情報科学 2010 年度 冬学期 解答＆解説

問題

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 `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 ${}_nC_k$ を求める関数 `combination_loop(n,k)` に関して以下の問に答えよ。

```
def combination_loop(n,k)
  c=make2d(n+1,n+1)
  for i in 0..n
    c[i][0]=1
    for j in 1..(i-1)
      c[i][j]=c[i-1][j-1]+c[i-1][j] # (ア)
    end
    イ
  end
  c[n][k]
end
```

(a) プログラムを正しく動かすために、イ に何を入れればよいか。

一問目からソースコードを書かせる問題です。単純な動的計画法の問題なので簡単でしょう。

まず、この関数が返すべき値は ${}_nC_k$ です。そこで、この関数が返している文を見てみると、`c[n][k]` となっています。どうやら、`c[i][j]` は ${}_iC_j$ の計算結果を表しているような気がします。また、(ア) を見てみると、${}_nC_k$ の関係式である、${}_nC_k = {}_{n-1}C_{k-1} + {}_{n-1}C_k$ と一致しているように見えます。どうやら、`c[i][j]` が ${}_iC_j$ を表していることと、(ア) で ${}_nC_k = {}_{n-1}C_{k-1} + {}_{n-1}C_k$ を利用しているのは間違いなさそうです。

さて、上の関係式は、$0 < j < i$ の場合しか利用できないので、$j = 0, i$ の時には別の方法で計算する必要性があります。ソースコードを見てみると、`c[i][0]=1` という行があり、どうやらここで $j = 0$ の場合の計算[*1]を行っているようです。しかし、このソースコードを読んでも、$j = i$ の場合の計算は行われていないようです。よって、この場合の計算を行えばこのソースコードは目的通りの動きをしそうです。よって、出題者が求めている解答は`c[i][i]=1`であると考えられます。

問題

(b) `combination_loop(3,2)` と呼び出したとき、(ア) の行の文が実行されて、どのような代入がされるかを、代入される順に、添字 i, j の値と代入される値を示せ。

Ruby の中の人通りに実行していけば良いでしょう。

講義をとった人ならばわかると思いますが、`for i in 0..n` は変数 `i` を `0` から `n` まで変化させながら、対応する `end` までの間を繰り返し実行させます。この点に注意しながらワンステップずつ実行される様子を見ていきます。

[*1] 計算と言っても 1 なので、ただの代入にすぎないのですが。

ソースコード	i	j	その他
c=make2d(n+1,n+1)	nil	nil	c へ 4 * 4 配列を代入
for i in 0..n	0	nil	
c[i][0]=1	0	nil	c[0][0] = 1
for j in 1..(i-1)	0	nil	1..-1 より実行せず
end	0	nil	
c[i][i]=1	0	nil	c[0][0] = 1
end	0	nil	
for i in 0..n	1	nil	
c[i][0]=1	1	nil	c[1][0] = 1
for j in 1..(i-1)	1	nil	1..0 より実行せず
end	1	nil	
c[i][i]=1	1	nil	c[1][1] = 1
end	1	nil	
for i in 0..n	2	nil	
c[i][0]=1	2	nil	c[2][0] = 1
for j in 1..(i-1)	2	1	
c[i][j]=c[i-1][j-1]+c[i-1][j]	2	1	c[2][1] = c[1][0] + c[1][1] = 2
end	2	1	
c[i][i]=1	2	1	c[2][2] = 1
end	2	1	
for i in 0..n	3	1	
c[i][0]=1	3	1	c[3][0] = 1
for j in 1..(i-1)	3	1	
c[i][j]=c[i-1][j-1]+c[i-1][j]	3	1	c[3][1] = c[2][0] + c[2][1] = 3
end	3	1	
for j in 1..(i-1)	3	2	
c[i][j]=c[i-1][j-1]+c[i-1][j]	3	2	c[3][2] = c[2][1] + c[2][2] = 3
end	3	2	
c[i][i]=1	3	2	c[3][3] = 1
end	3	2	
c[n][k]	3	2	c[3][2] = 3

