アルゴリズムとデータ構造

分割統治法とソート

森 立平
mori@c.titech.ac.jp

2018 年 6 月 26 日

今日のメッセージ

- **分割統治法の時間計算量のは漸化式を解けば得られる**
- **クイックソートはランダムにピボットを選べば平均 $O(n \log n)$ 時間で計算できる**
- **中央値の中央値アルゴリズムを使えば $O(n)$ 時間で k 番目に小さい値が計算できる**

今日の目標

- クイックソート、クイックセレクト、中央値の中央値アルゴリズムを理解する

今日の演習の目標

- クイックソート、クイックセレクトなどのプログラムを書けるようになる

今日の主な演習課題 (提出締切は来週火曜日正午 (12:00))

1. クイックソートとクイックセレクトのプログラムを書く

今日の演習時間のワークフロー

1. この資料をよく読み、`alg2018/sort` にある課題をやる

1 分割統治法の時間計算量

分割統治法の時間計算量は漸化式を立てることによって見積る。長さ n の配列に対するマージソートの時間計算量 $\chi(n)$ はある定数 $c > 0$ について

$$\chi(n) = \chi(\lfloor n/2 \rfloor) + \chi(\lceil n/2 \rceil) + cn$$

を満たす。

補題 1. $\chi(n)$ は n について単調非減少である。

Proof. 明らかに $\chi(0) \le \chi(1)$ が成り立つ。$k \ge 1$ とおく。$n+1 \le k$ について $\chi(n) \le \chi(n+1)$ が成り立つとするときに、$\chi(k) \le \chi(k+1)$ を示せばよい。k が偶数のとき、$\chi(k) = 2\chi(k/2) + ck$ と $\chi(k+1) = \chi(k/2) + \chi(k/2+1) + c(k+1)$ が成り立つ。帰納法の仮定より $\chi(k/2) \le \chi(k/2+1)$ なので $\chi(k) \le \chi(k+1)$ が成り立つ。k が奇数のとき、$\chi(k) = \chi((k-1)/2) + \chi((k+1)/2) + ck$ と $\chi(k+1) = 2\chi((k+1)/2) + c(k+1)$ が成り立つ。帰納法の仮定より $\chi((k-1)/2) \le \chi((k+1)/2)$ なので $\chi(k) \le \chi(k+1)$ が成り立つ。 □

よって $m = 2^{\lceil \log n \rceil}$ とおけば (m は n を 2 冪に切り上げたもの)、$\chi(n) \leq \chi(m)$ が成り立つ。m が 2 冪であることから $\chi(m) = O(n \log n)$ が分かる。よって $\chi(n) = O(n \log n)$ である。

長さ n のソート済み配列に対する二分探索の時間計算量 $\chi(n)$ はある定数 $c > 0$ について

$$\chi(n) \leq \chi(\lceil n/2 \rceil) + c$$

を満たす。マージソートの場合と同様に χ の単調性が示せるので、$\chi(n) \leq \chi(m)$ であり、

$$\chi(m) \leq \chi(m/2) + c \leq \chi(1) + c \log m = \chi(1) + c\lceil \log n \rceil$$

が得られる。よって $\chi(n) = O(\log n)$ である。

2 クイックソート

クイックソートは次の漸化式に基づく。

$$\mathrm{Qsort}(A) = \begin{cases} [\,], & \text{if } |A| = 0 \\ (B, C) := \mathrm{split}(A \setminus a_p, a_p),\ \mathrm{Qsort}(B) \circ \mathrm{Qsort}(C) & \end{cases}$$

ここで、a_p は配列 A の要素の一つであり、$\mathrm{split}(A, a)$ は配列 A を a 以下の値からなる配列 B と a より大きい値からなる配列 C へ分割する関数である。a_p の選択の仕方によって時間計算量は大きく変化する。

