No. H26 [I]

Date Jul 18 2019

H26 [I] アルゴリズムとプログラミング

[1] 【必須問題】アルゴリズムとプログラミング （情報工学 1

配点：(1) 30 点，(2) 20 点，(3) 15 点，(4) 25 点，(5) 20 点，(6) 15 点

図 1 は ANSI-C 準拠である C 言語のプログラム（program）である．このプログラムにおいて，insert 関数はある特定の規則に従ってデータ列（data sequence）に新しいデータ（data）を挿入（insert）する関数（function）であり，データ列を格納する配列（array）A，配列の大きさ SIZE，挿入するデータ d を引数（arguments）とする．delete 関数はデータ列からある特定の規則に従ってデータを一つ取り出し（retrieve），削除（delete）する関数であり，データ列を格納する配列 A を引数とし，取り出したデータを戻り値（return value）とする．main 関数は，それらの二つの関数を実行する一例を示している．以下の各問に答えよ．

(1) 図 1 のプログラムにおいて，43 行目および 46 行目の処理を実行した結果をそれぞれ示せ．

(2) 図 1 のプログラムにおいて，43 行目の処理を実行した直後の変数 front および変数 rear の値をそれぞれ示せ．

(3) 図 1 の insert 関数内の 11～21 行目では，新たに挿入するデータの挿入位置を決定している．この際，データ列を最初から一つずつ調べることなく，処理を効率化している．具体的にどのようなことをしているかを簡潔に説明せよ．

```
1  #include <stdlib.h>
2  #include <stdio.h>
3
4  int front = 0;
5  int rear = 0;
6
7  void insert(int A[], int SIZE, int d) {
8    int p, left, right, m, i;
9    if (rear > SIZE-1) { printf ("Overflow !!\n"); exit(1); }
10
11   p = -1; rear = SIZE で True
12   if (front == rear) p = front;
13   else {
14     left = front; right = rear-1;
15     while (left < right) {
16       m = (left+right)/2;
17       if (A[m] == d) { p = m+1; break; }
18       if (d < A[m]) right = m-1; else left = m+1;
19     }
20     if (p == -1) { if (A[left] > d) p = left; else p = left+1; }
21   }
22
23   i = rear;
24   while (i > p) { A[i] = A[i-1]; i--; }
25   A[p] = d; rear++;
26 }
27
28 int delete(int A[]) {
29   int x;
30   if (front == rear) { printf ("Underflow !!\n"); exit(1); }
31   x = A[front]; front++;
32   return x;
33 }
34
35 int main () {
36   int i;
37   int A[20]; int SIZE = 20;
38
39   int a1[5] = {10, 5, 20, 6, 13};
40   int a2[5] = {2, 5, 18, 7, 5};
41
42   for (i = 0; i < 5; i++) insert(A, SIZE, a1[i]);
43   for (i = 0; i < 3; i++) printf("%d ", delete(A)); printf ("\n");
44
45   for (i = 0; i < 5; i++) insert(A, SIZE, a2[i]);
46   for (i = 0; i < 7; i++) printf("%d ", delete(A)); printf ("\n");
47
48   return 0;
49 }
```

図 1：プログラム

(1)(解)
43行目： 5, 6, 10
46行目： 2,5,5,7,13,18,20

(2)
42行目が終了後のAの状態
A=[5, 6, 10, 13, 20, ○]
↑front ↑rear

43行目が終了すると
A=[○, ○, ○, 13, 20, ○]
↑front ↑rear

(解) front=3
rear=5

(3)(解) 11行目でAが空の場合の処理を行い，13～21行目で二分探索を用いて探索範囲を1/2に減らすことを繰り返して、効率化をしている。

(4) データ列のデータ数を n としたとき，insert 関数および delete 関数を実行する際の時間計算量（time complexity）のオーダー（order）を示せ．その理由も簡潔に説明せよ．

