情報科学シケプリ

2009 年度入学理科一類 13 組三浦拳

範囲と傾向と対策

基本的に、(1)完成しているプログラムが何をしているかなどについて答える問題と、(2)未完成のプログラムを適切に完成させる問題と、(3)アルゴリズムについて理解を問う問題と、(4)プログラミングに関する用語や概念を問う問題（特にオブジェクト指向について）、でできています。(1),(2),(3)は基本的に授業で扱ったプログラムなので、ちゃんと毎回理解していればできるはずです。授業のプログラムを全部復習しなくても、使っている命令などは基本的なものばかりなので、一度理解してしまえば、プログラムを演繹的に読めば特別な知識が必要なものはないです。(4)については基本的なことなので、理解して暗記しましょう。

公式サイトのほうの解説が充実しているので、とりあえず読んでおくべき部分のリンクを張っておきます。

(1),(2),(3)

isrb を使った画像については、ここ 3 年間は出ていないようですが、出ても知りません。ソートの具体的なのは難しいから出ないんじゃないかな、と。

http://lecture.ecc.u-tokyo.ac.jp/~shagiya/

というわけでそこを除いたスライドを全部読んでおいてください。

(4)

レコードとオブジェクト

http://lecture.ecc.u-tokyo.ac.jp/~shagiya/z8.pdf

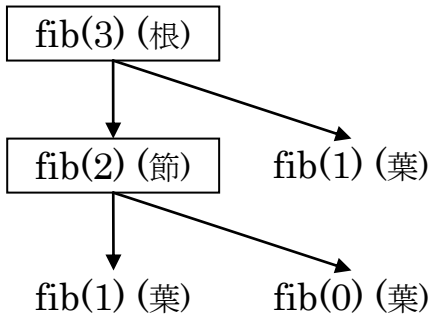
オブジェクト指向については確実に出て、しかもたぶんあんまり理解しなくてもプログラミングはできてしまっていたと思うので、ちゃんと理解しておいたほうがいいと思います。

06 冬の過去問の解答例は載せませんが、@hoge というのは self.hoge と同じで、インスタンス変数のことです。

過去問解答例　07 冬

共通問題 1

(a)



(b)

$$F(n) = \begin{cases} 2 + F(n-1) + F(n-2) & (n>1) \\ 0 & (n=0,1) \end{cases}$$

(c) f(n) 個

(d) 解答： $F(n) = 2\,f(n) - 2$

理由１：$\{F(n) + 2\} = \{F(n-1) + 2\} + \{F(n-2) + 2\}$

$F(n) + 2 = g(n)$ とおいて、同様に木構造を書いてみれば、やはり葉の数は f(n) 個。

葉は g(0) または g(1) で、$g(0) = g(1) = 2$ だから、$g(n) = 2\,f(n)$ よって $F(n) = 2\,f(n) - 2$

理由２：最下層の葉は f(n)個。

矢印が下に延びると１つの（枝が出ていない節又は葉）が２つの枝が出て２つの（枝が出ていない節又は葉）になるので１つ（枝が出ていない節又は葉）が増えると考えればいい。これを繰り返して１つの根が f(n)個の葉になるのだから、f(n)-1 回これが行われたということ。したがって $F(n) = 2\{f(n) - 1\} = 2\,f(n) - 2$

共通問題２

(a) ア．a[i][j] - a[k][j] * aik

イ．a[i][col-1]

(b) ・係数行列の対角要素が 0 になると 0 で割ることになり正規化できない

・割算の分母が 0 に近い値になるとき割算の結果非常に大きな数字が結果として表れ、数値計算の精度が下がる

⇒行または変数を、対角要素ができるだけ大きいものになるように行を入れ替えればよい

共通問題３

(a)

		A	**G**	**T**	**G**	**G**	**G**	**A**
	0	-2	-4	-6	-8	-10	-12	-14
A	-2	2	0	-2	-4	-6	-8	-10
G	-4	0	4	2	0	-2	-4	-6
G	-6	-2	-2	3	4	2	0	-2
T	-8	-4	0	4	2	3	1	-1
A	-10	-6	-2	2	3	1	2	3
A	-12	-8	-4	0	1	2	0	4

(b) AGTGGGA
AG-GTAA

共通問題４

(a)

(ア) データとメソッドをもつオブジェクトの内部構造を隠し、外部からは与えられたメソッドだけを操作できるようにして、オブジェクト内部のデータやオブジェクト内部の処理を意識させないようにすること。

(イ) あるメソッドの処理が変わると、それを利用している他のプログラムの部分をすべて変えなくてはならないという問題。

(b)

(ア) あるオブジェクトが、他のオブジェクトの、メソッドや変数などの性質を引き継ぐこと。

(イ) 既存ものと同じようなクラスを作るときに、また同じコーディングをしなくてはならないという問題。

(c)

(ア) メソッドの型が様々でも、動的結合によって同じ名前のインスタンスメソッドを呼び出して実行できるという性質

(イ) 同じようなメソッドを持つ異なるクラスを、同時に扱いにくいという問題

過去問解答例　08 冬

問題 1

(a)

```
def reverse(x, p, q)
        for i in p..((p+q)/2)
                t = x[i]
                x[i] = x[q-(i-p)]
                x[q-(i-p)] = t
        end
end
```

