情報科学 共通問題 (2013 年度冬学期試験)

[科目名: 情報科学, 試験実施日: 2014 年 2 月 12 日第 4 限, 答案用紙: 1 部, 計算用紙: 1 枚, 持込み: 一切不可]

- 内容に関する質問は受け付けない。問題の記述があいまいな場合は、適切な仮定を置いて解答し、どのような仮定を置いたかを明記せよ。

以下の問題では、関数 make2d、関数 make1d、関数 is_even は教科書で扱ったものと同じ関数で、定義済みとしてよい。また、include(Math) が実行されており、関数 sqrt も定義済みとしてよい。

第 1 問

(a) 以下に示すのは、教科書で扱った併合整列法のプログラムである。空欄 ア ～ ウ を埋めよ。関数 merge を呼び出すと、昇順の配列を 2 つ併合したものが返る。例えば merge([1,3,5,7],[2,4,6,8]) と呼び出すと、[1,2,3,4,5,6,7,8] が返る。

```
def mergesort(a)
  n = a.length()
  from = make2d(n,1)
  for i in 0..(n-1)
    from[i][0] = a[i]
  end
  while n > 1                    # (エ)
    to = make1d((n+1)/2)
    for i in 0..(n/2-1)
      to[i] = merge(from[i*2],from[i*2+1])
    end
    if !is_even(n)
      to[(n+1)/2-1] = ア
    end
    from = to
    n = イ
  end
  ウ
end
```

(b) mergesort([7,2,8,3,1,4,6,5]) の計算の途中で (エ) が実行される時の from の内容を、実行されるたびごとに書き出せ。

(c) 教科書で扱った単純整列法と併合整列法を計算量のオーダーと使用するメモリ量の観点で比較せよ。

第 2 問

以下の問いに答えよ。

(a) 次の関数は配列 a の要素の並び順を変えるものである。

```
def x_sort(a)
  i = 0
  while (i+1 < a.length())
    if a[i] > a[i+1]
       s = a[i]
       a[i] = a[i+1]
       a[i+1] = s
    end
    i = i+1
    # (オ)
  end
  a
end
```

x 君は、この関数により引数の配列 a の要素を小さい順に並べ替えることができると主張している。

i. 配列 a=[3,1,2,5,4] に対して x_sort(a) と呼び出したときに、関数内で a が書き換えられる様子を調べてみよう。各反復において、(オ) での i の値と a の内容を示せ。

ii. 実際には、この関数で並べ替えた後に要素が小さい順に並ぶとは限らない。そうならない配列の具体例と、並べ替えた後の配列の内容を示せ。

(b) 配列 a の i 番目から j 番目までの要素 (i, j を含む) の最小値を求める関数 min(a,i,j) を以下のように再帰的に記述した。

```
def min(a,i,j)
  if i == j
    a[i]
  else
    if a[j] < min(a,i,j-1)
      a[j]
    else
      min(a,i,j-1)
    end
  end
end
```

配列 a に対して min(a,0,a.length()-1) を実行すると、その最小値が求まる。

i. min(a,0,a.length()-1) を実行したとき、関数 min の呼び出し回数は配列 a によって変化する。min の呼び出し回数が最大となるような長さ 4 の配列 a の例を与えよ。
また、呼び出し回数が最大となる場合の計算量のオーダーを、配列の長さ n を用いて表せ。

ii. このプログラムを再帰を用いつつ計算量を減らして高速化するための改善案を示せ。そして計算量がどう変わるかを説明せよ。

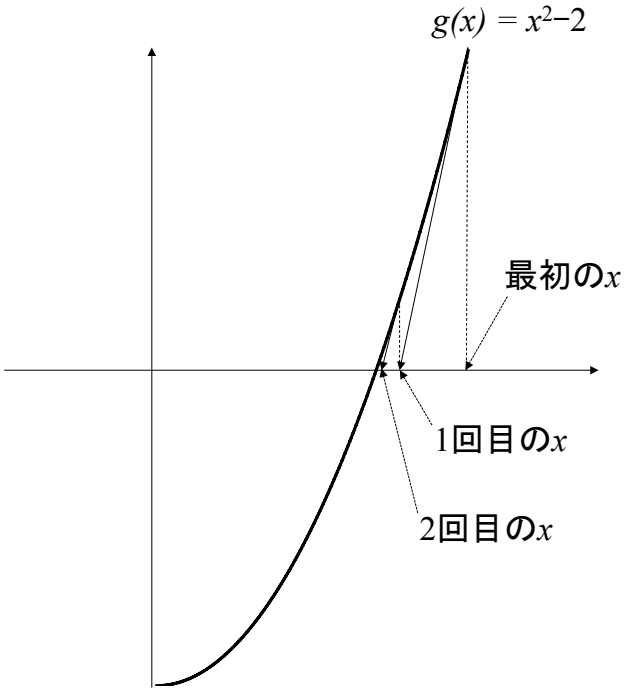
第 3 問

複素数 $a+bi$ を長さが 2 の配列 [a,b] によって表現する。二つの複素数 x, y を足した結果を返す関数 plus(x,y)、x, y を掛けた結果を返す関数 mul(x,y) は、以下のように定義することができる。

```
def plus(x,y)
  [x[0]+y[0],x[1]+y[1]]
end
def mul(x,y)
  [x[0]*y[0]-x[1]*y[1],x[0]*y[1]+x[1]*y[0]]
end
```