(5) 図 1 の main 関数において，47 行目以降に insert 関数および delete 関数を多数回実行した場合，データ列のデータ数に関わらず，実行途中でデータの挿入に失敗してしまう．その原因，および，データの挿入が失敗する条件を，データ挿入回数の観点から簡潔に説明せよ．

(6) 上記(5)の問題を解決するために，配列を十分大きくする，および，データの挿入回数に制限を設ける以外に，どのような方法があるか，簡潔に説明せよ．ただし，その方法が insert 関数および delete 関数の時間計算量のオーダーを変えることがあってはならない．また，できるだけ時間計算量の増加が少ない方法を示すこと．なお，その方法を実現するために，関数に新たな引数を追加することがあってもよいが，新たな関数は追加してはならない．

(4)

(解) insert 関数は 15行目に 最大 n 回のループと
24行目に 最大 n 回のループがある．
よって計算量のオーダーは $O(n+n) = O(n)$

delete 関数はループを含まず，定数時間処理
のみなので $O(1)$．

(5)

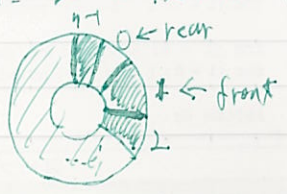
(解) 挿入，削除後の front と rear のポインタの更新を以下のように
配列要素数で割ったあまりを用いる，

25行目 rear ++ ⇒ rear = (rear+1) % SIZE
31行目 front++ ⇒ front = (front+1) % SIZE
※delete 関数の引数に SIZE を追加．

またこれにともなって，Overflow の判定条件を以下のように
変更する
9行目 rear ? SIZE-1 ⇒ (rear+1) % SIZE == front

[リングバッファ]

「リングバッファ」では 少なくとも
1つ空の要素をつくる．
（今回は rear が指す要素は
常に空き領域）．



No. H26 [2].

H26 [2] 計算機システムとシステムプログラム

[2] 【必須問題】計算機システムとシステムプログラム (情報工学 3/

配点：(1-1) 36 点，(1-2) 8 点，(1-3) 8 点，(1-4) 18 点，(2-1) 20 点，(2-2) 35 点

(1) 計算機の演算器に関する以下の小問に答えよ．解答は全て解答欄に書くこと．

(1-1) 以下に示す仕様の計算機およびＣ言語コンパイラを考える．

- 演算器 (ALU: arithmetic logic unit)，レジスタ，メモリの語 (word) 長は全て 8 ビット．
- int 型は 8 ビット符号付き (signed) 整数 (integer) で，2 の補数系 (2's complement system) 表現．
- 整数演算時にはオーバーフローは無視され，演算器での演算結果がそのまま格納される．
- float 型は 8 ビット浮動小数点数 (floating point number)．表現したい数を $(-1)^s \times (1+m) \times 2^{(e-3)}$ の形式で表し，最上位ビット (MSB: most significant bit) から s，e，m の順にメモリに格納する．s は 1 ビット，e は 3 ビット，m は 4 ビットである．仮数部 (significand) は正規化 (normalize) ($1 \leq 1+m < 2$) されており，整数部 (integer part) の 1 はメモリには格納されない (隠しビット (hidden bit))．仮数部の 2 進数表現の小数部 (fractional part) 上位 4 桁をメモリに格納する．5 桁目以降は切り捨て (round down) により丸める．

この計算機およびＣ言語コンパイラを用いて，図 1 に示すプログラムをコンパイル，実行した．return 文の時点での各変数 (variable) の値を 10 進数で示せ．また，それらを格納するメモリの内容をビット列 (bit string) で示せ．なお，float 型変数の値は，指数表記 (scientific notation) を用いずに表すこと．また，メモリの内容 (ビット列) は，左端を最上位ビットとして示すこと．

```
int main()
{
  int i, j, k, m, n;
  float d, e;

  i = 90;
  j = -40;
  k = i + j;
  m = i - j;
  n = j - i;
  d = 0.4;
  e = d + 0.8;

  return 0;
}
```

