アルゴリズム入門　共通問題（2016年度Aセメスター試験）

[試験日時：2017年1月25日第3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．以下の問題で用いられている `make1d`(n) は長さ n の配列を作り，`make2d`(m,n) は $m \times n$ の2次元配列を作る．これらは定義済みとしてよい．

問題1

整数を要素とする配列を小さい順に整列する方法の一つに併合整列法がある．これは

㈱既に整列された二つの配列 p と q とを併合し，一つの整列された配列 r を作る

という操作を再帰的に用いることにより配列を整列する．これに関して以下の問に答えよ．

(1) まず㈱の手順を行う関数 `merge`(p, q) を作ろう．例えば `merge([0, 4, 5, 7], [1, 3, 6, 9])` の結果は新たな配列 `[0, 1, 3, 4, 5, 6, 7, 9]` となる．

```
1  def merge(p, q)
2    r = make1d(p.length() + q.length())
3    i = 0
4    j = 0
5    for k in 0 .. (r.length() - 1)
6      if i < p.length() && (j == q.length() || p[i] < q[j])
7        r[k] = p[i]
8        i = i + 1
9      else
10       r[k] = [ (A) ]
11       j = [ (B) ]
12     end
13   end
14   r
15 end
```

(i) 空欄 [(A)]・[(B)] を埋めよ．

(ii) `merge([0, 4, 5, 7], [1, 3, 6, 9])` の計算において，第6行目から12行目までの繰り返し部分は何回実行されるか．

(2) この `merge` を再帰的に使うことにより，配列 s の要素を整列した配列を返す関数 `mergesort`(s) を作ろう．例えば `mergesort([4, 5, 0, 7, 6, 3, 9, 1])` の結果は新たな配列 `[0, 1, 3, 4, 5, 6, 7, 9]` となる．この `mergesort` を作るために，右のプログラムでは，与えられた配列 s の第 a 要素から第 $b-1$ 要素までを整列した配列を返す関数 `mergesort_sub`(s, a, b) を定義している．

```
1  def mergesort(s)
2    mergesort_sub(s, 0, s.length())
3  end
4  def mergesort_sub(s, a, b)
5    if b - a == 1
6      [s[a]]
7    else
8      c = (a + b) / 2
9      merge(mergesort_sub(s, a, c), [ (C) ])
10   end
11 end
```

(i) `mergesort_sub([4, 5, 0, 7, 6, 3, 9, 1], 0, 1)` の結果は何か．

(ii) `mergesort_sub([4, 5, 0, 7, 6, 3, 9, 1], 0, 4)` の結果は何か．

(iii) 空欄 [(C)] を埋めよ．

(3) 長さ 2^m の配列 s について `mergesort`(s) を実行すると，`mergesort_sub` を通じて関数 `merge` が頻りに呼び出されることになるが，それによって `merge` のプログラム中の第 6 行目から 12 行行目までの繰り返し部分が実行される総回数を $T(m)$ とする．

(i) $T(m+1)$ と $T(m)$ の間に成立つ関係を述べよ．

(ii) $T(m)$ を求めよ．

(4) 整列を行うには，併合整列法の他にも様々なアルゴリズムがある．例えば単純整列法では，与えられた配列全体を調べることでまず最も小さい要素を見つけて左端に置き，次にその他の要素のうち最も小さい要素を見つけて次の位置に置き，というように，小さい要素から順に見つけてゆく．この単純整列法の計算量について述べ，併合整列法と比較せよ．なお比較に際しては (3) のように配列の長さが 2^m の場合のみについて述べても構わない．

問題 2

次に示すのは，Gauss-Jordan 法を用いて連立一次方程式の解を求める Ruby のプログラムである．このプログラムに関して以下の問に答えよ．

```
1  def gj(a)
2    row = a.length()
3    col = a[0].length()
4    for k in 0 .. (col - 2)
5      akk = a[k][k]
6      for i in 0 .. (col - 1)
7        a[k][i] = a[k][i] * 1.0 / akk
8      end
9      for i in 0 .. (row - 1)
10       if i != k
11         aik = a[i][k]
12         for j in k .. (col - 1)
13           a[i][j] = a[i][j] - aik * a[k][j]
14         end
15       end
16     end
17   end
18   a
19 end
```