以上より、$i=2, j=1$ の時 2、$i=3, j=1$ の時 3, $i=3, j=2$ の時 3です。

問題

(c) n に関して、この計算方法の計算量を示せ。求める過程も示せ。

計算のオーダーを求める問題です。

今回のプログラムで計算を行っているのは、(ア) の行ぐらいですので、ここが何回実行されるかを n の式で表したいと思います。

外側の for ループは、i を 0 から n に変化させながら $n+1$ 回実行されます。また、内側の for ループについては、i に依存し、$i \geq 2$ の時 $i-1$ 回実行されます。以上より、$n \geq 2$ の時、$1+2+3+4+\cdots+(n-1) = \frac{1}{2}n(n-1)$ 回計算されることが分かります。よって、解答はオーダーで答えれば$O(n^2)$となります。

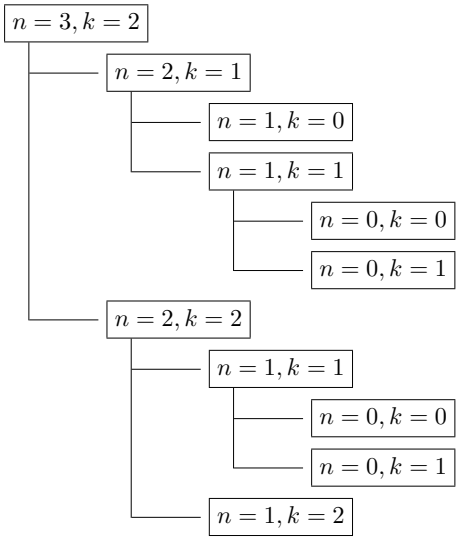
問題

${}_nC_k$ を求める関数 `combination(n,k)` について以下の問に答えよ。

```
def combination(n,k)
  if k > n
    0
  else
    if k == 0
      1
    else
      combination(n-1,k-1)+combination(n-1,k)
    end
  end
end
```

(d) `combination(3,2)` と呼び出したとき、`combination(n,k)` がどのように再帰的に呼び出されるかを示せ。ただし、式 `a+b` は `a` を `b` より前に評価するものとする。

上のソースコードの条件分岐と、再帰呼び出しの部分のみに注目すれば、$0 \neq k \leq n$ の時に再帰呼び出しを必ず二回行うようです。そういう訳で、呼び出し関係をツリー構造で示しましょう。



問題

(e) ${}_nC_n$ の値は、`combination_loop(n,n)` では、どのように求めているか。`combination(n,n)` では、どのように求めているか。

出題者がどの様な解答を求めているかは自分には分かりませんが、自分ならば以下のような書く気がします。

`combination_loop(n,n)` では、${}_nC_k = {}_{n-1}C_{k-1} + {}_{n-1}C_k$ という関係式を用いて、動的計画法によって ${}_iC_j$ $(0 \leq i \leq n, 0 \leq j \leq i)$ を全て求め、最後に必要な値を返している。一方、`combination(n,n)` では、上と同じ関係式を用いて、再起呼び出しによって必要な ${}_iC_j$ だけを求めて計算を行って値を返している。

まあ、どの様に求めているかと聞かれたら、私には日本語能力が欠如しているので上のようにしか書けません。“ 比較検討せよ ” と言われたら突っ込みどころは色々とあって、書きやすいのですが……

問題

問題 2 大きさ n の配列 a に m 種類の正の整数が格納されているとき、a に現れる整数を配列 b に求め、各整数の a に現れる回数を配列 c に求めたい。たとえば、配列 c に求めたい。たとえば、配列 a が [3,1,4,1,5,9,2,6,5,3] であるとき、n は 10、m は 7 である。このとき、b には [3,1,4,5,9,2,6]、c には [2,2,1,2,1,1,1] を返すことができる。

(a) 次のプログラムは、a から b と c を求めるプログラムの一例である。このプログラムの計算量を n と m を用いて表せ。なお、引数 b と c には大きさ m の配列が渡される。配列 b の各要素は 0 で初期化されていると仮定する。

```
def intcount(a, b, c)
  for i in 0..(a.length()-1)
    x = a[i]
    j = 0
    while b[j] != 0 && b[j] != x
      j = j + 1
    end
    if b[j] == 0
      b[j] = x
      c[j] = 1
    else
      c[j] = c[j] + 1
    end
  end
end
```

プログラムを見て、オーダーを求める問題です。普通に書けば、$O(n \log m)$ ぐらいになると思うんですが、上のソースコードでは線形探索を行っているので、$O(nm)$ になってますね。