図 1

(1-1)

$i = 90, = (0101\ 1010)_{2c}$

$40 = 32+8 = (0010\ 1000)_2$

$-40 = (1010\ 1000)_2$

$= (1101\ 0111)_{1c}$

$= (1101\ 1000)_{2c}$, $= j$

2)50 2)25…0 2)12…1 2)6…0 2)3…0 2)1…1 0…1

2)90 2)45…0 2)22…1 2)11…0 2)5…1 2)2…1 2)1…0 0…1

$k = 90 + (-40) = 50 =$

$= (0011\ 0010)_{2c}$

$m = (0101\ 1010)_{2c} - (1101\ 1000)_{2c}$

$= (1000\ 0010)_{2c}$

$= -126$〃

```
  0101 1010
- 1101 1000
 (1)1000 0010
```

$n = (1101\ 1000)_{2c} - (0101\ 1010)_{2c}$

$= (0111\ 1110)_{2c}$

$= 64 + 32 + 16 + 8 + 4 + 2$

$= 126$〃

```
  0010 01
  1101 1000
- 0101 1010
  0111 1110
```

$d = 0.4 \simeq (0.0110011\cdots)_2$

$= (-1)^0 \times (1.1001) \times 2^{-2}$

$e = -2 + 3 = 1$

0.4 ×2 = 0.8 ×2 = 1.6 ×2 = 1.2 ×2 = 0.4

よって

$d \simeq (-1)^0 \times (1.1001) \times 2^{1-3}$

$= (1 + \frac{1}{2} + 0 + 0 + \frac{1}{16}) \times 2^{-2}$

$= \frac{16+8+1}{16} \times \frac{1}{4} = 0.390625.$

$0.8 = (-1)^0 \times (0.1100\ 10)_2$ 2−3

$= (-1)^0 \times (1.1001) \times 2^{-1}$

よって

$e = (-1)^0 \times (1.1001) \times 2^{-1} + (-1)^0 \times (1.1001) \times 2^{-2}$

$= [(1.1001 + 1.1001 \times 2^{-1}), -1]$

$= [(1.1001 + 0.11001), -1]$

$= [(10.01011), -1]$

$= (-1)^0 \times (10.01011)_2 \times 2^{-1}$

$= (-1)^0 \times (1.0010)_2 \times 2^0$ → 3−3

$=$

$[0\ 011\ 0010]$〃

```
  1.1001
+ 0.11001
 10.01011
```

(解)

	10 進表記	8ビット2進表記
i	90	0101 1010
j	-40	1101 1000
k	50	0011 0010
m	-126	1000 0010
n	126	0111 1110
d	0.390625	0001 1001
e	1.125.	0011 0010

No. H26 [2]

Date

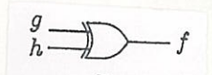


図2

(1-2) 計算機において，符号付き整数の表現に2の補数系表現を用いることの利点を述べよ．

(1-3) 入力として8ビット符号なし(unsigned)整数 y(ビット列：$y_7 y_6 \cdots y_0$)と1ビット制御信号(control signal) w が与えられたとき，出力 y'(ビット列：$y'_7 y'_6 \cdots y'_0$)として，$w=0$ ならば y そのものを，$w=1$ ならば y の1の補数(1's complement)を出力する回路を作ることを考える．この回路は，同じ構成の回路を8ビット分並べて構成することができる．第 i ビット($i \in [0,7]$)1ビット分の回路を，図2に示す排他的論理和(exclusive OR)を用いて構成し，その回路図を示せ．なお，排他的論理和の論理式は $f = g\bar{h} + \bar{g}h$ である．必要ならばNOT回路も用いてよい．

