大阪大学大学院情報科学研究科
令和 2 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：市川直人，出田涼子，藤本章良，前島葵

2019 年 9 月 9 日

1　アルゴリズムとプログラミング

解答

(1) ヒープソート
(2) 図 1 参照．
(3) 節点番号が `current` の節点のデータが，その子のデータ以上の値となる．
(4) 最悪時間計算量：$O(n \log n)$
理由：解説参照．
(5)
(5-1) (あ) `n / 2 - 1`　(い) `d`
(う) `n`　(え) `i`
(5-2) $T(n)$：$O(n)$
理由：解説参照．
(6) (ア)：`child + 1 < n` （変更なし）
(イ)：`d[child] > d[child + 1]`
(ウ)：`d[current] > d[child]`
(エ)：`d[parent] > d[current]`

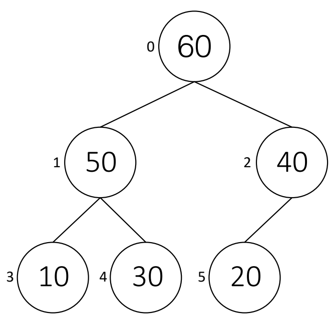

図 1　(2) の図

解説

(1)

二分木という言葉があるので 2 分探索木のように思えるが，プログラムをよく見てみると，常に親が子供より大きくなるように木を構成している．よって，これはヒープソートである．

(2)

関数 `uph` では，節点番号が `k` である節点を必要に応じて親と入れ替えることで，親のデータが子のデータ以上になるようにしている．つまり，親より大きいデータを格納している節点が上に移動していく．

初期状態は問題文の図 3 に示されているので，上記の考え方で節点番号 1 から順番に木を変更すると図 1 が得られる．

(3)

11 行目によって，節点番号 `current` の子のうち，データが大きい方の節点番号が変数 `child` に格納される．12 行目では，節点番号 `current` のデータが節点番号 `child` のデータよりも小さい時に，その 2 つのデータが入れ替えられる．よって，11 行目と 12 行目が実行されることにより，節点番号 `current` とその子のデータのうち，一番大きいデータが節点番号 `current` に格納される．

(4)

関数 sort で実現されている整列アルゴリズムは2つの過程からなる．1つ目が28行目で実現されている，ルールを満たすようにヒープを整形する過程である．2つ目が29行目で実現されている，ヒープから順番に最大要素を取り出していく過程である．

まず，28行目の最悪時間計算量を考える．外側の for ループは $n-1$ 回処理を行う．また，内側の while ループは最悪の場合，現在の節点が根になるまで入れ替え続けるので，各 i の高さである $\lfloor \log_2 i \rfloor$ 回処理が実行される．よって，最悪時間計算量は $O(n \log n)$ となる．

次に，29行目の最悪時間計算量を考える．外側の for ループは $n-1$ 回処理を行う．また，内側の while ループは最悪の場合，現在の根が葉になるまで入れ替え続けるので，その時点の葉の高さである $\lfloor \log_2 i \rfloor$ 回処理が実行される．よって，最悪時間計算量は $O(n \log n)$ となる．

28行目と29行目は独立なので，関数 sort で実現されているアルゴリズムの最悪時間計算量は $O(n \log n)$ である．

(5)

(5-1)

変更前のプログラムでは，整形対象の要素を1つ追加するたびに，親子に関するヒープの制約を完全に満たすようにヒープを整形した．しかし，あくまでも28行目を実行し終わった時にヒープの制約が満たされていればいい．なので，ヒープを下から見ていき，節点番号が大きい方から順次整形していけばいい．(詳しくは教科書「アルゴリズム論」5.5節ヒープソートの最後の数段落を参照)

for ループのすでに埋まっている部分から，i は1ずつ小さくなりながら0まで変化することが分かる．子を持つ全ての節点を確認し，必要に応じて，子より小さいデータを格納している節点を下に移動させることを考えているため，i は節点番号を表し，(あ) には子を持つ中で最大の節点番号を入れればいいことが分かる．

