専門科目（午前）　　23　大修

情報工学

時間　9：30 ～ 12：30

集積システム専攻・計算工学専攻

注　意　事　項

1. 各解答用紙に受験番号を記入せよ．
2. 次の7題の中から4題を選択して解答せよ．5題以上解答した場合はすべて無効とする．
3. 解答は1題ごとに別々の解答用紙に，問題番号を明記した上で記入せよ．必要であれば，解答用紙の裏面に記入してよいが，解答用紙の表面にその旨を明記すること．
4. 1枚の解答用紙に2題以上の解答を記入した場合はそれらの解答を無効とすることがある．
5. 1題の解答を2枚以上の解答用紙に記入した場合はその解答を無効とすることがある．
6. 電子式卓上計算機等の使用は認めない．

---

1. 実数の数列全体の集合を V とする．和とスカラー倍を次のように定めると，V は実ベクトル空間になることが知られている．

$$\{a_n\}+\{b_n\}=\{a_n+b_n\}$$
$$\alpha\{a_n\}=\{\alpha a_n\}$$

ただし，α はスカラーであるとする．$\{a_n\}$ は，$a_0, a_1, a_2, \ldots$ という数列を表し，$\{a_n+b_n\}$ は，$a_0+b_0, a_1+b_1, a_2+b_2, \ldots$ という数列を表している．

以下の問に答えよ．

1) 漸化式 $a_{n+1}=3a_n-2a_{n-1}$ $(n\geq 1)$ をみたす数列全体 W_1 が V の部分ベクトル空間であることを示せ．
2) W_1 の基底を求めよ．
3) 写像 f を $f(\{a_n\})=\{a'_n\}$ をみたす実数の数列上の写像とする．ここで $\{a'_n\}$ は $a'_n=a_{n+1}$ と定義される数列である．この写像 f が線形写像であることを示せ．
4) 漸化式 $b_{n+1}=b_n+2b_{n-1}$ $(n\geq 1)$ をみたす数列全体を W_2 とする．W_1 の元と W_2 の元の和として表すことができるベクトルの全体は，V の部分ベクトル空間となることが知られている．これを和空間と呼び，W_1+W_2 と書く．W_1+W_2 の基底を求めよ．
5) 次の文が正しくなるような p, q, r の値を求めよ．ただし，p, q, r はいずれも 0 ではないものとする．

「和空間 W_1+W_2 は漸化式

$$c_{n+2}=pc_{n+1}+qc_n+rc_{n-1}\quad(n\geq 1)$$

をみたす数列全体の集合と一致する．」

2. 確率変数 X, Y に関する同時確率密度関数 $p_{X,Y}(x, y)$ が次式で与えられるとする．

$$p_{X,Y}(x, y) = \frac{1}{2\pi\sigma^2\sqrt{1-r^2}} \exp\left[-\frac{(x+m)^2 - 2r(x+m)y + y^2}{2\sigma^2(1-r^2)} \right] \tag{2.1}$$

ただし，$0 \leq r < 1$, $0 < \sigma$, $-\infty < X < \infty$, $-\infty < Y < \infty$ とする．また，m は実数である．以下の問では，式番号を参照して次式を用いてもよい．

$$\int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2} \right] dx = 1 \tag{2.2}$$

1) Y の周辺確率密度関数を $p_Y(y)$, $Y = y$ が与えられたときの X に関する条件付き確率密度関数を $p_{X|Y}(x|y)$ とする．

 a) $p_Y(y)$ を求めよ．

 b) $z = x + m - ry$ として，$p_{X|Y}(x|y)$ を z, σ および r で表せ．

2) 確率変数 X, Y と確率変数 U, V との変換が以下の 2 つの変換関数で表されるとする．

$$\begin{cases} X = g_1(U, V) = U - aV - m \\ Y = g_2(U, V) = U + bV \end{cases} \tag{2.3}$$

ただし，$a > 0$, $b > 0$, $-\infty < U < \infty$, $-\infty < V < \infty$ とする．

 a) U, V の同時確率密度関数を $q_{U,V}(u, v)$ とする．$q_{U,V}(u, v)$ は $p_{X,Y}(g_1(u, v), g_2(u, v))$ とヤコビアンの積となる．ヤコビアンを求めよ．ただし，計算過程を示すこと．

 b) 確率密度関数 $q_U(u), q_V(v)$ により $q_{U,V}(u, v) = q_U(u)q_V(v)$ となるとき，式 (2.3) の変換が満たすべき条件を求めよ．また，$q_U(u), q_V(v)$ を求めよ．

3. 以下の問に答えよ.