オーダーの求め方は、多くの場合はループが何回程度繰り返されるかを見れば大丈夫です。今回の場合、外側の for ループでは a.length 回、つまり n 回繰り返されます。ただ、内側の while ループは何回程度繰り返されるかソースをぱっと見た感じではわかりません。このような時にはソースコードの意味を考えることが重要です。今回の場合は、b の中に、今見ている数字が既に登録されているかを線形探索で行っています。線形探索では中のデータの数を m とすれば、平均 $\frac{m}{2}$ 回程度です。今回の問題では、捜索を行うごとに b 内に登録されているデータが増えていくので常に平均 $\frac{m}{2}$ ではありませんが、いずれにしても $O(m)$ でしょう。よって、外側のループを考慮に入れれば、$O(nm)$ となります。

問題

(b) a が整列されていると仮定する。この場合に計算量が $O(n)$ で済むように、上のプログラムを参考に新たに intcount(a,b,c) の定義を書け。

$O(n)$ ということは、上のコードのオーダーと比較して考えればわかるように、探索に時間をかけることができないということです。

さて、ソートが行われているということは数字が順番に並んでいるということです。今回の場合は、順番に並んでいるというのは嬉しくは無いのですが、同じ数字は必ず連続して並んでいるというのが重要です。つまり、配列を一番初めから順番に見ていったとき、今見ている値と次の値が異なっていたら、次の値はまだ一度も出現していないことが保証されています。よって、新しい値が出てきたらそれを b の一番後ろに追加し、連続した数を c の一番後ろに追加すればいい事になります。これならば $O(n)$ となりそうです。

ある程度分かりやすく書いたと思われるソースは以下のようなものでしょう。b[index] = last 及び c[index] = count が二回書かれていてあまり綺麗じゃ無いような気もしますが、あまり気にしないようにお願いし

ます……。

```ruby
def intcount(a, b, c)
  last = nil
  count = 0
  index = 0
  for i in 0..(a.length()-1)
    x = a[i]
    if last != x
      if last != nil
        b[index] = last
        c[index] = count
        index += 1
      end
      last = x
      count = 1
    else
      count += 1
    end
  end
  b[index] = last
  c[index] = count
end
```

問題

問題 3 以下の問に答えよ。

(a) 丸め誤差とは何か、例を挙げて説明しなさい。

Wikipedia によれば、

> 数値を、どこかの桁で端数処理（切り上げ・切り捨て・四捨五入・五捨六入・丸めなど）をしたときに生じる誤差

と書かれています。具体例を書けとのことなので具体例を挙げるとすると、0.1 や 0.3 を浮動小数点で表現した際にこの誤差が発生します。なぜそのような誤差が発生するのかと言うと、その値を 2 進数で表現すると、$0.1_{10} = 0.00011001100110011\cdots_2$ のように、無限小数になってしまい、有限のビット数では正確な値を表せない為です。

より詳しいこともかけますが、大体上のようなことが書いてあれば正解になると思います。10 進数から 2 進数への変換の方法などは、教科書や Wikipedia 内の二進法のページを御覧ください。

問題

(b) Ruby (irb) において，以下の下線部のような入力をしたところ、結果はどのようになるか述べなさい。また、この理由を簡単に述べなさい。

```
irb(main):001:0> 0.3==0.1*3
```

true になってしまったら、ただの当たり前の算数の問題になので、false になります。理由は先程説明させられた丸め誤差で 0.1 の値がずれており、それを 3 倍したら更にずれて、同じく丸め誤差でずれている 0.3 と比較したら値が合うわけがなかろう、ということです。

恐らく上のような解答で点数は貰えると考えますが、実際はここまで簡単ではありません。実は `0.2 == 0.1 * 2` は true になります。理由は自分でお考えください。

問題

(c) 桁落ち誤差とは何か、例を挙げて説明しなさい。

Wikipedia によれば、

> 桁落ち（けたおち）とは、値がほぼ等しく丸め誤差をもつ数値どうしの減算を行った場合、有効数字が減少する

こと。絶対値がほぼ等しく符号が異なる数値どうしの加算の場合も同様。 浮動小数点数では、上位の桁がゼロになると、正規化によってそれを詰め、以下の桁に”0” を強制的に挿入するので、下位の桁が信頼できないものになる。

