大阪大学大学院情報科学研究科
平成 29 年度　博士前期課程　入試問題
(A) 情報工学　解答・解説

楠本研究室：有馬諒，佐々木美和，谷門照斗，松尾裕幸，山田涼太

2017 年 5 月 30 日

1　アルゴリズムとプログラミング

■■■ 解答 ■■■

(1)

	Len[0]	Len[1]	Len[2]	Len[3]
1 巡目	0	2	5	INF
2 巡目	0	2	3	3

(2) **時間計算量**：$O(n^2)$

理由：内側の for ループは n 回処理を行う．外側の while ループでは visited がすべて 1 となるまで処理を行うため n 回実行される．よって時間計算量は $O(n^2)$ となる．

(3) (ア) `Prev[j] = i;`　(イ) `printpath(3);`

(4)

(4-1) まず，compute(w,n,0,0) を実行する．Len[0] に頂点 0 から 0 への距離，Len[1] に頂点 0 から 1 への距離，Len[2] に頂点 0 から 2 への距離，Len[3] に頂点 0 から 3 への距離が求められているのでそれぞれ出力する．
次に，compute(w,n,1,1) を実行する．Len[0] に頂点 1 から 0 への距離，Len[1] に頂点 1 から 1 への距離，Len[2] に頂点 1 から 2 への距離，Len[3] に頂点 1 から 3 への距離が求められているのでそれぞれ出力する．
以下同様に，compute(w,n,2,2) を実行して頂点 2 を始点したときの距離を出力し，compute(w,n,3,3) を実行して頂点 3 を始点としたときの距離を出力する．
以上より，実行回数 $T = 4$ である．

(4-2) compute(w,n,0,1) を実行する．無向グラフなので，Len[1] の値は頂点 0 から 1 および頂点 1 から 0 の距離となり，それを出力する．以下同様に，compute(w,n,0,2)，compute(w,n,0,3)，compute(w,n,1,2)，compute(w,n,1,3)，compute(w,n,2,3)を呼び出し，値を出力する．以上より呼び出し回数 $T = 6$ である．

■■■ 解説 ■■■

(1)

配列 Len の初期値は Len[0] のみ 0 で他は INF である．1 巡目では頂点 0 を始点に Len を更新し，Len[1] $= 0 + 2$，Len[2] $= 0 + 5$ となる．

visited が 0 の要素の中で Len が一番小さいものは頂点 1 であるので，2 巡目は頂点 1 を始点に Len を更新し，Len[3] $= 2 + 1$ となる．

(3)

Prev[i] には，最短経路を通った時に頂点 i の前に通る頂点の番号を保存する．(ア)の行では頂点 j までの現在の最短距離よりも頂点 i から (i, j) 間にある辺を通って頂点 j に行く方が距離が短くなるため，その値に Len[j] を更新している．よって，頂点 j の前の頂点は頂点 i となるため Prev[j] = i とする．

関数 printpath は再帰的に Prev をたどり，終点から戻るように経路を復元するので，終点の頂点 3 で printpath を呼び出すものが(イ)となる．

(4)

問題のプログラムでは，与えられたグラフによっては実行が終了しないバグがあるため注意が必要である[*1]．

図 1　25 行目：

誤　`if (visited[i]==1||w[i*n+j]==INF) continue;`

正　`if (visited[i] == 1) continue;`

(4-1)

compute 中の while の終了条件は visited の要素がすべて 1 になることであるため，一度の compute の呼び出しで始点からすべての頂点への最短距離が Len に求まる．よって，呼び出し回数は 4 回でよい．

(4-2)

プログラムの変更により，終点までの最短距離が求まると while を抜けるようになり，必ずしも一度の呼び出しですべての頂点への最短距離が求まるとは限らなくなった．そのため，すべての頂点対ごとに compute を呼び出す必要がある．

ただし，無向グラフでは頂点 i から j への最短距離と頂点 j から i への最短距離は同じなので，呼び出し回数は 6 回でよい．

[*1] 平成 29（2017）年度博士前期課程筆記試験における出題ミスについて
http://www.ist.osaka-u.ac.jp/japanese/news/2016/08/292017.html

2　計算機システムとシステムプログラム

■■■ 解答 ■■■

(1)

(1-1)

(1-1-1) 表 1 参照.