1) 表 3.1 と表 3.2 の表記による正規表現を考える．ただし，閉包，連接，和では，閉包の優先順位が最も高く，和の優先順位が最も低いとして，意味が変化しない括弧を省略してもよい．

表 3.1 正規表現

表記	意味
$r_1 + r_2$	正規表現 r_1 と r_2 の和
r_1 *	正規表現 r_1 の閉包
r_1 r_2	正規表現 r_1 と r_2 の連接
(r_1)	正規表現 r_1 自身（グループ化）

表 3.2 補助的な正規表現

表記	意味
•	入力アルファベット中の任意の一文字
[^*]	文字*を除く入力アルファベット中の任意の一文字
[^/]	文字/を除く入力アルファベット中の任意の一文字
*	* 自身（閉包を表す*と区別するために*と表記）

例えば次の文字列の集合を考える.「先頭が『/以外の文字』で始まり，次に『任意の一文字』が続き，その後に 0 文字以上の『*と/の任意個の組み合わせ』が続く文字列」の集合は，[^/]•(*+/)* という正規表現で表せる．また，図 3.1 は，この正規表現と等価で ε 遷移を含まない決定性有限状態オートマトン (DFA) である．

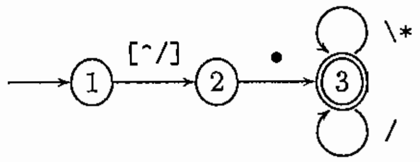


図 3.1 正規表現 [^/]•(*+/)*と等価な DFA

a) C 言語のコメントを正規表現で表せ．ここで，C 言語のコメントとは，/*で始まり，0 個以上の任意の文字が続き，*/で終わる．ただし，最初の/*と最後の*/の間に*/が出現してはならない．また，表 3.1 と表 3.2 の正規表現だけを使うこと．

b) 図 3.2 は，a) の C 言語のコメントの正規表現と等価な，ε 遷移を含む非決定性有限状態オートマトン (ε-NFA) である．

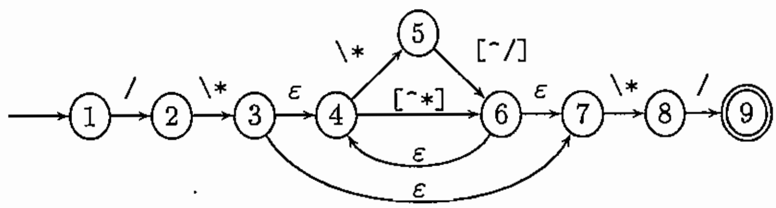


図 3.2 a) の C 言語のコメントの正規表現と等価な ε-NFA

図 3.2 の ε-NFA と等価で状態数が最小の，ε 遷移を含まない決定性有限状態オートマトン (DFA) を作れ．また，その導出過程を説明せよ．

(次ページにつづく)

（前ページのつづき）

2) 次の G_1 は C 言語の式を簡略化した文法，G_2 は C 言語の宣言を簡略化した文法である．

文法 G_1（式）：

E → E + E
E → E * E
E → (E)
E → id

文法 G_2（宣言）：

S → int D ;
D → id
D → * D
D → (D)
D → D [ ]
D → D ( )

ここで，E と S と D は非終端記号，それ以外はすべて終端記号である．id は識別子（変数名）を表す終端記号とする．なお，文法 G 中の生成規則の個数を $|G|$ と表記する．例えば，$|G_1| = 4$, $|G_2| = 6$ となる．また，文法 G が生成する文字列の集合を $L(G)$ と表記する．

a) 文法 G_1 があいまいであることを示せ．

b) 文法 G_2 もあいまいである．G_2 の文法を書き直して，ポインタ (*) の優先順位が，関数 (()) や配列 ([]) よりも低くなるように，つまり優先順位の低いものが先に導出されるように，等価であいまいではない文法 G_3 を作れ．例えば,「int *id(); 」が,「int (*id)();」（int 型を返す関数へのポインタ）ではなく,「int *(id()); 」（int 型へのポインタを返す関数）と同じ意味になるように，つまり関数 (()) よりもポインタ (*) を先に導出するようにせよ．ただし，$|G_3| \leq 7$ とすること．

c) 「文法 G_1 が生成する式のうち，余分な括弧を含まない式」をすべて生成する文法 G_4 を作れ．ただし，G_4 はあいまいでもよい．また，$|G_4| \leq 6$ とすること．$L(G_4)$ は $L(G_1)$ の真部分集合となることに注意せよ．

