情報科学 共通問題 (09年度冬学期試験)

[科目名: 情報科学, 試験実施日: 2010年2月10日 (水) 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回答し，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min(x,y), max(x,y) はそれぞれx, yの小さい方，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．解答の際，これらの関数は定義済みであるとしてよい．

問題1　aリットル入るバケツAとbリットル入るバケツBがある．このバケツで水を正確にqリットル計り取りたい．ただし，a, b, qは正の整数とする．可能な操作は以下であるとき(a)から(c)の問に答えよ．

fill(X)	バケツXに一杯の水を汲む．ただし，XはAまたはB.
empty(X)	バケツXを空にする．ただし，XはAまたはB.
move(X,Y)	バケツXの水を可能な限りYに移す．可能な限りという意味は，Xが空になるかYが一杯になるまでという意味である．X, YはそれぞれA,BまたはB,A.

(a) 目標は，AおよびBが空の状態から始めて，AまたはBにqリットルの水が入っている状態を作ることである．$a = 5, b = 3, q = 4$として，その目標状態に至る操作の列の一つの例を示しなさい．たとえば以下のように記述すればよい．

fill(B), move(B,A), empty(A), ...

(b) 一般のa, b, qに対して，目標状態に至る操作の列を発見するために，この問題を次のようにモデル化する．まず，ある時点における問題の状態を，AとBに入っている現在の水の量を(x, y)として，xy平面上の点によって表現する．以下ではこれを問題平面と呼ぶ．

(i) 問題平面上で，fill, empty, moveという操作が一般にどのような動きとなるかを述べなさい．

(ii) 問題平面上で，目標となる状態はどのような点に対応するかを述べなさい．

(iii) 解が存在すると仮定して，目標状態に至る操作列を問題平面上で探索する方法について，簡単に述べなさい．

(c) $a = 8, b = 5, q = 4$の場合について，目標状態に至る最短の操作列を1つ示しなさい．

(以下余白)

問題 2　関数 $f(x)$ の数値積分 $\int_{x_s}^{x_e} f(x)dx$ を求めたい．これに関して以下の問に答えよ．

(a) 台形公式について，計算式を示しながら，考え方を説明せよ．

以下，台形公式は授業で説明したプログラムの計算順序で計算するものとする．

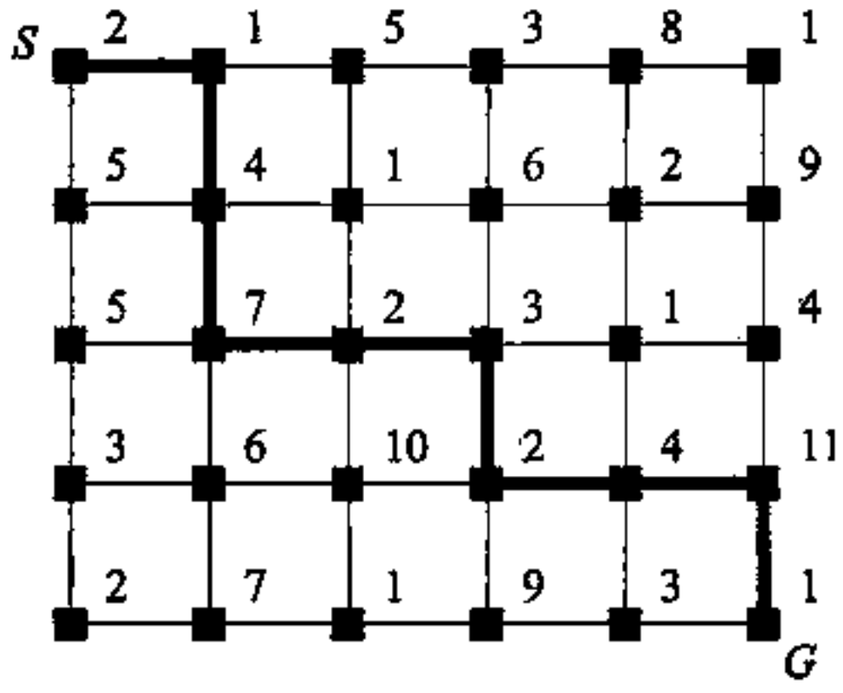
(b) 台形公式を用いて $f(x) = 1$ を $x_s = 0$ から $x_e = 1$ まで積分する場合，どのような誤差が生じる可能性があるかを，その誤差が発生する理由を含めて，述べよ．

(c) 台形公式を用いて $f(x) = x^2$ を $x_s = 0$ から $x_e = 1$ まで積分する場合，問 (b) に加えてどのような誤差が生じる可能性があるかを，その誤差が発生する理由を含めて，述べよ．

(d) 台形公式以外の公式を用いて，問 (c) で生じる誤差を減らすことを考える．そのような公式の名前を一つあげ，考え方を説明せよ．

(以下余白)

問題 3　右図のように m 行 n 列の格子状に並んだ都市 (■) が道路で結ばれている．西北の町 S から出発して，東南の町 G へ最小費用で行きたい．S, G を含め，町では必ず一泊し，1G 以上の宿泊料を払う (都市の右肩の数値)．以下の各問に答えなさい．



(a) 町から南の道へ行くのを 0, 東の道へ行くのを 1 と表わせば，最小宿泊経路はたどった順に 0 と 1 からなる 1 次元の配列で表現できる．図の中で太字で描いた経路を，Ruby の配列の表記法 ([1，0，…] のように要素を順に書く) に従って書きなさい．また，この経路での費用 (宿泊料の合計) を書きなさい．