(1-4) 図3に示す8ビット符号なし整数加算器を用いて，2の補数系表現による8ビット符号付き整数の加算ならびに減算を行う演算器を作ることを考える．被演算数(operand) x(ビット列：$x_7 x_6 \cdots x_0$)，y(ビット列：$y_7 y_6 \cdots y_0$)，1ビットの演算制御信号 w が与えられ，演算結果 z(ビット列：$z_7 z_6 \cdots z_0$)が出力される．$w=0$ ならば $z=x+y$，$w=1$ ならば $z=x-y$ となる．この回路を，排他的論理和と8ビット符号無し整数加算器を用いて構成し，その回路図を書け．なお，必要ならばNOT回路も用いてよい．

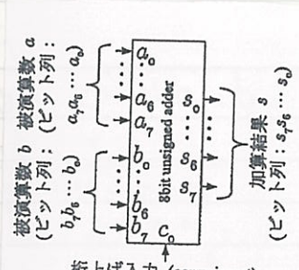


図3

(1-2)

(解) ・(例1) 加減減をするときに、符号を意識しなくてよい。

・(例2) 正，負，加減に関わらず、同じ1つの加算器で演算ができる点。

オーバーフローに注意が必要。

(1-3)

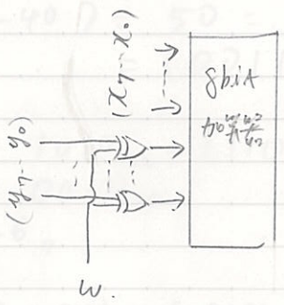
	a_i 0	a_i 1
w 0	0	1
w 1	1	0

$y'_i = y_i \oplus w$

(解)

(1-4)

(解)



② ガードビット と 丸め.

② ガードビット

・正規化された浮動小数点2進数 A と B の減算.

・仮数部の減算を正しく行うために、可数部より長さ1ビット右に延長した減算器が必要

② 丸め.

・真の数値から、それより有効桁数の小さい桁数を作る.

① 切り捨て　0.01** ⇒ 0.01　誤差 $0 \leq \delta < 1$

② フォン/イロン丸め　(a-3 ~ a-n) が全て0の場合を除いて、最下位桁を1にする.　0.1010 → 0.11　0.1000 → 0.11　誤差 $-1 < \delta < +1$

③ 丸め！

⇒ 最下位桁の、次の桁に1を加えて切り捨て

0.**(0…) → 0.**

0.**(1…) → 0.** + 0.01　誤差 $-0.1 \leq \delta < +0.1$

※ 中央値0.1は切り上げ.

④ 丸め2

0.**(00) → 0.**

0.**(01) → 0.**

0.*0(10) → 0.*0

0.*1(10) → 0.*1 + 0.01

0.**(11) → 0.** + 0.01.

No. H26[3]

Date

テストアンドセット etc.

(2) 排他制御(mutual exclusion)に関する以下の小問に答えよ.

(2-1) 以下の空欄 [a] ～ [c] を適切な用語で埋めよ.

共有資源(shared resource)の排他制御を実現する方法としてテストアンドセット(test and set)やセマフォ(semaphore)等がある. テストアンドセットは一つしかない共有資源の排他制御を実現する機械語命令(machine instruction)である. これは,「共有資源の [a] 状態と [b] 状態とを監視し制御する」機能を不可分操作(atomic operation) として実行する命令である. 不可分操作として実行されるのは以下の二つである.

1) テスト : 共有資源が [a] 状態か [b] 状態かをチェックする. [b] 状態の場合, 終了する.
2) セット : 共有資源を [b] 状態に変更する.

ここでテストとセットは不可分操作のため, [c] が発生しない.

(2-2) テストとセットを不可分操作とする必要性を, 不可分操作でなかった場合に排他制御が失敗する状況を例示することにより明らかにせよ. なお回答では以下の状況を仮定せよ.

- 二つのプロセスXとYが一つしかない共有資源Zを取り合う.
- プロセスXが先にテストを実行する.