(い) は関数 downh に与える配列であり，他の配列が登場しないため，d が入る．

ヒープを整形する段階では全ての節点を見ているので，ヒープの最後の要素の添字を示す (う) には n が入る．

(え) は確認を始めたいデータの節点番号なので，i が入る．

(5-2)

配列 d の要素を子の個数で分類すると表1のようになる．

表1　配列 d の要素の分類

番号	添字	説明	swap 回数	個数
①	$n-1 \sim \frac{n}{2}$	葉	最大0回	$\frac{n}{2}$
②	$\frac{n}{2}-1 \sim \frac{n}{4}$	①の親	最大1回	$\frac{n}{4}$
③	$\frac{n}{4}-1 \sim \frac{n}{8}$	②の親	最大2回	$\frac{n}{8}$
		$\vdots$		

よって，

$$
\begin{aligned}
T(n) &= 0 \times \frac{n}{2} + 1 \times \frac{n}{4} + \cdots + (\log_2 n - 1) \times \frac{n}{2^{\log_2 n}} \\
&= \frac{n}{2} \sum_{i=0}^{\log_2 n - 1} \frac{i}{2^i} \\
&= \frac{n}{2} \times \left(2 - \frac{2 + \log_2 n - 1}{2^{\log_2 n - 1}}\right) \\
&= n - \log_2 n - 1 \\
&\in O(n)
\end{aligned}
$$

(6)

ヒープの根が最小値を格納するように変更すれば良い．このためには，親のデータが子のデータより小さくなるようにすれば良い．よって，データの比較に関係する条件式の不等号を逆向きにすれば良い．

■■■ 所感 ■■■

関連する授業はおおよそ次の通り．

- データ構造とアルゴリズム (2年後期) - 5.5節

2　計算機システムとシステムプログラム

解答

(1)

(1-1) (a) エ　(b) イ　(c) ウ
(d) ア　(e) ク

(1-2)

(1-2-1) 255.75

(1-2-2) 0.015625

(1-2-3) 1 1100 001001010

(1-2-4) 切り捨てた分がそのまま本来の値からの誤差となるため，本来等しいはずの値が異なると判定される．また，繰り返すたびに誤差が大きくなるため，とても大きな誤差になる．

(2)

(2-1) (a) キ　(b) カ　(c) オ
(d) ア　(e) コ　(f) ソ
(g) シ　(h) ウ　(i) イ

(2-2) LRU：○○○×○○○×○×○×○
FIFO：○○○×○×○×××○○×

(2-3) 2.5×10^{-5}

(2-4) ページ枠を大きくする．

解説

(1)

(1-1)

演算に関して，固定小数点の加減算は，通常の整数と同じように行える．一方，浮動小数点演算は，桁合わせ等が必要になるため固定小数点よりも演算が複雑になる．

また余談であるが，問題にもあるように 2 進数では 10 進数の 0.1 を誤差なく表現できない．したがって

```
(0.1 + 0.2 == 0.3) => false
```

と判定されることがあるので，小数を扱うコードを書く場合は気を付けたい．

(1-2)

(1-2-1)

無限大を除く正の最大値は次の表現で表される数である．

$$
\begin{aligned}
&(-1)^0 \times 2^{14-7} \times (1.111111111) \\
&= 2^7 \times 1.111111111 \\
&= 11111111.11
\end{aligned}
$$

(1-2-2)

正の最小値は次の表現で表される数である．

$$
\begin{aligned}
&(-1)^0 \times 2^{1-7} \times (1.000000000) \\
&= 2^{-6} \times 1.000000000 \\
&= 0.000001
\end{aligned}
$$

(1-2-3)

36.66 を 2 進数で表現すると $100100.1010\ldots$ となる．よって，-36.66 を浮動小数点表現にすると次のようになる．

$$(-1)^1 \times 2^{12-7} \times (1.001001010\ldots)$$

(1-2-4)