(b) S から G に行く最小宿泊数の経路が何通りあるかを m と n の式で書きなさい．

(c) 以下では町の座標を (行番号,列番号) のように書く．最小費用を求める問題は，町 S, G の座標をそれぞれ $(0,0)$, $(m-1,n-1)$ としたとき，町 (i,j) の宿泊料を c(i,j) で表わす関数でモデル化できる．経路が問 (a) のように表現されているとき，費用を求めるプログラムを Ruby で下図 (左) のように書いた．空欄 ア , イ , ウ に適当なプログラム断片を入れなさい．path は上記のように表現された経路を，m(), n() は m, n をそれぞれ意味する．

(d) S から G へ行く最小費用を求めるために，町 (i,j) から G へ行く最小費用を求める関数 get_min_cost を下図 (右) のように定義した．get_min_cost(0,0) を計算するのに get_min_cost(4,5) が何回呼ばれるかを答えなさい ($4<m$, $5<n$ とする)．なお，1000000 はここでは無限大を意味する．

(e) 一度計算した get_min_cost(i,j) の値を，関数の外側で定義した m 行 n 列の配列 $min_cost[i][j] に入れておくと無駄な計算が省ける．このアイデアに基づいて問 (d) のプログラムを手直ししたプログラムと，その外側で行う $min_cost の定義や初期化の文を書きなさい．関数名は同じとし，下図 (右) のプログラムのうち縦棒で示された範囲の行を必ずそのまま再利用すること．(注意: Ruby では，ある関数の中で，その関数の外や他の関数で定義された変数を使うためには，変数名を $min_cost のように $ではじめなければいけない．)

```
def get_cost(path)
  cost=c(0,0)
  i=0
  j=0
  for k in 0..(m()+n()) ア )
     if path[k]==0
        i=i+1
     else
        イ
     end
     cost= ウ
  end
  cost
end
```

```
def get_min_cost(i,j)
  if i==m()-1 && j==n()-1
     c(i,j)
  else
     south=1000000
     east=1000000
     if i<m()-1
        south=get_min_cost(i+1,j)
     end
     if j<n()-1
        east=get_min_cost(i,j+1)
     end
     c(i,j)+min(south,east)
  end
end
```

問題 4　図 1(i),(ii),(iii) はそれぞれ −2 以上 0 以下，2 以上 3 以下，5 以上 8 以下の数から成る集合，図 1(iv) は，これらの三つの集合の合併を示している．このような数直線上の数の集合をオブジェクトとして表現したい．なお，この問題では数値は全て整数で与えられるものと仮定せよ．

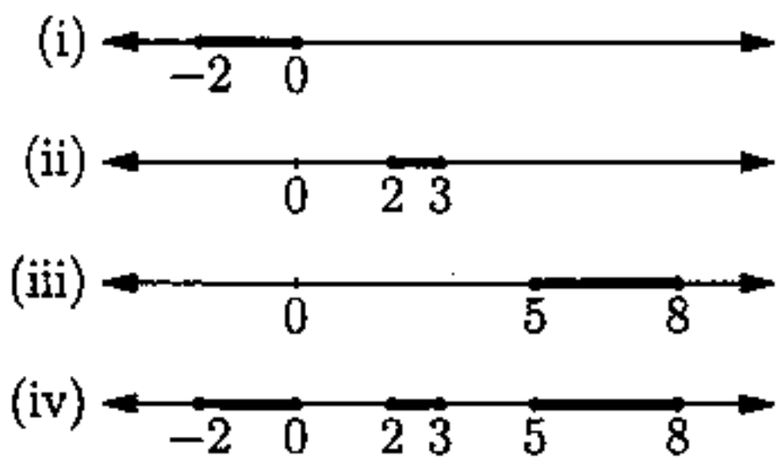


図 1: 数直線上の数の集合

(a) 図 3(左) のように Interval クラスを定義する．このクラスのオブジェクトは図 2 のように，二つの数を指定して生成する．

```
s1 = Interval.new(-2,0)
s2 = Interval.new(2,3)
s3 = Interval.new(5,8)
```

図 2: Interval オブジェクトの生成

s1, s2, s3 のオブジェクトは，それぞれ図 1(i),(ii),(iii) の集合を表す．

is_member(x) というメソッドが，各々のオブジェクトが表す集合に x が属しているときに true，属していないときに false を返すように，図 3(左) の ア の部分を埋めよ．たとえば，s1.is_member(-1) は true，s2.is_member(-1) は false となる．

(b) 図 3(右) のように Union クラスを定義する．このクラスのオブジェクトは以下のように，集合を表す二つのオブジェクトを指定して生成する．

```
s4 = Union.new(Union.new(s1,s2), s3)
```

s4 のオブジェクトは，s1, s2, s3 の表す集合の合併を表す．s1, s2, s3 が問 (a) のオブジェクトを表すとすると，s4 のオブジェクトは，図 1(iv) の集合を表す．たとえば，s4.is_member(-1) は true となる．問 (a) と同様にして，図 3(右) の イ を埋めよ．

```
class Interval
  attr_accessor("from", "to")

  def initialize(f,t)
    self.from = f
    self.to = t
  end

  def is_member(x)
    ア
  end
end
```

```
class Union
  attr_accessor("set1", "set2")

  def initialize(s1,s2)
    self.set1 = s1
    self.set2 = s2
  end

  def is_member(x)
    イ
  end
end
```

図 3: 数の集合を表わすクラスの定義

以上