平 成 ３０ 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

平成２９年８月３日（木）
１２：３０～１５：３０

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．外国人留学生は英語で解答してよい。また、和英辞書などの辞書を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．問題冊子、解答用紙3枚、草稿用紙3枚が配布されていることを確認せよ。

5．問題は(1)解析・線形代数、(2)確率・統計、(3)プログラミング、(4)計算機理論、(5)ハードウェア、(6)ソフトウェアの６科目がある。(1)~(3)の３科目から１科目以上、(4)~(6)の３科目から１科目以上を選択し、(1)~(6)の６科目から合計３科目を選択して解答せよ。
なお、選択した科目名を解答用紙の指定欄に記入せよ。
(1)(2)(3)や(4)(5)(6)の組み合わせを選択した場合は0点となる。

6．解答用紙は指定欄に受験番号を必ず記入せよ。解答用紙に受験者の氏名を記入してはならない。

7．解答用紙表面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せよ。

8．解答用紙はホッチキスを外さず、試験終了後に3枚とも提出せよ。

9．問題冊子、草稿用紙は試験終了後に持ち帰ってよい。

解析・線形代数

（解の導出過程も書くこと）

[1] 次の関数 f について、以下の問いに答えよ。

$$f(x,y) = \sin x + \sin y + \sin(x+y) \qquad (0 < x < \pi,\ 0 < y < \pi)$$

(a) f の停留点を求めよ。

(b) (a) で求めた停留点に対して極値をとるかどうかを判定し、f の極値を求めよ。

[2] 以下の問いに答えよ。ただし、$\mathrm{Im}(z)$ は複素数 z の虚部を表す。

(a) z 平面上の直線 $\mathrm{Im}(z) = \dfrac{1}{2}$ が複素関数 $w = \dfrac{1}{z}$ によって写される w 平面上の図形を求め、図示せよ。

(b) z 平面上の領域 $\mathrm{Im}(z) > 0$ が 1 次分数変換 $w = \dfrac{\alpha z + \beta}{z + \gamma}$ によって w 平面上の領域 $|w| < 1$ に写されるとき、複素数 α, β, γ を求めよ。

[3] 次の対称行列 A について、以下の問いに答えよ。

$$A = \begin{pmatrix} 5 & -3 \\ -3 & 5 \end{pmatrix}$$

(a) A のすべての固有値および各固有値に対する単位固有ベクトルを求めよ。

(b) A を対角化する直交行列 $P = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ のうち $a = d$ のものを求め、A を対角化せよ。

(c) ある 1 次変換 $\begin{pmatrix} x \\ y \end{pmatrix} = U \begin{pmatrix} x' \\ y' \end{pmatrix}$ により、2 次形式 $Q(x,y) = 5x^2 - 6xy + 5y^2$ を $x'y'$ の項を含まない 2 次形式に変形せよ。

(d) 2 次曲線 $5x^2 - 6xy + 5y^2 = 8$ の概形を図示せよ。

Translation of technical terms

関数	function	停留点	stationary point
極値	extreme value	複素数	complex number
虚部	imaginary part	平面	plane
直線	line	複素関数	complex function
図形	shape	領域	region
1 次分数変換	linear fractional transformation	対称行列	symmetric matrix
固有値	eigenvalue	単位固有ベクトル	unit eigenvector
対角化	diagonalization	直交行列	orthogonal matrix
1 次変換	linear transformation	2 次形式	quadratic form
項	term	2 次曲線	quadratic curve
概形	rough sketch		

確率・統計

解の導出過程も書くこと。

[1]　ある事象が2分間に平均1回発生する場合、10分間でその事象が3回発生する確率を有効数字2桁で求めなさい。ただし、その事象の生起回数の確率はポアソン分布に従うものとし、$e^{-1} = 0.37, e^{-2} = 0.14, e^{-3} = 0.050, e^{-4} = 0.018, e^{-5} = 0.0067$ とする。

[2]　統計学的仮説検定の手順を、帰無仮説および有意水準という用語を説明した上で、それらを用いて300字程度（about 250 words for English）で説明しなさい。

[3]　互いに独立な確率変数 X と Y がともに正規分布 $N(\mu, \sigma^2)$ に従うとする。この確率密度関数は $\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$ で表される。このとき、以下の問いに答えなさい。