(1-1-2) 符号絶対値表現: 10001 10100

1 の補数表現: 11110 01011

2 の補数表現: 11110 01100

(1-1-3) 符号なし整数: 691

符号絶対値表現: −179

1 の補数表現: −332

2 の補数表現: −333

(1-2)

(1-2-1) $s_i = \overline{a_i}\,\overline{b_i}c_i + \overline{a_i}b_i\overline{c_i} + a_i\overline{b_i}\,\overline{c_i} + a_ib_ic_i$

$c_{i+1} = a_ib_i + b_ic_i + c_ia_i$

(1-2-2) $c_3 = c_0p_0p_1p_2 + g_0p_1p_2 + g_1p_2 + g_2$

(1-2-3) 桁上げ伝搬加算器の最大遅延: 18

ｸﾘﾃｨｶﾙﾊﾟｽ: $a_0 \rightarrow c_1 \rightarrow c_2 \rightarrow c_3 \rightarrow s_3$

桁上げ先見加算器の最大遅延: 14

ｸﾘﾃｨｶﾙﾊﾟｽ: $a_0 \rightarrow p_0 \rightarrow c_3 \rightarrow s_3$

(2)

(2-1) ㋐ a　㋑ k　㋒ c　㋓ g　㋔ h　㋕ d　㋖ l

(2-2)

(2-2-1) (a) P1: 20, P2: 55, P4: 65, P5: 60, 平均: 50

(b) P1: 20, P2: 55, P4: 75, P5: 30, 平均: 45

(2-2-2) (c) P1: 20, P3: 90, P4: 30, P6: 20, 平均: 40, 利用率: 80%

(d) P1: 30, P3: 40, P4: 55, P6: 15, 平均: 35, 利用率: 100%

■■■ 解説 ■■■

(1)

(1-1)

符号絶対値表現　最上位ビットを符号ビットにする．最上位ビットが 1 の時は最上位ビット以下を 10 進数に変換しその絶対値にマイナスを付ける.

1 の補数表現　最上位ビットが 1 の場合ビット反転したビット列を 10 進数に変換しその絶対値にマイナスを付ける.

2 の補数表現　最上位ビットが 1 の場合ビット反転したビット列に 1 を加算した物を 10 進数に変換しマイナスを付ける.

(1-2)

(1-2-1)

表 2, 3 より求まる.

		$a_i\ b_i$			
		00	01	11	10
c_i	0	0	1	0	1
	1	1	0	1	0

表 2　s_i の状態遷移表

		$a_i\ b_i$			
		00	01	11	10
c_i	0	0	0	1	0
	1	0	1	1	1

表 3　c_{i+1} の状態遷移表

(1-2-2)

$c_{i+1} = a_ib_i + b_ic_i + c_ia_i$ より,

$$c_{i+1} = a_ib_i + c_i(a_i + b_i) = g_i + c_ip_i.$$

したがって,

$$c_3 = c_2p_2 + g_2 = (c_1p_1 + g_1)p_2 + g_2 = \ldots$$
$$= c_0p_0p_1p_2 + g_0p_1p_2 + g_1p_2 + g_2.$$

(1-2-3)

図 1, 2 を参照.

表 1　(1-1-1) の解答

	10 進数	ビット列	10 進数	ビット列
符号なし整数	0	00000 00000	1023	11111 11111
符号絶対値表現	−511	11111 11111	511	01111 11111
1 の補数表現	−511	10000 00000	511	01111 11111
2 の補数表現	−512	10000 00000	511	01111 11111

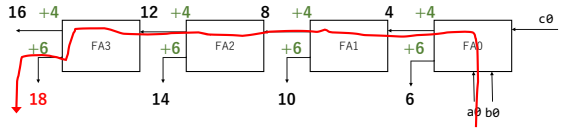


図 1　桁上げ伝搬加算器

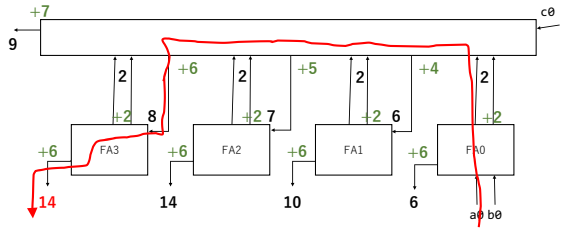


図 2　桁上げ先見加算器

(2)

(2-1)

省略.

(2-2)

(2-2-1)

図 3, 4 参照.

(2-2-2)

図 5, 6 参照. 赤字の矢印は I/O を表す.

利用率については，(c) の場合，実行時間 100 のうち 20 が I/O のみであるため 80% となる．また (d) の場合，実行時間 80 のうち I/O のみの時間がないため 100% となる.

