令 和７年 度

名古屋大学大学院情報学研究科
情報システム学専攻
入学試験問題（専門）

令和６年８月７日

注意事項

１．試験開始の合図があるまでは、この問題冊子を開いてはならない。

２．試験終了まで退出できない。

３．外国人留学生の志願者は、日本語と日本語以外の１言語間の辞書１冊に限り使用してよい。電子辞書の持ち込みは認めない。

４．日本語または英語で解答すること。

５．問題冊子、解答用紙６枚、草稿用紙３枚が配布されていることを確認すること。

６．問題は問１～８の８問がある。このうち６問を選択して解答すること。なお、選択した~~科目名~~問の番号を解答用紙の指定欄に記入すること。

７．全ての解答用紙の所定の欄に受験番号を必ず記入すること。解答用紙に受験者の氏名を記入してはならない。

８．解答用紙に書ききれない場合は、裏面を使用してもよい。ただし、裏面を使用した場合は、その旨、解答用紙表面右下に明記すること。

９．解答用紙は試験終了後に６枚とも提出すること。

１０．問題冊子、草稿用紙は試験終了後に持ち帰ること。

問1

U をすべての頂点の集合，W を加算と最小値演算が可能な集合とするとき，次の性質を満たす部分関数 $\psi : U \to 2^{U \times W}$ は重み付き無向グラフを表す．

$$\text{任意の } v, v' \in U \text{ と } w \in W \text{ に対して，} \quad (v', w) \in \psi(v) \iff (v, w) \in \psi(v')$$

実際に，ψ が表す重み付きグラフ $G_\psi = \langle V_\psi, E_\psi \rangle$ は次のように定まる．

$$V_\psi = \mathrm{Dom}(\psi)$$
$$E_\psi = \{(v, w, u) \mid v \in V_\psi,\ (u, w) \in \psi(v)\}$$

ここで，$\mathrm{Dom}(\psi)$ は ψ の定義域，すなわち，$\mathrm{Dom}(\psi) = \{v \mid \psi(v) \text{ が定義される}\}$ であり，また，$(v, w, u) \in E_\psi$ は，頂点 v と頂点 u を結ぶ重み w の辺が G_ψ に存在することを表している．以下では，定義域が空集合の部分関数を $\perp$ と書く．また，$F[v \to w]$ は，引数が v のときに w を返すように部分関数 F を変更して得られる部分関数を表し，以下のように定義される．

$$F[v \to w](u) = \begin{cases} F(u) & \text{if } u \neq v \\ w & \text{if } u = v \end{cases}$$

部分関数 $\psi : U \to 2^{U \times W}$ が与えられたとき，G_ψ において頂点 s と t を結ぶパスのうちで最小の重みを求めるためのアルゴリズムが以下のように与えられるとする．このとき，(1) から (5) に答えよ．

アルゴリズム

Input: ψ and $s, t \in V_\psi$

Step.1: $D := \perp$ and $N := \perp[s \to 0]$

Step.2: Repeat the following steps (a)–(c) until $t \in \mathrm{Dom}(D)$ or $\mathrm{Dom}(N) - \mathrm{Dom}(D) = \emptyset$:

(a) Choose some $v \in \mathrm{Dom}(N) - \mathrm{Dom}(D)$ such that

$$N(v) = \min\{N(u) \mid u \in \mathrm{Dom}(N) - \mathrm{Dom}(D)\}$$

(b) $D := D[v \to N(v)]$

(c) For each $(u, w) \in \psi(v)$, $N := N[u \to w']$, where

$$w' = \begin{cases} \min(D(v) + w, N(u)) & \text{if } u \in \mathrm{Dom}(N) \\ D(v) + w & \text{if } u \notin \mathrm{Dom}(N) \end{cases}$$

Output: $\begin{cases} D(t) & \text{if } t \in \mathrm{Dom}(D) \\ \text{no connection} & \text{if } t \notin \mathrm{Dom}(D) \end{cases}$

