教科書輪講
「アルゴリズムとデータ構造」
第6章　再帰・分割統治法

秋山研究室　M1　黄毅聰

2017/5/22

再帰

再帰関数:関数の中で自分自身を呼び出すような関数

- *e. g. n*の階乗を計算する再帰関数

```
factorial(n) //関数の定義
  if n == 1
    return 1
  return n * factorial(n - 1)
```

- 必ずどこかで終了するように実装

分割統治法

1. 問題を部分問題に分割 (Divide)
2. 部分問題を再帰的に解決 (Solve)
3. 得られた解を統合し、元の問題を解決 (Conquer)

e. g. 配列***A***の要素の最大値を線形探索

```
findMaximum(A, 1, r) //関数の定義
  m = (1 + r) / 2 //Divide
  if 1 == r-1 //要素数が1つ
      return A[1]
  else
     u = findMaximum(A, 1, m) // 前半の部分問題をSolve
     v = findMaximum(A, m, r) // 後半の部分問題をSolve
     x = max(u, v) // Conquer
  return x
```

例題1: 全探索

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の要素を足し合わせてmが作れるかどうかを判定するプログラムを作成してください。***A***の要素は1度だけ使うことができます。

数列***A***が与えられたうえで、質問としてq個のm_iが与えられるので、それぞれについてyesかnoと出力してください。

入力　1行目のn、2行目に***A***を表すn個の整数、3行目にq、4行目にq個の整数m_iが与えられます。

出力　各質問について***A***の要素を足し合わせてm_iを作ることができればyesと、できなければnoと出力してください。

制約　$n \leq 20$
$q \leq 200$
$1 \leq$ Aの要素 $\leq 2{,}000$
$1 \leq m_i \leq 2{,}000$

入力例
```
5
1 5 7 10 21
4
2 4 17 18
```

出力例
```
no
no
yes
yes
```

例題1解説(その1)

各要素について、選択するかしないかの2択となり、2^n通りの組み合せ

組み合せを列挙

```
makeCombination()
  for i が 0 から n-1  まで
   S[i] = 0 // i を選択しない
   rec(0) // Sに記録されるビット列

rec(i)
  if i が n に達した
    print S
    return
  rec(i + 1)
  S[i] = 1 // i を選択する
  rec(i + 1)
  S[i] = 0 // i を選択しない
```

例題1解説(その2)

前述の考え方を応用し、問題解決関数solve(*i*, *m*)を定義
solve(*i*, *m*): 「i番目の要素を使ってmを作れる場合trueを返す」という関数

$$\mathrm{solve}(i, m)$$

$$\|$$

$$\mathrm{solve}(i+1, m)$$

$$+$$

$$\mathrm{solve}(i+1, m - \boldsymbol{A}[i])$$

A[*i*]を引いていることが、*i*番目の要素を使うことに対応

```
// 整数が作れるかどうかを判断する再帰関数

solve(i, m)
  if m == 0
    return true
if i >= n
    return false
res = solve(i + 1, m) || solve(i + 1, m – A[i])
return res
```

部分問題がいずれtrueのときは、元の問題がtrueになる

例題1解説(その3)

- solve(i, m)において、与えられた整数を作ることができたとき、m = 0
- m > 0 and i >n のとき、与えられた整数は作成不可

例えば、次の例は数列A = {1, 5, 7}に対して、8が作れるかどうかを判定している様子

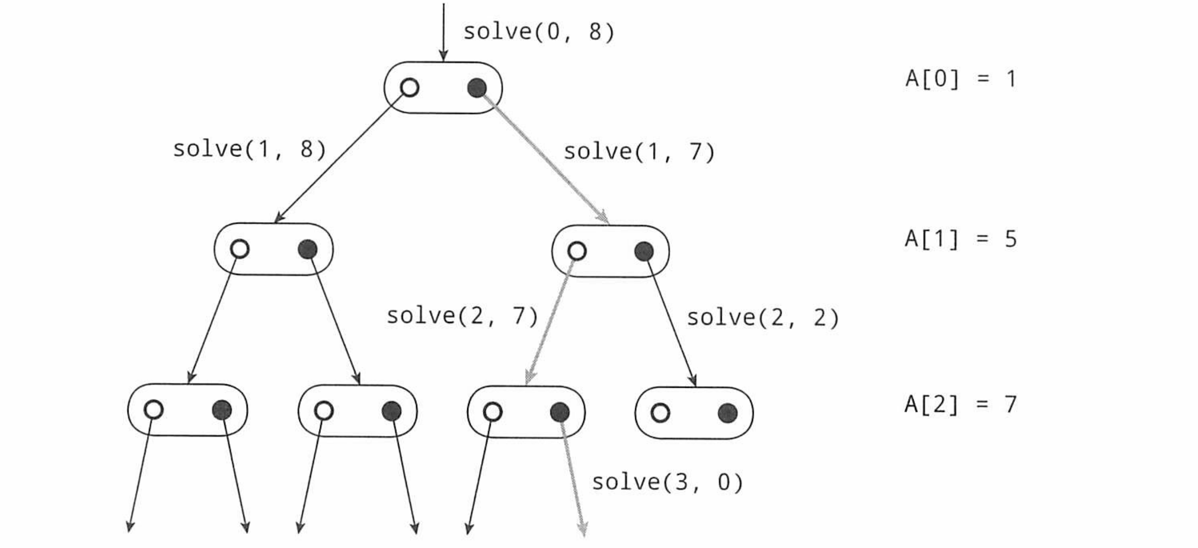

図6.2: 再帰による問題の分割

i番目の要素を選ぶか選ばないかについて2つの関数を呼び出す。計算量は$O(2^n)$になり、大きなnに対して適用不可、動的計画法によって改善できそう

例題2: コッホ曲線(その1)

