簡潔データ構造 第 0 回

marimo

2023 年 4 月 18 日

言葉と記号の準備

アルゴリズムとは

何かしらの問題を解くための，機械的な操作からなる有限の手続き
個別のプログラムとは異なる
例 : ユークリッドの互除法

- 教科書にのっている手続きはアルゴリズム
- これを C++ とかで書いたものは互除法の実装であってアルゴリズムではない

データ構造とは

普通，アルゴリズムはデータを扱う
機械の上でのデータの持ち方と索引を合わせて**データ構造**とよぶ
データ構造とアルゴリズムは不可分

アルゴリズムの良さ

アルゴリズムの良し悪しをどのように測るか？

- 速い (実行時間が短い)
- 使用メモリが少ない

実際の機械に依存する議論はしたくないので，理想化された計算機を考えたい

計算モデル

計算を理論的に考えるための数理モデル

現実の機械に似せたものから抽象的なものまでいろいろある

- チューリングマシン
- RAM(Random Access Machine)
- ラムダ計算
- 有限オートマトン
- etc

例 :RAM

計算量解析によく使うモデル。陽に書いてなければだいたいこれ。
おおざっぱに言えば

- メモリは沢山の「セル」から成る
- メモリの 1 つのセルには任意の桁数の整数が入る
- 1 つのセルの読み出し，書き込みは単位時間でできる
- 任意の桁数の整数の加減乗除およびビット演算が単位時間でできる
- 制御も単位時間でできる
    - 具体的には条件分岐とジャンプと停止

Word-RAM

さっきの RAM モデルは実際の計算機よりも強すぎるので，いくらか実際に寄せたものとして Word-RAM モデルというものが存在する。
雑に言えばさっきの RAM の操作を語長 (word-size) 単位に制限したもの。

- パラメータとして語長 w を持つ
- メモリはビット列
  - 長さは高々 $2^w \,\mathrm{bits}$
  - ポインタが $w \,\mathrm{bits}$ になる
- メモリの連続する $w \,\mathrm{bits}$ の読み出し，書き込みが単位時間でできる
  - 開始地点はどこでもいい
- $w \,\mathrm{bits}$ の整数の加減乗除およびビット演算が単位時間でできる
- 制御も単位時間

この講座？では計算量を議論するときこれを使います

計算量

ある計算モデルの上でアルゴリズムを動かしたとき，どれだけの時間と空間を使うか調べられる。これは入力に対する関数になる。
これらをそれぞれ**時間計算量**[*1]，**空間計算量**と呼ぶ。
大抵の場合は精密な値よりも入力サイズに対する増加のスピードの方が興味があるので，次に説明するオーダー記法で書かれがち。

1 しばしば時間計算量を単に計算量と言う

ランダウの記号

ある関数 $g(n)$ に対して次の集合を定める。

- $O\big(g(n)\big) = \left\{ f \;\middle|\; \exists c \geq 0, \lim_{n\to\infty} \dfrac{f(n)}{g(n)} \leq c \right\}$
- $o\big(g(n)\big) = \left\{ f \;\middle|\; \lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 0 \right\}$

ランダウの記号

ある関数 $g(n)$ に対して次の集合を定める。

- $O\big(g(n)\big) = \left\{ f \;\middle|\; \exists c \geq 0, \lim_{n\to\infty} \dfrac{f(n)}{g(n)} \leq c \right\}$
- $o\big(g(n)\big) = \left\{ f \;\middle|\; \lim_{n\to\infty} \dfrac{f(n)}{g(n)} = 0 \right\}$

たいていの場合

$$f(n) \in O\big(g(n)\big) \Leftrightarrow f(n) = O\big(g(n)\big)$$

$$\exists h(n) \in O\big(g(n)\big), f(n) = f'(n) + h(n) \Leftrightarrow f(n) = f'(n) + O\big(g(n)\big)$$

などと濫用される。

ランダウの記号

いくらか例を見ましょう。記号を濫用しているので注意。

$$
\begin{aligned}
n &= O(n) \\
n &\neq o(n) \\
\log n &= o(n) = O(n) \\
n \log n &= O\left(n^2\right) \\
O(n) \times O\left(\log n\right) &= O\left(n \log n\right) \\
O\left(\frac{n}{\log n}\right) \times O\left(\log n\right) &= O(n) \\
O(n) &= O\left(n^{1.5}\right) = O\left(n^2\right)
\end{aligned}
$$

簡潔データ構造とは

情報理論的下限

大きさ L の集合の中の 1 つの要素を表現するために必要なビット数の**情報理論的下限**を $\lceil \lg L \rceil \,\mathrm{bits}$ と定義する。

たとえば $A = \{1, 2, ..., \sigma\}$ の長さが n の列 $S \in A^n$ の情報理論的下限は $\lceil n \lg \sigma \rceil \,\mathrm{bits}$ である。一方で自明な文字列の表現は $n \lceil \lg \sigma \rceil \,\mathrm{bits}$ であるので，σ が 2 ベキのときだけ下限と一致する。

適当な表現によって集合の中の**ある**要素はこれより短く表現できるかもしれないが，**全て**の要素をこれより短く表現することはできない。*2

簡潔データ構造

簡潔データ構造 (succinct data structure) はデータの**簡潔表現** (succinct representation) と**簡潔索引** (succinct index) から成る。

簡潔データ構造

簡潔データ構造 (succinct data structure) はデータの**簡潔表現** (succinct representation) と**簡潔索引** (succinct index) から成る。

大きさ L の集合 U に対する**簡潔表現**とは，任意の $x \in U$ に対して x を表現するためのビット数が $\lg L + o(\lg L)$ bits である表現のことである。

任意の有限集合に対してこのような表現は存在する。

簡潔データ構造

簡潔データ構造 (succinct data structure) はデータの**簡潔表現** (succinct representation) と**簡潔索引** (succinct index) から成る。

大きさ L の集合 U に対する**簡潔表現**とは，任意の $x \in U$ に対して x を表現するためのビット数が $\lg L + o(\lg L)\,\mathrm{bits}$ である表現のことである。

任意の有限集合に対してこのような表現は存在する。
簡潔索引とは次の要件を満たす構造のことである。

- 構造のビット数が $o(\lg L)\,\mathrm{bits}$ である
- クエリの時間計算量が従来のデータ構造と同程度か $\log n$ のベキがかかる程度である

うれしさ

Q. そうやってメモリ切り詰めて何がうれしいの？今は富豪的プログラミングの時代だよ？
クソデカデータを扱うときにうれしかったりします。たとえば DNA の全文を検索可能な形で持ちたいとか，IME で大きめの辞書を持ちたいとか[*3]，そういうときに使います。

3 Google/mozc では LOUDS 表現による Trie 木が実装されています

予定

たぶん次回から具体的なデータ構造のおはなしをします。一旦静的なものだけを扱います。

- Succinct Bit Vector
- Fully Indexed Dictionary
  - これはやるかわからない
- 順序木の表現
  - LOUDS はやります。BP の $O(1)$ 索引は大変なのでしないと思います。
- Wavelet Matrix

データ構造をひとつお話をし終わったら実装パートをはさみたい

予定

実装パートをどうするかはまだ未定です

- もくもく作業会にしてしまう？
- プログラムを書くことについてどれぐらいサポートするか
- 言語は GC の無いものが良いと思います
- 実装上のテクニックの話はしたいです