ここで余分な括弧とは，+よりも*の方が優先順位が高いとした場合に，括弧を省略しても式の意味が変わらないものを指す．例えば,「id*(id+id)」の括弧は余分ではない．一方,「id+(id*id)」,「id+(id+id)」,「id*(id*id)」,「(id)」，及び「(id+id)」の括弧は余分である．また,「id*((id+id))」の外側（または内側）の括弧も余分である（一組の括弧に減らせる）．

4. 図 4.1 の変成器の電圧と電流は

$$v_1(t) = L_1\frac{di_1(t)}{dt} + M\frac{di_2(t)}{dt} \tag{4.1}$$

$$v_2(t) = M\frac{di_1(t)}{dt} + L_2\frac{di_2(t)}{dt} \tag{4.2}$$

の関係をもつ．ただし，L_1, L_2 は自己インダクタンス，M は相互インダクタンスである．

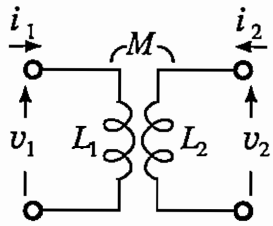


図 4.1 変成器

1) 変成器に入る瞬時電力は

$$p(t) = v_1(t)i_1(t) + v_2(t)i_2(t)$$

である．式 (4.1) と (4.2) を用いて，$p(t)$ から $v_1(t), v_2(t)$ を消去し，その結果を示せ．

2) 変成器に蓄えられるエネルギー

$$w(t) = \int_{-\infty}^{t} p(\tau)d\tau$$

を $i_1(t)$ と $i_2(t)$ を用いて表せ．ただし $i_1(-\infty) = i_2(-\infty) = 0$ とする．

3) どのような電流が流れても $w(t)$ が負になることはないという条件から，変成器の素子値 L_1, L_2, M に関する制約を求めよ．

4) $\frac{M^2}{L_1L_2} = 1$ のとき (これを密結合と言う)，$v_1(t)$ と $v_2(t)$ は独立ではあり得ない．$v_1(t)$ と $v_2(t)$ の関係を $n = \frac{M}{L_1} = \frac{L_2}{M}$ を用いて表せ．

5) 密結合の変成器に蓄えられるエネルギーを表す式を求め，蓄えられるエネルギーが 0 となるための $i_1(t)$ と $i_2(t)$ の関係を示せ．また，そのときの $v_1(t), v_2(t)$ を求めよ．

6) 図 4.2, 図 4.3 の回路の電圧と電流の関係がそれぞれ図 4.1 と同じになるように，インピーダンス Z_1, Z_2, Z_3 とアドミタンス Y_1, Y_2, Y_3 とをそれぞれ $j\omega L_1, j\omega L_2, j\omega M$ を用いて表せ．ただし密結合でないとする．

7) たとえ電圧と電流の関係が同じであっても，図 4.2 や 4.3 の回路は図 4.1 の変成器の代替となれないことがある．図 4.2 や図 4.3 ではなく図 4.1 を用いる場合の利点を簡単に説明せよ．

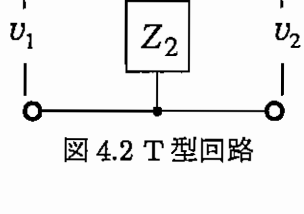


図 4.2 T 型回路

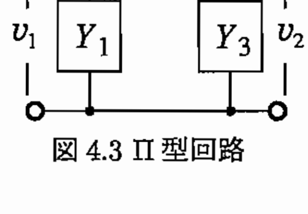


図 4.3 Π 型回路

5. 送信信号が $s \in \{0,1\}$ である 2 値通信システムがある．受信信号 r は，送信信号 s に統計的に独立な誤り e が加わって得られる．すなわち，

$$r = s \oplus e$$

である．ただし，$\oplus$ は排他的論理和をあらわし，また誤り e は図 5.1 のマルコフモデルであらわされる確率過程に従って与えられるものとする．図中において S0 → S0 および S1 → S0 と状態が遷移する場合に e は 0 となり，S0 → S1 および S1 → S1 の場合に e は 1 となる．また $p, q, 1-p, 1-q$ は遷移確率を示し，$0<p<1, 0<q<1$ とする．