(1) $\psi_1 : U \to 2^{U \times W}$ を集合 $S_{\psi_1} = \{(v, \psi_1(v)) \mid v \in Dom(\psi_1)\}$ により与える．このとき，部分関数 ψ_1 が表すグラフ G_{ψ_1} を図示せよ．

$$S_{\psi_1} = \left\{ \begin{array}{l} (a, \{(b,5),(c,1),(e,2)\}), \\ (b, \{(a,5),(d,1),(e,4)\}), \\ (c, \{(a,1),(d,2)\}), \\ (d, \{(b,1),(c,2)\}), \\ (e, \{(a,2),(b,4)\}) \end{array} \right\}$$

(2) (1) で与えられる ψ_1 に対して $s = a, t = b$ としてアルゴリズムを実行したときに，Step.2 の各繰り返しにおける部分関数 D と N の推移(すいい)を示せ．ここで，部分関数は (1) で ψ_1 を表すのに S_{ψ_1} を用いたのと同様に，集合 S_D および S_N によって示すこと．

(3) アルゴリズムの Step.2 の各繰り返しにおいて，次の性質 (*) がグラフ G_ψ 上で成立することを示せ．

(*) 任意の $v \in \mathrm{Dom}(N)$ に対して，s から v へ重み $N(v)$ のパスが存在する

(4) 重みが正整数(せいせいすう)であり，かつ ψ が常に有限(ゆうげん)集合を返す（すなわち任意の頂点に接続する辺の数が有限である）とする．このとき，無限(むげん)の頂点集合 V_ψ を持つグラフに対して，s と t の間にパスが存在するならばアルゴリズムが停止(ていし)することを説明せよ．また，s と t の間にパスが存在しない場合においてアルゴリズムが停止するかどうか論ぜよ．

(5) 重みが正の有理数(ゆうりすう)であり，かつ ψ が常に有限集合を返す（すなわち任意の頂点に接続する辺の数が有限である）とする．このとき，無限の頂点集合 V_ψ を持つグラフに対して，s と t の間にパスが存在する場合であってもアルゴリズムが停止しないことがあることを説明せよ．

Translation of technical terms

頂点	vertex	空集合	empty set
集合	set	引数	argument
加算	addition	パス	path
最小値演算	minimum value operator	アルゴリズム	algorithm
部分関数	partial function	推移	transition
重み	weight	繰り返し	repetition
無向グラフ	undirected graph	正整数	positive integer
定義域	domain	停止する	terminate
辺	edge	有理数	rational number

問2

アルファベット Σ 上の言語について以下のすべての問いに答えよ．(1), (3), (4) を解答する際に具体的に有限オートマトンを示す場合には状態遷移図を記述することとし，決定性オートマトンもしくは非決定性オートマトンのどちらでもよい．ただし，具体的に示すオートマトンの初期状態は1つとし，さらに ϵ 遷移を持たないこととする．

(1) Σ 上の語 w の接頭辞すべてからなる言語を $prefix(w)$ と表し，言語 L に含まれる語の接頭辞すべてからなる言語を $prefix(L)$ と表す．すなわち，$prefix$ は以下のように定義される．

$$prefix(w) = \{w' \in \Sigma^* \mid \exists w'' \in \Sigma^*.\ w = w'w''\} \qquad prefix(L) = \bigcup_{w \in L} prefix(w)$$

図1の状態遷移図で定義される，アルファベット $\{a, b\}$ 上の有限オートマトン A_1 により認識される正規言語 L_1 について，$prefix(L_1)$ が正規言語であるかどうか答えよ．正規言語である場合には $prefix(L_1)$ を認識するオートマトンを示し，そうでない場合には $prefix(L_1)$ が正規言語ではないことを説明せよ．

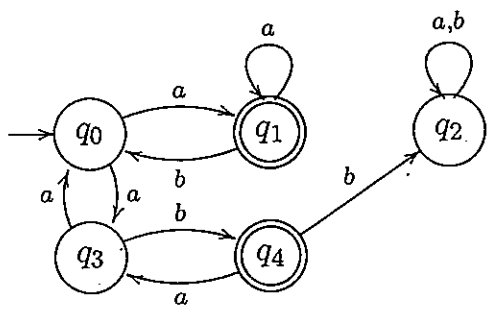

図 1: オートマトン A_1 の状態遷移図

(2) $prefix(L)$ が空集合となる正規言語 L は存在するか？存在する場合にはそのような L の例を1つ示し，そうでない場合には任意の正規言語 L について $prefix(L) \neq \emptyset$ であることを示せ．