と書かれています。

具体例を挙げましょう。2 進数で例を挙げろとは書かれてないので、10 進数で考えるとします。有効数字 3 桁で 1.00 から 9.90×10^{-1} を引くとすると、“ 有効数字 3 桁同士だから結果も 3 桁に決まっている ” ということは決してなく、1×10^{-2} で有効数字は 1 桁となってしまいます。この時に相対誤差が $\frac{1}{100}$ 程度だったのが、1 程度[*2]になってしまいます。この有効数字の現象が桁落ちです。パソコン上では常に一定の桁数値を持っていますが、その下の桁が実際は信用出来なくなってしまうという話です。

問題

(d) A 君と B 君が 2 次方程式の解を求める以下のような関数 `fa` と `fb` を作成した。(これらの関数は 2 次方程式が実数解を持つときに使われると仮定する。)

```
include(Math)

def fa(a,b,c)
  d=b**2-4.0*a*c
  x1=(-b+sqrt(d))/a/2
  x2=(-b-sqrt(d))/a/2
  [x1, x2]
end
  def fb(a,b,c)
  d=b**2-4.0*a*c
  x1=(-b+sqrt(d))/a/2
  x2=(-b-sqrt(d))/a/2
  if b > 0
    x1 = (1.0*c)/a/x2
  else
    x2 = (1.0*c)/a/x1
  end
  [x1, x2]
end
def check(a,b,c,x)
  [a*x[0]**2+b*x[0]+c,a*x[1]**2+b*x[1]+c]
end
```

2 次方程式 $x^2 - 100x + 1 = 0$ の解を両方のプログラムで計算し、結果を関数 `check` で検査したところ、結果は以下のようになった。

```
irb(main):002:0> check(1,-100,1,fa(1,-100,1))
[0.0, 1.22124532708767e-13]
irb(main):003:0> check(1,-100,1,fb(1,-100,1))
[0.0, 0.0]
```

B 君のプログラムでは A 君のプログラムにない部分が追加されているが、この部分は何を目的としているか述べなさい。

[*2] もはや値は全く信用出来ないな。

(a) は (b) への布石でしたが、この流れから (c) は (d) への布石だと読んで、“ 桁落ち誤差を防ぐため ” と書けば問題ないのではないでしょうか。理由も訊ねられてないのでいい考えだと思います。

しかし、これだけでは寂しいので解説を書きます。解答の際には以下の内容を掻い摘んで書けば、答案が短くて場合によっては減点されてしまうかもしれないと思っている人は安心できるでしょう。

2 次方程式の解の公式は、解を求めたい 2 次方程式を $ax^2+bx+c=0$ とすれば $x=\dfrac{-b\pm\sqrt{b^2-4ac}}{2a}$ です。桁落ち誤差が発生する条件は、“ 値がほぼ等しく丸め誤差をもつ数値どうしの減算を行った場合 ” でした。これはどのような時に発生するかというと、$|4ac|\ll b^2$ の時です[*3]。何故かというとこの時、$\sqrt{b^2-4ac}=|b|\sqrt{1-\dfrac{4ac}{b^2}}\fallingdotseq|b|$ となり、$-b\pm\sqrt{b^2-4ac}\fallingdotseq-b\pm|b|$ の値が、$b\geq 0$ の時は $+$ 側の答えで桁落ちが発生し、$b<0$ の時は $-$ 側の答えで桁落ちが発生します。

さて、ここで上のソースに合わせて、$x_1=\dfrac{-b+\sqrt{b^2-4ac}}{2a}, x_2=\dfrac{-b-\sqrt{b^2-4ac}}{2a}$ と置きます。この時、解と係数の関係によって、$x_1x_2=\dfrac{c}{a}$ が成立します。よって、それぞれ $x_1=\dfrac{c}{ax_2}, x_2=\dfrac{c}{ax_1}$ が成立します。掛け算、割り算の時は桁落ちは発生しないので、桁落ちした計算結果は用いずに上の式で桁落ちしていない計算結果を用いてもう片方の解を計算しようというのが追加された式の役割です。

