平成２５年度

名古屋大学大学院情報科学研究科
情報システム学専攻
入学試験問題

専門

平成２４年８月９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。(４)～(６)の３科目から少なくとも１科目を選択して解答し、(１)～(３)を含めた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。

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

（解の導出過程を書くこと）

[1] 2次形式$Q(\boldsymbol{x}) = 13{x_1}^2 + 6\sqrt{3}x_1x_2 + 7{x_2}^2$ を考える．ここで，$\boldsymbol{x}^\top = (x_1, x_2)$ とする．なお，$^\top$ は転置を表わす．このとき，次の問いに答えよ．

(a) ある行列 $\boldsymbol{A}$ を用いて，$Q(\boldsymbol{x}) = \boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x}$ と表すことができる．このような行列 $\boldsymbol{A}$ のうち，対称 行列であるものを求めよ．

(b) (a) で求めた行列 $\boldsymbol{A}$ の固有値を求めよ．また，各固有値に対応する固有ベクトルを求めよ．ただし，固有ベクトルの大きさを 1 に正規化せよ．

(c) $\boldsymbol{A}$ のような対称行列は，ある直交 行列 $\boldsymbol{U}$ により $\boldsymbol{D} = \boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{U}$ のように対角化できることが知られている．このような直交行列 $\boldsymbol{U}$ 及び対角 行列 $\boldsymbol{D}$ を求めよ．

(d) 線形変換 $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{y}$ を考えたときに，2 次形式 $Q(\boldsymbol{x})$ は $Q(\boldsymbol{y}) = \boldsymbol{y}^\top \boldsymbol{D}\boldsymbol{y}$ のように 標準形 で表せることが知られている．ここで，$\boldsymbol{y}^\top = (y_1, y_2)$ とする．

このことと，行列 $\boldsymbol{U}$ の性質をふまえ，$Q(\boldsymbol{x}) = 4^2$ を満たすすべての点の 集合 が描く図形について，その大きさや 傾 きが分かるように x_1-x_2 平面上に図示せよ．

[2] 複素平面上の点 $z = x + iy$ のうち，$|z| < 1$ を満たす点からなる 領域 Z を考える．このとき，次の問いに答えよ．

(a) $|z| < 1$ を満たす点における x と y の関係を表す式を求めよ．

(b) 領域 Z を複素平面上に図示せよ．

(c) 写像 $f : w = \frac{1}{z-1}$ を考える．このとき，領域 Z が f により写像される領域 W を複素平面上に図示せよ．ここで，$w = u + iv$ とする．

[3] 線形微分方程式 $y' - xy = x$ について考える．ここで $y' = \frac{dy}{dx}$ とする．

(a) $y' - xy = 0$ の一般解を求めよ．

(b) (a) の結果を用いて，$y' - xy = x$ の一般解を求めよ．

Translations of technical terms

2次形式	quadratic form	転置	transpose
行列	matrix	対称 行列	symmetric matrix
固有値	eigenvalue	固有ベクトル	eigenvector
大きさ	size	正規化	normalize
直交 行列	orthogonal matrix	対角化	diagonalization
対角 行列	diagonal matrix	線形 変換	linear transformation
標準形	normal form	点	point
集合	set	図形	shape
傾 き	tilt	平面	plane
複素平面	complex plane	領域	area
式	equation	写像	mapping
線形微分 方程式	linear differential equation	一般解	general solution

確率・統計　（解の導出過程も書くこと.）

［1］　以下の問いに答えよ.　下記(1),(2)の答えは既約分数で書け.

(1) あるコインを投げたときの表の出る確率は$\frac{1}{2}$である．このコインを3回投げた結果，3回とも表の出る確率を求めよ.

(2) あるコインを投げたときの表の出る確率は$\frac{1}{3}$である．このコインを10回投げたとき，表の出る回数を確率変数Xで表す．確率変数Xの平均と分散を求めよ.

(3) あるコインを投げたときの表の出る確率は不明である．このコインを3回投げたところ3回とも表であった.「このコインの表の出る確率は$\frac{1}{2}$である」という帰無仮説を立て，有意水準（危険率）10%で仮説検定を行い，その結果を述べよ.

［2］　確率変数X,Yの同時確率密度関数が$f_{X,Y}(x,y)=\begin{cases} ax & (0<y<x,\ 0<x<1) \\ 0 & (otherwise) \end{cases}$である（$a$は定数).

その際に以下の問いに答えよ.