(3) Σ 上の語 w を逆順にした語を w^{-1} と表す．すなわち，w^{-1} は以下のように定義される．

$$\epsilon^{-1} = \epsilon \qquad (aw)^{-1} = w^{-1}a \quad (\text{ただし，} a \in \Sigma,\ w \in \Sigma^*)$$

また，言語 L のすべての語を逆順にして得られる言語を L^{-1} と表す．すなわち，L^{-1} は以下のように定義される．

$$L^{-1} = \{w^{-1} \mid w \in L\}$$

(1) で与えた言語 L_1 について，L_1^{-1} が正規言語であるかどうか答えよ．正規言語である場合には L_1^{-1} を認識するオートマトンを示し，そうでない場合には L_1^{-1} が正規言語ではないことを説明せよ．

(4) Σ 上の語 w の接尾辞(せつびじ)すべてからなる集合を $suffix(w)$ と表し，言語 L に含まれる語の接尾辞すべてからなる言語を $suffix(L)$ と表す．すなわち，$suffix$ は以下のように定義される．

$$suffix(w) = \{w'' \in \Sigma^* \mid \exists w' \in \Sigma^*.\ w = w'w''\} \qquad suffix(L) = \bigcup_{w \in L} suffix(w)$$

(1) で与えた言語 L_1 について，$suffix(L_1)$ が正規言語であるかどうか答えよ．正規言語である場合には $suffix(L_1)$ を認識するオートマトンを示し，そうでない場合には $suffix(L_1)$ が正規言語ではないことを説明せよ．

(5) Σ 上の任意の言語 L について $suffix(L) = (prefix(L^{-1}))^{-1}$ が成り立つことを証明せよ．

(6) Σ 上の任意の正規言語 L について $suffix(L)$ が正規言語であることを証明せよ．

Translation of technical terms

アルファベット	alphabet	語	word
言語	language	接頭辞	prefix
有限オートマトン	finite automaton	認識する	recognize
状態遷移図	state transition diagram	正規言語	regular language
決定性オートマトン	deterministic automaton	空集合	empty set
非決定性オートマトン	non-deterministic automaton	逆順にする	reverse order
初期状態	initial state	接尾辞	suffix
ϵ 遷移	ϵ-transition		

問 3

n を 2 以上の整数として，次の命題論理式 A, B, C_n, D_n を考える．

$$
\begin{aligned}
A &= (x_2 \supset x_1) \wedge (x_2 \vee x_3) \wedge \neg(x_1 \wedge x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3) \\
B &= A \wedge (x_1 \vee x_2 \vee \neg x_3) \\
C_n &= \bigvee_{1 \le i \le n} x_i \\
D_n &= \bigwedge_{1 \le i \le n-1} \ \bigwedge_{i+1 \le j \le n} (\neg x_i \vee \neg x_j)
\end{aligned}
$$

ここで，各 x_i は命題変数を表す．論理結合子の結合の強さは，強いものから順に否定 ($\neg$)，論理積 ($\wedge$)，論理和 ($\vee$)，含意 ($\supset$) とする．各 P_i が命題論理式であるとき，$\bigvee_{1 \le i \le n} P_i$ は $P_1 \vee \cdots \vee P_n$ の略記である．同じように，$\bigwedge_{1 \le i \le n} P_i$ は $P_1 \wedge \cdots \wedge P_n$ の略記である．命題変数またはその否定をリテラルという．リテラルを論理和で結合した論理式を節という．以下の問いに答えよ．

(1) A の真理値表を作成せよ．

(2) A を同値な連言標準形に変換せよ．

(3) 導出原理を用いて，B が充足不能であることを示せ．

(4) D_n に含まれる節の個数を n を使って表せ．

(5) $C_n \wedge D_n$ を真にする命題変数への真理値割り当ての総数を n を使って表せ．また，その理由を述べよ．

Translation of technical terms

命題論理式	propositional formula	節	clause
命題変数	propositional variable	真理値表	truth table
論理結合子	logical connective	同値な	equivalent
否定	negation	連言標準形	conjunctive normal form
論理積	conjunction	導出原理	resolution principle
論理和	disjunction	充足不能	unsatisfiable
含意	implication	真	true
リテラル	literal	真理値割り当て	truth assignment

