令和 6 年 7 月 27 日（土）9:00～12:00

大阪大学大学院情報科学研究科

コンピュータサイエンス専攻
情報システム工学専攻
情報ネットワーク学専攻
マルチメディア工学専攻
バイオ情報工学専攻

令和 7 年度　博士前期課程　入試問題

（A）情報工学

【注意事項】

- 問題数は必須問題 2 題（問題 1～2），選択問題 5 題（問題 3～7），合計 7 題である．必須問題は 2 題すべて解答すること．また，選択問題は 2 題を選択して解答すること．
- 問題用紙の枚数は表紙と白紙を除いて 14 ページである．
- 解答用紙は全部で 4 枚ある．
  1 枚目（赤色）の解答用紙には問題 1（必須問題）の解答を
  2 枚目（青色）の解答用紙には問題 2（必須問題）の解答を
  3 枚目（白色）の解答用紙には問題 3～7（選択問題）から選択した 1 題の解答を
  4 枚目（白色）の解答用紙には問題 3～7（選択問題）から選択したもう 1 題の解答を
  それぞれ記入すること．
  解答用紙を間違えると採点されないことがあるので注意すること．
- 解答用紙は 4 枚すべてを回収するので，すべての解答用紙に受験番号を記入すること．
- 解答用紙の「試験科目」の欄には解答した問題の科目名（「アルゴリズムとプログラミング」など）を記入すること．
  また，選択問題調査票には，選択した問題の番号（3～7 から二つ）に○をつけること．
- 解答欄が不足した場合は裏面を使用すること．その際，表面末尾に「裏面に続く」と明記しておくこと．解答用紙の追加は認めない．
- 解答用紙には，日本語または英語で解答すること．これ以外の言語で記述されていた場合，採点されないことがある．

配点: (1) 45, (2-1) 20, (2-2) 30, (2-3) 30

二分ヒープ (binary heap) と呼ばれる，以下の二つの特性を持つ二分木 (binary tree) を考える．

- 特性 1: 各節点 (node) に格納された値は，その親 (parent) に格納された値以上であること．
- 特性 2: 木の高さ (height) を h とすると，深さ (depth) が h の節点は，左詰めに並んでいること．また，$h \geq 2$ の場合には，深さが $h-2$ 以下の節点はすべて 2 個の子 (child) を持つこと．ただし，ある節点の深さとは，根 (root) からその節点までの経路 (path) 上の辺 (edge) の数を指し，木の高さは深さの最大値を指す．

以降，節点番号 (node number) i を持つ節点を節点 i と呼ぶ．また，根の節点番号を 1 とし，節点 i の左の子の節点番号を $2i$，右の子の節点番号を $2i+1$ とする．図 1 に節点数が 6 の二分ヒープの例を示す．節点の左上にある数は節点番号を表し，節点内部の数は節点に格納されている値を表す．

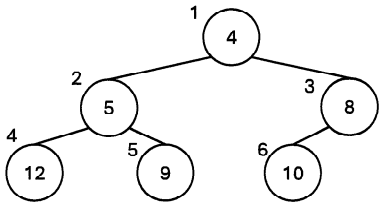


図 1 二分ヒープの例

以下の各問に答えよ．

(1) 以下の空欄 (A) ～ (J) に当てはまる最も適切な語句を，以下の選択肢 (a) ～ (l) から選び，記号で答えよ．なお，同じ選択肢を複数回用いてもよい．また，空欄 (α), (β), (γ) を適切な数式で埋めよ．

木の高さが h であるような二分ヒープが持つ節点の数は，最も少ない場合 [(α)] であり，最も多い場合 [(β)] である．

節点数が 1 以上の二分ヒープへの新しい値の追加は，次の手順で行うことができる．なお，以下の手順の Step 1 により [(A)] を満たすことができ，Step 2 ～ Step 3 により [(B)] を満たすことができる．

Step 1 子を一つしか持たない節点があれば，その節点の [(C)] の子として，新しい節点を追加する．子を一つしか持たない節点がなければ，子を持たない節点のうち，節点番号が [(D)] 節点の [(E)] の子として，新しい節点を追加する．追加された節点に新しい値を格納する．

Step 2 新しい値と新しい値が格納されている節点の [(F)] に格納されている値を比較する．もし，新しい値の方がより [(G)] 場合は，新しい値が格納されている節点とその [(H)] の値を入れ替えることにより，新しい値が格納されている節点を変更する．それ以外の場合は，処理を終了する．

Step 3 新しい値が格納されている節点が [(I)] であれば，処理を終了する．それ以外の場合は，Step 2 に戻る．

上記の手順で，二分ヒープへの値の追加を行っていくと，根の値は，二分ヒープ内の値のうち [(J)] ものとなる．また，上記の手順で，節点数が $n-1$ の二分ヒープへ値を追加し，節点数が n の二分ヒープを構成する際の，値の入れ替えの回数を $S(n)$ とすると，$S(n)$ の最大値の n に関するオーダ表記 (order notation) は，$O($ [(γ)] $)$ となる．