(2-1)

(解) a) フリー (アンロック)
b) ロック
c) 不整合.

(2-2)

今、共有資源はどのプロセスからもセットされていないものとする.
プロセスXがテストを実行した後、プロセスXがセットを実行する前にプロセスYがセットを行うと この時点ではどのプロセスもセットを行っていないので、プロセスXYはともにセット可能な状態となる. 後、プロセスXYが共にセットを行い共有資源Zにアクセスを行うと、排他制御は失敗する。

よって テストとセットは 不可分命令である必要がある.

No. H26 四

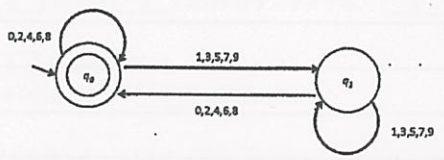
Date

H26 計算理論

4 【選択問題】計算理論 (情報工学 7/12

配点:(1-1) 25 点, (1-2) 25 点, (1-3) 10 点, (2-1) 15 点, (2-2) 30 点, (2-3) 20 点

(1) 有限オートマトン (finite automaton) は 5 項組 $(Q, \Sigma, \delta, q_0, F)$ で与えられる. ここで $Q, \Sigma, \delta, q_0, F$ は, それぞれ, 状態 (state) の有限集合, 入力記号 (input symbol) の有限集合であるアルファベット (alphabet), 遷移関数, 開始状態 (initial state) ($q_0 \in Q$), 受理状態集合 ($F \subseteq Q$) である. 有限オートマトンに関する以下の各小問に答えよ. アルファベット Σ を $\{0,1,2,3,4,5,6,7,8,9\}$ とする. 以下の各小問では Σ 上の語 (word) を先頭を上位桁とする 10 進数の非負整数とみなす. 通常の非負整数表現以外の語 (例えば 00045) についての動作は考慮しなくて良い. 次に与える有限オートマトン A は与えられた語が 2 で割り切れるときのみ受理 (accept) するオートマトンである.



(1-1) 与えられた語が 3 で割り切れるときのみ受理する有限オートマトン B の構築の方法を説明せよ. 説明にあたってはオートマトンの図を用いてもよい.

(1-2) 小問 (1-1) の構築方法で得られる有限オートマトン B を用いて 15 で割り切れるときのみ受理する有限オートマトン D を構築したい. まず与えられた語が 5 で割り切れるときのみ受理する有限オートマトン C を示せ. ついでこの 2 つの有限オートマトン B と C をどのように用いればオートマトン D を構築できるかを説明せよ.

(1-3) 小問 (1-2) の構築方法で得られる有限オートマトン D を少し変更し, 3 または 5 で割り切れるときのみ受理する有限オートマトン E を構築するにはどのようにすればよいか, 変更の方法を説明せよ.

(1-1) ある時点までに入力された数の3の剰余類を考える。よって状態は、3で割った余りをラベルにする。つまり $Q = \{0, 1, 2\}$

ある時刻 t までに入力された10進数を Q_t,

$Q_t \pmod 3 = r_t$ とする.

次に入力 x があるとき

$$Q_{t+1} = Q_t \times 10 + x$$

よって

$$\begin{aligned} r_{t+1} &= Q_t \times 10 + x \pmod 3 \\ &= Q_t + x \pmod 3 \\ &= r_t + x \pmod 3 \end{aligned}$$

