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

2019 年 10 月 1 日

1　アルゴリズムとプログラミング

(1)
ヒープソート

(2)

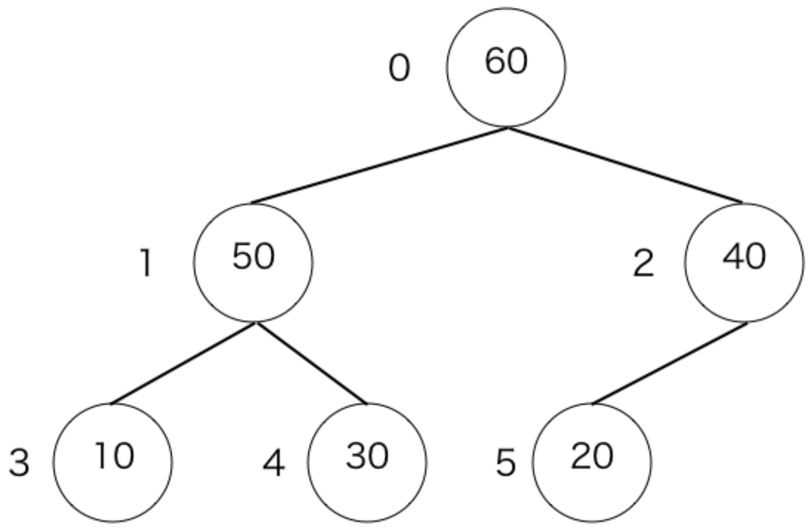

図 1　(2) の解答

(3)
親と存在すればその 2 つの子ノードの 3 つの中で、最大の値を持つものが親となる。

(4)
28 行目で uph という関数を n 回呼び出しており、この関数では最大で木の高さの回数だけ交換を行う。ここで木の高さは $\log_2 n$ で表されるため、最悪時間計算量は $O(n \log n)$ となる。

(5)
(5-1)
(あ)n/2-1
(い)d
(う)n
(え)i

(5-2)

簡単のためここでは木の高さを k として、ノード数を $n = 2^{k+1} - 1$ とする。このとき木の深さ $i(0 \leq i \leq k)$ での交換と比較の回数は高々 $k - i$ ずつで、合計 $2(k - i)$ 回である。また各深さにおけるノードの数は 2^{i-1} で表せる。よって最悪時間計算量は下のように導ける。

$$
\begin{aligned}
\sum_{i=0}^{k} 2(k-i) \cdot 2^{i-1} &= \sum_{h=0}^{k} 2h \cdot 2^{k-h-1} (h = k - i \text{ と置く}) \\
&= 2^k \sum_{h=0}^{k} \frac{h}{2^h} \text{（式変形）} \\
&= 2^k (2 - \frac{2+k}{2^k}) \text{（問題で与えられた式を用いる）} \\
&= 2^{k+1} - 2 - k \text{（式変形）} \\
&= n - log(n+1) - 2 \text{（}n = 2^{k+1} - 1\text{）}
\end{aligned}
$$

従って最悪時間計算量は $O(n)$ となる。

(6)
(ア)child + 1 < n
(イ)d[child] > d[child+1]
(ウ)d[current] > d[child]
(エ)d[parent] > d[child]

2　計算機システムとシステムプログラム

(1)

(1-1)

a) (エ) 表現できる数の範囲が広い

b) (イ) 浮動小数点

c) (ウ) 演算回路（arithmetic circuit）が簡素になる

d) (ア) 固定小数点

e) (ク) 0.1

(1-2)

(1-2-1)

正の数であるため符号部は 0、指数部は e=15 が $+\infty$ に用いられるため e=14 が最大で 1110、仮数部は全て 1 で埋めて 111111111。すると最大値の浮動小数点表記は 01110111111111 となり、隠しビットに注意して 2 進数で表すと 11111111.11 である。そして、これを 10 進数で表すと 255.75 となる。

(1-2-2)