(a) 右 (b) 左 (c) 大きい (d) 小さい (e) 最も大きい (f) 最も小さい
(g) 根 (h) 親 (i) 子 (j) 葉 (k) 特性 1 (l) 特性 2

（次ページへ続く）

(2)　図 2 に示す ANSI-C 準拠である C 言語のプログラム (program) により，図 3 に示される input.txt に記載された値を昇順 (ascending order) で出力 (output) したい．以下の各小問に答えよ．

(2-1)　関数 (function) build_heap は，配列 (array) data に二分ヒープの節点数と節点の値を格納する関数である．57 行目の処理を終えたときの data[0] ～ data[10] の値を示せ．

(2-2)　空欄 (A) ～ (F) を x，y あるいは z で埋めて，関数 down を完成させよ．

(2-3)　下図の関数 replace は，関数 build_heap により二分ヒープの節点数と節点の値が格納された配列である引数 (parameter) data に対し，引数 index で与えられた番号の節点が存在する場合に，引数 index で指定された節点の値を引数 e の値に変更した上で，配列 data が二分ヒープの特性を満たすための処理を行う．以下の図中の関数 replace にある空欄 (G) ～ (K) に当てはまる適切な式 (expression) を，以下の選択肢 (a) ～ (k) から選び，記号で答えよ．ただし，同じ選択肢を複数回用いてはならない．

```
void replace(int *data, int index, int e) {
    if ((index > [ (G) ]) || (index <= [ (H) ])) return;
    data[index] = e;
    if ((index != 1) && (data[index] < data[ [ (I) ] ])) {
        [ (J) ];
    } else {
        [ (K) ];
    }
}
```

(a) 0　　(b) data[0]　　(c) data[index]

(d) index / 2　　(e) 2 * index　　(f) 2 * index + 1

(g) up(data, index)　　(h) down(data, index)　　(i) add(data, e)

(j) del(data)　　(k) replace(data, index, e)

（次ページへ続く）

```
1  #include <stdio.h>
2  #define SIZE 10
3
4  void swap(int *data, int i, int j) {
5      int tmp = data[i]; data[i] = data[j]; data[j] = tmp;
6  }
7
8  void up(int *data, int i) {
9      int j;
10     while (i > 1) {
11         j = i / 2;
12         if (data[j] <= data[i]) break;
13         swap(data, i, j); i = j;
14     }
15 }
16
17 void down(int *data, int x) {
18     int y, z;
19     while (2 * x <= data[0]) {
20         y = 2 * x; z = 2 * x + 1;
21         if ((z > data[0])||(data[ (A) ] < data[ (B) ])) {
22             if (data[ (C) ] <= data[ (D) ]) break;
23             swap(data, x, y); x = y;
24         } else {
25             if (data[ (E) ] <= data[ (F) ]) break;
26             swap(data, x, z); x = z;
27         }
28     }
29 }
30
31 void add(int *data, int e) {
32     if ((data[0] + 1) > SIZE) return;
33     data[0]++; data[data[0]] = e;
34     up(data, data[0]);
35 }
36
37 void del(int *data) {
38     if (data[0] <= 0) return;
39     swap(data, 1, data[0]); data[0]--;
40     down(data, 1);
41 }
42
43 void build_heap(int *data) {
44     FILE *fp;
45     int i, d;
46     fp = fopen("input.txt", "r");
47     data[0] = 0;
48     for (i = 0; i < SIZE; i++) {
49         fscanf(fp, "%d", &d);
50         add(data, d);
51     }
52     fclose(fp);
53 }
54
55 int main(void) {
56     int data[SIZE + 1];
57     build_heap(data);
58     while (data[0] > 0) { printf("%d\n", data[1]); del(data); }
59     return 0;
60 }
```

図2 プログラム

```
9
17
65
60
64
39
36
8
16
39
```

図3
input.txt

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 4/14）

配点： (1-1) 10, (1-2) 10, (1-3) 10, (1-4) 15, (1-5) 20, (2-1) 20, (2-2) 20, (2-3) 20

同期型のパイプラインプロセッサ (synchronous pipelined processor) について，以下の各問に答えよ．パイプラインプロセッサは 1 クロックサイクル (clock cycle) で 1 ステージ (stage) を実行するものとする．

(1) 単一サイクルプロセッサ (single-cycle processor) のデータパス (datapath) を，n ($n \geq 5$) 個のステージに分割することでパイプラインプロセッサを設計する．以下の仮定のもと，小問 (1-1) 〜 (1-5) に答えよ．

- 制御回路 (control unit) の遅延 (latency) は無視できるほど小さい．
- 単一サイクルプロセッサの命令遅延 (instruction latency) は 900 [ピコ秒] である．
- 分割後の各ステージの遅延が等しくなるよう，データパスを任意のステージ数に分割できる．
- パイプラインレジスタ (pipeline register) の挿入に伴う遅延オーバヘッドは 40 [ピコ秒] である．
- パイプライン化に伴うその他の遅延オーバヘッドは無視できる．
- $n = 5$ のときパイプラインプロセッサの CPI (cycles per instruction) は 1.14 であり，ステージを一つ増やすたびに，パイプラインハザード (pipeline hazard) とその他の理由により CPI は 0.1 増加する．
- パイプラインプロセッサで実行する命令数を m とする．m は n より十分大きい．