また、複素数 x の絶対値 abs(x) は以下のように定義することができる。

```
def abs(x)
  sqrt(x[0]*x[0]+x[1]*x[1])
end
```



(a) 上と同様にして、複素数 x を複素数 y で割った結果を返す Ruby の関数 div(x,y) を定義せよ。

(b) plus(x,y) と abs(x) を使って、$|10^8+1|$ を abs(plus([100000000.0 , 0.0] , [1.0 , 0.0])) と計算したところ、abs([100000000.0 , 0.0]) と異なる値が得られた。一方、$|10^8+i|$ を abs(plus([100000000.0 , 0.0] , [0.0 , 1.0])) と計算したところ、abs([100000000.0 , 0.0]) と同じ値が得られた。この違いの理由として考えられる事を述べよ。

(c) 複素関数 f を $f(x)=x^2+x+1$ と定義する。関数 $f(x)$ とその導関数 $f'(x)=2x+1$ を計算する Ruby の関数 f(x) と fd(x) を、plus(x,y) と mul(x,y) を使って定義せよ。

(d) ニュートン法は関数 g の零点（$g(x)=0$ となる x）の近似値を計算するアルゴリズムである。右上の図は $g(x)=x^2-2$ の零点を求める様子を示している。この図にあるように、現在の x の値に対して、$(x, g(x))$ 上の接線と x 軸との交点により x の値を更新することを繰り返す。ニュートン法は複素関数に対しても適用することができる。以下の関数 newton(x) は、上で定義した複素関数 f に対して、初期値 x からニュートン法を実行して f の零点の近似値を返す。その近似値における f の値の絶対値は 0.000000000001 より小さくなる。

以下の［カ］を埋めよ。

```
def newton(x)
  while abs(f(x)) >= 0.000000000001
    x = [カ]
  end
  x
end
```

第4問

AとBの2チームが対戦を繰り返す。1試合の対戦結果は、チームAの勝利、敗北、引分のいずれかとする。

n試合を対戦した後で、チームAがi勝、j敗、k引分の状態に至るパターンの数を計算するRubyの関数 pattern(i,j,k)（ただし、$i+j+k=n$）について考える。たとえば、pattern(3,0,0)であれば、チームAの3連勝のみの1パターンであり、1を返す。また、pattern(1,2,0)であれば、チームAが3試合のいずれか1試合で勝利するパターンしかないので、3となる。このような関数 pattern(i,j,k) を再帰的に定義すると以下のようになる。

```
def pattern(i,j,k)
  if i < 0 || j < 0 || k < 0
    0
  else
    if i == 0 && j == 0 && k == 0
      1
    else
      pattern(i-1,j,k) + pattern(i,j-1,k) + pattern(i,j,k-1)
    end
  end
end
```

(a) n試合を対戦した後で、チームAがi勝、j敗、k引分の状態に至る確率を計算するRubyの関数 probability(i,j,k)（ただし、$i+j+k=n$）について考える。i勝、j敗、k引分の状態でチームAが勝利する確率が$w(i,j,k)$、敗北する確率が$l(i,j,k)$、引き分ける確率が$d(i,j,k)$（ただし、$w(i,j,k)+l(i,j,k)+d(i,j,k)=1.0$）とするとき、probability($i$,$j$,$k$)を、$n-1$試合目までの確率を用いて、再帰的に定義せよ。$w(i,j,k), l(i,j,k), d(i,j,k)$は関数 w($i$,$j$,$k$)、l($i$,$j$,$k$)、d($i$,$j$,$k$) として、定義済みとしてよい。

(b) 一般に上記の関数 pattern(i,j,k) や probability(i,j,k) は、試合数nが多くなると計算に時間がかかることが予想される。その理由を三行程度で簡潔に書け。

(c) 上記 (b) の問題を改善して高速化を実現した関数 fastProbability(i,j,k) を以下の右に示す。［キ］・［ク］を埋めよ。fastProbability(i,j,k) で使用する関数 make3d(l,m,n) と value(a,i,j,k) は以下の左に示されている。

```
def make3d(l,m,n)
  a = make1d(l)
  for i in 0..(l-1)
    a[i] = make2d(m,n)
  end
  a
end

def value(a,i,j,k)
  if i < 0 || j < 0 || k < 0
    0.0
  else
    a[i][j][k]
  end
end
```

```
def fastProbability(i,j,k)
  n = i+j+k
  a = make3d(n+1,n+1,n+1)
  for z in 0..n
    for y in 0..(n-z)
      for x in 0..(n-z-y)
        if x == 0 && y == 0 && z == 0
          a[x][y][z] = [キ]
        else
          a[x][y][z] = [ク]
        end
      end
    end
  end
  a[i][j][k]
end
```