問4

頂点の集合が V, 辺の集合が E の無向グラフを $G = (V, E)$ と表記する．頂点 v_1, v_2 を端点とする辺は $\{v_1, v_2\} \in E$ として表す．この問題では，ループや多重辺を持たない単純グラフのみを考え，グラフに関する用語を以下のように定義する．

- $G = (V, E)$ が完全グラフ: 異なる任意の頂点の組 $v_1, v_2 \in V$ に対し，辺 $\{v_1, v_2\} \in E$ が存在する．
- $G = (V, E)$ が正則グラフ: V の全ての頂点が同一の次数を持つ．頂点の次数とは，その頂点を端点に持つ辺の本数である．
- $G' = (V', E')$ が $G = (V, E)$ の補グラフ: $V' = V$ であり，異なる頂点の組 $v_1, v_2 \in V$ に対し，$\{v_1, v_2\} \notin E$ のとき，かつそのときのみ $\{v_1, v_2\} \in E'$．
- グラフ $G' = (V', E')$ が $G = (V, E)$ に同型: 次の性質を満たす全単射 $f : V \to V'$ が存在する; $\{v_1, v_2\} \in E$ のとき，かつそのときのみ $\{f(v_1), f(v_2)\} \in E'$．

(1) 頂点数 5 の完全グラフを示せ．

(2) 頂点数 n の完全グラフが持つ辺の本数を答えよ．

(3) 図 1 で与えられるグラフの補グラフを示せ．解答のグラフには頂点名を記すこと．

(4) 図 2 の (a) から (d) に示す 4 つのグラフのそれぞれについて，図 1 のグラフと同型であるかどうかを答え，同型の場合は頂点間の全単射を示し，同型でない場合は，同型でないと判断する理由を述べよ．

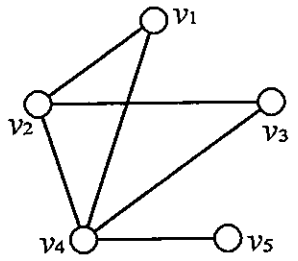

図 1

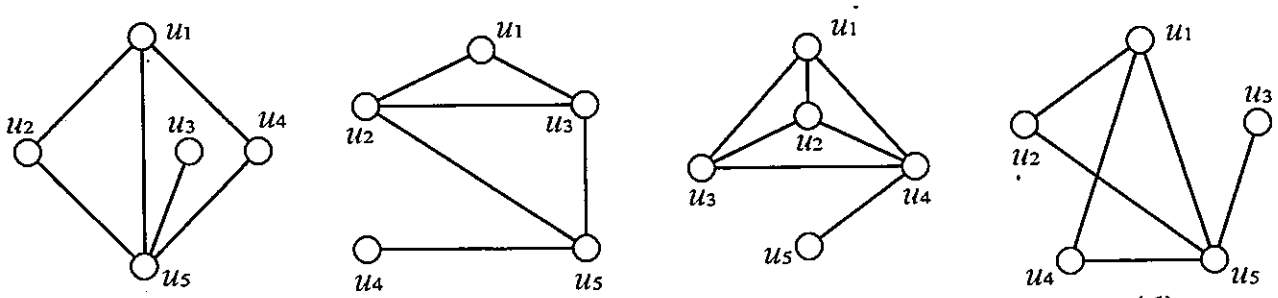

図 2

(5) $G = (V, E)$ の頂点数を n とする．G の補グラフが G 自身と同型となるとき，G の頂点数 n は非負整数 k を用いて $n = 4k$ または $n = 4k + 1$ として表されることを示せ．

(6) 前問 (5) において，さらに G が正則グラフであれば，$n = 4k + 1$ として表されることを示せ．

Translation of technical terms

頂点	vertex	完全グラフ	complete graph
辺	edge	組	pair
無向グラフ	undirected graph	正則グラフ	regular graph
端点	end point	次数	degree
ループ	loop	補グラフ	complement graph
多重辺	multiple edges	同型	isomorphic
単純グラフ	simple graph	全単射	bijective mapping