省略．

(2)

(2-1)

マッピング方式は次の 2 つが代表的である．

ページング　固定長．マッピングが簡単でメモリ使用効率良好．内部断片化が発生する．

セグメンテーション　可変長．論理的まとまりを活用できる．外部断片化が発生．

実際には，双方の長所を利用できる“ページセグメンテーション方式”が採用されていることが多い．

(2-2)

LRU と FIFO でのページ枠の内容の遷移はそれぞれ表 2，表 3 のようになる．

表 2　LRU でのページ枠

ページ参照列	0	1	2	0	3	1	4	3	2	3	1	2	4
ページフォルト	◯	◯	◯	×	◯	◯	◯	×	◯	×	◯	×	◯
ページ枠の内容	0	1	2	0	3	1	4	3	2	3	1	2	4
		0	1	2	0	3	1	4	3	2	3	1	2
			0	1	2	0	3	1	4	4	2	3	1

表 3　FIFO でのページ枠

ページ参照列	0	1	2	0	3	1	4	3	2	3	1	2	4
ページフォルト	◯	◯	◯	×	◯	×	◯	×	×	×	◯	◯	×
ページ枠の内容	0	0	0	0	1	1	2	2	2	2	3	4	4
		1	1	1	2	2	3	3	3	3	4	1	1
			2	2	3	3	4	4	4	4	1	2	2

(2-3)

単位に注意する．主記憶へのアクセスは 2 **マイクロ** ($=10^{-6}$) 秒，ページフォルトのオーバーヘッドは 8 **ミリ** ($=10^{-3}$) 秒である．

ページフォールトが発生しない場合の 1 命令あたりのメモリアクセス時間 T_m は次のようになる．

$$T_m = 2 \times 2 \times 10^{-6}[s]$$

ページフォールトの確率を P とおく．この時，1 命令あたりのページフォールトを考慮したメモリアクセス時間 T_e は次のようになる．

$$T_e = 2 \times (2 \times 10^{-6} + P \times 8 \times 10^{-3})[s]$$

性能低下係数 α は $\alpha = T_e/T_m$ で表すことができる．性能低下を平均 10% 以下に抑えるということは $\alpha \leq 1.1$ である．

よって，

$$\frac{2 \times (2 \times 10^{-6} + P \times 8 \times 10^{-3})}{2 \times 2 \times 10^{-6}} \leq 1.1$$

$$\frac{10^{-3} + 4P}{10^{-3}} \leq 1.1$$

$$4P \leq 0.1 \times 10^{-3}$$

$$P \leq 2.5 \times 10^{-5}$$

(2-4)

「ページ枠を大きくする＝主記憶の容量を大きくする」

また，ページサイズを大きくするという方法も考えられる．例えば，ページサイズを 3 倍にすれば，ページ番号 0-2 と 3-5 でまとめられ，LRU を用いるとページフォールト回数は 8 回になる．ただしこの方法はページフォールト時のページ入れ替え時間が増加するという欠点がある．

■■■ 所感 ■■■

関連する授業はおおよそ次の通り．

(1) ディジタル回路（2 年後期） - 5.3 節

(2) オペレーティングシステム（3 年前期） - 3.2 節

(2-3) の解説作成には，明星大学の講義資料 (`http://www.hino.meisei-u.ac.jp/is/iga/lecture/os/No7org.pdf`) を参考にした．

3 離散構造

解答

(1)

(1-1) C：どのような状態においても，ロボットが台に乗っていないならば，任意の位置 x に台を移動させ設置することができ，移動させ設置した状態に遷移する．

D：任意の状態 s からロボットが台に乗った状態 s' では，ロボットは台に乗っている．

(1-2)

