平成２１年度

名古屋大学大学院情報科学研究科
情報システム学専攻
入学試験問題

専　　門

平成２０年８月１１日（月）
１２：３０～１５：３０

注意事項

１．試験開始の合図があるまでは、この問題冊子を開いてはならない。

２．試験終了まで退出できない。

３．外国人留学生は、日本語から母国語への辞書１冊に限り使用してよい。
電子辞書の持ち込みは認めない。

４．問題冊子、解答用紙３枚、草稿用紙３枚が配布されていることを確認せよ。

５．問題は(１)解析・線形代数、(２)確率・統計、(３)プログラミング、
(４)計算機理論、(５)ハードウェア、(６)ソフトウェアの６科目がある。
このうち３科目を選択して解答せよ。なお、選択した科目名を解答用紙の
指定欄に記入せよ。

６．ITスペシャリストコースへの配属を希望する者は、必ず、「プログラミング」
を選択して解答すること。

７．解答用紙は指定欄に受験番号を必ず記入せよ。解答用紙に受験者の氏名を
記入してはならない。

８．解答用紙は試験終了後に３枚とも提出せよ。

９．問題冊子、草稿用紙は試験終了後に持ち帰ってよい。

解析・線形代数

(解の導出過程も書くこと)

[1] 次の微分方程式の一般解を求めたい.

$$2x^2\frac{d^2y}{dx^2} - 3x\frac{dy}{dx} - 3y = 0$$

(1) $x = e^t$ とおき, 上の方程式を y の t に関する微分方程式に書き換えよ.
(2) (1) で得られた微分方程式を解け.
(3) 上の方程式の一般解を求めよ.

[2] 次の曲線で囲まれる領域の面積を求めたい.

$$x^{2/3} + y^{2/3} = 1$$

(1) この曲線で囲まれた領域を D, この曲線を C (D を左に見て進む方向を正とする) として, 求めたい面積を C 上の積分として表せ.
(2) $x = \cos^3\theta$, $y = \sin^3\theta$ として, (1) で得られた積分式を θ に関する積分式に変換せよ.
(3) 上の曲線で囲まれる領域の面積を求めよ.

[3] 次の行列 A について, 以下の問いに答えよ.

$$A = \begin{pmatrix} 0 & 2 & 2 \\ 2 & 1 & 0 \\ 2 & 0 & -1 \end{pmatrix}$$

(1) 行列 A の固有値 $\lambda_1, \lambda_2, \lambda_3$ (ただし, $\lambda_1 > \lambda_2 > \lambda_3$) と対応する単位固有ベクトルを求めよ.
(2) (1) で得られた単位固有ベクトルを, それぞれ列ベクトルとして並べた行列 L, ならびに, 固有値 $\lambda_1, \lambda_2, \lambda_3$ を用いて, 行列 A^n を表せ. ただし, $n = 1, 2, \cdots$ の整数である.
(数値の代入は不要である)

【専門用語の英訳】

微分方程式	differential equation	一般解	general solution
面積	area	積分	integral
行列	matrix	固有値	eigenvalue
単位固有ベクトル	unit eigenvector	列ベクトル	column vector

確率・統計

［1］ 正6面体のサイコロを6480回投げたときに，1の目が1056回以下現れる確率を近似的に計算し，数値で答えよ．考え方の手順も書くこと．表1を用いても良い．

［2］ 次の問いに答えよ．解の導出過程も書くこと．

(1) 確率変数 X_1, X_2 が独立で，区間[0，1]における一様分布に従うとき，
確率変数 $T = \max\{X_1, X_2\}$ の確率密度関数 $f_T(t)$ を求めよ．

(2) 区間[0，1]から無作為に2点 x_1, x_2 を独立に選ぶ．$w = |x_1 - x_2|$ の平均と分散を求めよ．

［3］ 確率変数 X が平均 μ，分散 $\sigma^2 (\neq 0)$ をもつとき，任意の正の定数 k に対し次の不等式が成り立つ．$P(|X-\mu| \geq k\sigma) \leq 1/k^2$．この式を，$X$ の確率密度関数を $f(x)$ とし，次の手順で式変形することで証明せよ．

$$\sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)^2 f(x)dx = \cdots \geq \cdots \geq \cdots = k^2\sigma^2 P(|X-\mu| \geq k\sigma)$$

$\cdots \geq \cdots \geq \cdots$ の部分に適切な式変形を書け．

【専門用語の英訳】

確率 probability, 確率変数 random variable, 独立 independent, 密度関数 density function,
無作為 random, 平均 mean, 分散 variance

表1

z	0.2	0.4	0.6	0.8	1.0	1.2	1.4	1.6
$\phi(z) = \int_z^{\infty} \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx$	0.421	0.345	0.274	0.212	0.159	0.115	0.081	0.055

プログラミング

以下に示すC言語プログラム中の関数quick_sortは，クイックソートアルゴリズムにより，ポインタの配列aのleft番目からright番目までの要素を，比較関数compareが定める順序によりソートする関数である．(行頭の数字は行番号を表し，プログラムには含まれない.)

