情報科学 共通問題 (2011 年度冬学期試験)

[科目名: 情報科学, 試験実施日: 2012 年 2 月 13 日第 2 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．

以下の問題で用いられている Ruby の式の意味は次のとおりである．x `&&` y は x と y の値がともに真の場合にのみ真となる．x `||` y は x と y の値がどちらか真の場合にのみ真となる．`!`x は x の真偽を逆転させる．また，関数 `max(`x`,`y`)` は計算量 $O(1)$ で x, y の大きい方の値を返す関数である．関数 `make2d(`h`,`w`)` は h 行 w 列の 2 次元配列を作る関数である．問題においても解答の際も，これらの関数は定義済みであるとしてよい．

問題 1　N 個の整数の列 $a = [x_0, x_1, \ldots, x_{N-1}]$ の i 番目から $(j-1)$ 番目 $(0 \le i \le j \le N)$ までの数の和を $s(a, i, j)$ とする (ただし $i = j$ のとき $s(a, i, j) = 0$)．以下の問いに答えよ．

(a) $a = [8, -4, -5, 2, 4, -5, 5, 3, -7, 8]$ のとき $s(a, 0, 0) = 0$, $s(a, 0, 1) = 8$ である．$s(a, 0, 2)$, $s(a, 0, 3)$, $s(a, 0, 4)$ を求めよ．

(b) $s(a, 0, N)$ を単純な反復計算によって求めるのに必要な計算量のオーダーを N の式で書け．

(c) $x \le i \le j \le y$ を満たす全ての i, j について $s(a, i, j)$ の最大値を $mss(a, x, y)$ とする (ただし $0 \le x \le y \le N$ とする)．右のように全ての i, j の組み合わせについて $s(a, i, j)$ を計算し，その最大値を求める場合の `mss(`a`,0,`N`)` の計算量のオーダーを N の式で書け．

```
def mss(a,x,y)
  m = 0
  for i in x..y
    for j in i..y
      m = max(m,s(a,i,j))
    end
  end
  m
end
```

(d) $mss(a, 0, z-1) = s(a, x, z-1)$ かつ $s(a, x, z) < 0$ のとき，a の $z-1$ 番目は $mss(a, 0, y)$ を与えるような区間に含まれないことが知られている (つまり $mss(a, 0, y) = s(a, i, j)$ ならば，$i \le z \le j$ ではない)．このことを利用して $mss(a, 0, m)$ を求める関数 `mss0(a,m)` は右のように定義できる．

```
def mss0(a,m)
  t = 0
  sum = 0
  for i in 0..(m-1)
    sum = sum + a[i]
    if sum > t
      t = sum
    else
      if sum < 0
        sum = 0
      end
    end
    #※
  end
  t
end
```

`a=[8,-4,-5,2,4,-5,5,3,-7,8]` の下で `mss0(a,10)` を実行したとき，繰り返し各回の最後 (※) における `sum`, `t` の値を下表のような形で書け．

i	0	1	2	3	4	5	6	7	8	9
sum										
t										

(e) 小問 (d) で示した `mss0(`a`,`N`)` の計算量のオーダーを N の式で書け．

問題 2　右の Ruby プログラムを読んで以下の問いに答えよ．

```
def f(x,y)
  z = 1
  while y != 0
    while y % 2 == 0
      x = x * x
      y = y / 2
    end
    y = y - 1
    z = z * x
  end
  z
end
```

(a) この関数を

`f(2,4)` および `f(3,5)`

のように呼び出したら，それぞれどのような値が得られるか．結果のみ答えよ．

(b) `c = f(a,b)`

としたとき，`c` と `a`, `b` の関係を，簡潔に述べよ．ただし，`a` と `b` は非負の整数とする．

問題 3　下図のように，平面上の x 座標が非負の領域の上を，一回のステップでランダムな方向に直線距離で 1 だけ移動する点がある．y 軸にぶつかる場合は，y 軸によって完全反射する．この場合，ぶつかるまでに移動した距離とぶつかってから移動した距離の和が 1 であるとする．ただし，$(0,-1)$ から $(0,1)$ までは穴が空いている仮定する．この点の初期値を $(\mathtt{x},\mathtt{y})$ として，穴を抜けるまでに移動するステップ数を求める手続き `escapesteps(x,y)` を右に掲げる．以下の問いに答えよ．