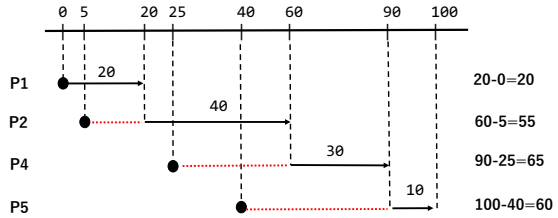


図 3　(a) の図解

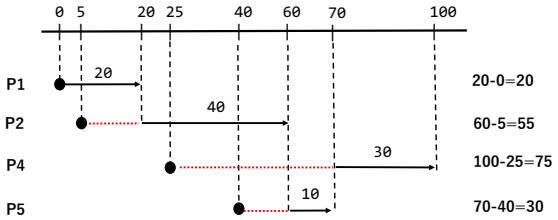


図 4　(b) の図解

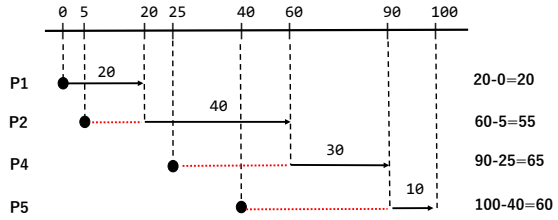


図 5　(a) の図解

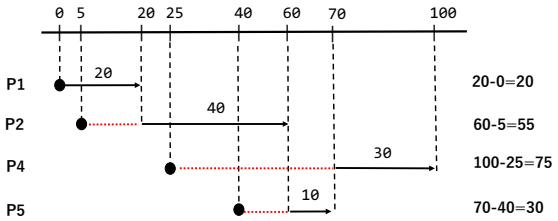


図 6　(b) の図解

3　離散構造

■■■ 解答 ■■■

(1)
　(1-1) α：D1　　β：A1，A2　（**自信なし**）
　(1-2) 演繹(えんえき)定理
　(1-3) **分かりませんでした.**
(2)
　(2-1) 解説参照.
　(2-2) 図 7 に示すグラフ G の通り．ただし，
　　$V = \{1, 2, 3, z\}$ とする.
　(2-3)
　　(2-3-1) $\forall x\, \exists y\, (\neg Z(x) \to Z(y) \wedge e(y, x))$
　　(2-3-2) 解説参照.
(3) 以下の解答には各命題の真偽のみを記す．具体的な証明および反例は解説を参照のこと.
　(3-1) 真
　(3-2) 偽
　(3-3) 偽
　(3-4) 真

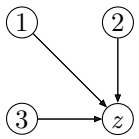


図 7　(2-2) のグラフ G

■■■ 解説 ■■■

(2)
(2-1)
　述語

$$\exists y\, \forall x\, e(x, y) \to \forall x\, \exists y\, e(x, y)$$

が恒真であることを示す.
　十分条件 $\exists y \forall x\, e(x, y)$ は，「適当な y を選べば，すべての x において $e(x, y)$ が真となる」，すなわち「すべてのノードへの有向辺を持つ，あるノードが存在する」ことを表している.
　必要条件 $\forall x\, \exists y\, e(x, y)$ は，「すべての x について，適当な y を選べば $e(x, y)$ が真となる」，すなわち「すべてのノードは，あるノードへの有向辺を持つ」ことを表している.
　十分条件が真となるとき，グラフ G には，すべてのノードへの有向辺を持つノードが 1 つ以上含まれている．このとき必要条件も真となるため，与えられた命題は恒真となることがわかる．よって示せた.

(2-2)
　述語

$$\exists z\, \forall x\, \forall y\, [\, n(x, z) \wedge n(y, z) \\ \to e(x, z) \wedge \neg e(z, x) \wedge \neg e(x, y) \wedge \neg e(y, x)\,]$$

について考える．この論理式を読み解くと，「あるノード $z \in V$ について，z 以外のすべてのノードから z への有向辺が存在し，かつ z への有向辺以外は存在しないようなグラフ G」を図示すればよいことがわかる．頂点数が 4 で，かつこのような条件を満たすグラフの一例を考えると，図 7 のようになる.

(2-3)
(2-3-1)
　閉論理式とは，すべての変数が束縛され，自由変数を持たないような述語論理式のことをいう.
　(条件 X) は，「すべての x について，$x \notin Z$ のとき，$y \in Z$ かつ y から x への有向辺が存在するような適当な y が存在する」と読み替えることができる．これを論理式 A として表すと，

$$A = \forall x\, \exists y\, [\, \neg Z(x) \to Z(y) \wedge e(y, x)\,]$$

となる.

(2-3-2)
　本問で考えるグラフ G は，図 8 に示すようなグラフである.
　まず，問題文に示された仮定（前提条件）を論理式として表していく.
　今回考えるグラフ G に含まれる有向辺は，順序対 $E = \{(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_1)\}$ で与え

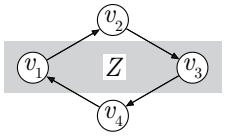


図 8　(2-3-2) のグラフ G

られている．これは，論理式で

$$e(v_1,v_2)\wedge e(v_2,v_3)\wedge e(v_3,v_4)\wedge e(v_4,v_1)\ \cdots\cdots ①$$

と表せる．

次に，（条件 X）を表す述語論理式 A 中の変数に，今回の題意に合致する値を代入し，命題論理式を導く．A 中の変数 x には全称作用素 $\forall$ が付いているため，v_1 から v_4 のすべての値をそれぞれ代入する．つまり，全部で 4 種類の命題論理式が導かれる．また，変数 y には存在作用素 $\exists$ が付いているため，各 x の値に対して，題意に合致する適当な値を y に代入すればよい．

$x=v_1$ のとき，A 中の十分条件は $\neg Z(v_1)$ となる．これは仮定と合致しないため，後で導出原理を適用する際には用いない．同様に，$x=v_3$ のときに導かれる命題論理式も用いないため，これら 2 つの命題論理式は無視する．

