2012 年度情報科学　期末試験　答え

文責　愛知

問題 1

(a) あえて再帰的であるということを示すなら 5*(4*(3*(2*1)))ですが、普通に階乗の計算ですね。

答え…5 の階乗を計算し、120 を出力する。

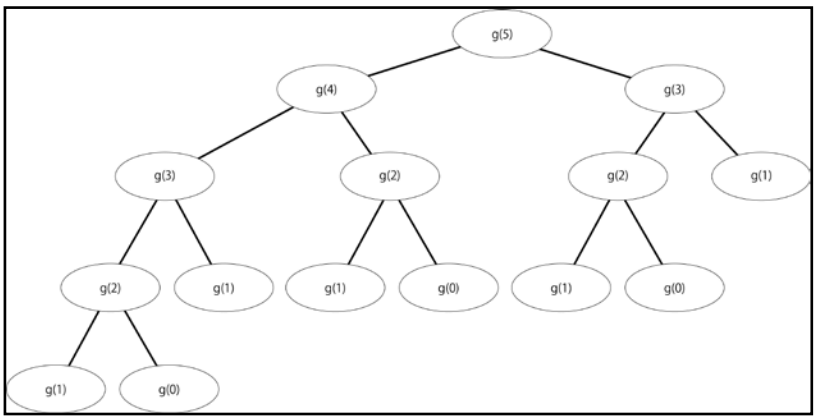
(b) n!=1*2*…*n をそのまま使いましょう。かける数字が大きくなっているので for 文がいいかなと考えましたが、もちろん while 文でも可能です。

答え

```
def f(n)
  fact = 1
  for i in 1..n
   fact = fact*i
  end
end
```

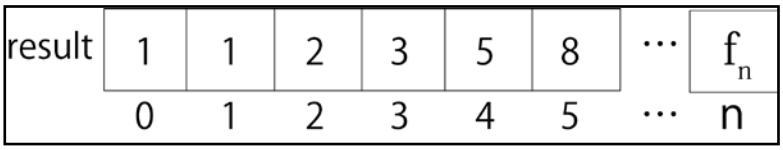
(c) おおまかな構造が g(n)=g(n-1)+g(n-2)になっていることから、フィボナッチ数の計算をしていることが分かると思います。しかし注意すべきなのが、ここでは if 文が「n>=2」となっているので n=2 の時も if 文の中に入る、つまり g(0)=1 が使われるということです。よってこのフィボナッチ数は n=0 から始まっています。数学で使われるフィボナッチ数と一つずれるので注意しましょう。

答え…下図のような再帰呼び出しによって n=5 のフィボナッチ数を求めており、f=8 を出力する。



(d) (n+1)個の成分を持つ配列を作り、そこに順番にフィボナッチ数を入れていくというプログラムです。アは i の初期化、イは while 文を繰り返す条件を示し、実際にフィボナッチ数を計算しているのはウ、次の数に対象をずらしているのがエです。最後の出力が配列の何番目かを指定しているのがおオになります。配列は result[0]から始まっているので n+1 個の配列の最後は result[n]であることに注意しましょう。

答え…ア… 0
イ…i=<n
ウ…result[i]=result[i-1]+result[i-2]
エ…i+1
オ…n



問題 2

(a)任意の二数の選び方は全部で ${}_nC_2$、つまり O(n^2)個ありますから、これを全部求め、その中から最大のものを求めるというのが一番考えやすく、ここの問題にあうアルゴリズムになります。

答え…

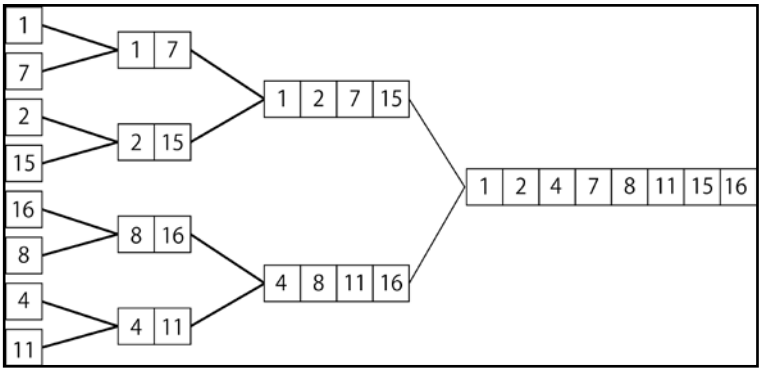
```
def minabs(a)
  k=abs(a[1]-a[0])
  for i in 1..n
    for j in 0...i-1
      if k>abs(a[i]-a[j])
        k=abs(a[i]-a[j])
      end
    end
  end
  k
end
```