ここで，配列の添字（index）は0から始まるものとし，配列の最初の要素を0番目の要素と呼ぶ.

quick_sort.h

```
1     typedef int BOOL;
2
3     #define TRUE  1
4     #define FALSE 0
5
6     extern void quick_sort(void *a[], int left, int right,
7                                 BOOL compare(void *, void *));
```

quick_sort.c

```
8     #include "quick_sort.h"
9
10    void quick_sort(void *a[], int left, int right,
11                          BOOL compare(void *, void *))
12    {
13        void *p, *t;
14        int  i, j;
15
16        p = a[(left + right) / 2];
17        i = left;
18        j = right;
19        while (i <= j) {
20            while (compare(a[i], p)) i = i + 1;
21            while (compare(p, a[j])) j = j - 1;
22            if (i >= j) break;
23            t = a[i];
24            a[i] = a[j];
25            a[j] = t;
26            i = i + 1;
27            j = j - 1;
28        }
29        if (left < i - 1) quick_sort(a, left, i - 1, compare);
30        if (j + 1 < right) quick_sort(a, j + 1, right, compare);
31    }
```

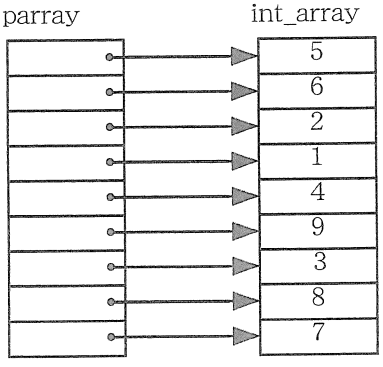
関数 quick_sort を用いて，9 つの整数値をソートして表示するプログラムを，次のように作成した．

main.c

```
32    #include <stdio.h>
33    #include "quick_sort.h"
34
35    BOOL int_compare(int *a, int *b)
36    {
37        if (*a < *b) return(TRUE);
38        else return(FALSE);
39    }
40
41    int int_array[9] = { 5, 6, 2, 1, 4, 9, 3, 8, 7 };
42    int *parray[9];
43
44    main()
45    {
46        int i;
47
48        for (i = 0; i < 9; i++) {
49            parray[i] = &int_array[i];
50        }
51        quick_sort(parray, 0, 8, int_compare);
52        for (i = 0; i < 9; i++) {
53            printf("%d\n", *(parray[i]));
54        }
55    }
```

（1）このプログラムは，与えられた整数値をどのような順序にソートするか．また，ソートする順序を逆にするには，**main.c** をどのように修正すればよいか．

（2）このプログラム（前問で修正する前のプログラム）が最初に関数 quick_sort を呼ぶ時の配列 parray および配列 int_array を図示したものを下に示す．これを参考に，プログラムが最初に 29 行目を実行する直前および最初に 30 行目を実行する直前の配列 parray および配列 int_array を下に示す形式で図示せよ．



（3）学生の学籍番号（id フィールド），氏名（name フィールド），成績（score フィールド）を管理するための構造体型 student を下のように定義した．関数 quick_sort を用いて，構造体型 student の値を，成績の高い順（成績が同じ学生の間では，学籍番号の小さい順）にソートするために用いる比較関数 student_compare を完成させよ．

```
56    typedef struct {
57        int     id;
58        char    *name;
59        int     score;
60    } student;
61
62    BOOL student_compare(student *a, student *b)
63    {

          . . . .

...   }
```

（4）関数 quick_sort の問題点として，型の整合性（consistency）がチェックできないことが挙げられる．この問題により不具合が生じるのは，関数 quick_sort をどのように呼び出した場合か述べよ．

また，関数 quick_sort を型の整合性がとれていない状況で呼び出した結果，関数 quick_sort 中で不正なメモリアドレスに対するアクセスが発生し，プログラムの実行が中断される可能性がある．不正なメモリアドレスに対するアクセスが発生する理由を具体的に説明せよ．