問5

1ビットの入力Xと1ビットの出力Zを有する同期式順序回路を考える。この回路は、動作開始前にはすべてのクロックサイクルにおいてXに0が入力されており、Zに0を出力している。この状態から回路が動作を開始しXに1が3回連続して入力されるとZに1を出力する。以降Xに0が2回連続して入力されるまでZに1を出力し続ける。Xに0が2回連続して入力されたあとは動作開始直後と同様にXに1が3回連続して入力されるまでZに0を出力し続ける。この回路の動作例を下表に示す。以下のすべての問いに答えよ。

クロックサイクル	1	2	3	4	5	6	7	8	9	10
入力 X	0	1	1	1	0	1	0	0	1	0
出力 Z	0	0	0	1	1	1	1	0	0	0

(1) この回路の次状態と出力を、右表にならって表形式で示せ。なお、(1)では各状態を右表のカッコ内に示す2進数の符号により定義する。この符号は最も左のビットが1クロックサイクル前のZの値を表し、右の2ビットが現在のクロックサイクルを含まない直近の2クロックサイクルのXの値（左ビットが古い）を表す。この回路において符号に対する上述の定義に従うと、符号（100）は存在しない。また、(1)では状態数が最小である必要はない。

現状態 (符号)	入力 X=0		入力 X=1	
	次状態	出力	次状態	出力
S_0(000)				
S_1(001)				
S_2(010)				
S_3(011)				
S_5(101)				
S_6(110)				
S_7(111)				

(2) この回路の状態数を最小化した場合の状態数を求めよ。状態数が最小であることを確認した方法も明示すること。また、状態数を最小化した場合の次状態と出力を、上表にならって表形式で示せ。なお、各状態への符号の割り当ては(1)の定義から自由に変更してよい。

(3) この回路を必要最小個数のDフリップフロップを用いて実現する。各Dフリップフロップの D 入力の論理値を与える論理関数の最小積和形表現を示せ。

(4) 出力Zを表す論理関数の最小積和形表現を示せ。

Translation of technical terms

同期式順序回路	synchronous sequential circuit	クロックサイクル	clock cycle
2進数	binary number	論理関数	logic function
最小積和形表現	minimum sum-of-products form		

問 6

5 段パイプライン処理プロセッサ（以下，プロセッサ A と呼ぶ）において，算術論理演算命令，ロード命令，ストア命令，および，条件分岐命令から構成される命令列（プログラム 1）を実行する．なお，# 以降は各行の命令を説明したコメントである．各パイプラインステージにおける処理内容について，表 1 に示す．プロセッサ A では，アドレス計算用と算術論理演算用の ALU（Arithmetic Logic Unit）が独立しているものとする．また，命令メモリとデータメモリについても独立しており，両者に同時に 1 クロックサイクルでアクセスできるものとする．レジスタへの書き込みについてはクロックサイクルの前半に完了し，読み出しについてはクロックサイクルの後半から開始できるものとする．このとき，以下の問いに答えよ．

プログラム1

```
命令1      sub  r3,r1,r4          # r3 = r1 - r4
命令2      add  r2,r3,r5          # r2 = r3 + r5
命令3   L1: addi r2,r2, 4          # r2 = r2 + 4
命令4      lw   r3,100(r2)        # r3 にアドレス [r2+100] のデータをロード
命令5      sub  r3,r3,r6          # r3 = r3 - r6
命令6      add  r4,r2,r6          # r4 = r2 + r6
命令7      sw   r3,100(r2)        # r3 のデータをアドレス [r2+100] にストア
命令8      bne  r2,r7,L1          # r2 と r7 が等しくなければ、L1に分岐
命令9      sub  r9,r1,r8          # r9 = r1 - r8
```

表1: 各パイプラインステージにおける処理内容

	IF	ID	EX	MEM	WB
算術論理演算命令	命令フェッチ	命令デコードおよびレジスタ読み出し	算術論理演算	N/A	レジスタ書き込み
ロード命令			アドレス計算	メモリ読み出し	
ストア命令				メモリ書き込み	N/A
条件分岐命令		命令デコード，レジスタ読み出し，および，分岐先アドレス計算	分岐条件の判定	プログラムカウンタの更新	