(1) $\begin{pmatrix} 1 & 1 & -1 \\ 3 & 5 & -7 \\ 2 & -3 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 5 \end{pmatrix}$ を満たす x, y, z を求めることを考える．

(i) 関数 `gj` を呼び出す際の引数の値を示せ．

(ii) (i) で呼び出される関数 `gj` の 11 行目で `aik` に代入される値を代入が実行されるごとに示せ．

(iii) (i) で呼び出される関数 `gj` の返す値と，そのどれが求めたい x, y, z に対応するかを示せ．

(2) 関数 `gj` の 7 行目で，代入の右辺の式のなかで `1.0` をかけている理由を述べよ．

(3) $\begin{pmatrix} 1 & 1 & -1 \\ 3 & 3 & -6 \\ 2 & -3 & 3 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ 9 \end{pmatrix}$ を満たす x, y, z を求めることを考える．

(i) 関数 gj を用いるとうまく解くことができない．その理由を問題が生じるプログラムの行と関連づけて説明せよ．

(ii) (i) の問題を解決する pivoting についてそのアイデアを 3 行程度で述べよ．

(4) $\begin{pmatrix} 2^{64}+1 & 2^{64} & 2^{64} \\ 2^{64} & 2^{64}+1 & 2^{64} \\ 2^{64} & 2^{64} & 2^{64}+1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 3\cdot 2^{64}+1 \\ 3\cdot 2^{64}+1 \\ 3\cdot 2^{64}+1 \end{pmatrix}$ を満たす x, y, z は pivoting を用いた関数 gj でも求めることができなかった．その理由を述べよ．

問題 3

駅 $s_0, s_1, \ldots$ がこの順に一列に並んだ鉄道路線を使って駅 s_0 から駅 s_n まで移動したい．この路線では，駅 s_i から駅 s_j までの移動には $\texttt{cost}(i,j)$ 円の運賃がかかる（$\texttt{cost}(i,i) = 0$ であるとする）．運賃の計算方法はやや複雑で，そのためわざわざ複数枚の切符を使う方が 1 枚の切符を買うより安い場合がある．例えば，駅 s_i から駅 s_k を経由して駅 s_j へ向かうとき，$\texttt{cost}(i,j) > \texttt{cost}(i,k)+\texttt{cost}(k,j)$ であれば，s_i から s_k までと s_k から s_j までの 2 枚の切符を買う方が安くなる．このことをふまえ，以下の問に答えよ．なお，逆方向の列車に乗るような経路は考えないものとする．

(1) 右の関数 ticket2(n) は駅 s_0 から駅 s_n まで高々2 枚の切符を使って移動する場合の最安費用を計算する．空欄 (A) を埋めよ．

```
1  def ticket2(n)
2    m = cost(0,n)
3    for i in 1 .. (n - 1)
4      x = (A)
5      if x < m
6        m = x
7      end
8    end
9    m
10 end
```

(2) 駅 s_0 から駅 s_n までの最安費用を $ticket(n)$ とする．切符は何枚使っても良いとすると，$ticket(n)$ は以下の漸化式を満たす．空欄 (B) を埋めよ．その理由も簡単に説明すること．

$$ticket(0) = 0$$
$$ticket(n) = \min_{0 \le i \le n-1} (\boxed{\text{(B)}} + \texttt{cost}(i,n))$$

(3) 上記漸化式をもとに定義した，$ticket(n)$ を求める再帰関数 ticket_r(n) を左下に示す．空欄 (C)・(D) を埋めよ．

(4) 再帰関数 ticket_r の効率を動的計画法を用いて改善したものが右下の ticket_dp である．空欄 (E) ～ (G) を埋めよ．

(5) ticket_dp の時間計算量をオーダ表記を用いて表せ．導出の過程も簡単に述べること．

```
1  def ticket_r(n)
2    if n == 0
3      (C)
4    else
5      m = cost(0,n)
6      for i in 0 .. (n - 1)
7        x = (D)
8        if x < m
9          m = x
10       end
11     end
12     m
13   end
14 end
```

```
1  def ticket_dp(n)
2    table = make1d(n + 1)
3    for k in 0 .. n
4      if k == 0
5        (E)
6      else
7        m = cost(0,k)
8        for i in 1 .. (k - 1)
9          x = (F)
10         if x < m
11           m = x
12         end
13       end
14       (G)
15     end
16   end
17   table[n]
18 end
```