正の数であるため符号部は 0、指数部は e=0 が- ∞ に用いられるため e=1 が最小で 0001、仮数部は全て 0 で埋めて 000000000。すると最小値の浮動小数点表記は 00001000000000 となり、隠しビットに注意して 2 進数で表すと 0.000001 である。そして、これを 10 進数で表すと 0.015625 となる。

(1-2-3)

-36.66 を 2 進数で示すと-100100.10100001... で、これは丸め処理を施して $(-1)^1 \times 2^5 \times (1 + 0.001001010)$ と表せる。この表記から s=1、e=12、m=001001010 であることがわかり、求めるビット列は 11100001001010 である。

(1-2-4)

切り捨ては最接近丸めと比べて丸め誤差の上界が 1 と大きく、誤差が常に同じ符号であるため、丸めた数を繰り返し足したり、繰り返し引いたりすると個数に比例して丸め誤差が累積し、浮動小数点を用いた計算結果と本来の計算結果との差が大きくなる。

(2)

(2-1)

a) (キ) アドレス空間（address space）

b) (カ) 仮想アドレス（virtual address）

c) (オ) 実アドレス（real address）

d) (ア) 主記憶

e) (コ) ページング（paging）

f) (ソ) セグメンテーション（segmentation）

g) (シ) 外部断片化（external fragmentation）

h) (ウ) 内部断片化（internal fragmentation）

i) (イ) ページ（page）

(2-2)

ページ番号	0	1	2	0	3	1	4	3	2	3	1	2	4
ページ枠内容	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

図 2　(2-2-LRU)

ページ番号	0	1	2	0	3	1	4	3	2	3	1	2	4
ページ枠内容	0	0	2	2	3	3	4	4	4	4	1	2	2
		1	1	1	2	2	3	3	3	3	4	1	1
			0	0	1	1	2	2	2	2	3	4	4

図 3　(2-2-FIFO)

(2-3)

ページフォールトが発生しない場合のスループットは $\frac{1}{2\times(2\times10^{-6})}$（命令/秒）、一方ページフォールトが発生する場合のスループットは $\frac{1}{2\times(2\times10^{-6}+P\times8\times10^{-3})}$（命令/秒）。ページフォールトによる性能低下を 10% 以下に抑えるためには以下の式を満たせば良い。

$$\frac{\frac{1}{2\times(2\times10^{-6})}-\frac{1}{2\times(2\times10^{-6}+P\times8\times10^{-3})}}{\frac{1}{2\times(2\times10^{-6})}} \leq 0.1$$

$$P \leq \frac{1}{36}\times10^{-3}$$

このため P の上限は $\frac{1}{36}\times10^{-3}$

(2-4)

時間的局所性を利用できるように、ページ参照列に存在する参照の周期の中で、同じページを参照する間隔までページ枠を増やす。

3 離散構造

(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ければ、ロボットが台をどのような位置に移動させ設置させたとしても台はその位置に設置される。

D：どのような状態においても、ロボットが台に登った状態において、ロボットは台に乗っている。

(1-2)

(1-2-1) $\exists s\, have(s)$

(1-2-2) $\forall s \forall x\, ((\neg sl(s,x) \vee sl(climb(s),x)) \wedge (\neg sl(s,goal) \vee \neg on(s) \vee have(grasp(s))) \wedge (on(s) \vee sl(move(s,x),x)) \wedge on(climb(s)) \wedge \neg have(s))$

(1-2-3) $I = \neg on(s_0)$

初期状態としてロボットが台に乗っていないことを示す必要があるため。

(1-2-4) grasp(climb(move(s0,goal)))

(2)

(2-1) G1：連結である

G2：辺 be を取り除くと連結でなくなる

(2-2)

(2-2-1)

G’が連結でないと仮定する。

つまり、辺(u,v)を取り除くことによってグラフ G は 2 つのグラフ H と I に分けられる。なお、グラフ H,I のそれぞれに頂点 u,v が含まれるとする。