計算機理論

記号 (symbol) の有限集合 (finite set) をアルファベット (alphabet) という．アルファベット Σ の記号からなる有限の長さの系列を Σ 上の語 (word) という．すべての語の集合を Σ^*，空語（empty word）を ε で表す．語 w, w' の連接 (concatenation) を $w \cdot w'$ と書く．$\cdot$ を略して $w \cdot w'$ を ww' と書くことがある．記号 a が n 個連なった語を a^n と書く．Σ^* の部分集合 (subset) を Σ 上の言語 (language) という．言語 L, L' の連接を $L \cdot L'$ で表す．

Σ_1，Σ_2 をアルファベットとする．写像 (mapping) $h : \Sigma_1 \to \Sigma_2^*$ から次のように定められる写像 $\overline{h} : \Sigma_1^* \to \Sigma_2^*$ を**準同型写像** (homomorphism) という．

$$\overline{h}(w) = \begin{cases} \varepsilon & w = \varepsilon \text{ のとき} \\ h(a) \cdot \overline{h}(w') & w = a \cdot w',\ a \in \Sigma_1,\ w' \in \Sigma_1^* \text{ のとき} \end{cases}$$

以下では，h と $\overline{h}$ を同一視して，両方を h で表すことにする．Σ_1 上の言語 L に対して $h(L) = \{h(w) \mid w \in L\}$ とする．

$\Sigma_{exp} = \{\underline{\emptyset}, \underline{\varepsilon}, *, \bullet, +, [,]\}$ とする．Σ は $\Sigma \cap \Sigma_{exp} = \emptyset$ を満たすアルファベットとする．Σ 上の**正規表現** (regular expression) $E \in (\Sigma \cup \Sigma_{exp})^*$ とそれが表す言語 $L(E) \subseteq \Sigma^*$ は次のように帰納的に (inductively) 定義される．

(i) $\underline{\emptyset}$ は正規表現である．$L(\underline{\emptyset}) = \emptyset$
(ii) $\underline{\varepsilon}$ は正規表現である．$L(\underline{\varepsilon}) = \{\varepsilon\}$
(iii) $a \in \Sigma$ は正規表現である．$L(a) = \{a\}$
(iv) E と F が正規表現ならば $[E{+}F]$ は正規表現である．$L([E{+}F]) = L(E) \cup L(F)$
(v) E と F が正規表現ならば $[E{\bullet}F]$ は正規表現である．$L([E{\bullet}F]) = L(E) \cdot L(F)$
(vi) E が正規表現ならば $[E*]$ は正規表現である．$L([E*]) = L(E)^*$

ここで，L^* は言語 L のクリーネ閉包 (Kleene closure)，すなわち，$L^* = \{w_1 \cdot w_2 \cdot \ldots \cdot w_n \mid n \geq 0,\ w_i \in L\ (i = 1, 2, \ldots, n)\}$ とする．Σ 上のすべての正規表現の集合を $\mathcal{E}(\Sigma)$ で表す．言語が**正規言語** (regular language) であることとその言語が正規表現で表されることとは互いに必要十分条件 (necessary and sufficient condition) であることが知られている．

[1] 準同型写像 $h : \{a, b, c\}^* \to \{0, 1\}^*$ は次の条件を満たすとする．

$$h(a) = 01, \qquad h(b) = \varepsilon, \qquad h(c) = 10$$

以下の問いに答えよ．

(1) 言語 $L_0 = \{\varepsilon, abc, bca, cab\}$ に対して，$h(L_0)$ を求めよ．

(2) 正規表現 $E = [[[a*] \bullet [b*]] \bullet [c*]]$ が表す言語 $L(E)$ に属し，かつ，長さ (length) が 3 の語をすべて列挙せよ．

(3) (2) で与えた正規表現 E について，$h(L(E))$ を表す正規表現を書け．

[2] アルファベット $\{a, b, c, d\}$ 上の言語 L_1 を次のように定める．

$$L_1 = \{a^k b^{n-m} c^m d^n a^k \mid k \geq n \geq m \geq 0\}$$

アルファベット $\{0, 1\}$ 上の言語 $L_2 = \{0^n 1^n \mid n \geq 0\}$ は正規言語ではないことが知られている．この事実と次の定理 (theorem) を用いて，L_1 が正規言語ではないことを証明せよ．

> **【定理】** $h : \Sigma_1^* \to \Sigma_2^*$ を任意の準同型写像とするとき，$L \subseteq \Sigma_1^*$ が正規言語ならば $h(L)$ も正規言語である．