(1) プロセッサ A がフォワーディング機構を持たないとき，プログラム 1 の命令 1 から命令 7 までを実行する際の 平均 CPI（Clock cycles Per Instruction）を計算し，有効数字 2 桁で回答せよ．クロックサイクルごとのパイプライン処理の状況を図 1 の書式にならって示すこと．また，実行時にパイプラインストールが発生する命令を全て列挙し，その理由について説明せよ．

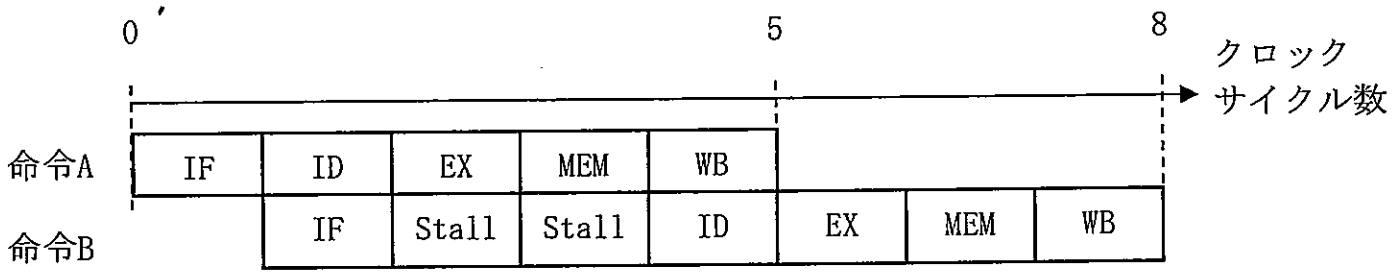

図1: パイプライン処理の状況

(2) プロセッサ A がフォワーディング機構を持つとき，プログラム 1 の命令 1 から命令 7 までを実行する際の平均 CPI を計算し，有効数字 2 桁で回答せよ．クロックサイクルごとのパイプライン処理の状況を図 1 の書式にならって示すこと．また，パイプラインストールが発生するか否かを理由と共に説明せよ．パイプラインストールが発生すると考える場合は，ストールが発生する命令を全て列挙するとともに，各ストールを解消する方法を示せ．

(3) フォワーディング機構を持つプロセッサ A 上で，命令 1 から命令 9 までを実行する．ただし，(2)において，命令 1 から命令 7 までを実行する際にパイプラインストールが発生すると考えた場合は，各ストールを解消する方法を適用しているものとする．このとき，パイプラインハザードが発生するか否かを理由と共に説明せよ．発生すると考える場合は，ハザードの名称とそのハザードを引き起こす命令を答え，そのハザードにより発生するパイプラインストールを低減する方法を 2 つ示せ．

Translation of technical terms

パイプライン処理	pipeline processing	命令デコード	instruction decode
プロセッサ	processor	レジスタ	register
算術論理演算	arithmetic-logic operation	アドレス	address
		プログラムカウンタ	program counter
命令	instruction	命令メモリ	instruction memory
ロード	load	データメモリ	data memory
ストア	store	クロックサイクル	clock cycle
条件分岐	conditional branch	フォワーディング	forwarding
プログラム	program	パイプラインストール	pipeline stall
パイプラインステージ	pipeline stage	パイプラインハザード	pipeline hazard
命令フェッチ	instruction fetch		

問7

オペレーティングシステムにおける断片化に関する以下の問いに答えよ。

(1) 動的メモリ割当てにおいて発生するメモリの外部断片化について，どのような現象であるか，それが発生すると何が問題であるかについて説明せよ。

(2) メモリの外部断片化を解決する方法の1つに，メモリの詰め直しがある。メモリの詰め直しについて説明し，その問題点について述べよ。

(3) メモリ割当てアルゴリズム（ファーストフィット，ベストフィットなど）が，メモリの外部断片化に与える影響について説明せよ。

(4) メモリの内部断片化について，外部断片化との違いが明確になるように説明せよ。

(5) ファイルの断片化は，メモリの断片化とは異なる現象である。ファイルの断片化について，メモリの断片化との違いが明確になるように説明せよ。

Translation of technical terms

断片化	fragmentation
動的メモリ割当て	dynamic memory allocation
外部断片化	external fragmentation
メモリの詰め直し	memory compaction
メモリ割当てアルゴリズム	memory allocation algorithm
ファーストフィット	first fit
ベストフィット	best fit
内部断片化	internal fragmentation