$x=v_2$ のとき，A 中の十分条件は $\neg Z(v_2)$ となり，これは仮定と合致する．ノード v_2 への有向辺は (v_1,v_2) であるから，このときの y の値として v_1 を選択すればよい．よって，導かれる命題論理式は

$$\neg Z(v_2)\rightarrow Z(v_1)\wedge e(v_1,v_2)\qquad\cdots\cdots ②$$

となる．同様に，$x=v_4$ のとき，y の値として v_3 を選択して，

$$\neg Z(v_4)\rightarrow Z(v_3)\wedge e(v_3,v_4)\qquad\cdots\cdots ③$$

を得る．

最後に，本問で証明したい命題「論理式①〜③が成り立ち，かつ v_2 および v_4 が集合 Z に含まれなければ，$Z=\{v_1,v_3\}$ となる」を P とすると，P は次のようになる．

$$\begin{aligned}P=\ & e(v_1,v_2)\wedge e(v_2,v_3)\wedge e(v_3,v_4)\wedge e(v_4,v_1)\\ & \wedge\{\,\neg Z(v_2)\rightarrow Z(v_1)\wedge e(v_1,v_2)\,\}\\ & \wedge\{\,\neg Z(v_4)\rightarrow Z(v_3)\wedge e(v_3,v_4)\,\}\\ & \wedge\neg Z(v_2)\wedge\neg Z(v_4)\rightarrow Z(v_1)\wedge Z(v_3)\end{aligned}$$

以下では，P の否定 $\neg P$ が恒偽であることを示すことで，元の P が恒真であることを示す．ここで，

$$\neg(\,\alpha\rightarrow\beta\,)=\neg(\,\neg\alpha\vee\beta\,)=\alpha\wedge\neg\beta$$

$$\begin{aligned}&\neg Z(x)\rightarrow Z(y)\wedge e(y,x)\\ &=\neg\{\,\neg Z(x)\,\}\vee\{\,Z(y)\wedge e(y,x)\,\}\\ &=\{\,Z(x)\vee Z(y)\,\}\wedge\{\,Z(x)\vee e(y,x)\,\}\end{aligned}$$

等の変形を用いると，$\neg P$ は

$$\begin{aligned}\neg P=\ & e(v_1,v_2)\wedge e(v_2,v_3)\wedge e(v_3,v_4)\wedge e(v_4,v_1)\\ & \wedge\{\,Z(v_2)\vee Z(v_1)\,\}\wedge\{\,Z(v_2)\vee e(v_1,v_2)\,\}\\ & \wedge\{\,Z(v_4)\vee Z(v_3)\,\}\wedge\{\,Z(v_4)\vee e(v_3,v_4)\,\}\\ & \wedge\neg Z(v_2)\wedge\neg Z(v_4)\wedge\{\,\neg Z(v_1)\vee\neg Z(v_3)\,\}\end{aligned}$$

と表せる．導出原理を用いて，図 9 のように空節が導かれるため，$\neg P$ は恒偽である．よって，P は恒真であり，題意が示された．

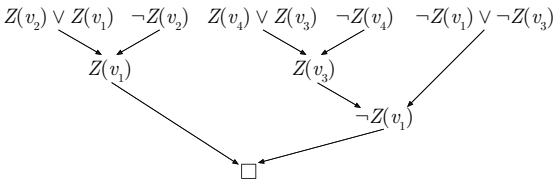


図 9　(2-3-2) の導出原理

(3)

関数 $f:X\mapsto Y$ が**単射**であるとは，

$$(\forall x_1\in X)\,(\forall x_2\in X)\,[\,f(x_1)=f(x_2)\rightarrow x_1=x_2\,]$$

を満たすことをいう．すなわち，同じ関数値に対応する変数値は高々一つであることをいう．

また，関数 $f:X\mapsto Y$ が**全射**であるとは，

$$(\forall y\in Y)\,(\forall x\in X)\,[\,f(x)=y\,]$$

を満たすことをいう．すなわち，どの $y \in Y$ についても，ある $x \in X$ が対応していることをいう．

(3-1)「h が全射ならば，f が全射かつ g が全射」

命題は真である．以下からその証明を示す．

h は $X \mapsto Y \times Y$ の写像であるため，h が全射ならば，h は $Y \times Y$ の全要素を関数値として持つことになる．ここで，h は f, g の関数値の順序対 $(f(x), g(x))$ として与えられているため，h が全射となるためには，f, g のそれぞれが Y の全要素を関数値として持つ必要がある．よって，このとき必ず f, g もそれぞれ全射となる．

(3-2)「f が全射かつ g が全射ならば，h が全射」

命題は偽である．以下から反例を示す．

次のような状況を考える．

$$\begin{aligned} X = Y &= \{1, 2\} \\ f(x) = g(x) &= x \\ Y \times Y &= \{(1,1), (1,2), (2,1), (2,2)\} \end{aligned}$$

