情報科学 2011 年度共通試験問題 解答例・解説

2012 年度入学理科一類 10 組情報シケ対 編

2013 年 1 月 8 日

※この解答例と解説の正確さは保証しかねます.

問題 1

(a) $s(a,0,2) = \mathbf{4}$, $s(a,0,3) = \mathbf{-1}$, $s(a,0,4) = \mathbf{1}$

(b) $\boldsymbol{O(N)}$

(c) $s(a,i,j)$ は，(b) のような単純な反復計算によって求めるとする．このとき，$\boldsymbol{O(N^3)}$

(d)

i	0	1	2	3	4	5	6	7	8	9
sum	**8**	**4**	**0**	**2**	**6**	**1**	**6**	**9**	**2**	**10**
t	**8**	**8**	**8**	**8**	**8**	**8**	**8**	**9**	**9**	**10**

(e) $\boldsymbol{O(N)}$

解説 これらは主に教科書第 5 章からの出題である.

(a) については，例えば $s(a,0,3)$ であれば実際に与えられた数列 a の 0 番目から 2 番目の和である $8 + (-4) + (-5)$ を計算すればよい．問題中で，数列の最初は「0 番目」であることに注意すること.

(b) は，「単純な反復計算」というのはただ `a[0]` から `a[N-1]` までを順番に足していく，ということになるので，例えば次のようなプログラムがこれを実現する.

```
def s(a,i,j)
  sum = 0
  for k in i..(j-1)
    sum = sum + a[k]
  end
  sum
end
```

この場合，$s(a,0,N)$ を求めるなら N 回の足し算を行うことになる.

(c) では，`mss(a,0,N)` を求める場合，二重になっている繰り返しの外側が 0 から N まで，内側が i から N までとなるので，全体で二重ループの中身は $O(N^2)$ 回実行されることになり，その中身の計算量が $O(N)$ なので全体として計算量は $O(N^3)$ となる.

(d) では，プログラムを読むとループの中では次のような処理をしていることが分かる.

- `sum` に `a[i]` を足す.
- もし `sum` が `t` より大きいならば，`t` を `sum` にする.
- もし `sum` が 0 未満ならば，`sum` を 0 にする.

これに従って，$t = 0$,$sum = 0$ という初期条件の下で数列を順番に見ていって処理すればよい．

(e) で，関数 `mss0` 内の繰り返しは 0 から $N-1$ までとなるので，ループの回数は $O(N)$ 回で，ループの中では足し算と条件分岐をそれぞれ (N に関係のない) 定数回行なっているだけなので，全体の計算量は $O(N)$ となる．

問題 2

(a) `f(2,4) = 16` , `f(3,5) = 243`

(b) $c = a^b$ である．

解説 この問題の題材は「繰り返し二乗法」や「バイナリ法」などと呼ばれる，冪乗の「早い」計算法で，教科書では 5 章の「5.3.2 冪乗の計算アルゴリズム」というところで紹介されている．教科書中では行列の冪乗を計算しているが，整数の冪乗であっても原理は同じである．

例として，3^{21} の値を求めたいとしよう．まず，$27 = 1 + 4 + 16 = 2^0 + 2^2 + 2^4$ なので，

$3^{21} = 3^{2^0+2^2+2^4} = 3^{2^0} \times 3^{2^2} \times 3^{2^4}$

となる．ここで，再右辺に出てきた 3^{2^n} という形の冪乗は，$3^{2^{n+1}} = 3^{2^n+2^n} = (3^{2^n})^2$ であることから，$3^{2^0}(= 3)$ から $3^{2^1}, 3^{2^2}, 3^{2^3}$... と各々 1 回の掛け算で順番に求めていくことができて，それらのうち必要なもの (この場合は $3^{2^0}, 3^{2^2}, 3^{2^4}$) を掛けあわせれば求めたい冪乗の値が求まったことになる．

普通に反復で冪乗 a^N を求める場合，その計算量は $O(N)$ となるが，この繰り返し二乗法の場合，a^{2^n} たちを求めていくのに $O(\log N)$，それらを掛けあわせるのも $O(\log N)$ で，全体の計算量も $O(\log N)$ となり，この手法が単純な反復よりかなり「早く」計算できることがわかる．(詳しくは教科書を参照)

このようなことを知らなくても，プログラムを丹念に追っていけば **(a)** は解けるし，**(a)** の結果から **(b)** の答えの予想はつくのではないだろうか．

参考までに，`f(3,21)` を呼び出した場合，一番外側の `while` ループが一度実行されるたびに各値がどのように変わっていくかを表にすると次のようになる．一番上の行は初期値である．

z	x	y
1	$3 = 3^{2^0}$	$17 = 2^0 + 2^2 + 2^4$
$3 = 3^{2^0}$	$3 = 3^{2^0}$	$16 = 2^2 + 2^4$
$243 = 3^{2^0+2^2}$	$81 = 3^{2^2}$	$4 = 2^2$
$10460353203 = 3^{2^0+2^2+2^4}$	$43046721 = 3^{2^4}$	0

問題 3

(a) 一見すると乱数列のように見えるが，実際には確定的な計算により求めている数列中の数のこと．

(b) 疑似乱数列の周期が十分に長いこと．

(c) $(0, 1)$ 及び $(0, -1)$ をちょうど通過した際も通り抜けられると仮定して，

```
y - x * dy / dx <= 1 && y - x * dy / dx >= -1
```