r_{t+1} が次の状態になる。つまりこれが遷移規則

以上よりオートマトン B は

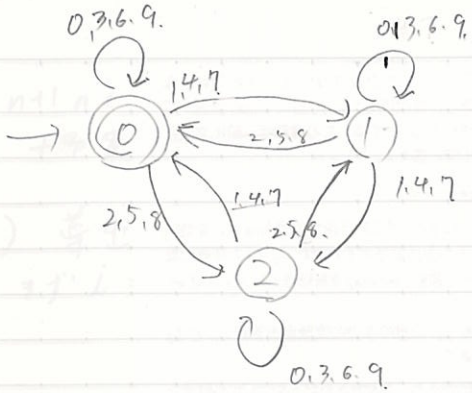
$B = (Q_B, \Sigma, \delta_B, q_0, F)$

$Q_B = \{0, 1, 2\}$, $\Sigma = \{0,1,2,3,4,5,6,7,8,9\}$

$q_0 = 0$, $F = 0$,

$\delta(q, x) = \{$ $r_t + x \pmod 3$ をラベルにもつ状態 $\}$

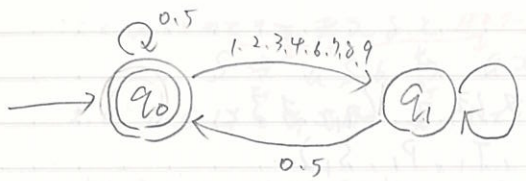
B は.



(2). 5で割り切れるのは、入力が0または5の場合のみ.

よって

C



$C = (Q_C, \Sigma, \delta_C, q_0, F)$

$Q_C = \{q_0, q_1\}$

$\delta_C = \begin{cases} q_0 & x = 0,5 \\ q_1 & \text{others.} \end{cases}$

15の倍数を受理するオートマトン D は、BとCの直積を考えればよい.

よって $D = (Q_D, \Sigma, \delta_D, q_0, F)$ は.

$Q_D = Q_B \times Q_C = \{(0,q_0), (0,q_1), (1,q_0), (1,q_1), (2,q_0), (2,q_1)\}$

$\delta_D = \delta_D((q_B, q_C), x) = (\delta_B(q_B, x), \delta_C(q_C, x))$

$q_0 = (0, q_0)$ $\forall q_B \in Q_B, \forall q_C \in Q_C, x \in \Sigma$.

$F = (0, q_0)$ //

(3). 状態集合, 遷移関数は D のままで、受理状態の集合を次のように変えればよい.

$F = \{(0,q_0), (0,q_1), (1,q_0), (2,q_0)\}$ //

H26 [4].

(2) 文脈自由文法 (context-free grammar) G は 4 項組 (N, T, P, S) で与えられる．ここで N, T, P, S は，それぞれ，非終端記号 (non-terminal symbols) 集合，終端記号 (terminal symbols) 集合，生成規則 (production rules) 集合，開始記号 (start symbol) である．文脈自由文法 G の生成文全体を表す言語を $L(G)$ とする．この言語は一般に文脈自由言語 (context-free language) と呼ばれる．これらに関する以下の各小問に答えよ．

(2-1) 文脈自由言語 $L_1 = \{c^m \mid m \geq 1\}$ を生成する文脈自由文法 G_1 を示せ．

(2-2) 言語 $L_2 = \{a^n b^n c^m \mid n \geq 1, m \geq 1\}$ および $L_3 = \{a^m b^n c^n \mid n \geq 1, m \geq 1\}$ が文脈自由言語であることを証明せよ．言語 L_2, L_3 を生成する文法を示して証明する場合は，それらの生成言語がそれぞれ L_2, L_3 と等価になることを証明すること．なお必要であれば小問 (2-1) の結果と以下の補題 1(lemma) を証明なしに用いてよい．

[補題 1]
文脈自由言語全体の集合は連接演算・について閉じている．すなわち，任意の 2 つの文脈自由言語 L_α と L_β に対して言語 $L_\alpha \cdot L_\beta = \{xy \mid x \in L_\alpha, y \in L_\beta\}$ も文脈自由言語である．

(2-3) 文脈自由言語全体の集合が積演算 ∩ について閉じていないことを証明せよ．必要であれば小問 (2-2) の結果と以下の補題 2 を証明なしに用いてよい．