(b)

```
def transpose1(x,k)
        for i in 1..k
                rotate(x, 0, x.length()-1)
        end
end
```

計算量は　k(n+1)

情報科学シケプリ　09-S1-13　三浦拳

(c)

```
def transpose2(x,k)
        reverse(x, 0, k-1)
        reverse(x, k, x.length()-1)
        reverse(x, 0, x.length()-1)
end
```

計算量は　3n

問題２

(a) 出現確率が値によらず一定になっている乱数

(b) 乱数のように見えるが、実際には確定的な計算によって求めている数列に含まれる数

(c) 厳密な解が存在するが正確に求めるのに時間がかかるので近似的な解を求めたいような問題や、もともと確率的な変動を含み実際に実験を行うことが困難であったり多額の費用がかかったりするような問題

(d) 均一に分布しない。
r～r+dr に含まれる点の数が確率的に r によらないので、r に反比例して密度が小さくなってしまう。
r = rand()　を　r = rand()**(1/2)　（または　r = sqrt(rand())）　とすればよい。

問題３

(a)　([2,3],7)　([2,3],5) ([2,3],3) ([2,3],1)
([2,3],0)
([2,3],2) ([2,3],0)
([2,3],4) ([2,3],2) ([2,3],0)
([2,3],1)

より 11 回

(b) 呼び出したときの結果を保存しないため、たとえば rec([2,3],5)も rec([2,3],4)も rec([2,3],2)を呼び、そのたびに重複して計算をするから。

(c) ア：0　　イ：k　　ウ：n

(d) b[m]は rec(a,m)に相当するが、これは配列として保存されているので、再度計算する必要がないから。

問題４

(a) 得点の配列を入れ替えるのと同時に、同じインデックスの氏名と学生証番号も入れ替えるようにする。

(b) 氏名、学生証番号、得点

(c) 氏名を返すメソッド、学生証番号を返すメソッド、得点を返すメソッド、コンストラクタとして氏名と学生証番号を設定するメソッド、得点を設定するメソッド

(d) 配列の値の代わりに、配列のオブジェクトがもつメソッドの戻り値をもとに、オブジェクトごとの整列を行えばよい。

(e) 配列がずれることによってすべてのデータがずれてしまうことがない点や、学生単位の処理をする際に、オブジェクト単位で考えればよく、学生の追加、削除、コピーなどが行いやすい点。

情報科学シケプリ　09-S1-13 三浦拳

過去問解答例　06 冬

問題１

1.

```
def advance(t)
        @second = @second + t
        @minute = @minute + ( @second - @second%60) / 60
        @second = @second%60
        @hour = @hour + ( @minute - @minute%60) / 60
        @minute = @minute%60
end
```

2.

```
def back(t)
        @second = @second - t
        while @second < 0
                @second = @second + 60
                @minute = @minute – 1
        end
        while @minute < 0
                @minute = @minute + 60
                @hour = @hour – 1
        end
end
```

問題２

1.

```
def linear(a, v)
        i = 0
        while a[i] != v
                i = i + 1
        end
        i
end
```

2.

```
def binary(a, v)
        l = 0
        h = a.length() - 1
        while a[(l + h) / 2]! = v
                if a[c] > v
                        h = (l + h) / 2
                else
                        l = (l + h) / 2
                end
        end
end
```

3. アルゴリズム１　：　a.length()
   アルゴリズム２　：　log(2, a.length())

問題３

1.
   (ア) 10 進数を 2 進数に変換するときに割り切れなくなるために生じる誤差
   例：double 型で扱うと　0.1 × 3　が　0.3 にならない
   (イ) オーダーが全く違う数同士を足し合わせると、小さな数が大きな数の有効数字外になってしまうことにより生じる誤差
   例：1 × 10^100 + 1 を計算すると 1 × 10^100 になる
   (ウ) 無限級数などで与えられる値を、計算しても無限には計算できないので、途中で打ち切ってしまった際に生じる誤差
   (エ) 近い値の差をとった際に有効数字が減少してしまうことによる誤差
   例：sqrt(1.001) – sqrt(0.999)を計算するときに、有効数字を 7 桁としても
   sqrt(1.001) – sqrt(0.999) =　1.000500 - 0.9994999 = 0.0010001
   と 5 桁になってしまう
2. やってないから出ないと思う

問題４

問題が意味不明でしたが、よく考えたらこんな感じということになりそうです
〇または◎に行くたびに 3 進数で書かれた数字（120012 とか）の各桁の数字が与えられて、全部の桁が与えられたときに、奇数なら◎のところに偶数なら〇のところにいるようにしなさい。ってことですね。
ところで、3 進数は t(1) + t(2) * 3 + t(3) * 9 + t(4) * 27・・・　＝　Σt(n)*3^(n-1)と書けます
これの偶奇は　Σt(n)と一致するので

a, b　：　0, 2　（順不同）
c, f　：　1
e, d　：　0, 2　（順不同）

ですね。なんか不親切な問題。

情報科学シケプリ　09-S1-13　三浦拳