(1-2-1) $E = \exists s\ have(s)$

(1-2-2) $\neg F_p =$

$$\forall s\ \forall x((\neg sl(s,x) \lor sl(climb(s),x)))$$
$$\land (\neg sl(s,goal) \lor \neg on(s)$$
$$\lor\ have(grasp(s)))$$
$$\land (on(s) \lor sl(move(s,x),x))$$
$$\land\ on(climb(s))$$
$$\land \neg have(s))$$

(1-2-3) $I = \neg sl(s_0, goal) \land \neg on(s_0)$

(1-2-4) $grasp(climb(move(s_0, goal)))$

(2)

(2-1) G_1：連結

G_2：辺 (b,e) を取り除いたら連結でなくなる

(2-2)

(2-2-1) 解説参照．

(2-2-2) 解説参照．

解説

(1)

(1-1)

A と B を参考にして，C と D に用いられている関数および述語の説明をうまくまとめる．

(1-2)

(1-2-1)

「状態 s においてロボットがバッテリーを取得している」ことを表す述語は $have(s)$ なので，それと存在作用素 $\exists$ を組み合わせる．

(1-2-2)

$$\begin{aligned}
F &= (A \land B \land C \land D) \to E \\
\neg F &= A \land B \land C \land D \land \neg E \\
&= \forall s\ \forall x(sl(s,x) \to sl(climb(s),x)) \\
&\quad \land \forall s((sl(s,goal) \land on(s)) \to have(grasp(s))) \\
&\quad \land \forall s\ \forall x(\neg on(s) \to sl(move(s,x),x)) \\
&\quad \land \forall s\ on(climb(s)) \\
&\quad \land \neg(\exists s\ have(s)) \\
&= \forall s\ \forall x(\neg sl(s,x) \lor sl(climb(s),x)) \\
&\quad \land \forall s(\neg(sl(s,goal) \land on(s)) \lor have(grasp(s))) \\
&\quad \land \forall s\ \forall x(\neg\neg on(s) \lor sl(move(s,x),x)) \\
&\quad \land \forall s\ on(climb(s)) \\
&\quad \land \neg(\exists s\ have(s)) \\
&= \forall s\ \forall x(\neg sl(s,x) \lor sl(climb(s),x)) \\
&\quad \land \forall s(\neg sl(s,goal) \lor \neg on(s) \lor have(grasp(s))) \\
&\quad \land \forall s\ \forall x(on(s) \lor sl(move(s,x),x)) \\
&\quad \land \forall s\ on(climb(s)) \\
&\quad \land \forall s\ \neg have(s) \\
&= \forall s(\forall x(\neg sl(s,x) \lor sl(climb(s),x)) \\
&\quad \land (\neg sl(s,goal) \lor \neg on(s) \lor have(grasp(s))) \\
&\quad \land \forall x(on(s) \lor sl(move(s,x),x)) \\
&\quad \land on(climb(s)) \\
&\quad \land \neg have(s)) \\
&= \forall s\ \forall x((\neg sl(s,x) \lor sl(climb(s),x)) \\
&\quad \land (\neg sl(s,goal) \lor \neg on(s) \lor have(grasp(s))) \\
&\quad \land (on(s) \lor sl(move(s,x),x)) \\
&\quad \land on(climb(s)) \\
&\quad \land \neg have(s)) = \neg F_p
\end{aligned}$$

(1-2-3)

台の初期位置は $goal$ でないことを表す論理式は $\neg sl(s_0, goal)$ となる．

また，ロボットが台を位置 $goal$ に移動させるためには，初期状態ではロボットが台に乗っていないことが必要となる．すなわち，$\neg on(s_0)$ である．

この 2 つの条件は初期状態でともに成り立っている必要があるため，

$$I = \neg sl(s_0, goal) \land \neg on(s_0)$$

(1-2-4)