[補題 2]
言語 $L_4 = \{a^n b^n c^n \mid n \geq 1\}$ は文脈自由言語ではない．

(2-1)
(解) L_1 を生成する文法を G_1 とすると

$G_1 = (N_1, T_1, P_1, S_1)$

$N_1 = \{S\}$　$T_1 = \{c\}$

$P_1 = \{S \to c,\ S \to cS\}$,

$S_1 = S$ //

(2-2)
(解) 次のような言語 L_5 とそれを生成する文法 G_5 を以下のように定義する．

$L_5 = \{a^n b^n \mid n \geq 1\}$

$G_5 = (N_5, T_5, P_5, S)$

$N_5 = \{S\}$, $T_5 = \{a, b\}$,

$P_5 = \{S \to ab,\ S \to aSb\}$,

ここから $L_5 = L(G_5)$ つまり 任意の語 w に対して

$w \in L_5 \iff w \in L(G_5)$

を示す． ⇐この証明がメイン．

(十分条件) n に関する帰納法で証明する．

$n = 1$ のとき　$w = ab \in L_5$，

これは P_5 の生成規則 $S \to ab$ で導出できる．

n のとき　$a^n b^n \in L_5$ ならば $w \in L(G_5)$ と仮定すると

$S \overset{*}{\underset{P_5}{\Rightarrow}} a^n b^n$

となる．このとき $n+1$ つまり $w = a^{n+1} b^{n+1}$ について考えると，

$S \underset{P_5}{\Rightarrow} aSb \overset{*}{\underset{P_5}{\Rightarrow}} a(a^n b^n)b = a^{n+1} b^{n+1} = w$

よって $n+1$ のときも成立する．

従って，十分条件は $n \geq 1$ で成立する．

(必要条件) 導出のステップ数 i に関する帰納法で示す．

まず $i = 1$ のとき，1ステップで導出できるのは 生成規則 $S \to ab$ より $w = ab$．これは L_5 の $n=1$ に該当．

i のとき $\forall x \in L(G_5) \Rightarrow x \in L_5$ と仮定する．つまり

$S \overset{i}{\underset{P_5}{\Rightarrow}} x$．また ある $k \geq 1$ が存在して $x = a^k b^k$ とかける．Total $i+1$ ステップ

ここで $i+1$ のときを考えると 1ステップ　iステップ

$S \underset{P_5}{\Rightarrow} aSb \overset{i}{\underset{P_5}{\Rightarrow}} axb = a(a^k b^k)b = a^{k+1} b^{k+1} = w$

よって，$w \in L_5$ であるので $i+1$ でも成立する．

以上より，必要条件も $i \geq 1$ で成立．

したがって　$L_5 = L(G_5)$．

ここで G_5 は文脈自由文法 (CFG) であるので L_5 は文脈自由言語である．L_2 は L_5 と L_1 の連接である．したがって補題 1 より文脈自由言語の連接 L_2 は文脈自由言語である．

また L_1 で $c \to a$，L_5 で $a \to b$, $b \to c$ とおきかえ，$L_1 \cdot L_5$ の順で連接することで同様に L_3 も文脈自由言語である． ∎

(2-3)
(解) CFL L_2 と L_3 の積演算について考える．

L_2 は a と b，L_3 は b と c の数が等しい．

よって $L_2 \cap L_3$ は a と b かつ b と c の数が等しい，つまり a, b, c の数が等しい言語となる．よって $L_2 \cap L_3 = L_4$．

補題より L_4 は文脈自由言語でないので，積演算については閉じていない．

∎

※PDAを構成して L_5 が文脈自由という風でOK
(その場合は証明までするべきか？)

No. H26 6

Date

H26 6 電子回路と論理設計.