(1) 確率変数 X のモーメント母関数（積率母関数）は e^{tX} の期待値、つまり $E[e^{tX}]$ で定義される。X のモーメント母関数を μ と σ を使って表しなさい。

(2) 確率変数 Z を $Z = X + Y$ とする。Z のモーメント母関数を求め、Z も正規分布に従うことを示しなさい。

Translation of Technical Terms

- 事象　event
- 平均　average
- 有効数字　significant digit
- 生起回数　number of occurrences
- ポアソン分布　Poisson distribution
- 統計学的仮説検定　statistical hypothesis testing
- 帰無仮説　null hypothesis
- 有意水準　level of significance
- 互いに独立　mutually independent
- 確率変数　random variable
- 正規分布　normal distribution
- 確率密度関数　probability density function
- モーメント母関数（積率母関数）　moment-generating function
- 期待値　expected value

プログラミング

ソースコード 1 は整数の集まりを操作するためのデータ構造を扱う C言語プログラムである．このプログラムについて以下の問いに答えよ．

(1) ソースコード 1 を実行した際に標準出力に出力される文字列を書け．

(2) ポインタ head から参照される整数の集まりは，関数 insert，top，eliminate が実現している操作の下でどのようなデータ構造をなしているか？ 下記の中から最も適切なものを 1 つ選択せよ．

(a) LIFO　(b) FIFO　(c) ヒープ　(d) 2分木　(e) DAG

(3) 関数 top はポインタ head の値が NULL であるときに -1 を返す．この実装では，保持されているデータが存在するかどうかを関数 top が返す値からは必ずしも正しく判断できない．正しく判断できない場合を 30 文字（英語の場合は 20 単語）以内で説明せよ．

(4) ソースコード 1 では関数 insert によってデータを追加するときにすべてのデータを走査する必要があり効率が悪い．これを改善する方法として，ポインタ head とは別に末端の要素のアドレスを保持するポインタ tail を導入して以下のように改変することが考えられる．以下の［（ア）］～［（ス）］を埋めて関数 insert，eliminate を完成させよ．

- 11 行目に以下の文を追加する．

```
CELL tail = NULL;
```

- 関数 insert，eliminate のソースコードを以下のように変更する．

```
void insert(int i) {
  CELL c = (CELL) malloc(sizeof(struct cell));
  c->num = i;  c->next = NULL;
  if(tail != NULL) {
    （ア） = （イ）;
  } else {
    （ウ） = （エ）;
  }
  （オ） = （カ）;
}

void eliminate() {
  if(head != NULL) {
    if( （キ） != NULL) {
      （ク） = （ケ）;
    } else {
      （コ） = （サ）;
      （シ） = （ス）;
    }
  }
}
```

(5) ソースコード1の関数 eliminate ではメモリ領域を解放すべきところで解放していない．適当な2行を追加して，不要になったメモリ領域を解放するように修正せよ．なお，解答は「○○行目と次の行の間に△△を挿入」という形式で記し，追加する2行が連続していない場合は追加する行ごとに「○○行目と次の行の間に△△を挿入」という形式で記すこと．また，メモリ解放には標準ライブラリ関数である free を使用すること．

(6) 保持するデータの個数の最大数を決められる場合には，ソースコード1のような動的にサイズが変化するデータ構造ではなく，配列を用いても同様の処理を実現できる．ソースコード1の実行結果と一致するように，ソースコード2の中の［（セ）］～［（タ）］を埋めよ．

Translation of technical terms

ソースコード	source code	値	value
整数	integer	返す	return
データ構造	data structure	実装	implementation
C 言語	C programming language	走査する	scan
プログラム	program	保持する	store
実行する	execute	文	statement
標準出力	standard output	メモリ領域	memory space
出力する	output	解放する	free
文字列	string	標準ライブラリ	standard library
ポインタ	pointer	動的に	dynamically
参照する	refer to	配列	array
関数	function	処理	processing
ヒープ	heap	実行結果	result of execution
2分木	binary tree		

ソースコード 1:

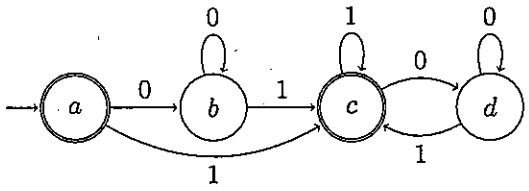
```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct cell {
5    int num;
6    struct cell *next;
7  };
8
9  typedef struct cell *CELL;
10 CELL head = NULL;
11
12 void insert(int i) {
13   CELL c = (CELL) malloc(sizeof(struct cell));
14   CELL tmp = head;
15   c->num = i; c->next = NULL;
16
17   if(head != NULL) {
18     while(tmp->next != NULL) tmp = tmp->next;
19     tmp->next = c;
20   } else {
21     head = c;
22   }
23 }
24
25 int top() {
26   if(head != NULL) {
27     return head->num;
28   } else {
29     return -1;
30   }
31 }
32
33 void eliminate() {
34   if(head != NULL) {
35     head = head->next;
36   }
37 }
38
39 void display() {
40   CELL tmp = head;
41   while(tmp != NULL) {
42     printf("%d;", tmp->num);
43     tmp = tmp->next;
44   }
45   printf("\n");
46 }
47
48 int main() {
49   insert(0); insert(4); insert(9); insert(3);
50   display();
51   printf("%d\n", top());
52   eliminate(); eliminate(); insert(7); insert(2);
53   display();
54   return 0;
55 }
```