(1-1) m 個の命令を，元の単一サイクルプロセッサで実行する場合の実行時間 [ピコ秒] を示せ．

(1-2) $n = 5$ のとき，パイプラインプロセッサの最小クロックサイクル時間 [ピコ秒] を示せ．

(1-3) $n = 5$ のとき，最小クロックサイクル時間で動作するパイプラインプロセッサの平均実行時間 [ピコ秒] を示せ．

(1-4) パイプラインプロセッサの平均実行時間が最小となる n を示せ．

(1-5) 分岐予測 (branch prediction) という観点から，ステージ数が増えるほど CPI が増加する理由を一つ説明せよ．なお，ステージ数が変化しても分岐予測の精度は変わらないものとする．

(次ページへ続く)

(2)　5つのステージ (IF, ID, EX, MA および WB) からなるパイプラインプロセッサを考える．以下の仮定のもと，小問 (2-1) ～ (2-3) に答えよ．

- このプロセッサは主要な部品として，プログラムカウンタ (program counter)，命令メモリ (instruction memory)，データメモリ (data memory)，レジスタファイル (register file) および ALU (arithmetic logic unit) をそれぞれ一つ備える．
- その他の資源は，パイプライン処理のために十分な数がある．
- レジスタファイルは，レジスタ `r1` ～ `r7` を持つ．
- 下表に，各ステージにおけるプロセッサの動作を示す．空欄 (—) は，主要な部品をいずれも使用しないことを表す．

	演算命令 `add, sub`	ロード命令 `lw`	ストア命令 `sw`
IF	命令メモリから命令の読み出し，プログラムカウンタに命令長を加算		
ID	命令のデコード，レジスタファイルからオペランドの読み出し		
EX	ALU を用いた演算	ALU を用いたアドレス計算	
MA	—	データメモリから読み出し	データメモリに書き込み
WB	結果をレジスタファイルに書き込み		—

- このプロセッサは，WB ステージでレジスタファイルに書き込んだ値を，同じクロックサイクルの ID ステージですぐ読み出して使用できる．
- このプロセッサはフォワーディング (pipeline forwarding) の機能をもち，MA ステージあるいは WB ステージの入力を，同じクロックサイクルの EX ステージの入力としても利用できる．
- 構造ハザード (structural hazard) およびデータハザード (data hazard) を解消するために，このプロセッサは必要に応じてストール (pipeline stall) を発生させ，IF ステージおよび ID ステージをやり直す．

(2-1)　このプロセッサを用いて以下の命令列 A を順に実行する．解答欄の命令 1 の例を参考に，各クロックサイクルでプロセッサが実行するステージ名を解答用紙に記入せよ．ステージの実行にストールが必要な場合は，ストールが発生するクロックサイクルにもそのステージ名を記入せよ．

命令列 A

命令 1	`add r1, r2, r3`	`r2` の値と `r3` の値を加算し，結果を `r1` へ格納
命令 2	`add r4, r5, r1`	`r5` の値と `r1` の値を加算し，結果を `r4` へ格納
命令 3	`lw r7, (r4)`	`r4` の値が指すデータメモリのアドレスから値を読み出し，`r7` に格納
命令 4	`sw r6, (r7)`	`r7` の値が指すデータメモリのアドレスに，`r6` の値を書き込む

(2-2)　このプロセッサを用いて以下の命令列 B を順に実行する．小問 (2-1) と同様に，各クロックサイクルでプロセッサが実行するステージ名を解答用紙に記入せよ．

命令列 B

命令 1	`lw r1, (r2)`	`r2` の値が指すデータメモリのアドレスから値を読み出し，`r1` に格納
命令 2	`add r3, r1, r4`	`r1` の値と `r4` の値を加算し，結果を `r3` へ格納
命令 3	`sub r5, r6, r7`	`r6` の値から `r7` の値を減算し，結果を `r5` へ格納
命令 4	`sw r5, (r3)`	`r3` の値が指すデータメモリのアドレスに，`r5` の値を書き込む

(2-3)　命令列 A および命令列 B のそれぞれについて，アウトオブオーダ (out-of-order) 実行を用いて実行時間を削減可能か理由とともに答えよ．削減可能な場合は，命令列の実行に必要な最小クロックサイクル数を示せ．

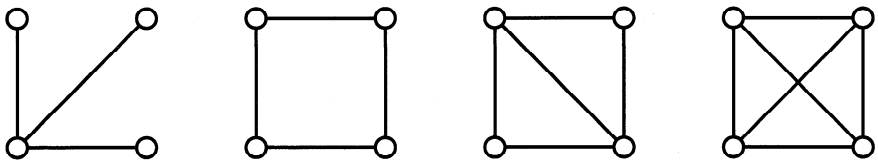
配点：(1-1) 15, (1-2) 20, (1-3) 20, (2-1) 15, (2-2) 15, (2-3) 20, (2-4) 20