ヒント： $h(L_1) = L_2$ となる準同型写像 $h : \{a, b, c, d\}^* \to \{0, 1\}^*$ を発見する．

[3] 準同型写像 $h : \Sigma_1^* \to \Sigma_2^*$ に対して，写像 $\tilde{h} : \mathcal{E}(\Sigma_1) \to \mathcal{E}(\Sigma_2)$ を次のように帰納的に定める．

$$\tilde{h}(E) = \begin{cases} \underline{\emptyset} & E = \underline{\emptyset} \text{ のとき} \\ \underline{\varepsilon} & E = \underline{\varepsilon} \text{ のとき} \\ g(h(a)) & E = a \in \Sigma_1 \text{ のとき} \\ [\tilde{h}(F) \bullet \tilde{h}(G)] & E = [F \bullet G] \text{ のとき} \\ [\tilde{h}(F) + \tilde{h}(G)] & E = [F + G] \text{ のとき} \\ [\tilde{h}(F) *] & E = [F *] \text{ のとき} \end{cases}$$

ただし，$g : \Sigma_2^* \to \mathcal{E}(\Sigma_2)$ は次のように定められる写像である．

$$g(w) = \begin{cases} \underline{\varepsilon} & w = \varepsilon \text{ のとき} \\ [b \bullet g(w')] & w = b \cdot w', b \in \Sigma_2, w' \in \Sigma_2^* \text{ のとき} \end{cases}$$

次の補題 (lemma) について考える．

> **【補題】** $h : \Sigma_1^* \to \Sigma_2^*$ を任意の準同型写像とする．Σ_1 上の任意の正規表現 E に対して，次が成り立つ．
>
> $$h(L(E)) = L(\tilde{h}(E))$$

以下の問いに答えよ．

(1) この補題が成り立てば，[2] で与えた定理が成り立つことを証明せよ．

(2) この補題は，正規表現の構造に関する帰納法 (structural induction) により証明できる．そのとき，基底段階 (base step) で証明すべきことを書け．

(3) この補題の証明の帰納段階 (induction step) で証明すべきことを書け．ただし，帰納法の仮定 (induction hypothesis) を明示せよ．ヒント：場合分けを行う．

(4) (3) の帰納段階で証明すべき場合の 1 つを選んで証明せよ．

ハードウェア

[1] データキャッシュ (data cache) の書き込みの際のデータ更新方式 (write policy) について，主要な 2 つの方式を挙げてそれぞれを説明し，両方式を比較し得失を述べよ．

[2] 表 1 の真理値表 (truth table) (‘$*$’ はドントケア (don't care)) で与えられる不完全定義論理関数 (incompletely specified logic function) $f(x, y, z, w)$ について以下の問いに答えよ．

(1) $f(x, y, z, w)$ を表すカルノー図 (Karnaugh map) を示せ．

(2) $f(x, y, z, w)$ のドントケアな組に対する出力をすべて 1 とした論理関数の主項 (prime implicant) をすべて求めよ．

(3) $f(x, y, z, w)$ の最小積和形表現 (minimum sum-of-products form) を求めよ．

[3] 表 2 の状態遷移表 (state transition table) で示される順序機械 (sequential machine) M と等価な状態数最小の順序機械を求め，その状態遷移図 (state transition graph) を示せ．

表 1: 不完全定義論理関数 $f(x, y, z, w)$ の真理値表

x	y	z	w	$f(x, y, z, w)$
0	0	0	0	1
0	0	0	1	$*$
0	0	1	0	0
0	0	1	1	$*$
0	1	0	0	1
0	1	0	1	$*$
0	1	1	0	0
0	1	1	1	$*$
1	0	0	0	0
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

表 2: 順序機械 M の状態遷移表

現状態	次状態/出力	
	入力	
	0	1
S_1	$S_7/0$	$S_6/1$
S_2	$S_3/0$	$S_5/0$
S_3	$S_4/0$	$S_2/0$
S_4	$S_1/0$	$S_5/1$
S_5	$S_1/0$	$S_4/1$
S_6	$S_3/0$	$S_4/0$
S_7	$S_4/0$	$S_6/0$

ソフトウェア

n番目のフィボナッチ数(Fibonacci number)をF_nとする．F_nは以下のような漸化式(recursion formula)で定義される．

$$F_n = \begin{cases} 0 & n = 0 \text{ のとき} \\ 1 & n = 1 \text{ のとき} \\ F_{n-2} + F_{n-1} & n \geq 2 \text{ のとき} \end{cases}$$

[1] F_nを`fib1(n)`として計算する以下のC言語のプログラムP1について問いに答えよ．(行頭の数字は行番号を表し，プログラムには含まれない．)