6 【選択問題】電子回路と論理設計 (情報工学 11/12)

配点：(1-1) 25 点，(1-2) 10 点，(1-3) 20 点，(1-4) 20 点，(2-1) 20 点，(2-2) 30 点

2 入力マルチプレクサ（multiplexor）について，次の各問に答えよ．

2 入力マルチプレクサは，3 個の入力 select，a，b と 1 個の出力 out をもち，入力 select が 1 のとき out に a の値を出力し，select が 0 のとき out に b の値を出力する．

(1) 2 入力マルチプレクサを次の手順で，論理回路（logic circuit）で実現する．

(1-1) 2 入力マルチプレクサの真理値表（truth table）を作成せよ．

(1-2) 2 入力マルチプレクサのカルノー図（Karnaugh map）を作成せよ．

(1-3) (1-2)で作成したカルノー図を利用して，out の最簡積和形（最小積和形，minimal sum-of-products expression）の論理式（logic expression）を導出せよ．

(1-4) (1-3)で求めた最簡積和形の論理式を用いて得られる 2 入力マルチプレクサの論理回路図を示せ．ただし，論理ゲート（logic gate）は，2 入力 NAND ゲートのみが使用できるものとする．

(2) (1-4)で設計した 2 入力マルチプレクサの遅延時間（delay）について考える．遅延時間とは，入力が変化してから出力が変化するまでに要する時間である．ただし，設計に用いた 2 入力 NAND ゲートの遅延時間は，出力が 0 から 1 に変化するときは T，出力が 1 から 0 に変化するときは 2T であるとする．

(2-1) 入力 (select, a, b) が (0, 0, 0) から (0, 0, 1) に変化するときの，マルチプレクサの遅延時間を T を用いて表せ．

(2-2) select，a，b のいずれか一つの入力が変化するときの，マルチプレクサの遅延時間の最大値を求めよ．また，そのときの入力の変化を下の例にならって答えよ．

例：(select, a, b) が (1, 1, 1) から (0, 1, 1) に変化するとき．

(1)

(1-1) (角子)

select	a	b	out
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

(1-2)

(角子)

		b		a
		00	01	11	10
select	0		○	○	
	1			○	○

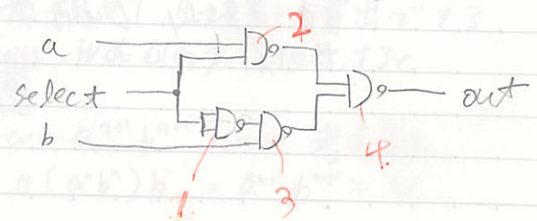
(1-3)

(1-2)のカルノー図より

(角子)

$out = select \cdot a + \overline{select} \cdot b$ //

(1-4) $out = \overline{\overline{sel \cdot a} \cdot \overline{sel \cdot b}}$ より



(2)

(2-1)(1-4)の図のように NAND ゲートに番号をつける．

それぞれのゲートの出力値は以下のように変化する．

	NAND Gate 1	2	3	4
(0 0 0)	1	1	1	0
(0 0 1)	1	1	0	1

(Sel, a, b)

よって

2T + T = 3T (角子)

ゲート3 ゲート4

(2-2)

最大の遅延時間をおける上で以下の点に注目する．

① a, b は最大2つのゲートの値を変えるが，select は最大3つ

② クリティカルパスになるのはゲートが3つある！ select → NAND(2) → NAND(3) → NAND(4)

よって，select の値が変わるパターンで比較する．

	NANDのゲート 1	2	3	4
(0 0 0)	1	1	1	0
(1 0 0)	0	1	1	0
(0 0 1)	1	1	0	1
(1 0 1)	0	1	0	1
(0 1 0)	1	1	1	0
(1 1 0)	0	0	1	1
(0 1 1)	1	1	0	1
(1 1 1)	0	0	1	1

1→0 (2T) が，最多

2が 1→0 でも クリティカルパスでない

以上より

最大の遅延時間は

2T × 2 + T = 5T //

これは (select, a, b) が (0 0 1) から (1 0 1) に変化するとき．