このとき，f, g はそれぞれ全射である．ところが，

$$\begin{aligned} h(1) &= (f(1), g(1)) = (1,1) \\ h(2) &= (f(2), g(2)) = (2,2) \end{aligned}$$

より，$(1,2), (2,1) \in Y \times Y$ に対応する変数値がそれぞれ存在しないため，h は全射ではない．

(3-3)「h が単射ならば，f が単射または g が単射」

命題は偽である．以下から反例を示す．

次のような状況を考える．

$$\begin{aligned} X &= \{0, 1, 2, 3\} \\ Y &= \{0, 1, 2\} \\ f(x) &= x \bmod 2 \\ g(x) &= x \bmod 3 \end{aligned}$$

このとき，

$$\begin{aligned} h(0) &= (f(0), g(0)) = (0,0) \\ h(1) &= (f(1), g(1)) = (1,1) \\ h(2) &= (f(2), g(2)) = (0,2) \\ h(3) &= (f(3), g(3)) = (1,0) \end{aligned}$$

より，同じ関数値に写像している変数値が存在しないため，h は単射である．ところが，f, g はそれぞれ同じ関数値に写像している変数値が複数存在し，単射ではない．

(3-4)「f が単射または g が単射ならば，h が単射」

命題は真である．以下からその証明を示す．

$h(x) = (f(x), g(x))$ より，$h(x)$ は $f(x), g(x)$ の順序対で与えられる．順序対においては，一番目の値と二番目の値を入れ替えたものは区別される．すなわち $(a, b) \neq (b, a)$ である．

関数 f または g の少なくとも一方が単射である場合，すべての変数値は異なる関数値に写像するため，$(f(x), g(x))$ は x にどのような値を代入してもそれぞれ区別可能となる．よって関数 h は単射である．

4　計算理論

■■■ 解答 ■■■

(1)

(1-1)

(1-1-1) ㋐ q　㋑ p　㋒ p　㋓ p
㋔ q　㋕ q　㋖ q　㋗ q

(1-1-2) 引数 x によっては，関数 a を一度も呼び出さないことがある．

(1-2)

(1-2-1) 関数 a〜d はそれぞれ，以下のように呼び出される．

関数 a　最初に呼び出されるか，関数 b または関数 d の次に呼び出される

関数 b　関数 a の次に呼び出される

関数 c　最初に呼び出されるか，関数 b または関数 d の次に呼び出される

関数 d　関数 c の次に呼び出される

(1-2-2) **遷移表**　表 4 を参照．

求め方　言語 L_2 を受理する DFA の遷移表を求め，その受理・非受理を入れ替えた．

(2)

(2-1) z **の長さの最大値**　2^k

理由　z を生成する文法がチョムスキー標準形であると仮定する．チョムスキー標準形では，1 つの変数から 2 つの変数もしくは 1 つの終端記号が生成される．よって，$|z|$ が最大となるのは，z の構文木において，葉がすべて深さ $k+1$ にあるような場合である．このとき，深さ k には変数が 2^k 個あり，これらの変数それぞれから終端記号が 1 つ生成されるので $|z| = 2^k$ となる．

(2-2) **関係式**　$k \geq |V|$

理由　鳩の巣原理より．

(2-3) **空欄㋐**　$2^{|V|}$

理由　(2-2) より，$k \geq |V|$ となるような n の値を求めればよい．ここで，$|z| \geq n$, $\log_2 |z| \leq k$ より，$n = 2^{|V|}$ とすれば，$\log_2 n = |V| \leq \log_2 |z| \leq k$ となるから．

表 4　(1-2-2) M_2 の遷移表

	a	b	c	d
$\rightarrow \{A\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\{B\}$	$\varnothing$	$\{C\}$	$\varnothing$	$\varnothing$
$\{C\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\{D\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{E\}$
$\{E\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

■■■ 解説 ■■■

(1)

(1-1)

(1-1-1)

プログラム Q が関数 a を 1 回以上呼び出すということは，$P(x)$ が少なくとも 1 つの a を含むということである．よって，a を 1 つ以上含む語を受理する，状態数 2 のオートマトンを構成すればよい．

このようなオートマトンの遷移図を考えると図 10 のようになる．これを遷移表に直すと表 5 のようになる．

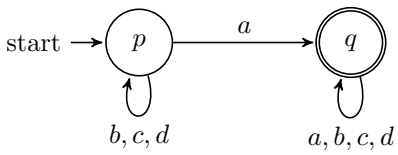


図 10　M_1 の遷移図

(1-1-2)

言語 $\overline{L_1}$ は a を 1 つも含まない語の集合で，$\{P(x)|x \in \mathbf{N}\}$ はプログラム Q の実行における関数 a〜d の呼び出し順序の全パターンを網羅したものである．これらの共通部分が空でないというこ

表 5　M_1 の遷移表

	a	b	c	d
$\to p$	q	p	p	p
$^*q$	q	q	q	q

とは，プログラム Q を実行したときに，関数 a を一度も呼び出さないような引数 x が存在するということを意味する．

(1-2)