この時、頂点 u に注目すると、辺(u,v)が取り除かれたことにより頂点 u の次数は 1 減っている。よって頂点 u の次数は奇数となっている。

G の各頂点の次数は偶数だったので、H の u 以外の頂点の次数は偶数である。したがって、グラフ H のすべての頂点の次数の和は奇数になってしまう。これは、補題に矛盾する。

以上より、G’は連結である、つまりグラフ G’は頂点 v から頂点 u への経路を持つ。

(2-2-2)

(2-2-1)より、辺(u,v)を取り除いても、頂点 u から頂点 v への経路が存在する。つまり、その u から v への経路と辺(u,v)を繋げると u を含む閉路となる。

4 計算理論

(1)

(1-1)

(A)(a, 0)/00

(B)(a, 0)/00 または (b, 0)/ε

(C)(b, 0)/ε

(1-2)

(1-2-1)

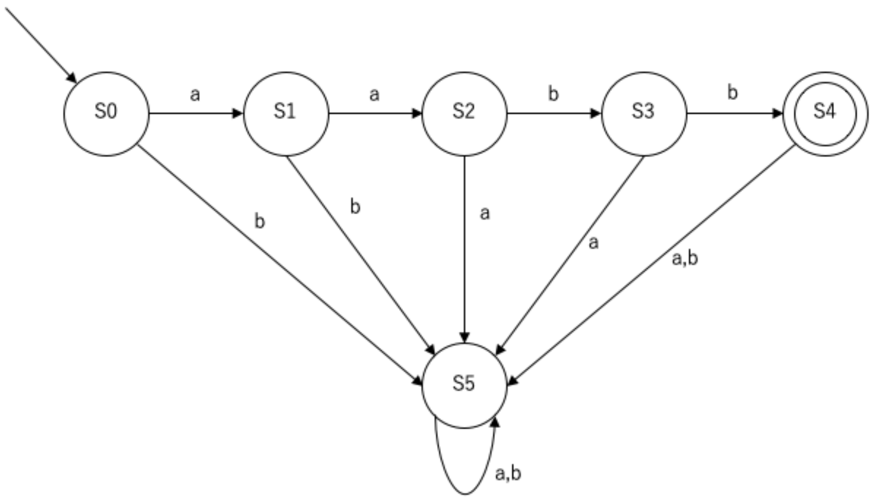

図 4 (1-2-1) の言語 L_2 を認識する DFA

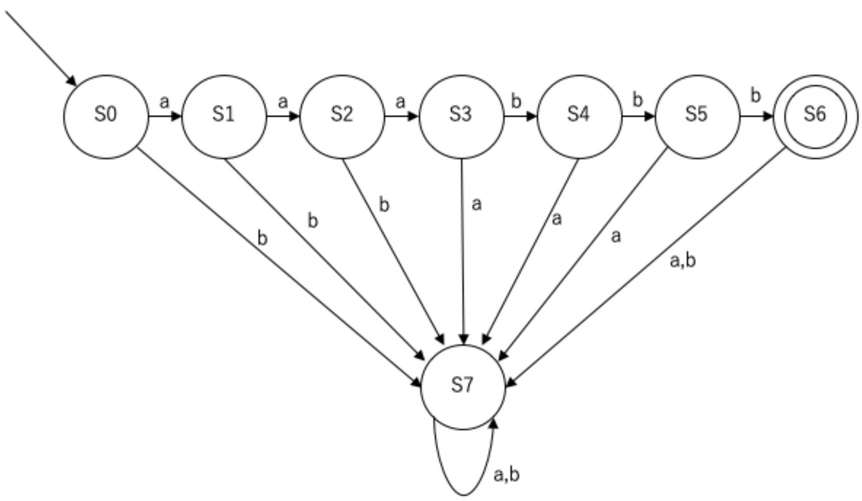

図 5 (1-2-1) の言語 L_3 を認識する DFA

(1-2-2) a のとりうる数は 0∼k 個の k+1 種類あるため、これらの判別のためには k+1 個の状態が必要となる。ここでこの判別を k 個の状態で行おうとすると、どこかで $p_i = p_j$ なる $j(i \leq j \leq k, i \neq j)$ が存在することになる。すると DFA にループが生じ、受理する文字列の a の部分が $a^{i+l(j-i)+(n-j)}(0 \leq l)$ となり、$a^n b^n$ 以外の文字列も受理してしまうことになる。従ってオートマトンの状態数は k より多くなければならない。