本問題で取り扱われるすべてのグラフ (graph) は無向グラフ (undirected graph) であり，多重辺 (parallel edge) や自己ループ (self-loop) を持たないものとする．グラフ G は有限 (finite) の頂点集合 (vertex set) V，および異なる頂点の非順序対 (unordered pair) の集まりである辺集合 (edge set) E の対 (pair) により $G = (V, E)$ と表される．記号 $\emptyset$ は空集合 (empty set) を表すものとする．

k を正の整数 (positive integer) とする．グラフ $G = (V, E)$ が与えられたとき，その頂点集合を定義域 (domain) とする写像 (mapping) $c : V \to \{0, 1, 2, \ldots, k-1\}$ が任意の $\{u, v\} \in E$ に対して $c(u) \neq c(v)$ を満たすとき，写像 c を G の k-彩色 (k-coloring) と呼ぶ．また，k-彩色が存在するようなグラフ G を k-彩色可能 (k-colorable) なグラフと呼ぶ．$\mathcal{G}_k$ を k-彩色可能なすべてのグラフからなる集合とする．また，グラフ G の異なる k-彩色の個数を $\gamma(G, k)$ で表す．以下の各問に答えよ．

(1) 以下の各小問に答えよ．

(1-1) 以下のグラフ G_1，G_2，G_3，G_4 のうち，3-彩色可能だが 2-彩色可能でないものをすべて選べ．



G_1 G_2 G_3 G_4

(1-2) 2-彩色可能な頂点数 6 のグラフで，辺数が最も多いグラフを一つ図示せよ．

(1-3) すべての頂点の次数 (degree) が 2 の連結 (connected) なグラフを閉路グラフ (cycle graph) と呼ぶ．また，閉路グラフを部分グラフ (subgraph) として含まない連結なグラフを木 (tree) と呼ぶ．$\mathcal{C}$ をすべての閉路グラフからなる集合，$\mathcal{T}$ をすべての木からなる集合とする．$\mathcal{G}_3$，$\mathcal{G}_2$，$\mathcal{C}$，$\mathcal{T}$ の 4 つの集合について，それらの包含関係 (containment relationship) および交わりの関係 (intersection relationship) が分かるようなベン図 (Venn diagram) を図示せよ．

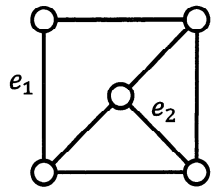
(次ページへ続く)

(2)　グラフ $G=(V,E)$ および辺 $e=\{x,y\}\in E$ について，$G-e$ をグラフ G から辺 e を取り除いて得られるグラフとする．また，グラフ G/e を，辺 e を縮約 (contract) して得られるグラフ，すなわち，以下のように定義される頂点集合 $V_{G/e}$ および辺集合 $E_{G/e}$ により定まるグラフとする．

- $V_{G/e}=(V\setminus\{x,y\})\cup\{v_e\}$．ただし v_e は V に含まれない要素とする．
- $E_{G/e}=\{\{v,w\}\in E\mid\{v,w\}\cap\{x,y\}=\emptyset\}\cup\{\{v_e,w\}\mid(\{x,w\}\in E\setminus\{e\})\vee(\{y,w\}\in E\setminus\{e\})\}$.

このとき，以下の各小問に答えよ．

(2-1)　以下のグラフ H と H 中の辺 e_1，e_2 について，H/e_1 および H/e_2 を図示せよ．



H

(2-2)　E が空集合であるような頂点数 n のグラフ $G=(V,E)$ について，$\gamma(G,k)$ の値を n，k を用いて表せ．

(2-3)　$E\neq\emptyset$ である任意のグラフ $G=(V,E)$ および任意の辺 $e\in E$ に対して，以下の式が成立することを示せ．

$$\gamma(G,k)=\gamma(G-e,k)-\gamma(G/e,k) \tag{1}$$

(2-4)　小問 (2-3) の式 (1) を用いて，頂点数 n の任意の木 T について $\gamma(T,k)=k\cdot(k-1)^{n-1}$ が成り立つことを，n についての数学的帰納法 (mathematical induction) により証明せよ．

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

(1) 特定の条件を満たす言語（language）を受理（accept）するような決定性有限オートマトン（deterministic finite automaton）を構成したい．決定性有限オートマトンの状態遷移図（state transition diagram）は，図 1 に示すように開始状態（start state）には太い矢印（thick arrow）を付与し，最終状態（final state）は二重丸（double circle）で表現する．以下の各小問に答えよ．

(1-1) 言語$\{w \in \{a,b\}^* |$文字列（string）wは奇数個のaを含み，かつ，高々一つのbを含む$\}$を受理する状態数 5 の決定性有限オートマトンの状態遷移図を示せ．