確認すればわかるように、$b\geq 0$ の時[*4]は $+$ 側の解の x_1 が桁落ちする可能性があるので、x_2 の値を利用して、x_1 を再計算しています。また、$b<0$ の時も同様に x_2 の値を x_1 で再計算しています。

このようにすることによって、誤差が小さくなります。それを証明するように、`check` で解を方程式に代入すると、愚直な式の場合 `1.22124532708767e-13` という 0 には近いですが、0 ではない値になってしまいますが、工夫した場合 `0` となり、コンピューターで値を表現する上ではほぼ厳密な 0 になるようになるのです。

問題

問題 4 横 $(M+1)$、縦 $(N+1)$ 個のマス目を持つ盤がある。駒が、座標 $(0,0)$ で表された左下のマス目から、座標 (M,N) まで移動するとする。一度に駒が動ける範囲は 3 通りに制限されており、右、上、斜め右上、だけとする。このとき以下の問に答えなさい。

(a) 座標 $(0,0)$ から座標 (m,n) まで駒が取り得る経路の数を $T_{m,n}$ とする（ただし $1\leq m\leq M, 1\leq n\leq N$ とする)。とのとき

$$T_{m,n}=T_{m,n-1}+T_{m-1,n}+T_{m-1,n-1}$$

となることを示しなさい。このとき初項

$$T_{0,0}, T_{0,i}, T_{j,0}\qquad (1\leq i\leq M, 1\leq j\leq N)$$

はどのように決めたらよいか示しなさい。

滅茶苦茶簡単な入試問題ですね。座標 (m,n) へ移動するためには、1 つ前に $(m,n-1)$、$(m-1,n)$、$(m-1,n-1)$ のいずれかの座標に必ずいるので、(m,n) への移動の通り数は、$T_{m,n}=T_{m,n-1}+T_{m-1,n}+T_{m-1,n-1}$ と表すことができます。また、$T_{0,0}$ については、初めからそこにいるので 1 通りであり、$T_{0,i}, T_{j,0}$ $(1\leq i\leq M, 1\leq j\leq N)$ については、横、または縦からの移動しか存在しないので、$T_{0,i}=T_{0,i-1}, T_{j,0}=T_{j-1,0}$ となります。よって、$T_{0,i}=T_{j,0}=T_{0,0}=0$ となります。

問題

(b) $T_{M,N}$ を求めるために動的計画法を用いたアルゴリズムの指針を示しなさい。

動的計画法は簡単に言えば、わかる部分を先に埋めて漸化式で答えを出す、という方法です。今回の場合は、$T_{0,0}, T_{0,i}, T_{j,0}$ $(1\leq i\leq M, 1\leq j\leq N)$ が 0 とわかっています。後は漸化式 $T_{m,n}=T_{m,n-1}+T_{m-1,n}+T_{m-1,n-1}$ を利用して $T_{M,N}$ まで計算すればいいですが、どのような順番で計算するかが問題です。しかし、今回はそれほど悩むこ

[*3] $b^2\fallingdotseq 4ax$ の時も b^2-4ac を計算する際に桁落ちが発生しますが、その時は、$|-b|\gg\sqrt{b^2-4ac}$ となっているはずなので、問題ないはずです。

[*4] ソース上では $b>0$ ですが……

ともなく、m が小さい方から大きい方へ、n が小さい方から大きい方へ順番に計算していけば何も問題ありません。アルゴリズムの指針としては、

1. T を $M \times N$ の配列で初期化
2. T[0][0]、T[0][i]、T[0][j] を 0 で初期化
3. for ループを二重にして、m を小さい方から大きい方へ、n を小さい方から大きい方へ変化させつつ、漸化式 $T_{m,n} = T_{m,n-1} + T_{m-1,n} + T_{m-1,n-1}$ を利用して $T_{M,N}$ を利用して、配列を埋める。
4. T[N][M] を返す。

というようになるでしょう。

具体的にソースを書くならば、

```
def calc(m, n)
  t = make2d(m + 1, n + 1)
  t[0][0] = 1
  for i in 1..m
    t[i][0] = 1
  end
  for j in 1..n
    t[0][j] = 1
  end
  for i in 1..m
    for j in 1..n
      t[i][j] = t[i][j - 1] + t[i - 1][j] + t[i - 1][j - 1]
    end
  end

  t[m][n]
end
```