3 クイックソートの時間計算量

クイックソートの漸化式に現れる a_p のことをピボットと呼ぶ。ピボットとしては配列の中央値を選ぶのが最適である。この章では配列に含まれる要素は全て異なる ($\mathtt{A[i]} \leq \mathtt{A[j]}$ かつ $\mathtt{A[i]} \geq \mathtt{A[j]}$ ならば $i = j$) と仮定する。仮に中央値を $O(n)$ 時間で選択できるとすると、マージソートと同じ漸化式が得られるので時間計算量は $O(n \log n)$ である。仮に定数 $\epsilon \in (0, 1/2)$ について ϵn 番目以上 $(1-\epsilon)n$ 番目未満の順番であるものをピボットとして $O(n)$ 時間で選択できるとするとクイックソートの時間計算量は

$$\chi(n) = \chi(\epsilon n) + \chi((1-\epsilon)n) + cn$$

となる (簡単のため切り捨てや切り上げの影響は無視することにする)。多少天下り的ではあるが、ある定数 $d > 0$ について帰納法で $\chi(n) \leq dn \log n$ を示す。$\chi(k) \leq dk \log k$ が $k < \lceil 1/\epsilon \rceil$ に対して成り立つように $d > 0$ をとることができる。このとき、$\chi(k) \leq dk \log k$ が $\lceil 1/\epsilon \rceil \leq k < n$ について成り立っているとすると,

$$\begin{aligned} \chi(n) &= \chi(\epsilon n) + \chi((1-\epsilon)n) + cn \\ &\leq d\epsilon n \log(\epsilon n) + d(1-\epsilon)n \log((1-\epsilon)n) + cn \\ &= dn \log n - dn \left(-\epsilon \log \epsilon - (1-\epsilon) \log(1-\epsilon)\right) + cn \end{aligned}$$

ここでバイナリエントロピー関数 $h(\epsilon) := -\epsilon \log \epsilon - (1-\epsilon) \log(1-\epsilon)$ は $\epsilon \in (0, 1)$ について正である。よって d を十分大きく取り直せば、$dh(\epsilon) \geq c$ となる。よって、$\chi(n) \leq dn \log n$ が得られる。よって、任意の自然数 n について $\chi(n) \leq dn \log n$ である。

実際のクイックソートの実装ではピボットはランダムに選ぶことが一般的である。ランダムに選べば高い確率で ϵn 番目以上 $(1-\epsilon)n$ 番目未満のピボットが選べるので、$O(n \log n)$ 時間で動作する。一様な確率でピボットを選択するクイックソートにおける平均比較回数が $O(n \log n)$ であることを示す。ソートしたい配列に含まれる i 番目に小さい要素を a_i とする。$1 \leq i < j \leq n$ について確率変数 X_{ij} を

$$X_{ij} := \mathbb{I}\{\text{クイックソートの中で } a_i \text{ と } a_j \text{ が比較される}\}$$

と定義する。するとクイックソートの中で比較されるペアの個数は $X := \sum_{1\le i<j\le n} X_{ij}$ で表わされる。

$$\begin{aligned}\mathbb{E}[X] &= \mathbb{E}\left[\sum_{1\le i<j\le n} X_{ij}\right] = \sum_{1\le i<j\le n} \mathbb{E}[X_{ij}] \\ &= \sum_{1\le i<j\le n} \Pr\left(\text{クイックソートの中で } a_i \text{ と } a_j \text{ が比較される}\right)\end{aligned}$$

もしも a_i や a_j よりも先に $a_i < a_k < a_j$ となる a_k がピボットとして選ばれてしまったとすると、a_i と a_j が比較されることはない。すべての要素は一様な確率でピボットとして選ばれるので、

$$\Pr\left(\text{クイックソートの中で } a_i \text{ と } a_j \text{ が比較される}\right) = \frac{2}{j-i+1}$$

となる。よって

$$\begin{aligned}\mathbb{E}[X] &= \sum_{1\le i<j\le n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k} \\ &\le \sum_{i=1}^{n-1} \sum_{k=1}^{n} \frac{2}{k} \\ &\le n \sum_{k=1}^{n} \frac{2}{k} = O(n \log n).\end{aligned}$$