(1-2) 言語$\{w \in \{a,b\}^* |$文字列wは部分文字列（substring）babを含まない$\}$を受理する状態数 4 の決定性有限オートマトンの状態遷移図を示せ．

(2) 特定の条件を満たす言語に対して，最終状態による受理（acceptance by final state）を行うプッシュダウンオートマトン（pushdown automaton）を構成したい．プッシュダウンオートマトンは$(Q, \Sigma, \Gamma, \delta, q_0, Z, F)$で表される．$Q$は状態（state）の有限集合（finite set），Σは入力アルファベット（input alphabet），Γはスタックアルファベット（stack alphabet），δは遷移関数（transition function），q_0は開始状態，$Z \in \Gamma$はスタックの開始記号（initial pushdown symbol），Fは最終状態の集合である．プッシュダウンオートマトンの状態遷移図は，図 2 に示すように開始状態には太い矢印を付与し，受理状態は二重丸で表現する．遷移に付与されているラベル $r, s/t$（$r \in \Sigma \cup \{\varepsilon\}$，$s \in \Gamma$，$t \in \Gamma^*$）は，入力から読み出す記号が$r$でスタックから取り出す記号が$s$のとき，スタックからその$s$を取り去り，$t$をスタックに押し込むことを意味する．$t$が複数の記号の場合は，右側の記号から順にスタックに押し込む．rがεの場合は，入力から記号を読み出さずに遷移を行う．tがεの場合は，スタックに記号を押し込まない．$\Gamma = \{A, B, Z\}$とする．以下の各小問に答えよ．

(2-1) 言語$\{w \in \{a,b\}^* | N_a(w) = 2N_b(w)\}$を受理する状態数 3 のプッシュダウンオートマトンの状態遷移図を示せ．ここで，$N_a(w)$，$N_b(w)$は，それぞれ文字列wに含まれるa，bの数とする．

(2-2) $\Sigma = \{a, b, \#\}$とする．回文（palindrome）とは，前から読んでも後ろから読んでも同じ文字列のことである．図 2 は言語$\{w\#w' | w, w' \in \{a,b\}^*$，$w\#w'$は回文ではない$\}$を受理するプッシュダウンオートマトンの状態遷移図である．①～③に記載すべき全てのラベルを$r, s/t$の形式で答えよ．

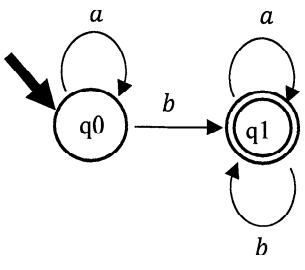


図 1

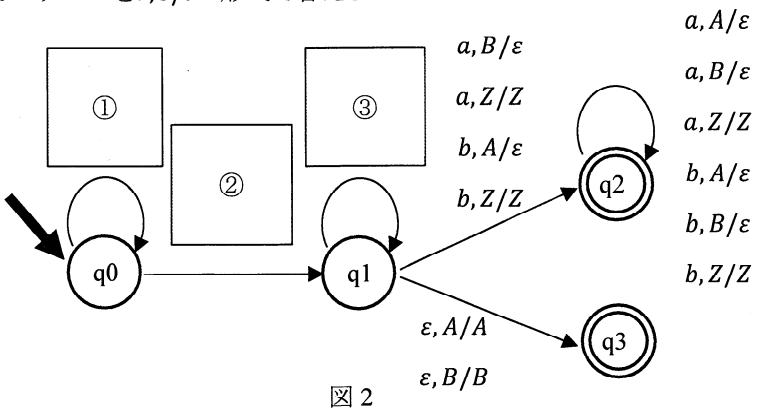


図 2

（次ページへ続く）

(3) 文脈自由文法（context-free grammar）は，一般に$G = (V, T, P, S)$で定義される．ここでVは変数（variable）の集合，Tは終端記号（terminal symbol）の集合，Pは生成規則（production rule）の集合，Sは開始記号（start symbol）であり，$S \in V$である．終端記号列（sequence）$t \in T^*$をi個（$i \geq 0$）並べたものをt^iと表記する．また，空文字列（empty string）をεと表記する．$T = \{a, b, c\}$のとき，文脈自由言語（context-free language）$L_1 = \{a^n b^n c^m \mid m, n \geq 0\}$および$L_2 = \{a^n b^m c^m \mid m, n \geq 0\}$を考える．以下の各小問に答えよ．

(3-1) L_1を生成する文法G_1を以下のように定める．

$G_1 = (\{S, A, B\}, T, P_1, S)$

P_1の要素を全て示せ．P_1にはε-規則（ε-production）を含めても良い．生成規則の数は 5 以下とする．

(3-2) 文脈自由言語は積集合（intersection）について閉じていないことを，文脈自由言語に対する反復補題（pumping lemma）を用いて示したい．文脈自由言語に対する反復補題は以下に示す通りである．

文脈自由言語に対する反復補題

Lを任意の文脈自由言語とする．Lに対してある定数pが存在し，Lに属する長さp以上の任意の文字列sは，以下の条件 1〜3 を満たす$s = uvxyz$の形に分解（decompose）できる．