(1-2-1)

正則表現 $(ab + cd)^*$ を言葉で説明すればよい．

(1-2-2)

言語 $\overline{L_2}$ は L_2 の補集合である．よって，L_2 に含まれる語は $\overline{L_2}$ には含まれず，逆に L_2 に含まれない語はすべて $\overline{L_2}$ 含まれる．

このような言語を受理するオートマトンを構成するには，まず言語 L_2 を受理するオートマトンを構成し，その受理状態を非受理状態に，非受理状態を受理状態にすればよい．

というわけで，まず L_2 を受理する ε-NFA を構成する．これは図 11 のようになる．

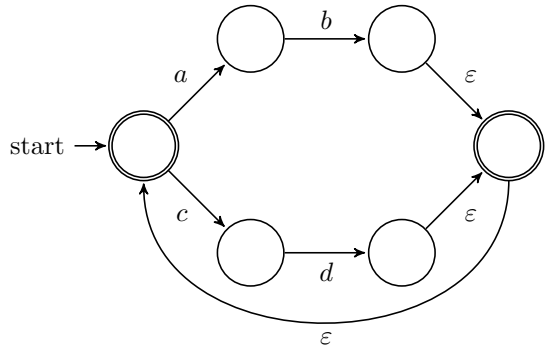


図 11　L_2 を受理する ε-NFA

次に，この ε-NFA から ε-遷移を除去する．ε-遷移を除去した NFA を図 12 に示す．

サブセット構成法を用いて，この NFA を DFA に変換する．変換結果を表 6 に示す．

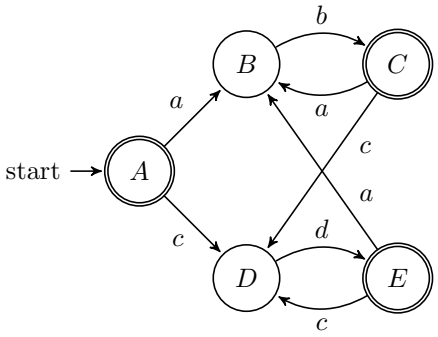


図 12　L_2 を受理する NFA

表 6　L_2 を受理する DFA

	a	b	c	d
$\to\ ^*\{A\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$\{B\}$	$\varnothing$	$\{C\}$	$\varnothing$	$\varnothing$
$^*\{C\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$\{D\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{E\}$
$^*\{E\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

最後に，この遷移表の受理・非受理を入れ替える．その結果を表 7 に示す．これが求める M_2 の遷移表である[*2]．

表 7　M_2 の遷移表

	a	b	c	d
$\to \{A\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\{B\}$	$\varnothing$	$\{C\}$	$\varnothing$	$\varnothing$
$\{C\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\{D\}$	$\varnothing$	$\varnothing$	$\varnothing$	$\{E\}$
$\{E\}$	$\{B\}$	$\varnothing$	$\{D\}$	$\varnothing$
$^*\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

[*2] この DFA は最小ではないが，この問題では最小の DFA を答える必要はないため，このままでよい．また，これと異なる DFA でも，等価な言語を受理するのであれば正解である．等価かどうかは，2 つの DFA の初期状態が同値（区別不能）であるかどうかで判定できる．

(2)

(2-2)

$A_0, A_1, A_2, \ldots, A_k$ は V の元である．よって，$\{A_0, A_1, A_2, \ldots, A_k\}$ から V への対応関係（写像）を考えなければならない．このとき，写像の定義域（$\{A_0, A_1, A_2, \ldots, A_k\}$）の要素数が値域（$V$）の要素数より多ければ必ず重複が発生する（単射でなくなる）．よって，求める不等式は以下のようになる．

$$k+1 > |V| \Longleftrightarrow k \geq |V| (\because k \in \mathbf{N})$$

(2-3)

$$|z| \geq n \Longrightarrow A_0, A_1, A_2, \ldots, A_k \text{に} \\ \text{同一の変数が 2 回以上現れる}$$

という命題を成立させるような n の値を求めればよい．ここで，(2-2) より，

$$k \geq |V| \Longrightarrow A_0, A_1, A_2, \ldots, A_k \text{に} \\ \text{同一の変数が 2 回以上現れる}$$

であることは分かっているから，

$$|z| \geq n \Longrightarrow k \geq |V|$$

となるような n の値を求めれば十分である．

さて，今考えている文法はチョムスキー標準形であるから，次の式が成り立つ[*3]．

$$|z| \leq 2^{h-1}$$

ただし，h は z の構文木の高さである．ここでは，$h = k+1$ であるから，

$$|z| \leq 2^k \Longleftrightarrow \log_2 |z| \leq k$$

となる．また，$|z| \geq n$ より，$\log_2 |z| \geq \log_2 n$ であるから，

$$\log_2 n \leq \log_2 |z| \leq k$$

である．よって，$n = 2^{|V|}$ とすれば，

$$\log_2 n = |V| \leq \log_2 |z| \leq k$$

となる．つまり，

$$|z| \geq 2^{|V|} \Longrightarrow k \geq |V|$$

であるから，$n = 2^{|V|}$ が求める n の値の一つである．

[*3] h に関する帰納法で証明可能．

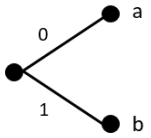
5 ネットワーク

■■■ 解答 ■■■

(1)