修正後の $\neg F$ の冠頭標準形 $\neg F'_p$ は

$$
\begin{aligned}
\neg F'_p = \forall s\ \forall x((&\neg sl(s,x) \lor sl(climb(s),x)) \\
&\land (\neg sl(s,goal) \lor \neg on(s) \lor have(grasp(s))) \\
&\land (on(s) \lor sl(move(s,x),x)) \\
&\land on(climb(s)) \\
&\land \neg sl(s_0,goal) \land \neg on(s_0) \\
&\land \neg have(s))
\end{aligned}
$$

と表せる．これは存在作用素 ∃ を含まないので，導出原理を用いる際にスコーレム化を行う必要はない．導出原理を用いて，図 2 のように空節が導かれる．

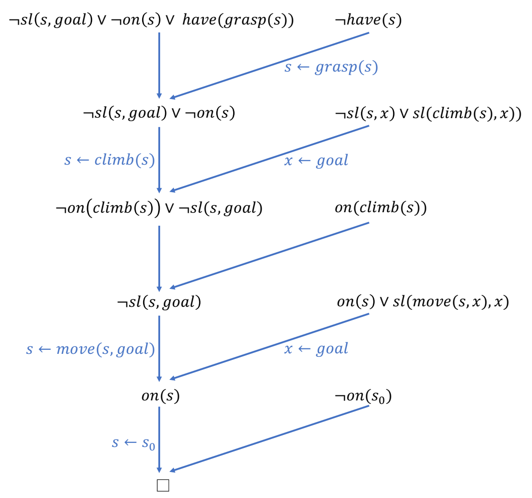

図 2　(1-2-4) の導出原理

この時，論理式 E に該当する節の変数への代入は，図 2 右上の $\neg have(s)$ への代入なので，上から順に

$$s \leftarrow grasp(s)$$

$$s \leftarrow climb(s)$$

$$s \leftarrow move(s, goal)$$

$$s \leftarrow s_0$$

である．よって，代入された項は

$$grasp(climb(move(s_0, goal)))$$

である．

(2)

(2-1)

省略．

(2-2)

(2-2-1)

G' が連結でないと仮定する．すなわち，G から辺 (u,v) を取り除いたことで，G' は 2 つの連結部分に分かれた．

ここで，G' の頂点 u を含む連結部分は，u 以外の頂点の次数は偶数のままで，u のみが奇数次数となる．これは問題文の補題に矛盾する．よって，背理法により，G' は連結であり，頂点 v から頂点 u への経路を持つことが示された．

(2-2-2)

(2-2-1) は成り立つので，G からある辺 (u,v) を取り除いて得られる G' は頂点 v から頂点 u への経路を持つ．この経路を $(v,a_1),\dots,(a_n,u)$ とする．$a_i = a_j (i < j)$ であれば，$(v,a_1),\dots,(a_i,a_{j+1}),\dots,(a_n,u)$ という経路も存在する．この操作を繰り返すことによって，各辺を高々一つだけ含む G' 上の頂点 v から頂点 u への経路 $(v,a_1),\dots,(a_m,u)$ が得られる．

G' 上にあって G 上にない辺はないため，この経路は G 上にも存在する．よって，G は閉路 $(u,v),(v,a_1),\dots,(a_m,u)$ を持つ．経路 $(v,a_1),\dots,(a_m,u)$ は辺 (u,v) を除く各辺を高々一つだけ含む．よって，この閉路は各辺を高々一つだけ含む．

■■■ 所感 ■■■

関連する授業はおおよそ次の通り．

(1) 情報論理学（3 年前期） - 5 章

(2) 情報数学基礎（2 年前期） - 第 6 回

4 計算理論

解答

(1)

(1-1) (A) $(a, 0)/00$

(B) $(\varepsilon, 0)/0$ もしくは $(b, 0)/\varepsilon$

(C) $(b, 0)/\varepsilon$

(1-2)

(1-2-1) L_2：図 3 参照.

L_3：図 4 参照.

(1-2-2) 解説参照.

(1-3) 図 5 参照.

(2)

(2-1) 表 4 参照.

(2-2) 解説参照.