条件 1. 任意の$i \geq 0$に対して，$uv^i xy^i z \in L$

条件 2. $|vy| > 0$

条件 3. $|vxy| \leq p$

ここで，$|vy|$，$|vxy|$はそれぞれvy，vxyの長さである．

以下は，L_1，L_2が文脈自由言語であることを用いた証明である．適切な記述で空欄④〜⑦を埋めよ．

証明：文脈自由言語が積集合について閉じていると仮定する．L_1，L_2が文脈自由言語であることから，$L_1 \cap L_2 =$ {　　④　　}も文脈自由言語である．$L_1 \cap L_2$に対して文脈自由言語に対する反復補題を適用する．このとき，$L_1 \cap L_2$に対してある定数pが存在し，$L_1 \cap L_2$に属する長さp以上の文字列$s =$　　⑤　　を考える．$s = uvxyz$と分解したとき，条件 1 より，任意の$i \geq 0$に対して，$uv^i xy^i z \in L_1 \cap L_2$である．ここで，条件 3 より，$vxy$が$a$および$c$をともに含むことはできないため，以下の 2 通りに分けて考える．

・vxyがcを含まない場合

⑥

これは反復補題に矛盾する．

・vxyがaを含まない場合

⑦

これは反復補題に矛盾する．

以上より，仮定が誤りであり，背理法により$L_1 \cap L_2$は文脈自由言語ではない．従って，文脈自由言語は積集合について閉じていない．（証明終）

配点：(1) 30, (2) 15, (3-1) 10, (3-2) 20, (3-3) 20, (4) 30

アプリケーション層プロトコル（application layer protocol）の 1 つに，HTTP (HyperText Transfer Protocol)がある．HTTP のクライアント（client）は，HTTP によって HTTP のサーバ（server）上のファイル（file）を取得する．トランスポート層プロトコル（transport layer protocol）に TCP (Transmission Control Protocol) を用いる HTTP に関する以下の各問に答えよ．

(1) クライアントが HTTP を用いて HTML (HyperText Markup Language)ファイルを受信したものとする．この時，HTML ファイルの先頭のビット（bit）を受信するまでの間に，クライアントおよびサーバで実行される HTTP と TCP の挙動を以下の語句を用いて説明せよ．すべての語句を 1 回以上使用すること．

語句：HTTP Request，HTTP Response，SYN フラグ（flag），ACK フラグ，スリーウェイハンドシェイク（three-way handshake），コネクション確立（connection establishment）

(2) HTML ファイルには，情報資源を表す URL が複数記載されることがある．その 1 つの URL が http://www.ist.osaka-u.ac.jp/eng/image.jpg であるとし，この URL を構成する「http」，「www.ist.osaka-u.ac.jp」，「/eng/image.jpg」のそれぞれの意味を述べることによって，URL の構造を説明せよ．

(3) サーバが情報資源のファイルの送信が完了した後も TCP コネクション（connection）を維持する手順を採用する HTTP を考える．この HTTP では，クライアントは，ファイルの受信を完了すると，一定時間 S [秒] 毎に制御メッセージ（control message）をサーバに送信する．サーバは，制御メッセージを受信するとカウントアップタイマ（count-up timer）を 0 に設定し，T [秒]を超えない間は TCP コネクションを維持する．以下の各小問に答えよ．

(3-1) クライアントからサーバまでのアプリケーション層における遅延とサーバからクライアントまでのアプリケーション層における遅延が等しく R [秒]であり，かつ，一定であるとする．また，制御メッセージの伝送遅延（transmission delay）と制御メッセージの処理に要する時間は 0 とする．この時，TCP コネクションを維持できる S の範囲を，T と R を用いて表せ．

(3-2) クライアントが Sを小さく設定する場合に，サーバで生じる問題を 1 つ述べよ．

(3-3) サーバがファイルの送信を完了した後に即座に TCP コネクションを解放する手順を採用する HTTP の場合，クライアントがサーバ上の複数のファイルを取得する際に受信が完了するまでの時間が増大する．増大する理由を説明せよ．

(4) サーバ上の動画像ファイルを順次取得し再生するストリーミング（streaming）に HTTP を使用すると，TCP プロトコルの挙動が原因で再生が途切れることがある．再生が途切れる理由を 1 つ述べよ．また，再生が途切れることがあるにもかかわらず，ストリーミングに HTTP を使用する利点を 1 つ述べよ．