(1) 周辺確率密度関数$f_X(x)$を求めよ.

(2) aの値を求めよ.

(3) 確率変数$Z=X-Y$の確率密度関数$g_Z(z)$を求めよ.

［3］　ポアソン分布は$f_X(x)=\frac{\mu^x}{x!}e^{-\mu}$（$\mu$は正の定数，$x=0,1,2,\ldots$）で表される.

その際に以下の問いに答えよ.

(1) ポアソン分布はどのような事象を表す分布かについて，例を一つ示して説明せよ.

(2) 確率変数Xが平均$E[X]=1$のポアソン分布に従うとき，確率変数$X(X-1)$の平均$E[X(X-1)]$を求めよ.

【専門用語の英訳】

確率 probability,　確率変数 random variable,　平均 mean,　分散 variance,　帰無仮説 null hypothesis,
有意水準 level of significance,　検定 test,　同時確率密度関数 joint probability density function,
周辺確率密度関数 marginal probability density function,　確率密度関数 probability density function,
ポアソン分布 Poisson distribution,　事象 event

プログラミング

それぞれの重量と価格が分かっている N 個の商品から，総重量が定められた上限値 limit 以下となるように任意個選択したときの合計価格のうちで，最大値を計算したい．リスト 1 は，この計算をするための C 言語プログラムである．構造体 item は商品を表し，そのメンバ w，p はそれぞれ重量と価格を表す．N 個の商品は配列 items に格納されている．maxtotal(num, start) を呼び出すことにより，総重量に start を加えた重量が limit 以内であるという条件のもとで，インデックスが num 以降の商品から任意個選択したときの合計価格の最大値が得られる．リスト 1 の左側の番号は行番号であり，プログラムの一部ではない．このプログラムについて，以下の問いに答えよ．

(1) リスト 1 の (a) から (e) を埋めてプログラムを完成せよ．

(2) リスト 1 の 33 行目が実行された際に 10 を入力した場合，関数 maxtotal が呼び出される回数をその理由とともに答えよ．

(3) リスト 1 の 33 行目が実行された際に 2 を入力した場合，関数 maxtotal が呼び出される回数を理由とともに答えよ．

次に，リスト 1 のプログラムの高速化を考える．プログラムの高速化を目的とした最適化技法の一つとしてメモ化がある．メモ化は，プログラム中の関数呼び出しの結果を呼び出し時の引数とともに記憶しておき，同じ引数で呼び出された際に再度計算せずに，記憶している値を利用する方法である．リスト 2 は，配列 memo を用いてリスト 1 の関数 maxtotal にメモ化を適用したものである．以下の問いに答えよ．

(4) リスト 2 の (f) から (i) を埋めてプログラムを完成せよ．(a) から (e) は (1) で埋めたものと同一である．

(5) リスト 2 の 48 行目が実行された際に 20 を入力したとき，maxtotal(4, 3) が呼び出される回数が，リスト 2 の 25 行目を行数を変化させずにコメントアウトした場合と比べて何回削減されるかを理由とともに答えよ．

(6) 関数の性質によっては，メモ化は適用できない場合がある．そのような場合を 2 つ挙げて説明せよ．

専門用語

関数（かんすう）	function	配列（はいれつ）	array
引数（ひきすう）	argument	最適化技法（さいてきかぎほう）	optimization technique
構造体（こうぞうたい）	structured type	メモ化（か）	memoization
メンバ	member	コメントアウト	comment out

リスト1

```
1  #include <stdio.h>
2
3  #define N 4
4
5  typedef struct _item {
6      int w;
7      int p;
8  } item;
9
10 int limit;
11 item items[N] = {
12   {1, 100}, {1, 50}, {2, 150}, {2, 100}
13 };
14
15 int maxtotal(int num, int start) {
16   int x, y, rval;
17   if (num == N) {
18     return 0;
19   }
20
21   if ( [(a)] > limit ) {
22     rval = maxtotal( [(b)] , [(c)] );
23   } else {
24     x = maxtotal(num + 1, [(d)] ) + items[num].p;
25     y = maxtotal(num + 1, [(e)] );
26     rval = x > y ? x : y;
27   }
28   return rval;
29 }
30
31 int main(void) {
32   printf("\nWeight limit?:");
33   scanf("%d", &limit);
34   printf("Maximum total price is %d\n", maxtotal(0, 0));
35   return 0;
36 }
```