ソースコード 2:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  #define MAX 5
5
6  int ar[MAX];
7
8  int head = 0;
9  int cnt = 0;
10
11 void insert(int i) {
12   if(cnt >= MAX) {
13     printf("error\n");
14     exit(1);
15   }
16   ar[  (セ)  ] = i;
17   cnt++;
18 }
19
20 int top() {
21   if(cnt > 0) {
22     return ar[head];
23   } else {
24     return -1;
25   }
26 }
27
28 void eliminate() {
29   if(cnt > 0) {
30     head =  (ソ)  ;
31     cnt--;
32   }
33 }
34
35 void display() {
36   int i = 0;
37   while(i < cnt) {
38     printf("%d;", ar[  (タ)  ]);
39     i++;
40   }
41   printf("\n");
42 }
43
44 int main() {
45   insert(0); insert(4); insert(9); insert(3);
46   display();
47   printf("%d\n", top());
48   eliminate(); eliminate(); insert(7); insert(2);
49   display();
50   return 0;
51 }
```

計算機理論

[1] 以下の問いに答えよ．ここで，M は次図で与えられる決定性有限オートマトンである．



オートマトン M

(1) M が認識する言語 $L(M)$ に属する長さ 2 以下の文字列をすべて示せ．

(2) 決定性有限オートマトンは、アルファベットの有限集合 Σ，状態の有限集合 Q，初期状態 q_0 $(\in Q)$，遷移関数 $\delta : Q \times \Sigma \to Q$，最終状態の集合 Q_f $(\subseteq Q)$ から定められる．決定性有限オートマトン $A = \langle \Sigma, Q, q_0, \delta, Q_f \rangle$ に対する関係 $\equiv_A$ $(\subseteq Q \times Q)$ を次のように定義する．

$$q \equiv_A q' \overset{\text{def}}{\iff} \forall w \in \Sigma^* \ (\bar{\delta}(q, w) \in Q_f \iff \bar{\delta}(q', w) \in Q_f)$$

ここで $\bar{\delta} : Q \times \Sigma^* \to Q$ は δ の自然な拡張である．このとき，オートマトン M について以下の各問に答えよ．

(a) $p \equiv_M p'$ を満たさない M の状態 p, p' を一組示し，その理由を説明せよ．

(b) $p \equiv_M p'$ を満たす M の状態 p, p' の組をすべて示せ．

(c) $L(M) = L(M')$ を満たす決定性有限オートマトン M' のうちで状態数が最小なものを図示せよ．

(3) (2) で定義される関係 $\equiv_A$ が推移律を満たすことを証明せよ．

Translation of technical terms

決定性有限オートマトン	deterministic finite automaton	初期状態	initial state
認識する	recognize	遷移関数	transition function
言語	language	最終状態	final state
長さ	length	関係	relation
文字列	string	自然な拡張	natural extension
アルファベット	alphabet	最小	minimum
有限集合	finite set	推移律	transitivity
状態	state		

[2] 命題論理に関する以下の問いに答えよ．以下で用いる論理結合子を以下のように記述する．

否定: $\neg$, 論理積: $\wedge$, 論理和: $\vee$, 含意: $\supset$

論理結合子の結合の強さは，強い方から $\neg, \wedge, \vee, \supset$ として括弧を省略して命題論理式を記述する．命題論理に対する証明系LKの推論規則を問題の最後に示す．

証明図の記法に関する注意:
解答において証明図 (およびその一部) は以下の例のように適用する推論規則を一段階ごとに記載すること．

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{A \to A}{\neg A, A \to}\,(\text{L-neg})}{\neg A, A \to B}\,(\text{R-weak}) \quad \dfrac{\dfrac{B \to B}{A, B \to B}\,(\text{L-weak})}{B, A \to B}\,(\text{L-ex})}{\neg A \vee B, A \to B}\,(\text{L-or})}{A, \neg A \vee B \to B}\,(\text{L-ex})}{\neg A \vee B \to A \supset B}\,(\text{R-imp})}{\to \neg A \vee B \supset (A \supset B)}\,(\text{R-imp})$$

なお同じ推論規則を連続して適用する場合には，以下のように推論規則の名前に $*$ をつけて推論規則を一回適用するように書いてもよい．例えば，