(1-3)

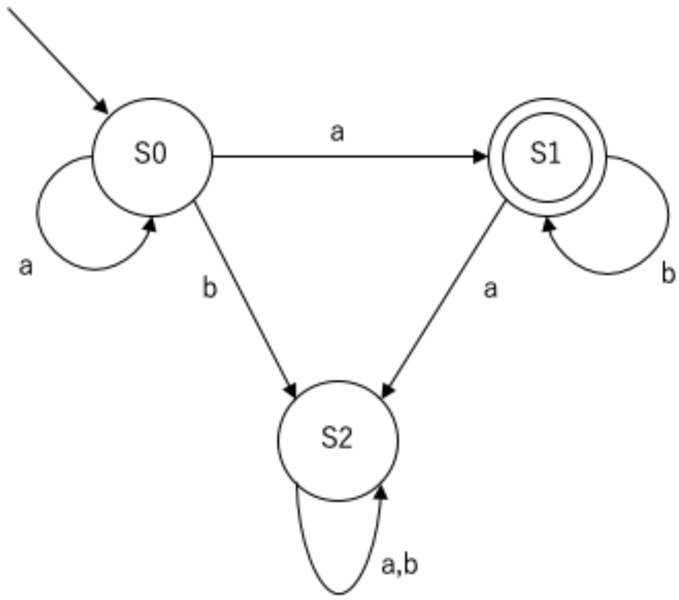

図 6　(1-3) の解答

(2)

(2-1)

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 7　(2-1) の解答

(2-2) aa は $S \xrightarrow{S \to SA} SA \xrightarrow{S \to SA} SAA \xrightarrow{S \to \varepsilon} AA \xrightarrow{A \to a} aA \xrightarrow{A \to a} aa$ と導出することができるが、これ

に対してアルゴリズムを実行すると下図のようになり、正しく動作しない。

M[i,j]	j=1	j=2
i=1	{A}	ø
i=2	-	{A}

図 8　aa に対してアルゴリズムを実行した結果

6　電子回路と論理設計

(1)

(2)

(2-1)

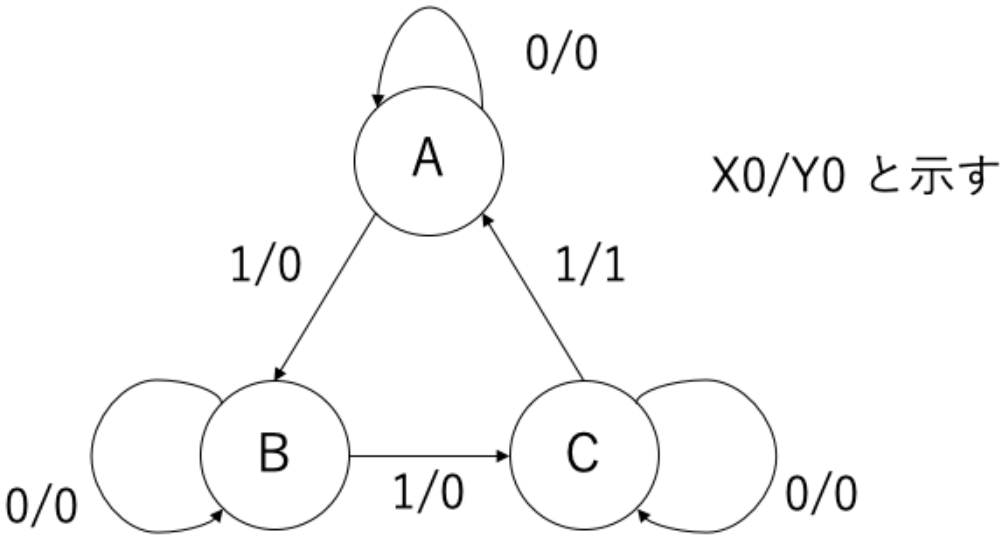

図 9　2-1 の解答

(2-2)