問 8

プログラム 1 は，配列 a に格納された 32 ビット符号付き整数を昇順に並べる C 言語プログラムである。以下の全ての問いに答えよ。

(1) プログラム 1 の [　　A　　] と [　　B　　] に単一の式を埋めよ。

(2) 配列 a に格納された整数を降順に並べるようプログラム 1 を 1 箇所だけ変更する。プログラム 1 の何行目をどのように変更すれば良いか答えよ。

(3) プログラム 1 の sort 関数では再帰呼出しを使用している。再帰呼出しを使用せずに同じ処理を実現する sort 関数をプログラム 2 のように実装した。プログラム 1 の sort 関数をプログラム 2 で置き換えた上で main 関数を実行したときに，プログラム 2 の 10 行目で標準出力に印字される実行結果を答えよ。

(4) (3) を踏まえて，プログラム 2 の merge 関数に関する冗長な関数呼出しを指摘し，改善方法を説明せよ。

(5) プログラム 1 の merge 関数では，配列 a と同じサイズの配列 b に処理結果を格納している。プログラム 1 の 5 行目を削除した上で，配列 a を直接操作する merge 関数をプログラム 3 のように実装した。プログラム 3 の [　　C　　] と [　　D　　] に単一の式を埋めよ。

(6) (5) のようにプログラムを実装することの利点と欠点を 1 つずつ答えよ。

Translation of Technical Terms

配列	array
符号付き整数	signed integer
昇順	ascending order
C 言語	C language
式	expression
降順	descending order
関数	function
再帰呼出し	recursive call
標準出力	standard output
印字する	print
冗長な	redundant

プログラム1

```
1  #include <stdio.h>
2  #define max 9
3
4  int a[max+1] = {10, -14, 19, -26, 27, 31, 33, 35, 42, 44};
5  int b[max+1];
6
7  void merge(int low, int mid, int high) {
8      int low1=low, low2=mid+1, i=low;
9      while (low1<=mid && low2<=high) {
10         if(a[low1] <= a[low2]) b[i++] = a[low1++];
11         else [    A    ];
12     }
13     while(low1 <= mid) b[i++] = a[low1++];
14     while(low2 <= high) b[i++] = a[low2++];
15     for(i = low; i <= high; i++) [    B    ];
16 }
17
18 void sort(int low, int high) {
19     int mid;
20     if(low < high) {
21         mid = (low + high) / 2;
22         sort(low, mid);
23         sort(mid + 1, high);
24         merge(low, mid, high);
25     }
26 }
27
28 int main() {
29     int i;
30     for(i = 0; i <= max; i++) printf("%d ", a[i]);
31     printf("\n");
32     sort(0, max);
33     for(i = 0; i <= max; i++) printf("%d ", a[i]);
34     printf("\n");
35 }
```

プログラム2

```
1  void sort(int low, int high) {
2      int size, low2, mid, high2;
3
4      for (size = 1; size <= (high - low); size = size*2) {
5          for (low2 = low; low2 < high; low2 += size*2) {
6              if ((low2 + size - 1) < high) mid = low2 + size - 1;
7              else mid = high;
8              if ((low2 + 2 * size - 1) < high) high2 = low2 + 2 * size - 1;
9              else high2 = high;
10             printf("(%d,%d,%d)", low2, mid, high2);
11             merge(low2, mid, high2);
12         }
13     }
14 }
```

プログラム3

```
1  void merge(int low, int mid, int high) {
2      int i, tmp, low1 = low, low2 = mid + 1, mid1 = mid;
3      if ([    C    ]) return;
4      while (low1 <= mid1 && low2 <= high) {
5          if (a[low1] > a[low2]) {
6              tmp = a[low2];
7              for (i = low2; low1 < i; i--) a[i] = a[i-1];
8              [    D    ];
9              low2++;
10             mid1++;
11         }
12         low1++;
13     }
14 }
```