$$\dfrac{\dfrac{A, B, C \to D}{B, A, C \to D}\,(\text{L-ex})}{B, C, A \to D}\,(\text{L-ex}) \quad \text{は} \quad \dfrac{A, B, C \to D}{B, C, A \to D}\,(\text{L-ex})^* \quad \text{のように書いてよい．}$$

(1) A, B を命題論理式，Γ, Δ を命題論理式の系列とするとき，空欄部分 (ア) に LK の推論規則を何度か適用してあらたな推論規則を作成せよ．

$$\dfrac{A, B, \Gamma \to \Delta}{\boxed{\text{(ア)}}}$$
$$A \wedge B, \Gamma \to \Delta$$

(2) (1) で導かれた推論規則を (L-seq) とする．

$$\dfrac{A, B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}\,(\text{L-seq})$$

p, q, r, s を命題変数とするとき，以下の命題論理式 P_1, P_2 が恒真であるかどうか示せ．恒真である場合は，LK の推論規則を用いて証明図を示せ．恒真でない場合は，論理式が偽となる付値を示せ．

(a) $P_1 : p \wedge (\neg r \vee s) \supset (((s \supset q) \wedge \neg p) \supset \neg r)$

(b) $P_2 : \neg(p \supset (q \vee r)) \supset \neg((q \wedge (p \vee r)) \supset (\neg r \supset \neg p))$

(3) 命題論理式 F に現れる命題変数の集合を $\mathsf{Var}(F)$ と書く．命題変数 p に対して，$p \vee \neg p$ を $\top$，$p \wedge \neg p$ を $\bot$ で表すことにする．命題変数 s を $\top$ および $\bot$ で置き換える操作をそれぞれ f_1, f_2 とし，f_1, f_2 を命題論理式 F に適用して得られる命題論理式をそれぞれ $f_1(F)$, $f_2(F)$ と書く．

(a) r, s に対する付値 v が与えられたとき，$v'(r) = v(r)$ であるような p, r に対する付値 v' を考える．このとき，$\mathsf{Var}(D) \subseteq \{r, s\}$ である任意の命題論理式 D について，$v(s)$ が真のとき $v(D) = v'(f_1(D))$，$v(s)$ が偽のとき $v(D) = v'(f_2(D))$ であることを命題論理式 D の構造に関する帰納法を用いて示せ．

(b) 命題論理式 A, B に対して $\mathsf{Var}(A) = \{q, r\}$, $\mathsf{Var}(B) = \{r, s\}$ とし，$A \supset B$ が恒真であるとする．$A \supset f_1(B)$，$A \supset f_2(B)$ はいずれも恒真となることを示すことによって，$A \supset f_1(B) \wedge f_2(B)$ が恒真となることを示せ．

Translation of technical terms

命題論理	propositional logic	証明図	proof diagram
論理結合子	logical connective	適用する	apply
否定	negation	命題変数	propositional variable
論理積	conjunction	恒真である	valid
論理和	disjunction	偽	false
含意	implication	付値	assignment
命題論理式	propositional logic formula	置き換える	substitute
証明系	proof system	真	true
推論規則	inference rule	構造に関する帰納法	structural induction

証明系 LK の推論規則

各規則の右側の括弧の中に各推論規則の名前を示す．

$$\frac{\Gamma \to \Delta}{A, \Gamma \to \Delta}\ (\text{L-weak}) \qquad \frac{\Gamma \to \Delta}{\Gamma \to \Delta, A}\ (\text{R-weak})$$

$$\frac{A, A, \Gamma \to \Delta}{A, \Gamma \to \Delta}\ (\text{L-cont}) \qquad \frac{\Gamma \to \Delta, A, A}{\Gamma \to \Delta, A}\ (\text{R-cont})$$

$$\frac{\Gamma, A, B, \Pi \to \Delta}{\Gamma, B, A, \Pi \to \Delta}\ (\text{L-ex}) \qquad \frac{\Gamma \to \Delta, A, B, \Sigma}{\Gamma \to \Delta, B, A, \Sigma}\ (\text{R-ex})$$

$$\frac{\Gamma \to \Delta, A \quad A, \Pi \to \Sigma}{\Gamma, \Pi \to \Delta, \Sigma}\ (\text{cut})$$

$$\frac{A, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}\ (\text{L-and1}) \qquad \frac{B, \Gamma \to \Delta}{A \wedge B, \Gamma \to \Delta}\ (\text{L-and2})$$