(1-1)

(1-1-1) **復号木：**

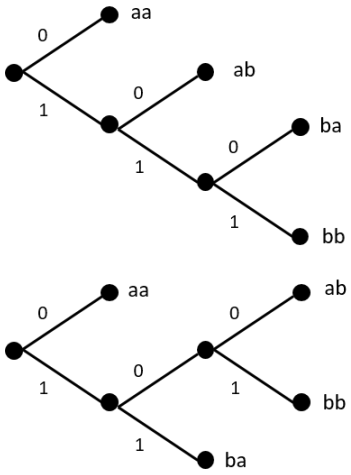


平均符号長：1

(1-1-2) **生起確率：**

$p(aa) = 0.64$
$p(ab) = 0.16$
$p(ba) = 0.16$
$p(bb) = 0.04$

復号木：



平均符号長：0.78

(1-2) (あ) $nH(S)$　(い) $\frac{1}{n}$　(う) $\frac{1}{\delta}$

(2)

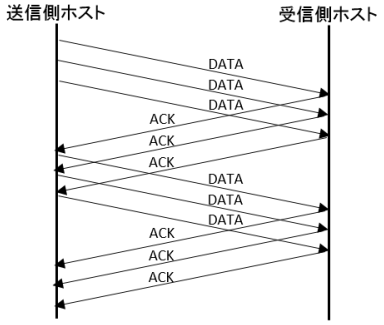
(2-1) (あ) (i) メモリサイズ　(い) (l) 処理速度
(う) (b) データ転送速度　(え) (c) 過負荷
(お) (d) 転送遅延　(か) (h) 喪失

(2-2)

(2-2-1) $\frac{8P}{T}$

(2-2-2) 確認応答セグメントを受け取るまで次のセグメントを送信できないため，スループットが低下する．

(2-2-3)



(2-3) 再送タイマの設定値が大きすぎる場合，セグメントが喪失したときに再送するまでの時間が長くなるため，スループットが低下する．
再送タイマの設定値が小さすぎる場合，セグメントが喪失していないのにもかかわらず再送してしまう場合が発生し，ネットワーク上に同じセグメントがあふれる．その結果，ネットワークが過負荷になり輻輳(ふくそう)が発生し，セグメントの喪失や転送遅延が発生する．
TCP の一般的な実装では，タイマの設定値はラウンドトリップ時間の平均および分散をもとに決定される．ただし，計算資源の制約のため，移動平均を用いた簡易な計算法が用いられる．

■■■ 解説 ■■■

(1)

(1-1)

(1-1-1)

情報源記号の数，通信路記号の数がともに 2 であるため，復号木は 1 通りしかない．このとき，平均符号語長は $0.8 \times 1 + 0.2 \times 1 = 1$ となる．

(1-1-2)

記憶のない情報源なので，$p(aa) = p(a)p(a) = 0.64$．他も同様に求めることができる．

このときの復号木は，初めに ab と bb を縮約するか，ba と bb を縮約するかの 2 通りがある．

(1-2)

X, Y が独立であるとき，

$$H(XY) = H(X) + H(Y)$$

が成り立つ．よって，

$$H(S^n) = nH(S)$$

である．

シャノンの情報源符号化定理は，

$$H(S) \leq l_1 < H(S) + 1$$

を満たす符号化が存在することを示したものである．n 次拡大を考えると

$$nH(S) \leq nl_n < nH(S) + 1$$

となり，これを n で割ると，

$$H(S) \leq l_n < H(S) + \frac{1}{n}$$

となる．

$l_n < l_{n_0}$ となるためには，

$$l_n < H(S) + \frac{1}{n}$$

より

$$H(S) + \frac{1}{n} \leq l_{n_0}$$

となればよい．ここで，

$$l_{n_0} = H(S) + \delta$$

より

$$H(S) + \frac{1}{n} \leq H(S) + \delta$$

である．これを解いて，

$$n \leq \frac{1}{\delta}$$

(2)

(2-1)

教科書や授業資料をよく確認しましょう．

(2-2)

(2-2-1)

1 度の送信で P [バイト]，つまり $8 \times P$ [ビット] のデータを送ることができ，それにかかる時間は T [秒] なので，$\frac{8P}{T}$ [ビット/秒] である．

(2-2-3)

スライディングウィンドウ制御では，ウィンドウサイズまでは確認応答セグメントを受け取るのを待たずに次のセグメントを送信する．よって，今回は 3 つのセグメントを確認応答なしに送信できる．

6　電子回路と論理設計

■■■ 解答 ■■■

(1) 表 8 を参照.