以下の問に答えよ．

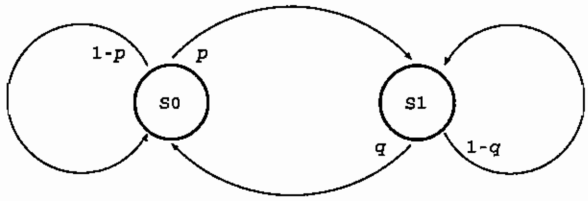


図 5.1 誤り e のマルコフモデル

1) S0 および S1 それぞれの定常確率を求めよ．

2) S0 および S1 の状態にあることを条件とした 誤り e の条件付きエントロピー $H(e|\mathrm{S0})$ および $H(e|\mathrm{S1})$ を求めよ．

3) 誤り e のエントロピーを求めよ．

4) $p = 1/4, q = 3/4$ とし，送信信号 s の発生確率を $P(s=0) = 1/4, P(s=1) = 3/4$ とする．この場合の送信信号 s と受信信号 r の相互情報量を示せ．ただし，$\log_2 3 = 1.6$ および $\log_2 5 = 2.3$ と近似せよ．

5) $p = q$ の場合，この 2 値通信システムの通信路容量が 0 となることを証明せよ．

6. 以下の仕様を満たす順序回路の設計を考える．

① 入力集合と出力集合は，ともに$\{1, 2, 3\}$の 3 つのシンボルのみからなる．

② この順序回路は，シンボル順列を状態変数として持ち，各入力シンボルについて，そのシンボルを順列の先頭（左端）に移動して新しいシンボル順列を作る．ただし，シンボル順列の初期状態は(1 2 3)とする．

③ この順序回路は，入力シンボルについて並び替えたシンボル順列の末尾（右端）を出力する．

表 6.1 は，入力系列(2 3 1 3 2 1)に対する順序回路の動作の様子を示す．初期状態(1 2 3)のシンボル順列において，最初の入力シンボル 2 について，これを順列の先頭に移動させて(2 1 3)に並び替え，その末尾シンボル 3 を出力する．さらに，次の入力シンボル 3 について，これを順列の先頭に移動させて(3 2 1)に並び替え，その末尾シンボル 1 を出力する．

図 6.1 は，この順序回路の回路構成を示す．ここでは，3 つのシンボル$\{1, 2, 3\}$が 2 ビット 2 進数$\{(01), (10), (11)\}$で表現されており，(x_1x_0)は 2 ビット入力変数，(z_1z_0)は 2 ビット出力変数である．(y_1y_0)，(y_3y_2)，(y_5y_4)は 2 ビットレジスタ R_1, R_2, R_3 がそれぞれ出力する 2 ビット状態変数であり，シンボル順列をそれぞれ保持している．また R_1, R_2, R_3 は，それぞれ(01), (10), (11)に初期化されているものとする．s_0, s_1 は入力変数と状態変数を入力とする組合せ回路が出力する 2 つの 2 ビットマルチプレクサへの制御変数である．図 6.2 に 2 ビットマルチプレクサの動作を示す．

なお，以下の問において論理式を解答する場合，最小数の AND 項からなる NOT-AND-OR 形式（「積項の和」形式）で論理式を表すこと．この順序回路の回路構成では，(x_1x_0)，(z_1z_0)，(y_1y_0)，(y_3y_2)，(y_5y_4)はいずれも(00)の値を取らないことに注意せよ．

表 6.1：動作例

入力	シンボル順列	出力
初期状態：(1 2 3)		
2	(2 1 3)	3
3	(3 2 1)	1
1	(1 3 2)	2
3	(3 1 2)	2
2	(2 3 1)	1
1	(1 2 3)	3