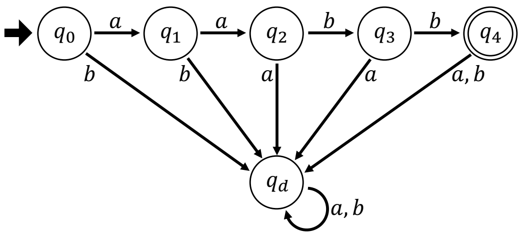

図 3 L_2 を認識するオートマトン

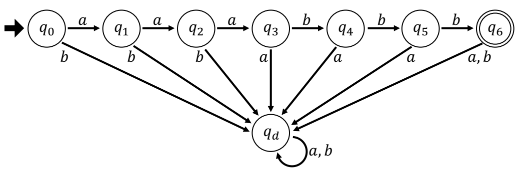

図 4 L_3 を認識するオートマトン

表 4 (2-1) の表

$M[i,j]$	$j=1$	$j=2$	$j=3$	$j=4$
$i=1$	$\{A, C\}$	$\{A\}$	$\{A\}$	$\{S\}$
$i=2$	–	$\{A, C\}$	$\{A\}$	$\{S\}$
$i=3$	–	–	$\{A, C\}$	$\{S\}$
$i=4$	–	–	–	$\{B, D\}$

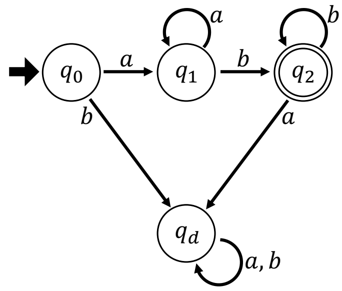

図 5 L'_{ab} を認識するオートマトン

解説

(1)

(1-1)

すでに q_0 での動作 $(a, Z)/0Z$ が与えられているため，このオートマトンは a の数だけ 0 をスタックに積むことがうかがえる．それならば，b を読み込むたびにスタックから 0 をポップし，入力列の終了とともにスタックから 0 がなくなった時にのみ最終状態に遷移するようにすれば良い．

(1-2)

(1-2-1)

省略．ドボン状態のループ，開始状態の矢印，最終状態の二重丸などをきちんとかけているか要確認.

(1-2-2)

異なる状態は全部で k 個しかないので，鳩の巣原理により，$k+1$ 個の $p_i (i = 0, 1, \ldots, k)$ が全て相異なるということはあり得ない．したがって，二つの異なる整数 i と $j (0 \leq i < j \leq k)$ で $p_i = p_j$ を満たすものが存在する．

このオートマトンは a^i と a^j を区別することができない．つまり，$a^k b^k$ を受理する時，$a^{k-(j-i)} b^k$ も受理してしまう．しかし，$i < j$ より $k-(j-i) \neq k$ なので，$a^{k-(j-i)} b^k$ は言語 L_{ab} に属さない．

(1-3)

省略．(1-2-1) 同様，ドボン状態のループ，開始状態の矢印，最終状態の二重丸などをきちんとかけているか要確認.

(2)

(2-1)

省略.

(2-2)

文字列 ba は次の導出によって得られるので，G_2 によって生成される．よって，出力されるべき判定は Yes である．

$$S \to SA \to BSA \to BA \to bA \to ba$$

与えられたアルゴリズムの終了時における $M[i,j]$ の内容は以下の表のとおりである．

$M[i,j]$	$j=1$	$j=2$
$i=1$	$\{B\}$	$\emptyset$
$i=2$	–	$\{A\}$

このとき，$S \notin M[1,2]$ であるため No が出力される．

所感

この大問は全て計算論 A（3 年前期）の範囲である．特に (2) は CYK アルゴリズムである．

5　ネットワーク

略.

6　電子回路と論理設計

解答

(1) (未回答)

(1-1)

(1-2)

(1-3)

(2)(2-1) 図 6 参照.