配点：(1-1) 10，(1-2) 25，(1-3) 60，(2-1) 15，(2-2) 15

以下の漸化式で定義される数列F_nを生成する専用ハードウェア (dedicated hardware) を設計したい．以下の各問に答えよ．

$$F_0 = 0$$

$$F_1 = 1$$

$$F_{n+2} = F_n + F_{n+1} \quad (n \geq 0)$$

(1) F_0，F_1，F_2，F_3，F_4を繰り返し生成する専用ハードウェアを図 1 に示す．この専用ハードウェアはデータパス (datapath) と制御回路 (control circuit) から構成され，クロック信号 (clock signal) CLK に同期して動作する．データパスは数列F_nを生成する演算処理を行う回路で 2 つの 4 ビットレジスタ (4-bit register) R0，R1，4 ビット加算器 (4-bit adder) ADD，4 ビットマルチプレクサ (4-bit multiplexer) MUX から構成される．レジスタ R0，R1 は 1 ビットの clear 入力を持ち，clear=0 の時はレジスタの値は CLK に同期して入力の値に更新されるが，clear=1 の時は CLK に同期して 0000_2 に設定される．加算器 ADD は 2 つの入力を加算して結果を出力するが，桁上げ信号 (carry signal) は出力しない．マルチプレクサ MUX は制御信号 s によって出力する値が切り替わる．F_0〜F_4は出力 y からそれぞれ 4 ビット符号無し 2 進数 (4-bit unsigned binary number) として得られる．制御回路はこのようなデータパスの制御信号 c_0，c_1，s を適切に切り替える．以下の各小問に答えよ．

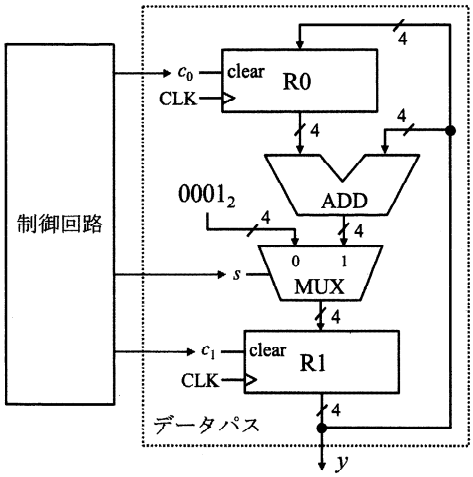


図 1. 対象とする専用ハードウェア

表 1. 制御回路 1 の状態遷移・出力表

状態	現状態 (k_0, k_1, k_2)	次状態 (d_0, d_1, d_2)	出力		
			c_0	c_1	s
S0	000	001	0	0	①
S1	001	010	0	0	②
S2	010	011	0	0	③
S3	011	100	0	0	④
S4	100	000	1	1	⑤

(1-1) F_0，F_1，F_2，F_3，F_4の値を 4 ビット符号無し 2 進数で求めよ．

（次ページへ続く）

(1-2)　図 1 の制御回路を同期式順序回路 (synchronous sequential circuit) として設計した．この制御回路を制御回路 1 と呼び，その状態遷移・出力表を表 1 に示す．制御回路 1 の初期状態 (initial state) は S0 で，初期状態ではデータパスのレジスタ R0，R1 の値は 0000_2 に初期化されている．また，制御回路 1 は状態 S0～S4 においてデータパスの出力 y から F_0～F_4 がそれぞれ得られるようにデータパスを制御したい．表 1 の①～⑤に当てはまる適切な値を答えよ．ドントケア (don't care) の場合は X と書くこと．

(1-3)　制御回路 1 の現状態 (current state) (k_0, k_1, k_2) に対する次状態 (next state) を (d_0, d_1, d_2) と表す．d_0，d_1，d_2，c_0，c_1，s の論理式の最簡積和形 (最小積和形，minimal sum-of-products expression) を k_0，k_1，k_2 を適切に用いてそれぞれ求めよ．

(2)　図 1 の専用ハードウェアが F_0，F_1，...，F_{14}，F_{15} を繰り返し生成するように制御回路とデータパスをそれぞれ改造したい．制御回路に関しては，制御回路 1 の状態を S0～S15 に増やした制御回路 2 を設計した．以下の各小問に答えよ．

(2-1)　F_0，F_1，...，F_{14}，F_{15} を繰り返し生成するために図 1 のデータパスに制御回路 2 を組み合わせた．制御回路 2 は状態 S0～S15 においてデータパスの出力 y から F_0～F_{15} がそれぞれ出力されるようにデータパスを制御する．図 1 のデータパスと制御回路 2 を組み合わせて動作させたところ途中から正しい値が生成できなくなった．最初に生成できない F_n の値を 10 進数 (decimal number) で書け．その理由も具体的に答えよ．

(2-2)　前問 (2-1) のデータパスをどのように改造すれば F_0，F_1，...，F_{14}，F_{15} を繰り返し生成できるか具体的に答えよ．

7 【選択問題】数学解析と信号処理 （情報工学 13/14）

配点：(1) 30, (2) 20, (3-1) 25, (3-2) 20, (4-1) 15, (4-2) 15

離散時間信号 (discrete-time signal) $x[n]$ を入力し，式 (1) で示される $H(\omega)$ にできるだけ近い周波数応答 (frequency response) をもつ低域通過フィルタ (low-pass filter) を設計したい．ただし，n は整数 (integer)，ω は正規化角周波数 (normalized angular frequency) [rad/sample]，ω_c はある正規化角周波数を表す正の実数 (positive real number) とする．

$$H(\omega) = \begin{cases} 1 & (|\omega| \leq \omega_c) \\ 0 & (\text{otherwise}) \end{cases} \tag{1}$$