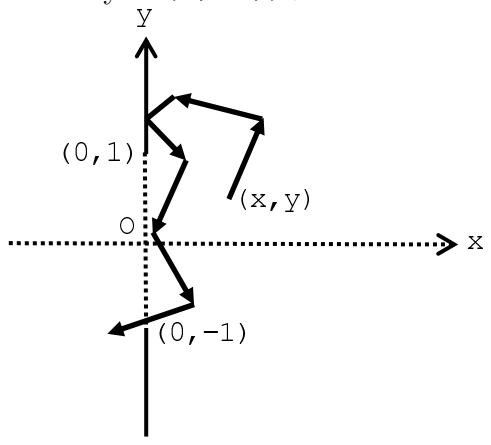
```
include(Math)

def escapesteps(x,y)
  n = 0
  escaped = false
  while !escaped
    r = rand()
    dx = cos(2*3.14159265358979*r)
    dy = sin(2*3.14159265358979*r)
    x1 = x + dx
    y1 = y + dy
    if x1 < 0
      if [ア]
        escaped = true
      else
        [イ]
        y = y1
      end
    else
      [ウ]
      y = y1
    end
    n = n + 1
  end
  n
end
```

(a) `escapesteps(x,y)` 関数で利用している `rand()` 関数は疑似乱数を実現している．疑似乱数とは何かを説明せよ．

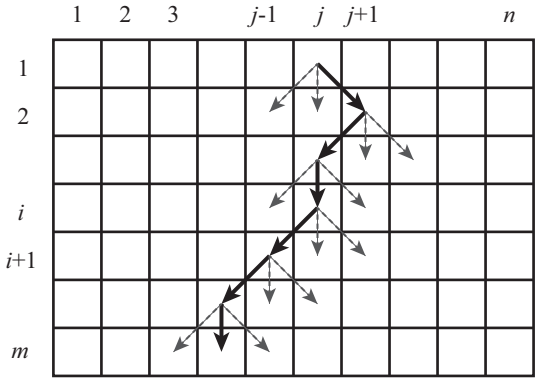
(b) 疑似乱数が満たすべき望ましい条件を一つ述べよ．

(c) [ア] は，点が穴を抜ける条件を表している．ここに入るべき Ruby の式を書け．

(d) [イ] と [ウ] は，x 座標の更新を行う．これらに入るべき Ruby の命令を書け．



問題 4　縦 m 行，横 n 列からなる格子領域において，i 行 j 列の位置を (i, j) と表記する．右図のように i 行目の (i, j) から移動する際には，$i+1$ 行目の $(i+1, j-1)$, $(i+1, j)$, $(i+1, j+1)$ のいずれかのみに移動して領域全体を縦断する．位置 (i, j) に存在することで正の利益 $gain(i, j)$ を得られるものとすると，ある経路で縦断する際の**累積利益**は $\sum_{i=1}^{m} gain(i, j_i)$ となる（j_i は当該の経路で i 行目における位置 (i, j_i) を表す）．出発点の 1 行目と終着点の m 行目の列の位置は任意として，縦断によって得られる累積利益の最大値を求めたい．以下の問いに答えよ．



(a) 1 行目の任意の位置から (i, j) までの経路は複数存在するが，それらの累積利益の最大値（以下，**最大累積利益**と呼ぶ）を $maxGain(i, j)$ とする．1 行目における最大累積利益 $maxGain(1, j)$ は，その位置 $(1, j)$ における利益のみであるから，

$$maxGain(1, j) = gain(1, j)$$

となる．$i > 1$ の場合の i 行目の最大累積利益 $maxGain(i, j)$ を，$i-1$ 行目の最大累積利益，$gain(i, j)$ ならびに `max(a,b)` を用いて漸化式で表現せよ．ただし，位置 (i, j) が範囲外の場合，すなわち $i < 1$，$i > m$，$j < 1$ ，$j > n$ のいずれかが真となる場合には，$maxGain(i, j)$ は 0.0 を返すと仮定して良い．

(b) 上記の議論をもとに最大累積利益 $maxGain(i, j)$ を求める Ruby の関数 `maxGain(i, j, m, n)` を再帰的に定義すると，左下のようになる．(引数として m, n を補っていることに注意せよ．) このような再帰関数は i の値が大きくなると急速に計算時間を必要とする．領域全体を縦断する際の最大累積利益 `maxGain(m, m, m, n)` を計算する際に，関数 `maxGain` は何回呼び出されるか？ただし，$n > 2m$ であるものとする．

```
def maxGain(i, j, m, n)
  if i < 1 || i > m || j < 1 || j > n
    0.0
  else
    if i == 1
      gain(1, j)
    else
      (a) の解答に相当する計算
    end
  end
end
```

(c) この計算を効率良く行うために，位置 (i, j) までの最大累積利益を与える $m+1$ 行 $n+2$ 列の 2 次元配列 `totalGain[i][j]` を動的計画法によって求める関数 `calculateTotal(m, n)` は右のようになる．［ア］～［ウ］を埋めよ．

```
def calculateTotal(m, n)
  totalGain = make2d(m+1, n+2)
  for i in 1..m
    totalGain[i][0]   = ［ア］
    totalGain[i][n+1] = ［ア］
    for j in 1..n
      if i == 1
        totalGain[i][j] = ［イ］
      else
        totalGain[i][j] = ［ウ］
      end
    end
  end
  totalGain
end
```

以上