(2-2)

$$
\begin{aligned}
{k_0}^+ &= k_0 \wedge \neg x_0 \\
&\quad \vee \neg k_1 \wedge x_0 \\
{k_1}^+ &= k_1 \wedge \neg x_0 \\
&\quad \vee \neg k_0 \wedge \neg k_1 \wedge x_0 \\
y_0 &= k_0 \wedge x_0
\end{aligned}
$$

(2-3) 図 8 参照.

(2-4) 機能:返金機能

理由: $x_1 = 0$ の場合は変更前と同じであり、$x_1 = 1$ の場合はすべて状態 A に遷移するため

解説

(1) 省略.

(2)

(2-1)

凡例:x_0/y_0

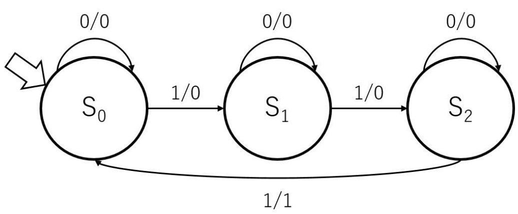

図 6　(2-1) 状態遷移図

k_0^+

x_0 \ $k_0 k_1$	00	01	11	10
0			X	1
1		1	X	

k_1^+

x_0 \ $k_0 k_1$	00	01	11	10
0		1	X	
1	1		X	

y_0

x_0 \ $k_0 k_1$	00	01	11	10
0			X	
1			X	1

図 7　(2-2) ${k_0}^+$,${k_1}^+$,y_0 のカルノー図

(2-2)

${k_0}^+$，${k_1}^+$，y_0 のカルノー図を図 7 に示す.

カルノー図より,

$$
\begin{aligned}
{k_0}^+ &= k_0 \wedge \neg x_0 \\
&\quad \vee \neg k_1 \wedge x_0 \\
{k_1}^+ &= k_1 \wedge \neg x_0 \\
&\quad \vee \neg k_0 \wedge \neg k_1 \wedge x_0 \\
y_0 &= k_0 \wedge x_0
\end{aligned}
$$

(2-3)

回路図から ${k_0}^+$，${k_1}^+$，y_0，y_1 の論理式は以下のようになる.

$$
\begin{aligned}
{k_0}^+ &= k_0 \wedge \neg x_0 \wedge \neg x_1 \\
&\quad \vee \neg k_1 \wedge x_0 \wedge \neg x_1 \\
{k_1}^+ &= k_1 \wedge \neg x_0 \wedge \neg x_1 \\
&\quad \vee \neg k_0 \wedge \neg k_1 \wedge x_0 \wedge \neg x_1 \\
y_0 &= k_0 \wedge x_0 \wedge \neg x_1 \\
y_1 &= x_1
\end{aligned}
$$

(2-2) の論理式とくらべると，$x_1 = 1$ の場合は変更前と同様である.

凡例:$x_0 x_1/y_0 y_1$

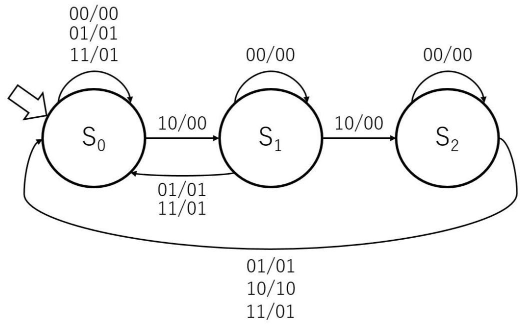

図 8　(2-3) 状態遷移図

(2-4)

省略.

■■■ 所感 ■■■

大問 1 が電子回路からの出題．例年の出題傾向とは大きく異なり，多くの受験生は別の科目を選択した．筆者は選択問題を 4,6 に絞って勉強していたため，大問 1 を白紙で提出した．大問 2 に関しては自動販売機の制御回路という比較的簡単な内容だが,回路図から状態遷移図を求めるというあまり見ない問題も含まれた.

7　数学解析と信号処理

略.