(2)
$$y = Q_0 + Q_2X$$
$$Q_2^+ = Q_0\overline{X} + Q_1Q_0$$
$$Q_1^+ = Q_1\overline{Q_0X} + Q_2\overline{Q_0}$$
$$Q_0^+ = \overline{Q_0}X + Q_1X$$

(3) CL0：3　CL1：4　CL2：3　CLY：2

(4) A S_2　B S_5　C S_3　D -　E S_4　F -

(5)
$$y = Q_1\overline{Q_0} + Q_0X + \overline{Q_1}Q_0$$
$$Q_1^+ = \overline{Q_1}Q_0\overline{X} + Q_1Q_0X + Q_1\overline{Q_0X}$$
$$Q_0^+ = Q_1\overline{Q_0} + \overline{Q_0}X + \overline{Q_1}Q_0\overline{X}$$

(6) (2) の順序回路：59　(5) の順序回路：55

(7) 状態数を減らすことにより，y や Q_1^+ などの最簡積和形が複雑化する．状態数を減らすことによって削減できる D フリップフロップなどの記憶回路のコストよりも，最簡積和形が複雑化したことによって増加する組み合わせ回路のコストが勝る場合があるため.

表 8　(1) の状態遷移出力表

状態	0	1
S_0	$S_0, 0$	$S_1, 0$
S_1	$S_4, 1$	$S_0, 1$
S_2	$S_2, 0$	$S_1, 0$
S_3	$S_4, 1$	$S_5, 1$
S_4	$S_2, 0$	$S_3, 1$
S_5	$S_4, 1$	$S_0, 1$

■■■ 解説 ■■■

(1)

入出力があることに注意.

状態遷移出力表の記述は，意味さえ伝われば書き方にこだわる必要はないようです．解答例のように状態と出力を同じ枠に書くのもアリ，出力を別枠で用意するのもアリ.

(2)

カルノー図さえ描ければ大丈夫です．本問におけるカルノー図を，図 13〜図 16 に示します.

y

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	d	d	d	d
10	0	1	1	1

図 13　y のカルノー図

Q_2^+

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	0	1
01	0	0	1	1
11	d	d	d	d
10	0	0	0	1

図 14　Q_2^+ のカルノー図

Q_1^+

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	d	d	d	d
10	1	1	0	0

図 15　Q_1^+ のカルノー図

Q_0^+

Q_2Q_1 \ Q_0X	00	01	11	10
00	0	1	0	0
01	0	1	1	0
11	d	d	d	d
10	0	1	0	0

図 16　Q_0^+ のカルノー図

(3)

$\overline{\overline{A}} = A$，および $A + B = \overline{\overline{A + B}} = \overline{\overline{A} \cdot \overline{B}}$（ド・モルガンの法則による変形公式）を用いると，NAND ゲートの組み合わせのみで実現することができます．この手法は過去問でも頻繁に出てきますので，覚え

ておくと吉です.

(4)

考え方:「入力 0 および 1 における，それぞれの遷移先が同一もしくは互いに遷移する」場合は共通化可能.

- S_0 と S_2 は，それぞれ 0 を入力すると互いに向かって，1 を入力すると S_1 へ遷移するため共通化可能
- S_1 と S_5 は，それぞれ 0 を入力すると S_4 へ，1 を入力すると S_0 へ遷移するため共通化可能
- S_3 および S_4 は，共通化可能でないのでそのまま

(5)

カルノー図を描きましょう．本問におけるカルノー図を，図 17〜図 19 に示します.

y	Q_0X 00	01	11	10
Q_1 0	0	0	1	1
Q_1 1	1	1	1	0

図 17　y のカルノー図

Q_1^+	Q_0X 00	01	11	10
Q_1 0	0	0	0	1
Q_1 1	1	0	1	0

図 18　Q_1^+ のカルノー図

Q_0^+	Q_0X 00	01	11	10
Q_1 0	0	1	0	1
Q_1 1	1	1	0	0

図 19　Q_0^+ のカルノー図

(6)

(5) においては，Q_2 を必要としないために D フリップフロップが一つ減っていることに注意しましょう.

(2) の順序回路：

2 入力 NAND ゲートが 10 個, 3 入力 NAND ゲートが 1 個，D フリップフロップが 3 個なので，コストは $2 \times 10 + 3 \times 1 + 12 \times 3 = 59$ となる.

(5) の順序回路：

2 入力 NAND ゲートが 5 個，3 入力 NAND ゲートが 7 個，D フリップフロップが 2 個なので，コストは $2 \times 5 + 3 \times 7 + 12 \times 2 = 55$ となる.

(7)

(6) を解いている際に，状態数が減っている (5) においてそこまでコストが削減されていないことに気がつくと，その原因として 3 入力 NAND ゲートの必要数の増加を発見し，最簡積和形の複雑化によるコストの増大化へと辿りつくかと思います.

7　数学解析と信号処理

略.