(b) 整列しているとは、数字が順番に並んでいるということですから、差の絶対値が最小になるのはいづれかの数字とその隣の数字との差でしかありえません。よって隣り合う数字同士の差だけを取ればよく、そのときオーダーは O(n)になります。

```
答え…
def minabs2(a)
  k=abs(a[1]-a[0])
  for i in 2..n                #i=1 の時は初期条件としてでやったので飛ばす
    if k>abs(a[i]-a[i-1])
      k= abs(a[i]-a[i-1])
    end
  end
end
```

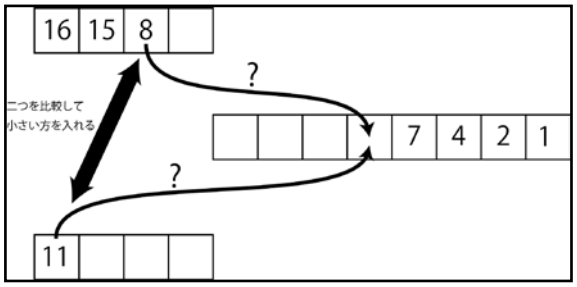
(c) 実は簡単です。a を整列させてから(b)を用いるのです。配列の整列方法は講義でやりましたが、併合配列法を用いればその計算量は $O(n \log n)$です。よって $O(n \log n)$と O(n)の計算をするので結局計算量は $O(n \log n)$です。

答え…併合配列法を用いて a を整列させてから(b)のアルゴリズムを使えばよい。併合配列法の計算量は $O(n \log n)$なので、O(n)の(b)と合わせて $O(n \log n)$のオーダーで計算できる。(併合配列法について説明すべきかどうかは分かりませんが、一応以下に説明と計算量の求め方を示します。)



併合配列法とは、上の図のように配列を小さな配列に分け、それを二つずつ組み合わせていき、最終的に求めたい整列を導く方法です。大きさ k の二つの配列を組み合わせて 2k の配列を作る時、はじめにある成分(下図では一番右の成分)から順に 2k の配列に入れていきますが、この時二つの配列は整列されているので、それぞれの配列の一番右に残っている二つの数を比較すれ

