アルゴリズム入門　共通問題（2019年度Aセメスター試験）

[試験日時：2020年1月30日第4限（60分），答案用紙：1部，計算用紙：1枚，持ち込み一切不可]

内容に関する質問は受け付けない．問題の記述が曖昧な場合は，適切な仮定をおいて回答し，どのような仮定をおいたのか明記せよ．

問題1

(1) 関数 `func1(x, y)` は，`x` も `y` も共に正なら0を，`x` が0以下で `y` が正なら1を，それ以外は−1を返す．下記のプログラムは `func1` を正しく実装できていない．どのような場合に正しくない値を返すか答えよ．またどのように修正すればよいか答えよ．

(2) 関数 `func2(data)` は，2次元配列 `data` 中の最大の値を返す．下記のプログラムは `func2` を正しく実装できていない．どのような場合に正しくない値を返すか答えよ．またどのように修正すればよいか答えよ．

```
def func1(x, y):
  if x > 0:
    if y > 0:
      return 0
  else:
    if y > 0:
      return 1
    else:
      return -1
```

```
def func2(data):
  m = data[0][0]
  for i in range(0, len(data)):
    for j in range(0, len(data[i])):
      if data[i][j] > m:
        m = data[i][j]
    return m
```

(3) 下記プログラム中の関数 `rev1(a)`，`rev2(a)`，`rev3(a)` はいずれも配列 `a` の要素を逆順に並び替えるために作成したものである．以下の問に答えよ．

(a) 関数 `rev1` を実行するとエラーが起こった．どのようなエラーが起こるか，理由も含め説明せよ．

(b) 関数 `rev2` は正しくない結果を返す．`rev2([1,2,3,4,5,6,7])` がどのような結果を返すか答えよ．

(c) 下記関数 `rev3` が正しい結果を返すように空欄 A ～ D を埋めよ．

```
def rev1(a):
  for i in range(0, len(a)):
    a[i] = a[len(a) - i]
  return a
```

```
def rev2(a):
  for i in range(0, len(a)):
    a[i] = a[len(a) - i - 1]
  return a
```

```
def rev3(a):
  for i in range(0, [ A ]):
    tmp = [ B ]
    a[len(a) - i - 1] = [ C ]
    a[i] = [ D ]
  return a
```

問題 2

下記プログラムは，教科書で説明された solve_linear である．これを用いて，様々な c について，$cx + (c+1)y = 2c+1$ と $(c+1)x + cy = 2c+1$ からなる連立一次方程式（解は $x = y = 1$）の解を求めてみた．以下の問に答えよ．

```
def fe(a):
  for i in range(0, len(a)):
    prepare(a, i, i)
    for j in range(i + 1, len(a)):
      erase(a, j, i)

def prepare(a, k, i):
  factor = a[k][i]
  for j in range(0, len(a[k])):
    a[k][j] = a[k][j] / factor

def erase(a, j, i):
  factor = a[j][i]
  for k in range(0, len(a[j])):
    a[j][k] = a[j][k] - factor * a[i][k]
```

```
def bs(a):
  n = len(a[0]) - 1
  result = ita.array.make1d(n)
  for i in range(0, n):
    calc_result(result, a, n - i - 1)
  return result

def calc_result(res, a, k):
  res[k] = a[k][len(a[0]) - 1]
  for j in range(k + 1, len(res)):
    res[k] = res[k] - res[j] * a[k][j]

def solve_linear(a):
  fe(a)
  return bs(a)
```

(1) 以下の実行結果を得た．c = 0.0 のときにゼロ割エラーが起きた理由を説明せよ．

```
> c = 1.0
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[1.0, 1.0]
> c = 0.0
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
ZeroDivisionError: float division by zero
```

(2) 上記の問題に対応すべく，以下のように関数 fe(a) を書き換えた．空欄 A ～ C を埋め，関数 maxrow(a, i) を完成させよ．

```
def fe(a):
  for i in range(0, len(a)):
    swap(a, i, maxrow(a, i))
    prepare(a, i, i)
    for j in range(i + 1, len(a)):
      erase(a, j, i)

def swap(data, i, j):
  x = data[i]
  data[i] = data[j]
  data[j] = x
```

```
def maxrow(a, i):
  m = [ A ]
  for j in range(i + 1, len(a)):
    if abs(a[m][i]) [ B ] abs(a[j][i]):
      m = [ C ]
  return m
```

(3) 上記の変更を加えたプログラムで，以下の実行結果を得た．c = 2.0 のときに実行結果が異なる理由を説明せよ．丸め誤差，桁落ち，情報落ち，打切り誤差などとの関係があれば述べること．

```
> c = 0.0
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[1.0, 1.0]
> c = 1.0
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[1.0, 1.0]
> c = 2.0
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[1.0000000000000002, 0.9999999999999999]
```