プログラムP1

```
1   #define N 3
2
3   int fib1(int n) {
4     int r;
5     if(n==0) r=0;
6     else if(n==1) r=1;
7     else r=fib1(n-2)+fib1(n-1);
8     return r;
9   }
10
11  main() {
12    printf("F%d=%d\n",N,fib1(N));
13  }
```

(1) F_nの計算における関数`fib1`の呼び出し回数を$c(n)$とする．

(ア) $c(n)$を漸化式を用いて表せ．

(イ) $c(n)$はnの増加に対してどのように増加するか，以下のA~Dから選択し，その理由を述べよ．

A. nに対して線形(linear)に増加する．

B. nに対して$n \log_2 n$の割合で増加する．

C. nに対してn^2の割合で増加する．

D. nに対して指数的(exponential)に増加する．

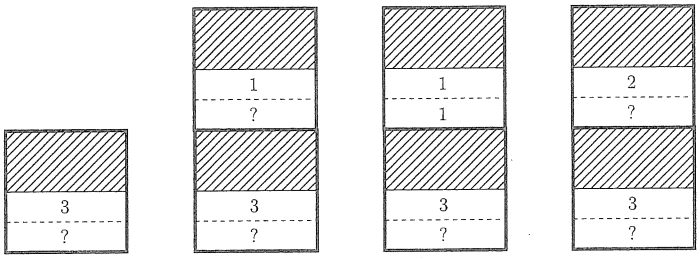
(2) 手続きや関数などをコンパイルして実行する際，駆動レコード(activation record)(またはスタックフレーム(stack frame))とよばれる領域がスタック上に生成される．

以下に駆動レコードの構造を示す．

局所データ領域 (local data area)
リンク情報 (dynamic link, static link, etc.)
戻り番地 (return address)
実引数 (actual parameters)
戻り値 (return value)

以下、この部分を斜線で表す．

スタック上の駆動レコードの生成と消去の様子を実引数領域と戻り値に注目して観測する．(他の領域については以下では斜線で表す．) 下の図は，`fib1(3)` の計算におけるスタックの様子の一部を示す．(k-c) は `fib1` が k 回目に呼出された直後のスタックにおける駆動レコードの様子を示す．また，(k-r) は，k 回目に呼出された `fib1` の実行が `return` の実行によって終了する直前の駆動レコードの様子を表わす．決定していない戻り値は '?' で表す．(3-c) 以降の `fib1(3)` の計算が終了するまでの (k-c) および (k-r) における駆動レコードの様子を順を追って記述せよ．



(1-c) (2-c) (2-r) (3-c)

[2] つぎに F_n を `fib2(n)` として計算する以下の C 言語のプログラム P2 について問いに答えよ．(行頭の数字は行番号を表し，プログラムには含まれない．)

プログラム P2

```
1   #define N 3
2
3   int a[N+1];
4
5   void fib2(int x) {
6     if(x==0) a[x]=0;
7     else if(x==1) a[x]=1;
8     else a[x]=a[x-2]+a[x-1];
9   }
10
11  main() {
12    int i;
13
14    for(i=0;i<=N;i++) fib2(i);
15    printf("F%d=%d\n",N,a[N]);
16  }
```

(1) P2 のコンパイル時に作成される記号表 (symbol table) について以下の問いに答えよ．

(ア) 記号表に記録することが必要な属性 (attribute) を 4 つ以上あげよ．

(イ) (ア) であげた属性に対して，`fib2` のコンパイル中に生成される変数 `a` および変数 `x` に対する記号表を作成せよ．必要に応じて属性値に対する説明を加えること．

(2) P1 の `fib1` および P2 の `fib2` の呼出し回数の違いについて説明せよ．さらに，そのような差が生じる理由を述べよ．

[3] n を fib3 の第一引数として F_n を計算する以下の C 言語のプログラム P3 について問いに答えよ．
(行頭の数字は行番号を表し，プログラムには含まれない．)

プログラム P3

```
1   #define N 3
2
3   int fib3(int n,int *p) {
4     int a;
5     if(n==0) { return(0); }
6     else if(n==1) { *p = fib3(0,&a); return(1); }
7     else { *p=fib3(n-1,&a); return(a+(*p)); }
8   }
9
10  main() {
11    int a;
12    printf("F%d=%d\n",N,fib3(N,&a));
13  }
```

(1) F_n の計算において，P2 の fib2 と P3 の fib3 の呼出し回数を比較せよ．

(2) P2 と P3 の記憶領域の使い方の違いについて説明せよ．