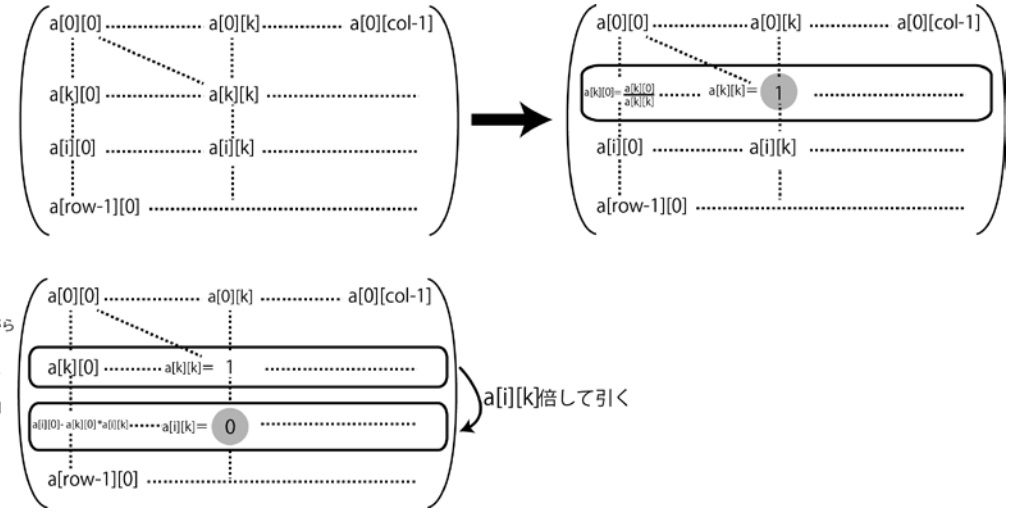
ば十分です。よってこの部分の計算量は 2k です。



また、$n=2^m$ と表されるとき、、成分が 1 個の配列同士の組み合わせ(計算量 2)が $2^{(m-1)}$回、成分が 2 個の配列同士の組み合わせ(計算量 4)が $2^{(m-2)}$回、…という風に計算が行われるので、全計算量は $2*2^{(m-1)}+4*2^{(m-2)}+\cdots 2^m*2^0 = m*2^m$ 回となります。$m = \log_2 n$なので、これは$n \log_2 n$と同じです。また、n が 2^m でないとき、$n=2^m+2^a+2^b+\cdots$ とおいて上と同じ議論をすれば $2^a+2^b+\cdots$の項は最高次数になりえないので、オーダー計算の時には無視されるということが分かります。よって一般的にどんな n に対しても併合整列法の計算量は $O(n \log_2 n)$ です。

問題 3

(a)連立一次方程式を解くためのガウス-ジョルダン法ですが、やっていることは行列の行基本変形による単位行列化です。プログラムでは行列の k 行目の k 列以外を全て 0 にするという操作を k を変えながら繰り返しています。以下の図を用いて見てみましょう。



図のようにして対角成分を 1、その他の成分を 0 にするような操作を繰り返します。右上の部分の操作がア、左下の部分の操作がイになります。

答え…ア…1.0/akk　(値に小数の形を保たせるために 1.0 をかけています。Ruby は「整数÷整数なら答えも整数だ！」と考える言語なので、1.0 をかけていない場合、akk が整数だと値が整数値しかとらなくなり正しい答えが導かれなくなります。が、おそらく 1/akk としても減点にはならないのではと思います。)

イ…aik*a[k][j]

(b) (c)への準備です。

答え…割り算をしている a[k][k]が 0 近くの時は割り算の結果が大きな数字になり誤差が大きくなる。

(c)Pivotiong が何かというのは知識の問題ですが、割り算をしている a[k][k]が 0 の時は計算ができず、また 0 に近い時は(c)で述べたように割り算の結果が大きな数字になり、誤差が大きくなるという問題を修正するためにする操作のことです。実際には a[k][k]の位置にくるものが k 列目で一番大きな数になるように行ごと交換してやればよいということになります。

答え…割り算をしている a[k][k]が 0 の時は計算不可能になり、また 0 に近い時は割り算の結果が大きな数字になり、情報落ち誤差が生じるという問題を修正するために行う操作のこと。よって、

for k in 0..(col-2) の後に

```
max=maxrow(a,k)
swap(a,k,max)
```

の二行を入れれば Pivotiong ができる。

(d)ここまでは講義そのものでしたが、この問題は新しい問題です。線形代数で学んだとおり、(拡大されていない)n×n 行列の Rank が n でないときに解が不定、もしくは不能になります。よって単純に akk=0 となったら abort()…かと思われるかもしれませんが、例えば以下のような場合、a[0][0]=0 ですが、解は一意に存在します。

$$\begin{pmatrix} 0 & 4 \\ 1 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} 8 \\ 6 \end{pmatrix}$$

しかし、ここで問題文を見返してみると、「(c) で修正したプログラムを適用した時に」

と書いてあります。0 は一番絶対値の小さい数ですから、もし同じ行に 0 以外の数が存在すれば必ずその行と入れ替わるということになります。つまり上のような場合、二つの行が入れ替わり、計算可能になるのです。このようにして、結局 akk=0 ならば abort()すればよいということになります。Pivoting には解の不定性を正しく判断する機能も持っていたということが分かりますね。

答え…(c)で追加した二行の下に

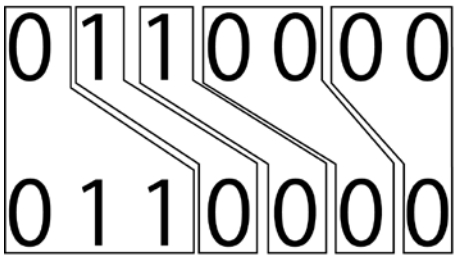
```
if a[k][k]==0
  abort()
end
```

を追加すればよい。

問題 4

(a) 文意が把握できれば問題ないでしょう。

答え…1,0,0,2,2(下図参照)