状態遷移表をかくと図 10 のようになり、これをもとに k_0^+、k_1^+、y_0 のカルノー図をかくと、図 11 のようになる。

	$X_0 = 0$		$X_0 = 1$	
現状態	次状態	出力	次状態	出力
0 0	0 0	0	0 1	0
0 1	0 1	0	1 0	0
1 0	1 0	0	0 0	0

図 10　状態遷移図

カルノー図から最簡積和系の論理式を導出すると、以下のようになる。

$$k_0^+ = k_0 \cdot \overline{x_0} + k_1 \cdot x_0$$
$$k_1^+ = k_1 \cdot \overline{x_0} + \overline{k_0} \cdot \overline{k_1} \cdot x_0$$
$$y_0 = k_0 \cdot x_0$$

k_0^+ x_0 \ k_0k_1	**00**	**01**	**11**	**10**
0	0	0	x	1
1	0	1	x	0

k_1^+ x_0 \ k_0k_1	**00**	**01**	**11**	**10**
0	0	1	x	0
1	1	0	x	0

y_0 x_0 \ k_0k_1	**00**	**01**	**11**	**10**
0	0	0	x	0
1	0	0	x	1

図 11　カルノー図

(2-3)

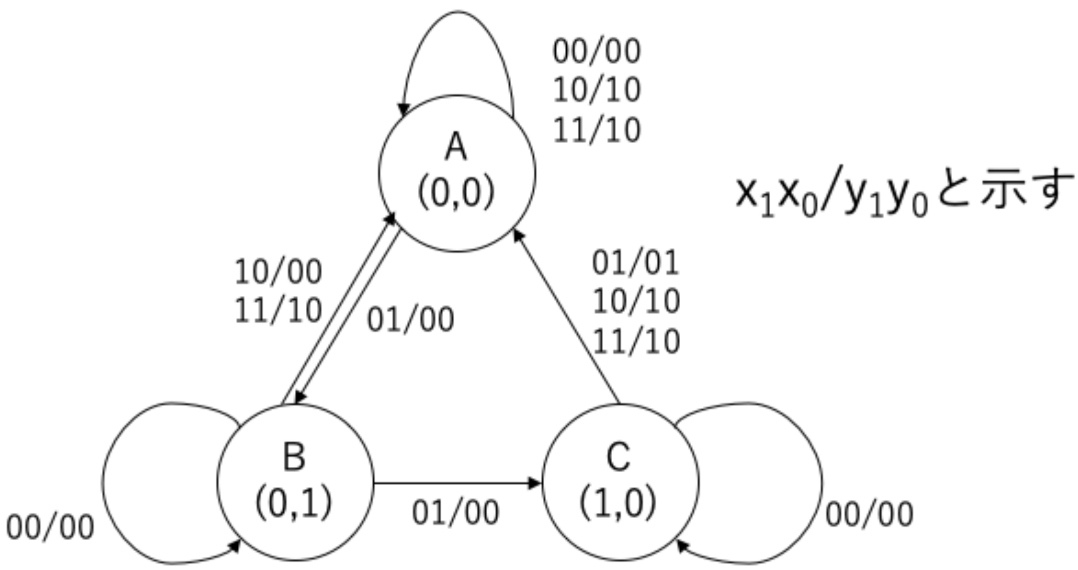

図 12　2-3 の解答

(2-4)

入力 x_1 が 0 であると通常の挙動をし、入力 x_1 が 1 であると現在の状態に関わらず初期状態に戻る、つま

り自販機に投入されたお金がなくなるためリセット機能、お釣り・返却機能だと考えられる。