分割統治法

競技プログラミング輪読回 第 3 回

田渕 智也 <t@tomoyat1.com>

June 02, 2018

分割統治法

概要

- divide and conquer
  - 政治・経済の「分割統治」がおそらく語源
- 問題を分割して統治する
  - 分割: 大きな問題を小さな部分問題に分割する
    - 部分問題: 同じ構造を持つ，より小さな問題
  - 解く: それぞれの部分問題を解く．必要に応じてこれらの部分問題も更に部分問題に分割する．
  - 統治: 個別に解いた部分問題の結果を組み合わせて最終的な答えを得る

例

```
fib 0 = 0
fib 1 = 1
fib n = fib(n-1) + fib(n-2)
```

- `fib(n)` は `fib(n-1) + fib(n-2)` と，部分問題の和で表す (分割)
- 部分問題をそれぞれ解いた上でそれらの結果を足す (統治)
- 分割は，`fib 0` または `fib 1` を求めるときにやめる．この時，それぞれ `0` と `1` を返す．
- 計算量は $O(2^n)$
  - 計算の木を書いたらそれっぽい
  - 証明はぐぐったら出てくる．帰納法を用いる．
  - より厳密な計算量は，Fibonacci 数列の閉じた形式と深く関連している．黄金数とか出てくる．
- これは「動的計画法」を用いることで高速される ->後日

最大値

```
findMaximum A =
    case length A in
        1 -> A[0]
        _ -> max (findMaximum front) (findMaximum end)
    where (front, end) = splitInHalf A
```

- A の前半にある最大の要素と A の後半にある最大の要素を求める（分割）
- 前半の最大と後半の最大値のうち，大きい方を A の最大値とする（統治）
- 分割は A の要素数が 1 になったときにやめる．この時，A の唯一の要素をそのまま返す．

速いソート

- $O(n \log n)$ なソートアルゴリズムたち
  - マージソート
  - クイックソート
- （ソートするリストを $\log n$ 個に分割し，それぞれに対して n の処理を行う）

-> 次週

考察

利点

- 問題を解く見通しが良くなる
  - 問題は同じ構造をしたより小さな問題の組み合わせで再定義できる
- (場合によっては) 計算量が削減できる
- 削減できなくてもより効率の良いアルゴリズムを作る手がかりになる
  - 部分問題が重複する場合，動的計画法を適用できるので
- 並列化の可能性
  - 部分問題を並列に解けば良さそう

注意点

- 終了条件を意識する
  - 無限に問題を分割し続けると無限に計算が続く.
- 再帰回数に注意
  - 例では再帰呼び出しを用いていた.
  - 無限に再帰呼び出しできない言語や処理系が多い.
  - ある程度まで分割し，そこから先は分割統治でない方法で解く
    - どこで戦略を切り替えるかはヒューリスティック・経験・勘・職人技

問題

の前に…

- スライドでは，疑似コードでアルゴリズムを示す
  - それとなく Haskell っぽい
  - 関数型っぽい
- ALDS-2018 のレポジトリの今日のディレクトリに，Golang でのちゃんとした実装がある
- これらは厳密に一致していない部分もある
  - 無駄な先や配列の結合を避けるために少し終了条件を変えたり…

ALDS1_5_A: Exhaustive Search

長さ n の数列 A と整数 m に対して，A の要素の中のいくつかを足し合わせて m を作れるかを判定する.

入力

```
5
1 5 7 10 21
4
2 4 17 8
```

出力

```
no
no
yes
yes
```

解法

```
solve i m =
  case m in
    m == 0 -> True
    i >= length A -> False -- A の要素を全部試した場合
    _ -> solve (i+1) m || solve (i+1) m - A[i]
```

- A の i 番目以降の要素を使って m が作れる場合，以下の 2 通りのいずれかが成立する (分割)
  - A の i+1 番目の要素を使って m が作れる
  - A の i+1 番目の要素を使って m - A[i] が作れる
- 答えは論理和とする (統治)
- 終了条件は，A の要素をすべて吟味し終えた時．この時，False を返す．

考察

- A の各要素に対して，それを用いる場合と用いない場合をそれぞれ試す
  - 2^n 通り試す（n は A の要素数とする）
- 実際はいくつかの m に対して前記の solve を行う
  - 同じ部分問題を複数回解くことになりそう
  - 動的計画法!
    - 実装例にはメモ化再帰した実装例がある

ALDS1_5_C: Koch Curve

整数 n を入力し，深さ n の再帰呼び出しによって生成されるコッホ曲線の頂点の座標を出力する

入力

```
1
```

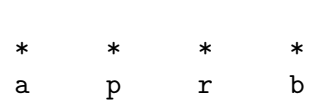
出力

```
0.000000 0.000000
33.333333 0.000000
50.000000 28.867513
66.666667 0.000000
100.000000 0.000000
```

解法

点の列を求める

```
koch d a b =
  case d in
    0 -> [a, b]
    _ -> init (koch (d-1) a p)
         ++ init (koch (d-1) p q)
         ++ init (koch (d-1) q r)
         ++ (koch (d-1) r b)
```



解法

```
  where
    p = (a * 2 + b * 1) / 3  -- (b-a) を 1:2 に内分する点
    r = (a * 1 + b * 2) / 3  -- (b-a) を 2:1 に内分する点
    q = p + rotate(q - p, pi/3)  -- p + (q-p) を半時計回りに pi/3 回転
```

解法

- p，q，r を以下のように取る
  - 1:2，2:1 のの内分点へのベクトルを p，r とする
  - p + $R(\frac{\pi}{2})$（q-p）を q
- (a，b) を両端とする Koch 曲線は (p，a)，(q，p)，(r，q)，(b，r) のそれぞれを両端とする Koch 曲線に分ける（分割）
- 部分の Koch 曲線をつなぐことで，全体を求める（統治）
- 再帰の止め時は，深さ d が 0 になった時．このとき，与えられた両端をそのまま返す．

考察

- Koch 曲線はフラクタルな図形…部分が全体と同じ構造をしている
    - よって座標も分割統治法で求まる

まとめ

まとめ

- 分割統治法：問題を同じ形をしたより小さな問題に分割し，それらを解き，最後に答えを組み合わせることで複雑な問題を解く
- 用いると複雑な問題の解法をシンプルに定義できることがある
- 用いると計算量が抑えられることがある
  - 抑えられなくても，抑える手がかりになることがある
- 分割の終了条件に注意．これが甘いと無限ループする．