(d) イ：`x = -x1` , ウ：`x = x1`

解説 この問題の前半は，疑似乱数に関する問題である．疑似乱数に関する記述は、教科書 6 章の「6.2.3 疑似乱数列」の中にある。疑似乱数列において，N 項目までの計算ができているときに (N+1) 項目の数

を求めるのは自らの直前の数項を用いること，および数列中の各々の値は有限の桁数の値であることから，疑似乱数列を計算し続けるといつか同じ部分の繰り返しになってしまう．よって，この繰り返しの周期をなるべく長くするのが望ましく，例えば「メルセンヌ・ツイスタ」という疑似乱数生成アルゴリズムは，$2^{19937}-1$ というとても長い周期を持つ．

(c) 以降は普通のプログラムの問題であるが，図を見て状況をつかむことが大切である．

空欄アに入れるべきは，点が y 軸を越えたとき (`x1 < 0`)，「きちんと穴が空いている部分を通ったか」，つまり「経路の y 軸との交点が $[-1,1]$ の中に入っているか」という条件である．これは，最後の移動が $(\mathtt{x},\mathtt{y})$ から $(\mathtt{x1},\mathtt{y1})=(\mathtt{x}+\mathtt{dx},\mathtt{y}+\mathtt{dy})$ への移動であったことから，$(\mathtt{x},\mathtt{y})$ と $(\mathtt{x}+\mathtt{dx},\mathtt{y}+\mathtt{dy})$ を結ぶ直線の y 切片が $[-1,1]$ に収まればよく，その y 切片は計算すると $\mathtt{y}-\mathtt{x}\times\dfrac{\mathtt{dy}}{\mathtt{dx}}$ となることから，解答のような条件式になる．

空欄イに入れるべきは，点が (見かけ上)y 軸を越え，かつ「穴が空いている部分を通らなかった」，つまり「y 軸により反射された」とき，`x` を更新する式である．図にもあるように，反射した際は y 軸より向こう側の部分を折り返してくればいいので，結局点の行き先は $(\mathtt{x1},\mathtt{y1})$ を y 軸で折り返した $(-\mathtt{x1},\mathtt{y1})$ であり，空欄に入れる式は解答のようになる．

空欄ウに入れるべき式は反射しなかった時の `x` であるから，これはそのまま `x1` を代入すればよい．

問題 4

(a) $maxGain(i,j)=$
`max(max(`$maxGain(i-1,j-1)$`,`$maxGain(i-1,j)$`),`$maxGain(i-1,j+1)$`)+`$gain(i,j)$

(b) 最初の `maxGain(m,m,m,n)` も 1 回の `maxGain` の呼び出しとカウントすると，$\mathbf{\frac{1}{2}(3^m-1)}$ 回

(c) ア：`0.0`

イ：`gain(1, j)`

ウ：`max(max(totalgain[i-1][j-1],totalgain[i-1][j]),totalgain[i-1][j+1])+gain(i,j)`

解説 この問題は主に第 7 章のうち，「7.4 動的計画法」に関しての出題である．

(a) $i>1$ のもとで，(i,j) というマスに進むには，$i-1$ 行目においては $(i-1,j-1),(i-1,j),(i-1,j+1)$ のいずれかのマスに居なければならない．逆にこの 3 つのマスのどれかにいれば，次の移動で (i,j) に進むことができる。$(i-1,j-1)$ を通った後 (i,j) に行く最大累積利益は，$(i-1,j-1)$ までの最大累積利益を用いて，$maxGain(i-1,j-1)+gain(i,j)$ と表せ，他の 2 つのマスについても同様である．従って，(i,j) までの最大累積利益は，これら 3 つのうち最大を返せばいいので，解答のようになる．

(b) `maxGain`(m,m,m,n) を計算する際の，m の値に対する `maxGain` の呼び出し回数を x_m とする．まず，$m>1$ では，$maxGain(m,j)$ を計算する際に $maxGain(m-1.j-1),maxGain(m-1,j),maxGain(m-1,j+1)$ を呼び出すが，$maxGain(m,m)$ の計算において $m>2n$ の場合、$i=1$ になるまで呼び出しても ((a) の問題文中でいう) 範囲外になることはない．よって，$m>1$ に対し，x_{m+1} は 1 行前の呼び出し回数 (x_m) を 3 倍したものに自らの呼び出し 1 回を加えたもの，つまり $x_{m+1}=3x_m+1$ という漸化式が成り立つ．さらに，$x_1=1$ であるので，この漸化式を解くと $x_m=\dfrac{1}{2}(3^m-1)$ となり，解答を得る．

(c) (a) の漸化式を見ると，$maxGain(i,j)$ を計算するためには，それぞれ引数が自分の i,j より小さい $maxGain$ の値が必要なことが分かる．よって，i,j が小さいほうから `totalGain[i][j]` を計算し

ていくことで，全ての i, j に対して `totalGain[i][j]`($=maxGain(i, j)$) を計算することができる．空欄アは，$j = 0, n + 1$ という「範囲外」のところの最大累積利益であるから，(a) に倣って `0.0` を入れればいいだろう．空欄イは，$i = 0$，つまり一番上の行の最大累積利益であるから，(b) に倣ってそのまま `gain(1, j)` を入れればよい．空欄ウは，それらがいずれも成立しないときの最大累積利益であるから，(a) の右辺の $maxGain$ を `totalGain` にしたものを入れればよい．