整数nを入力し、深さnの再帰呼び出しによって作成されるコッホ曲線の頂点の座標を出力するプログラムを作成してください。

コッホ曲線はフラクタルの一種として知られています。フラクタルとは再帰的な構造を持つ図形のことで、以下のように再帰的な関数の呼び出しを用いて描画することができます。

- 与えられた線分(p1,p2)を3等分する。
- 線分を3等分する2点s,tを頂点とする正三角形(s,u,t)を作成する。
- 線分(p1,s)、線分(s,u)、線分(u,t)、線分(t,p2)に対して再帰的に同じ操作を繰り返す。

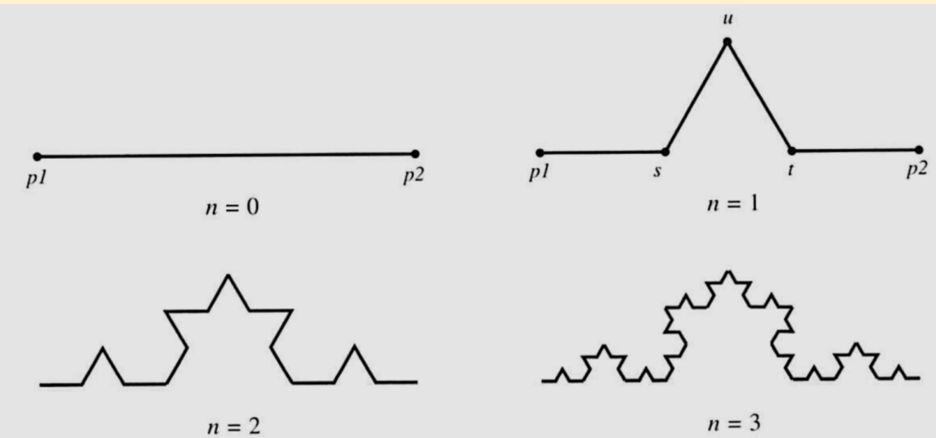

図6.3: コッホ曲線

例題2: コッホ曲線(その2)

(0,0), (100,0)を端点とします。

入力　1つの整数nが与えられます。

出力　コッホ曲線の各頂点の座標(x,y)を出力してください。1行に1点の座標を出力してください。端点の1つ(0,0)から開始し、一方の端点(100,0)で終えるひと続きの線分の列となる順番に出力してください。出力は0.0001以下の誤差を含んでいてもよいものとします。

制約　$0 \leq n \leq 6$

入力例
1

出力例
0.00000000 0.00000000
33.33333333 0.00000000
50.00000000 28.86751346
66.66666667 0.00000000
100.00000000 0.00000000

例題2解説(その1)

コッホ関数の頂点の座標を順番に出力する再帰関数

d: 再帰の深さ *p*1、*p*2: 線分の端点

Program 6.5: コッホ曲線の描画

```
koch(d, p1, p2)
  if d == 0
    return

  // p1, p2 から s, u, t の座標を計算

  koch(d-1, p1, s)
  print s
  koch(d-1, s, u)
  print u
  koch(d-1, u, t)
  print t
  koch(d-1, t, p2)
```

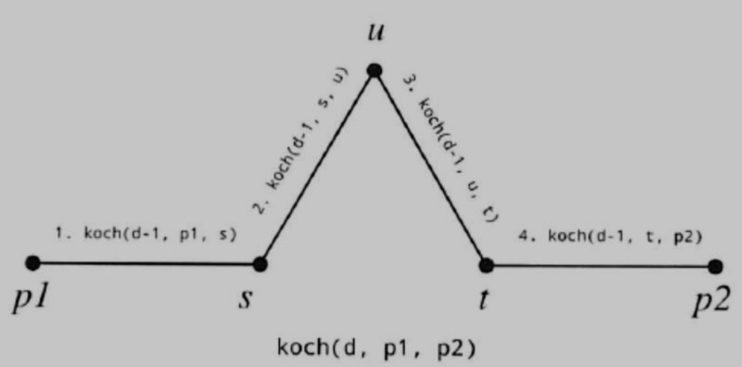

図6.4: コッホ曲線の描画

例題2解説(その2)

sとtの座標の求め方

$s.x = (2 \times p1.x + 1 \times p2.x)/3$
$s.y = (2 \times p1.y + 1 \times p2.y)/3$
$t.x = (1 \times p1.x + 2 \times p2.x)/3$
$t.y = (1 \times p1.y + 2 \times p2.y)/3$

u：tとsを起点として反時計回りに60度回転した位置
uの座標の求め方

$u.x = (2 \times p1.x + 1 \times p2.x)/3$
$u.y = (2 \times p1.y + 1 \times p2.y)/3$