入力の順番

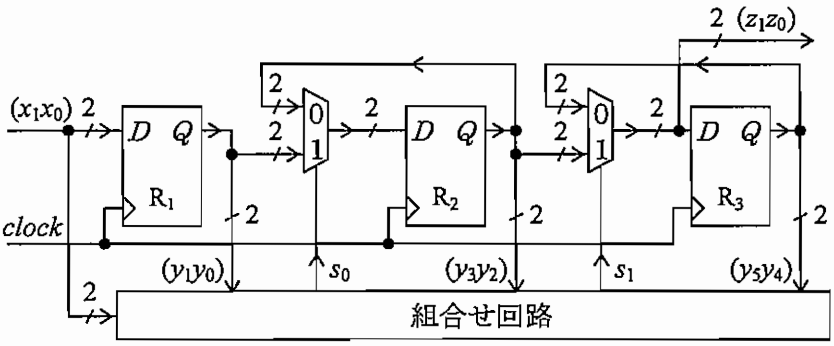
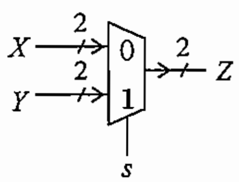


図 6.1 順序回路構成



$s=0$ のとき：$Z=X$

$s=1$ のとき：$Z=Y$

図 6.2　2 ビットマルチプレクサ

（次ページにつづく）

（前ページのつづき）

1) 初期状態(1 2 3)のシンボル順列において，入力系列 (3 3 1 2 3 1 3 1) に対するシンボル順列と出力を，表 6.1 と同じ形式の表で示せ．

2) 2 ビット入力変数(x_1x_0)は，(y_1y_0)，(y_3y_2)，(y_5y_4)のいずれかと値が一致し，その結果によっては，各レジスタの間でデータ転送が発生し，新しいシンボル順列に更新される．

   a) $(x_1x_0)=(y_1y_0)$ のとき 1 を出力し，$(x_1x_0)\neq(y_1y_0)$ のとき 0 を出力する論理関数 $F(x_1,x_0,y_1,y_0)$ の論理式と，その否定の関数 $\overline{F(x_1,x_0,y_1,y_0)}$ の論理式をそれぞれ示せ．

   b) $(x_1x_0)=(y_1y_0)$，$(x_1x_0)=(y_3y_2)$，$(x_1x_0)=(y_5y_4)$ の 3 つの場合について，レジスタ間転送が正しく行われるための s_0, s_1 それぞれの値を示せ．

   c) s_0, s_1 それぞれを，$x_0, x_1, y_0, y_1, y_2, y_3, y_4, y_5$ に関する論理式で表せ．

3) z_0, z_1 それぞれを，$x_0, x_1, y_2, y_3, y_4, y_5$ に関する論理式で表せ．

4) この順序回路の状態遷移図を示せ．ただし，状態遷移を示す枝のラベルは，{1, 2, 3}のシンボルで表記せよ．例えば，入力 1 に対し出力 3 である状態遷移枝のラベルは「1/3」となる．また，初期状態がどの状態か明示せよ．

7. 図7.1のCプログラムを読んで問に答えよ.

```
 1: #include <stdio.h>
 2: int a[1024];
 3:
 4: void printa(int n) {
 5:     int i;
 6:     if (n > 0) printf("%d", a[0]);
 7:     for (i = 1; i < n; i++)
 8:         printf(", %d", a[i]);
 9:     printf("\n");
10: }
11:
12: int check(int x, int y) {
13:     int k;
14:     for (k = 0; k < x; k++) {
15:         if (a[k] == y)
16:             return 0;
17:     }
18:     return 1;
19: }
20:
```

```
21: void f(int n, int x) {
22:     if (x >= n) {
23:         printa(n);
24:     }
25:     else {
26:         int y = 0;
27:         while (y < n) {
28:             if (check(x, y)) {
29:                 a[x] = y;
30:                 f(n, x + 1);
31:             }
32:             y = y + 1;
33:         }
34:     }
35: }
36:
37: int main(void) {
38:     f(3, 0);
39:     return 0;
40: }
```