(b) よくわからない再帰があるのでまずは問題文の式をプログラム通りに追ってみましょう。

上の図は分かりにくいかもしれませんので、自分でプログラムを追ってみてください。

答え…"11011"が返ってくる。

(c) (b)を解くと、どうやら初めに設定する変数 n が「許される最大の連結数」、as と bs が「連結の最初の部分」を表していると考えられます。as,bs に関しては""をはじめに与えておけば全ての場合を計算してくれるのだろうと考えられますが、問題は n です。今回は(a)で答えを知っているので n=5 とすればいいのですが、一般にこの問題を『解く』といった場合には n は未知であると考えるのが自然でしょう。うまく考えれば n の上限を与えられるのかなとも思ったのですが、分かりませんでした。よってここでは以下のような美しくない解答を書かせていただきます。申し訳ありません。

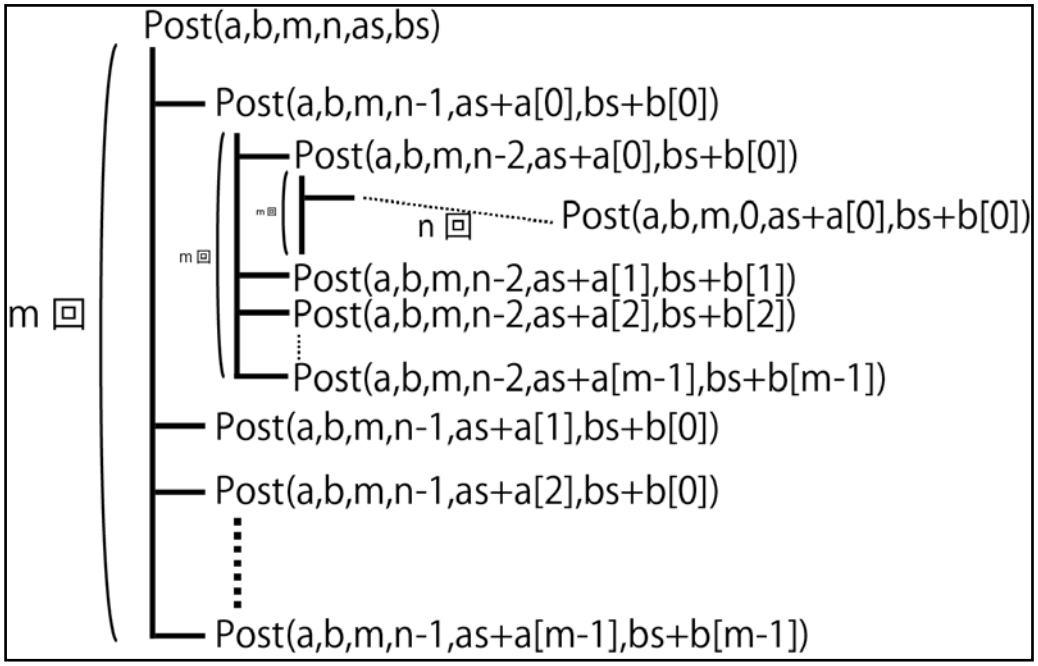
答え…Post(["1","0","00"],[["0","011","0"],3,n,"","")を n=1 から n を増やしながら順に実行し、空文字列でない文字列が出てきた時に終了すればよい。この時出力は"0110000"である。

(d) 繰り返しの計算が行われているのは while 文とその中の再帰です。決して同じ文字列にならないので、as,bs による例外的な状況は考えなくてよく、while 文を抜けるのは常に n=0 になった時、もしくは k>=m となった時です。よって n が 0 でないとき、Post(a,b,m,n-1,as+a[k],bs+b[k])は k=0～m-1 の
m 回実行されます。そこまで考えれば、自分で図を書いたり、小さい n と m で計算してみれば分かると思いますが、計算総数は

$$m^n + m^{n-1} + \cdots m^1 = \frac{m}{m-1} m^n$$

です。m/m-1 はほぼ 1 の係数とみなせるので、オーダーの時には省きます。

答え…$O(m^n)$



最後に

私は 2012 年度の問題はどれも重たいと感じましたが皆様はいかがでしたでしょうか。

「誤差」「再帰」あたりの概念は毎年出題されているようなので、対策をしておいた方がいいかもしれません。