$$\frac{\Gamma \to \Delta, A \quad \Gamma \to \Delta, B}{\Gamma \to \Delta, A \wedge B}\ (\text{R-and}) \qquad \frac{A, \Gamma \to \Delta \quad B, \Gamma \to \Delta}{A \vee B, \Gamma \to \Delta}\ (\text{L-or})$$

$$\frac{\Gamma \to \Delta, A}{\Gamma \to \Delta, A \vee B}\ (\text{R-or1}) \qquad \frac{\Gamma \to \Delta, B}{\Gamma \to \Delta, A \vee B}\ (\text{R-or2})$$

$$\frac{\Gamma \to \Delta, A \quad B, \Pi \to \Sigma}{A \supset B, \Gamma, \Pi \to \Delta, \Sigma}\ (\text{L-imp}) \qquad \frac{A, \Gamma \to \Delta, B}{\Gamma \to \Delta, A \supset B}\ (\text{R-imp})$$

$$\frac{\Gamma \to \Delta, A}{\neg A, \Gamma \to \Delta}\ (\text{L-neg}) \qquad \frac{A, \Gamma \to \Delta}{\Gamma \to \Delta, \neg A}\ (\text{R-neg})$$

ここで，A, B は命題論理式，$\Gamma, \Delta, \Pi, \Sigma$ は命題論理式の系列を表す．

ハードウェア

[1] プロセッサと主記憶との間のキャッシュに関する以下の問いに答えよ．

(1) セットアソシアティブ方式のキャッシュにおいて，キャッシュミス発生時に置き換え対象のブロックを選択する方式として，ランダム法と LRU 法がある．

(ア) 2 種類の方式のうち一方は，多くのプログラムに対し，他方よりもキャッシュミス率を低減することが経験的に知られている．その方式を，理由と共に答えよ．なお，理由は 60 字（英語の場合，25 語）程度で書け．

(イ) 実際のキャッシュではキャッシュの構成を考慮して，2 種類の方式から選択して適用される．その理由をキャッシュの連想度と関連させて 100 字（英語の場合，45 語）程度で説明せよ．

(2) キャッシュミスは次の 3 種類に分類可能である．
・初期参照ミス
・容量性ミス
・競合性ミス
それぞれのミスについて，キャッシュの構成を変更してミス率を低減させる最も効果的な方法について 20 字（英語の場合，10 語）程度で説明せよ．またその場合のデメリットについても 60 字（英語の場合，25 語）程度で説明せよ．

(3) キャッシュへの書き込み方式として，ライト・スルー方式とライト・バック方式がある．

(ア) それぞれの方式を 80 字（英語の場合，35 語）程度で説明せよ．

(イ) ライト・スルー方式がライト・バック方式より高速となる状況について 150 字（英語の場合，70 語）程度で説明せよ．

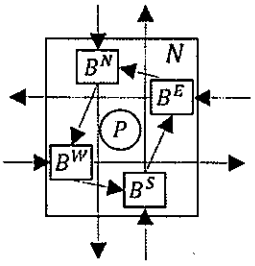


図1

[2] 1 台のプロセッサPと 4 個のバッファB^d $(d \in \{N,W,S,E\})$を有するノードN（図 1）を，縦n行×横m列$(n,m \geq 2)$の 2 次元メッシュ状に接続した並列プロセッサ$C(n,m)$を考える．図 2 に$C(2,2)$の例を示す．ここで，矢印で接続されている二つのバッファは隣接するという．なお，図 2 の×印で示されるように，並列プロセッサ$C(n,m)$の端には始終点のいずれかにバッファが存在しない矢印が存在するが，そのような矢印は無視して考える．

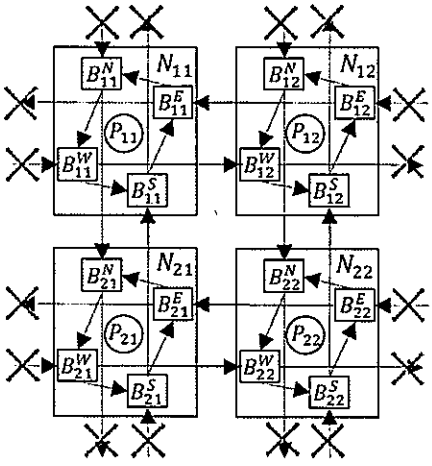


図2

このとき$C(n,m)$上の異なる 2 プロセッサ間におけるデータ通信を考える．1 回の通信においては，

まず送信元プロセッサP_{pq}が，送信先プロセッサ名P_{rs}を付与したデータ（以下では，通信データとよぶ）を生成する．そして，生成された通信データがバッファを経由して送信先プロセッサP_{rs}に送られる．バッファには最大１つの通信データを格納でき，通信データが存在しないバッファの状態を空という．並列プロセッサ$C(n,m)$は時刻$t=0,1,\cdots$を刻むクロック信号を持ち，プロセッサーバッファ間，バッファーバッファ間では，以下に説明するようにクロックに同期してデータが転送される．

- プロセッサーバッファ間通信：送信元プロセッサP_{pq}は，ノードN_{pq}内のいずれかのバッファB_{pq}^{d}に対し，時刻$t=k-1$にB_{pq}^{d}が空であれば通信データDを送ることができ，時刻$t=k$においてDはB_{pq}^{d}に格納される．また時刻$t=k-1$に，通信データDの送信先プロセッサP_{rs}を含むノードN_{rs}内のいずれかのバッファB_{rs}^{d}にDが格納された場合，時刻$t=k$にP_{rs}はDを受け取り，B_{rs}^{d}は空となる．

- バッファーバッファ間通信：隣接するバッファ間において，通信データを送ることが可能である．ここで，矢印の始点，終点に接続するバッファをそれぞれB，B'とする．時刻$t=k-1$にBに通信データDが格納されB'が空のとき，時刻$t=k$にBからB'にDを送ることができる．送信後，Bは空となり，DがB'に格納される．