4 クイックセレクト

配列の中で k 番目に大きい要素を計算する問題を「選択問題」と呼ぶ。選択問題はソートを用いれば $O(n\log n)$ 時間で解くことができるが、もっと効率のよい $O(n)$ 時間のアルゴリズムが存在する。

$$\mathrm{Qselect}(A,k) = \begin{cases} (B,C) := \mathrm{split}(A, a_p) \text{と置く} & \\ a_p, & \text{if } |B| = k+1 \\ \mathrm{Qselect}(B,k), & \text{if } |B| \le k \\ \mathrm{Qselect}(C, k-|B|), & \text{otherwise.} \end{cases}$$

```
#define N 10000000

int A[N];

/*
A[0], A[1], ..., A[n-1] の中でk+1番目に小さい値を返す関数。
n >= 1 && k >= 0 && k < n は保証されている。
ただし、Aの中身は並び換えてしまう。
*/
int quick_select(int A[], int n, int k){
  int i, j, pivot;
```

```
// 先頭の要素をピボットとする
  pivot = A[0];
  for(i = j = 1; i < n; i++){
    if(A[i] <= pivot){
      int z = A[j];
      A[j] = A[i];
      A[i] = z;
      j++;
    }
  }
  if(j == k+1) return pivot;
  else if(j < k+1) return quick_select(A+j, n-j, k-j);
  else return quick_select(A+1, j-1, k);
}
```

もしも $O(n)$ 時間で中央値が計算できたとすると、クイックセレクトの時間計算量 $\chi(n)$ は n を 2 の冪とすると

$$\chi(n) = \chi(n/2) + cn = c\left(n + \frac{n}{2} + \frac{n}{4} + \cdots + 2\right) + \chi(1) \leq 2cn + \chi(1) = O(n)$$

である。クイックソートと同様に大体中央にあるようなピボットを選んだ場合も $O(n)$ 時間であることはすぐに分かる。

5　中央値の中央値アルゴリズム

乱数を用いないで決定的に $O(n)$ 時間で中央値を計算することもできる。まず、n 個の要素を 5 つづつ $\lceil n/5 \rceil$ 個のグループに分ける。各グループの中央値を計算して、中央値を集めて長さ $\lceil n/5 \rceil$ の配列を作る。この長さ $\lceil n/5 \rceil$ の配列の中央値を再帰で計算し、その結果をピボットとして選択する。アルゴリズムを以下に示す。

アルゴリズム 1 中央値の中央値アルゴリズムの擬似コード (入力: 整数の配列 A, 非負の整数 k. 出力: 配列 A の $k+1$ 番目に小さい要素.)

if 配列 A の長さが 5 以下 **then**
　解を計算して出力する.
else
　配列 A を長さ 5 づつに分割してそれぞれ中央値を計算し，長さ $\lceil n/5 \rceil$ の配列 A' を作る.
　配列 A' の中央値をピボットとして選択 (再帰呼出).
　配列 A のピボット以外の要素を「ピボット以下のもの」からなる配列 B と「ピボットより大きいもの」からなる配列 C の 2 つの配列に分割する.
　配列 B の要素の数を r とおく.
　if $r == k$ **then**
　　ピボットを解として出力する.
　else if $r < k$ **then**
　　配列 C の中から $k-r$ 番目に小さい要素を解として出力する (再帰呼出).
　else
　　配列 B の中から $k+1$ 番目に小さい要素を解として出力する (再帰呼出).
　end if
end if

このアルゴリズムを「中央値の中央値アルゴリズム」と呼ぶ。中央値の中央値アルゴリズムの時間計算量 $\chi(n)$ は次の漸化式を満たす (切り捨て、切り上げの影響は無視することにする)。

$$\chi(n) = \chi(n/5) + \chi(7n/10) + cn$$

ある定数 $d > 0$ について $\chi(n) \leq dn$ を帰納法で示す。帰納法の仮定を用いると

$$\chi(n) \leq \frac{d}{5}n + \frac{7d}{10}n + cn = \frac{9d + 10c}{10}n.$$

よって $d \geq 10c$ となるような d を選べば、$\chi(n) \leq dn$ を帰納法で示せる。

6 比較に基づかないソートアルゴリズム: カウントソート、バケツソート、基数ソート

7 演習課題

1. クイックセレクトのソースコードを参考にしてクイックソートのプログラムを書け。
2. クイックセレクト及びクイックソートのプログラムを改良して、重複した要素を持つ配列に対しても、それぞれ $O(n)$ 時間、$O(n \log n)$ 時間で動くプログラムを書け。ただし、ソートする配列以外に配列を使ってはならない。
3. [発展的課題] 中央値の中央値アルゴリズムのプログラムを書け。