ここで，虚数単位 (imaginary unit) j を用いて，離散時間信号 $x[n]$ の離散時間フーリエ変換 (discrete-time Fourier transform) $X(\omega)$ とその逆変換は以下のように与えられるものとする．

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}, \qquad x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} X(\omega)e^{j\omega n}d\omega \tag{2}$$

以下の各問に答えよ．

(1) $H(\omega)$ の逆離散時間フーリエ変換 (inverse discrete-time Fourier transform) によりインパルス応答 (impulse response) $h[n]$ を求め，指数関数 (exponential function) を用いない形で表せ．

(2) 問 (1) で求めたインパルス応答 $h[n]$ は，実時間信号処理 (real-time signal processing) における畳み込みフィルタ (convolution filter) として用いるには，実装上の問題がある．因果性 (causality) および有限性 (finiteness) の観点から 2〜3 行程度で理由を説明せよ．

(次ページへ続く)

(3)　問 (1) で求めたインパルス応答 $h[n]$ に窓関数（window function）$w[n]$ を掛けたインパルス応答 $h'[n] = h[n]w[n]$ を畳み込みフィルタとして用いることを考える．ここでは，以下で定義される窓長（window length）M が奇数（odd number）の矩形窓（rectangular window）関数 $w[n]$ およびその離散時間フーリエ変換 $W(\omega)$ を用いることとする．

$$w[n] = \begin{cases} 1 & (0 \le |n| < \frac{M}{2}) \\ 0 & (\text{otherwise}) \end{cases}, \qquad W(\omega) = \begin{cases} M & (\omega = 2\pi k, \text{ ただし } k \text{ は任意の整数}) \\ \sin(\frac{M}{2}\omega)/\sin(\frac{1}{2}\omega) & (\text{otherwise}) \end{cases} \tag{3}$$

式 (3) に示したフーリエ変換 $W(\omega)$，式 (1) に示した周波数応答 $H(\omega)$ および，窓関数適用後のインパルス応答 $h'[n] = h[n]w[n]$ のフーリエ変換 $H'(\omega)$ の概形を図 1 に示す．ただし，図中のスケールは正確ではない．

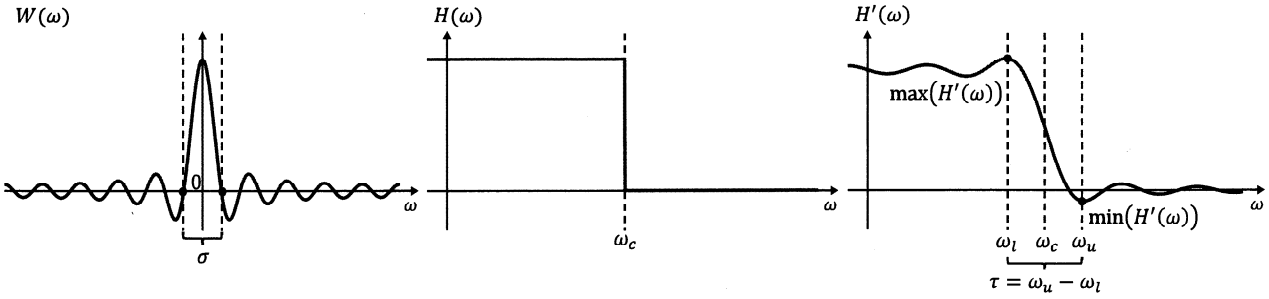


図 1　$W(\omega)$，$H(\omega)$，$H'(\omega)$ の概形

以下の各小問に答えよ．

(3-1)　$W(\omega)$ のメインローブ幅（main lobe width）を，図中に示すように，$W(\omega) = 0$ となる点のうち，原点を挟み，かつ原点から最も近い 2 点間の距離 σ として定義するとき，σ を求めよ．

(3-2)　$H'(\omega)$ の遷移域（transition band）の幅を，図中に示すように，$H'(\omega)$ が極小，極大となる ω のうち，それぞれ ω_c に最も近いものの差 $\tau = \omega_u - \omega_l$ として定義するとき，τ を求めよ．導出は記載しなくても良い．

(4)　窓関数がフィルタに与える実用上の影響について，以下の各小問に答えよ．

(4-1)　問 (3) の窓長 M を変えると，窓関数のメインローブ幅が変化する．「窓関数のメインローブ」と「周波数応答の遷移域」の関係および，これらがフィルタに与える実用上の影響を，窓長 M と関連付けて 2〜3 行程度で説明せよ．

(4-2)　問 (3) で用いた矩形窓関数の他にも様々な窓関数が用いられ，窓関数によってサイドローブ（side lobe）形状が異なる．一般的に，「窓関数のサイドローブ」と「周波数応答のリップル（ripple）」にどのような関係があるか，またこれらがフィルタに与える実用上の影響について，2〜3 行程度で説明せよ．