- ルーティング：各ノードN_{pq}において，プロセッサP_{pq}が送信先でない通信データDが各バッファB_{pq}^{d}に格納されると，あらかじめ決められた手順に従い，Dを送るべきバッファB'が一意に決定される．この手順をルーティングアルゴリズムとよぶ．なおB'はB_{pq}^{d}に隣接している必要があるが，N_{pq}以外のノードに含まれる場合もある．特に，通信データが通過する矢印の数を最小にするようなルーティングアルゴリズムを，最短経路ルーティングとよぶ．例えば図２において，B_{11}^{N} に格納されている通信データDの送信先プロセッサがP_{22}であるとき，最短経路ルーティングを用いると，B_{21}^{N}, B_{21}^{W}を経由してB_{22}^{W}に送られ，P_{22}がB_{22}^{W}からDを受け取る．

（注）同一時刻に，一つの空のバッファに対し複数の通信データが送られる場合がある．その場合にはいずれかの通信データのみが送られ，他は待たされるものとする．必要ならば，以下の問いにおいて，送られる通信データの決定方式について，問題の意味を変えない範囲内で仮定してよい．

これに関して次の問いに答えよ．

(1) 図２の並列プロセッサ$C(2,2)$において，時刻$t=0$に，プロセッサP_{21}に送られるべき通信データD_1がバッファB_{22}^{N}に，プロセッサP_{12}に送られるべき通信データD_2がバッファB_{22}^{W}にあるとする．その他のバッファが空であり，最短経路ルーティングが用いられているとき，D_1とD_2が送信先プロセッサに到着するまでに起こるバッファ間送信を時刻ごとに列挙せよ．

(2) 並列プロセッサ$C(n,m)$上に複数の通信が同時に存在するとき，バッファ内の通信データを，ルーティングアルゴリズムによって指定されたバッファに一つも送ることができず，永久に通信を進めることができなくなる状況が起こる場合がある．このような状況を表す用語を答えよ．

(3) (2)の状況が起こる例を一つ，図を用いて説明せよ．

(2)の状況を回避するルーティングアルゴリズムとして，次元順ルーティングとよばれる方法が知られている．例えば，X-Y 方向の 2 次元メッシュ構造を持つネットワークアーキテクチャにおいて，すべての通信データを，まず X 方向に送信した後に Y 方向に送る方法である．ここで，Y 方向に送信した後に X 方向に送ることができないようバッファ間接続関係を修正したノードN'を図 1'に示す．また，ノードN'を縦n行×横m列$(n, m \geq 2)$の 2 次元メッシュ状に接続した並列プロセッサを$C'(n, m)$とする．

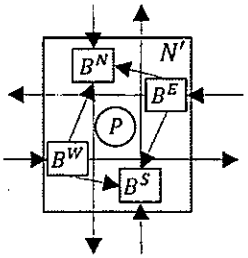


図1'

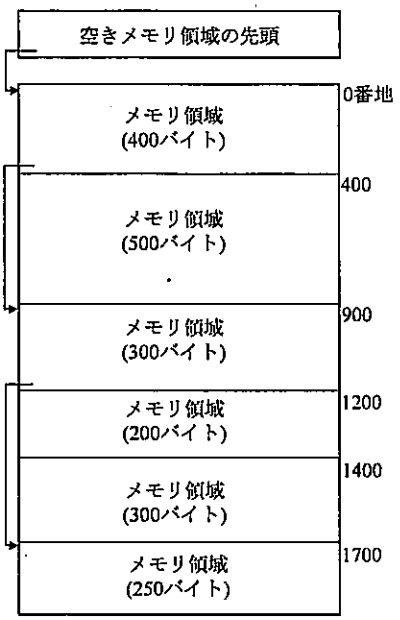
(4) $C'(n, m)$上で次元順ルーティングを適用した場合，(2)の状況が起こらない理由を，図を用いて説明せよ．

Translation of technical terms

主記憶	main memory	送信元	source
連想度	associativity	送信先	destination
初期参照ミス	compulsory miss	空	empty
容量性ミス	capacity miss	始点	start point
競合性ミス	conflict miss	終点	end point
隣接	adjacency	最短経路	shortest path
通信	communication	次元	dimension

ソフトウェア

[1] 連続した大きいメモリ領域から，指定したサイズの連続したメモリ領域の動的な割当てとその解放を繰り返し行う場合を考える．下図は，ある時点でのメモリ領域全体の状態とそれを管理するためのデータ構造を図示したものである．空きメモリ領域は空きメモリ領域の一部分を利用したリストによって管理しており，リストにつながれていないメモリ領域は使用中の領域である．図の右側の数字は10進数で表した番地を表している．



これに関して以下の問いに答えよ．なお，メモリ領域を管理するために必要な管理領域は無視してよい．

(1) この図の状態で，250バイトの連続するメモリ領域の割当てを要求した場合に，どのようにメモリが割り当てられるか．割当てアルゴリズムがファーストフィット(first-fit)の場合，ベストフィット(best-fit)の場合，ワーストフィット(worst-fit)の場合，それぞれについて次のような表記方法で答えよ．先頭の400バイトのメモリ領域の場合，「0番地からはじまる400バイトのメモリ領域」と表記する．

(2) この図の状態から1200番地からはじまる200バイトのメモリ領域を解放する．その後，200バイトの連続するメモリ領域を割当て，さらに100バイトの連続するメモリ領域を割当てたときの，メモリ領域全体の状態とそれを管理するデータ構造はどのようになるか．この図と同様に図示し，メモリ領域のバイト数とメモリ領域の開始番地を記入せよ．ただし，メモリ割当てアルゴリズムはベストフィットとする．

(3) ファーストフィット，ベストフィットを比較して，それぞれの優れている点を100字以内（英語の場合には50語以内）で説明せよ．

(4) メモリの割当てと解放を繰り返し行うと，合計すると必要な空きメモリ容量があるにもかかわらず，割当てに失敗することがある．こうした現象を何と呼ぶか．また，その現象を回避するための方法の一つを100字以内（英語の場合には50 語以内）で説明せよ．

Translation of technical terms

メモリ領域	memory area	リスト	list
動的	dynamic	10 進数	decimal number
割当て	allocation	番地	address
解放	release	バイト	byte
データ構造	data structure	アルゴリズム	algorithm
空き	free		

[2] コンパイラにおける最適化に関する以下の問いに答えよ．

(1) 以下の3番地コードを基本ブロックに分割し記述せよ．各基本ブロックは行番号の列として記述せよ．例えば，L1行目，L2行目，L3行目からなる基本ブロックであれば [L1, L2, L3] と記述せよ．