(3) さらに c を変更し，以下の実行結果を得た．c = 2.0 ** 53 のときにゼロ割エラーが起きた理由を説明せよ．丸め誤差，桁落ち，情報落ち，打切り誤差などとの関係があれば述べること．

```
> c = 2.0 ** 51
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[1.0, 1.0]
> c = 2.0 ** 52
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
[0.9999999999999998, 1.0]
> c = 2.0 ** 53
> solve_linear([[c, c + 1, 2 * c + 1], [c + 1, c, 2 * c + 1]])
ZeroDivisionError: float division by zero
```

問題 3

(1) 図 1 のように 11×11 の格子状の部屋があり，最も左上の部屋を $(0, 0)$ 番目の部屋，最も右下の部屋を $(10, 10)$ 番目の部屋とする．以下のプログラムは「$(0, 0)$ 番目の部屋から始め，右の部屋か下の部屋に移動することを繰り返し，$(10, 10)$ 番目の部屋まで辿り着く経路」の数を求める．num_of_routes(i, j) が再帰関数となるように空欄 A ～ C を埋めよ．

```
def num_of_routes(i, j):
  if i == 0 and j == 0:
    return 1
  elif i == 0:
    return [   A   ]
  elif j == 0:
    return [   B   ]
  else:
    return [   C   ]
num_of_routes(10, 10)
```

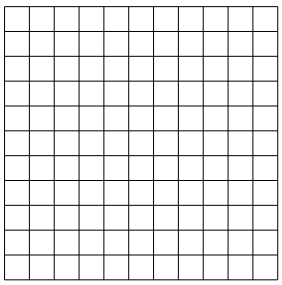

図 1

(2) (1) のプログラムの最終行を num_of_routes(n,n) とすれば，任意の正整数 n に対し，$(n+1) \times (n+1)$ の部屋がある場合の $(0, 0)$ 番目の部屋から (n, n) 番目の部屋への経路の数を求めることができる．この場合の時間計算量について適切なものを以下の選択肢から選び，理由を述べよ．

選択肢: n に比例　n^2 に比例　n^3 に比例　n^4 に比例　以上のいずれでもない

(3) 便宜的に，図 2 のように (0,10) より右の領域などにも部屋がある状況を考える．(0,0) 番目の部屋から右か下の部屋への移動を 20 回繰り返すとき，経路の総数は 2^{20} 通りである．そのうち (10,10) 番目の部屋に到達する経路の総数を X とすると，ランダムに右か下の部屋への移動を 20 回繰り返す時に (10,10) 番目の部屋に到達する確率は $X \,/\, 2^{20}$ である．

(a) 以下のプログラムは上記の考察に基づき，モンテカルロ法により (1) で求めた経路の数 X の近似値を求める．空欄 D を埋めよ．

(b) 以下のプログラムを実行した結果は 17550 であった．経路の数はおよそいくつであると考えられるか．有効桁数を 3 桁として求めよ．$2^{20} = 1.05 \times 10^6$ としてよい．

```
import random

def random_walk(n):
  c = 0
  for k in range(0, n):
    i = 0
    for step in range(0, 20):
      if random.randrange(0, 2) == 0:
        i = i + 1
    if [ D ]:
      c = c + 1
  return c

random_walk(100000)
```

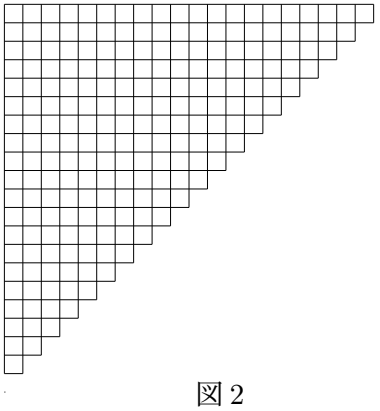

図 2

(4) 図 3 の黒色で示した部屋が入れないときの経路の数を関数 num_of_routes_2(10,10) で計算できるようにしたい.

(a) まずは入れない部屋がない場合について，入れない部屋がある場合にも部分的にプログラムを再利用できるよう，(1) のプログラムを以下のようにモジュール化したが，実行結果は正しくなかった．関数 num_of_routes_2 のどこをどう修正すればよいか答えよ.

(b) 関数 is_available を変更することで，黒色の部屋が入れないときの経路の数を求める．変更後の関数 is_available を書き下せ.

```
def is_available(i, j):
  return i >= 0 and j >= 0

def num_of_routes_2(i, j):
  if i == 0 and j == 0:
    return 1
  num = 0
  if is_available(i - 1, j):
    num = num + num_of_routes_2(i - 1, j)
  elif is_available(i, j - 1):
    num = num + num_of_routes_2(i, j - 1)
  return num

num_of_routes_2(10, 10)
```

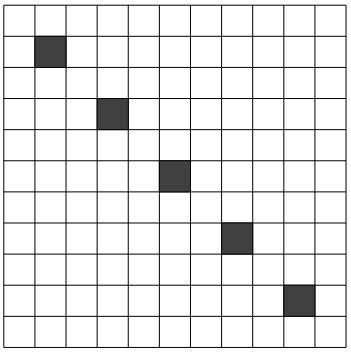

図 3