ **のとき** $T(n) = 2 \cdot T(n-1) + 1$
- $\therefore T(n) = 2^n - 1$

ハノイの塔に関するコメント

- ハノイに塔はない．フランスの数学者リュカによる創作である．
- 円盤が 2 枚 $\Rightarrow$ 円盤移動 3 回
- 円盤が 3 枚 $\Rightarrow$ 円盤移動 7 回
- 円盤が n 枚 $\Rightarrow$ 円盤移動 $2^n - 1$ 回
  - $2^{64} - 1 = 18,446,744,073,709,551,615$ であり，1 秒に 1 回の移動が可能だとしても，伝説における世界の終焉までには 5,800 億年以上かかる [2]．
- ルービックキューブなども原理的には同じ方法が適用可能だが・・・

[2] ハノイの塔の変種を考えるのも面白い．例えば，ハノイの塔の円盤移動法に以下のルールを加える．「柱の名前を A, B, C とする．円盤は A から B へ，B から C へ，C から A へのみ移動できる．」この時，n 枚の円盤を A から C へ全て移動するのに最低限必要な円盤移動回数は何回か？ なお，この問題の解答は少々複雑なので興味ある方々の自習に任せる．

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

問題: フィボナッチ数列の単純な計算の時間複雑度

漸化式

$$F_n = \begin{cases} 0 & (n = 0) \\ 1 & (n = 1) \\ F_{n-1} + F_{n-2} & (n \geq 2) \end{cases}$$

により定義される数列を Fibonacci **数列という．非負整数** $n \in \mathbb{Z}_{\geq 0}$ **を入力として** Fibonacci **数列の第** n **項を出力する単純なアルゴリズムを** `Fibonacci1` **として関数っぽく以下に記す．**

`Fibonacci1`(n):

Step 1 **もし** $n = 0$ **ならば，**0 **を出力して終了する．**

Step 2 **もし** $n = 1$ **ならば，**1 **を出力して終了する．**

Step 3 **もし** $n \geq 2$ **ならば，**`Fibonacci1`$(n-1)$ **と** `Fibonacci1`$(n-2)$ **の出力結果の和を出力して終了する．**

このアルゴリズム `Fibonacci1` **の時間複雑度を** n **の関数で表わせ．**

フィボナッチ数列の計算の解答例

Fibonacci1(n) の時間複雑度を $T(n)$ とすると

$$T(n) = \begin{cases} 1 & (n = 0) \\ 1 & (n = 1) \\ T(n-1) + T(n-2) + 4 & (n > 1) \end{cases}$$

となる．ここでは，比較，四則演算を基本演算としている．3 項間漸化式を解くと $T(n) = \mathrm{O}\left(\left(\frac{1+\sqrt{5}}{2}\right)^n\right)$ となる．なお，これは Fibonacci 数そのものの第 n 項を O 記法で表したものと同じである．

商と剰余の計算複雑度

商と剰余の計算を問題として正確に以下に記す．

問題 (商と剰余)

入力 n *bit* **で表された整数** $x, y \in \mathbb{Z}$ **（ただし** $y \geq 1$**）．**
出力 **商** q $(= \lfloor x/y \rfloor)$**，余り** r $(= x \bmod y)$**．**

商と剰余の計算複雑度の続き

商と剰余の計算を再帰的に行うアルゴリズム `ModuloRecursive` **を以下に記す．**

`ModuloRecursive`(x, y):

Step 1 **もし** $x = 0$ **ならば，**(q, r) **として** $(0, 0)$ **を出力して終了する．**

Step 2 (q, r) **に** `ModuloRecursive`$(\lfloor x/2 \rfloor, y)$ **の出力結果を代入する．**

Step 3 q **と** r **をそれぞれ** 2 **倍にする．**

Step 4 **もし** x **が奇数ならば，**r **を** 1 **増やす．**

Step 5 **もし** $r \geq y$ **ならば，**r **に** $r - y$ **を代入し** q **を** 1 **増やす．**

Step 6 (q, r) **を出力して終了する．**

このアルゴリズム `ModuloRecursive` **の時間複雑度を** n **の関数で表わせ．**

商と剰余の計算複雑度の解答例

n bit 整数を入力とする剰余アルゴリズム $\texttt{ModulaRecursive}(x, y)$ の時間複雑度を $T(n)$ とすると

$$T(n) = \begin{cases} \mathrm{O}(1) & (n = 1) \\ T(n-1) + \mathrm{O}(n) & (n > 1) \end{cases}$$

と表せる．よって，O 記法の定義より $\alpha, \beta \in \mathbb{R}_{>0}$ が存在して

$$\begin{aligned} T(n) &\leq T(n-1) + \alpha n + \beta \leq T(n-2) + \alpha(n-1) + \beta + \alpha n + \beta \\ &\leq T(1) + \alpha(2 + 3 + \cdots + n) + \beta(n-1) \end{aligned}$$

である．さらに O 記法の定義より $\gamma \in \mathbb{R}_{>0}$ が存在して

$$T(n) \leq \gamma + \alpha(2 + 3 + \cdots + n) + \beta(n-1)$$

である．よって

$$T(n) = \mathrm{O}(n^2)$$

である．

おまけ

ルール追加後のハノイの塔において，n 枚の円盤を A から B に移動するアルゴリズムの時間複雑度を $S(n)$，A から C に移動するアルゴリズムの時間複雑度を $T(n)$ とすると

$$S(n) = \begin{cases} 1 & (n = 1) \\ 2 \cdot T(n-1) + 1 & (n \geq 2) \end{cases}$$

$$T(n) = \begin{cases} 2 & (n = 1) \\ 2 \cdot T(n-1) + S(n-1) + 2 & (n \geq 2) \end{cases}$$

となる．よって漸化式を解くと

$$T(n) = \frac{9+5\sqrt{3}}{6}(1+\sqrt{3})^{n-1} + \frac{3-2\sqrt{3}}{6}(1-\sqrt{3})^{n} - 1 = O((1+\sqrt{3})^{n})$$

である．

計算複雑度の続き
計算複雑度とランダムアクセスの仮定

再帰アルゴリズム
再帰アルゴリズムの定義
ユークリッドの互除法
ハノイの塔
演習問題
文献紹介

さらなる勉強のために

- Gale-Shapley **アルゴリズムの時間複雑度の解析は引き続き** [Kleinberg and Tardos, 2005] **からの引用である．**
- **ユークリッドの互除法に関する定理は有名なので至るところで見かけるが，このスライドでは** [Dasgupta et al., 2006] **から引用した．剰余計算のアルゴリズムも同様である．**
- **ハノイの塔もまた有名であるが，このスライドでは** [**久保幹雄**, 2000] **から引用した．**

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.

[久保幹雄, 2000] 久保幹雄 (2000).
組合せ最適化とアルゴリズム.
共立出版.