のようになるでしょう。

あとがき

以上で、情報科学 2010 年度 冬学期試験の解答と適当な解説は終わりです。他にも色々と書こうとしましたが、自分には稚拙な文章しか書けないので、わざわざゴミのような文章をこの世の中に増やす必要性もなかろうということで、バッサリ削除しました。

言おうとしていたことを要約すると、問題 1 の (a) は自由度が大きすぎだと言うことです。return 文を使えればわざわざ他の部分を使わなくても値を計算して値を返すことができるということです。

問題 2 の (b) は分かりやすさ (笑) を重視してソースが長くなっていますが、以下のソースのほうがわかりやすいかもしれません。

```
def intcount(a, b, c)
  last = nil
  index = -1
  for i in 0...a.length
    if last != a[i]
      last = a[i]
      index += 1
      b[index] = last
      c[index] = 0
    end
    c[index] += 1
  end
end
```

うん。こっちのほうがわかりやすいね。今から書き直す気がないから今のままでいいや。

情報科学で学ぶ Ruby なんて、ほんの触りに過ぎません。そんなことを宣っている自分は Ruby の初心者ですが、“ ブロック付きメソッド呼び出し ” を利用していない時点で、「何のための Ruby だよ……」と小言を言いたくなります。もちろん Ruby の特徴はそれだけではなくて、豊富な組み込みライブラリ、標準添付ライブラリ、そして日本人が開発しただけあって、日本語のリファレンスの充実っぷりには眼を見張るものがあります。今回プログラミングをしてプログラミング面白い！と思ったならば、是非とも極めて欲しいと思います。

もちろん、プログラミング言語は Ruby だけではありません。TIOBE Index[*5]のトップ 20 を挙げると、Java, C, C++, PHP, C#, Objective-C, (Visual) Basic, Python, Perl, JavaScript, Lua, Ruby, Delphi/Object Pascal, Lisp, Transact-SQL, Ada, RPG (OS/400), Pascal, F#, Assembly、とこれだけの言語が存在します。シェアが多い言語が必ずしもいいとは言いませんが、ある程度の参考にはなるとおもいます。ちなみに自分は Basic から、C に入って、よくわからないまま C++ と JavaScript を並行で勉強して、大学に入学してから Ruby と Java と PHP を学びました。触っただけの言語も含めれば、Java, C, C++, PHP, C#, Basic, Perl, JavaScript, Lua, Ruby, Lisp, Assembly ぐらいですかね。

一度も訊かれたことは無いですが、“ プログラミング言語ってどれがいいんですか？ ” という質問がよくネット上にあがっています。自分はその質問に対しては、“ 何がしたいんですか？ ” と訊ね返したいです。それぞれの言語には得意、不得意があります。しっかりと目的を持ってプログラミングを学んでもらいたいと思います。

ブラウザじゃないゲームを作りたいのならば、C/C++, C#, Java あたりでしょうか。プラウザゲームを作りたいならば、PHP, Ruby, Perl + SQL + HTML + CSS + JavaScript で確定でしょう。作業の効率化、簡単なスクリプトだったら、Python, Perl, Ruby, Pascal あたりでしょうか。変態に出会いたいならば Lisp, Brainfuck, A[*6], Whitespace あたりでしょうか[*7]。

もちろんゲームを Ruby だけ、簡単なスクリプトを C/C++ で書くことも当然出来ます。しかし、明らかに作業効率が異なります。個人的には、C/C++, C#, Java, D あたりから 1 つと、Ruby, Perl, Pascal あたりから 1 つの計 2 つぐらい使えるようになると、それぞれの言語の特徴を生かして上手く使い分けることができると思います。何にして

[*5] http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html
[*6] http://www.sksk.info/a.html
[*7] Lisp はまだ普通ですよ。

も、何か 1 つぐらいプログラミング言語を触らないと勝手を知ることもできないと思うので、目的がないのだったら Ruby は適当な選択だとは思います。

なんだかよくわからないことを長々と書きました。最後に一言。

良いプログラミング生活をお送りください！

小中高校生の夏休みがあと一週間ぐらいで終わってしまう 2011 年 8 月 22 日 — ろくにプログラムも書けないのに偉そうなことを言う暇人

本書の初出は UTaisaku-Web[*8]です。

[*8] http://todai.info/