図7.1

1) 図7.1のプログラムをコンパイルして実行したところ,図7.2のような出力を得た.出力の各行は関数 printa の呼び出しに対応している.この実行では printa は6回呼び出され,それぞれの呼び出しにおいて 0, 1, 2 〜 2, 1, 0 という長さ3の数列を出力している.では,この実行において関数 f は何回呼ばれたかを答えよ.ただし関数 main による最初の呼び出しを含めるものとする.

```
0, 1, 2
0, 2, 1
1, 0, 2
1, 2, 0
2, 0, 1
2, 1, 0
```

図7.2

2) 図7.1の38行目にある関数 f の呼び出しを f(4, 0) に変更して実行したところ,出力の中に図7.3のような行があった.

```
1, 0, 3, 2
```

図7.3

a) これは出力の何行目に当たるかを答えよ.

b) この出力が行われた時点で関数 f は何回呼ばれているかを答えよ.ただし関数 main による最初の呼び出しを含めるものとする.終了していない呼び出しがある場合はその回数も含めること.

3) n を $0 \leq n < 1024$ をみたす整数とする.関数 f を f(n, 0) という形で呼び出したときの関数 f の総呼び出し回数を T_n とする.T_n について以下の漸化式がなりたつよう,空欄 $\boxed{A}$ および $\boxed{B}$ に入る式を答えよ.

$$
\begin{aligned}
T_0 &= 1 \\
T_{n+1} &= (\boxed{\quad A \quad})\,T_n + \boxed{\quad B \quad}
\end{aligned}
$$

(次ページにつづく)

(前ページのつづき)

4)　図 7.1 の関数 check を図 7.4 のものに変更して，n-クイーン問題の解を列挙するプログラムを作成した．空欄 $\boxed{C}$ および $\boxed{D}$ に入る式を答え，なぜそれらの式になったかを説明せよ．

n-クイーン問題とは，大きさ $n \times n$ のチェス盤上に n 個のクイーンを互いの効き筋上にないように配置する全ての方法を列挙する問題である．ここでは線対称および回転対称となる解同士は別のものとみなす．チェスのクイーンの効き筋は，縦，横および斜め 45° 方向の任意の場所である (図 7.5)．修正したプログラムの出力の各行は解となるクイーンの配置を表す．ある行が $p_0, p_1, \ldots, p_{n-1}$ $(0 \leq p_i < n)$ という数列であるとき，この数列は $0 \leq i < n$ をみたす各 i について座標 (i, p_i) にクイーンが配置されていることを表している．例えば図 7.5 の配置は 0，2，4，1，3 という行で表される．

5)　組込み系ソフトウェア会社に勤務するプログラマ A 氏が業務で図 7.1 の関数 f を書いたところ，会社から関数の再帰呼び出しの使用禁止を命ぜられた．そこで A 氏は，再帰呼び出しを用いずに元と同じ動作をするよう関数 f を書き直した．書き直した関数 f を図 7.6 に示す．この関数を f(n, 0) という形で呼び出すと，図 7.1 の関数 f を同じ引数で呼び出した場合と同じ出力を得ることができる．

a) 空欄 $\boxed{E}$ および $\boxed{F}$ に入る式を書け．

b) 一般に関数定義において再帰呼び出しを使うことの利点と欠点を理由と共に述べよ．利点と欠点はそれぞれ 2 つずつ挙げること．

```
int check(int x, int y) {
    int k;
    for (k = 0; k < x; k++) {
        if (a[k] == y ||
            a[k] == [    C    ] ||
            a[k] == [    D    ] )
            return 0;
    }
    return 1;
}
```

図7.4

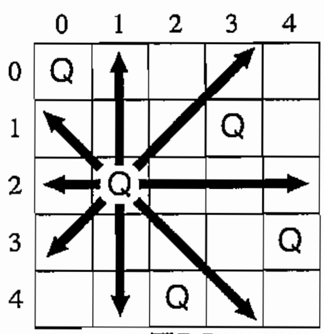


図7.5

```
void f(int n, int x) {
    int y = 0;
    while (1) {
        while (y < n) {
            if (check(x, y)) {
                a[x] = y;
                x = x + 1;
                if (x >= n) {
                    printa(n);
                    goto backtrack;
                }
                y = [    E    ];
            }
            else {
                y = y + 1;
            }
        }
    backtrack:
        x = x - 1;
        if (x < 0) return;
        y = [    F    ];
    }
}
```

図7.6