```
L1  i = 1
L2  j = 1
L3  a = 10 + i
L4  b = a * j
L5  c = 3 + b
L6  d = c + 7
L7  x[c] = 0.0
L8  j = j + 1
L9  if j <= 10 goto L3
L10 i = i + 1
L11 if i <=10 goto L2
L12 i = 1
L13 f = i - 1
L14 g = 32 * f
L15 x[g] = 1.0
L16 i = i + 1
L17 if i <= 10 goto L13
```

(2) (1)で記述した基本ブロックを $B_1, B_2, \ldots, B_{n-1}, B_n$ とおいたときに，これら基本ブロックを節点とする制御フローグラフを記述せよ．

(3) コンパイラにおける最適化では，基本ブロックをDAG（Directed Acyclic Graph, 有向非巡回グラフ）に変換することがある．例えば，以下の基本ブロックは，図1のDAGに変換される．

```
1 a = b + c
2 b = a - d
3 c = b + c
4 d = a - d
```

次の基本ブロックをDAGに変換せよ．

```
1 a = b + c
2 b = b - d
3 c = c + d
4 e = b + c
```

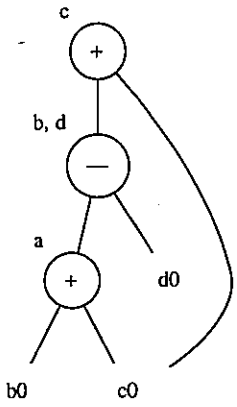


図 1: DAG

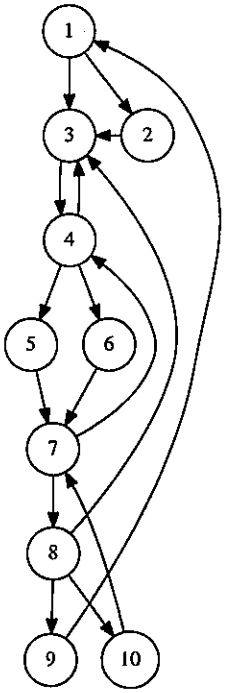


図 2: 制御フローグラフ

(4) 制御フローグラフの入口節から節点 n に到達するどの経路も，節点 d を通るとき，d は n を支配するという．支配木は，制御フローグラフの各節点を節点とする木である．支配木は，入口節を根とし，各節点 d は制御フローグラフにおいて支配する節点を子孫とする．図 2 の制御フローグラフについて，節点 1 を入口節とする支配木を記述せよ．

(5) 以下は，コンパイラの最適化において，制御フローグラフからループとして検出される部分グラフの定義である．

（ア）ヘッダと呼ばれるただひとつの入口節をもつ．

（イ）任意の節点からヘッダに戻る経路が存在する．

（ア）の入口節は何を支配するか？10 文字（英語の場合，5 語）程度で答えよ．

Translation of technical terms

コンパイラ	compiler	入口節	entry node
最適化	optimization	経路	path
3番地コード	three-address code	支配する	dominate
基本ブロック	basic block	支配木	dominator tree
行番号	line number	木	tree
列	list	根	root
制御フローグラフ	control flow graph	子孫	descendant
節点	node	ループ	loop