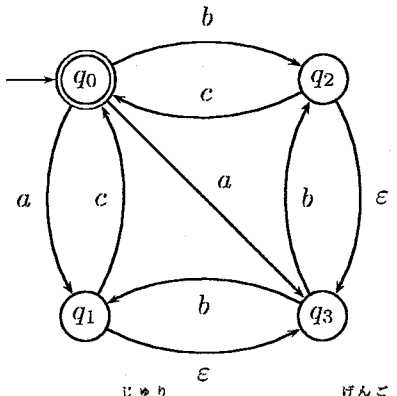
リスト2

```
1  #include <stdio.h>
2
3  #define N 6
4  #define TOTAL_W 15
5
6  typedef struct _item {
7      int w;
8      int p;
9  } item;
10
11 int limit;
12 item items[N] = {
13  {1, 100}, {2, 150}, {1, 50}, {2, 100}, {4, 200}, {5, 250}
14 };
15 int memo[N][TOTAL_W];
16
17 int maxtotal(int num, int start) {
18   int x, y, rval;
19
20   if (num == N) {
21     return 0;
22   }
23
24   if (memo[num][start] > -1) {
25     return   (f)   ;
26   }
27
28   if (  (a)   >  limit  ) {
29     rval = maxtotal(  (b)   ,   (c)   );
30   } else {
31     x = maxtotal(num + 1,   (d)   ) + items[num].p;
32     y = maxtotal(num + 1,   (e)   );
33     rval = x > y ? x : y;
34   }
35   memo[  (g)   ][  (h)   ] =  (i)   ;
36   return rval;
37 }
38
39 int main(void) {
40   int i, j;
```

```
41    for (i = 0; i < N; i++) {
42      for (j = 0; j < TOTAL_W; j++) {
43            memo[i][j] = -1;
44      }
45    }
46
47    printf("\nWeight limit?:");
48    scanf("%d", &limit);
49    printf("Maximum total price is %d\n", maxtotal(0, 0));
50    return 0;
51  }
```

計算機理論

[**1**] M_1 はアルファベット (alphabet) $\{a, b, c\}$ 上の有限オートマトン (finite automaton) である．M_1 の状態遷移図 (transition diagram) を以下のように定める．ここで，q_0 は受理状態 (acceptance state) であると同時に初期状態 (initial state) である．



以下では，$L(M)$ はオートマトン M が受理 (accept) する言語 (language) を表す．

(1) M_1 を等価な状態数最小の決定性有限オートマトン (deterministic finite automaton) に変換し，その状態遷移図を書け．

(2) $L(M_1)$ の補集合 (complement set) を受理する決定性有限オートマトン $\overline{M_1}$ の状態遷移図を示せ．

(3) 正規表現 (regular expression) で，'|' は選択 (alternation)，'•' は連接 (concatenation)，'$*$' は Kleene閉包 (closure) を表すとする．正規表現 $((a|b) \bullet c)^*$ が表す系列の集合を受理する $\{a, b, c\}$ 上で状態数最小の決定性有限オートマトン M_2 の状態遷移図を示せ．

(4) $\overline{M_1}$ の状態集合と M_2 の状態集合の直積 (direct product) を状態集合にもち，$L(\overline{M_1}) \cap L(M_2)$ を受理するオートマトン M_3 を構成し，M_3 が受理状態を持たないことを示せ．

(5) (4) の結果から，$L(M_1)$ と $L(M_2)$ との関係を説明せよ．

[2] 以下の記号 (symbol) を用いた一階述語論理式 (first-order formula) (以下，単に論理式という.) を考える.

- 定数 (constant) null
- 変数 (variable) x, y, z
- 1引数関数記号 (unary function symbol) s
- 3引数述語記号 (3-ary predicate symbol) add
- 論理結合子 (logical connective) ：否定 (negation) $\neg$, 連言 (conjunction) $\wedge$, 選言 (disjunction) $\vee$, 含意 (implication) $\Rightarrow$
- 限量子 (quantifier) ：全称限量子 (universal quantifier) $\forall$, 存在限量子 (existential quantifier) $\exists$
- 括弧 (parenthesis) (,)

ただし，論理結合子や限量子の結合の強さは，強い方から否定，連言，選言，全称限量子，存在限量子，含意の順とする．また，省略しても影響がない括弧は省略してよい.
次に，$\mathbf{N}_0$ をすべての非負整数 (non-negative integer) からなる集合とし，null, s, add に対して，$\mathbf{N}_0$ 上での解釈 (interpretation) I をそれぞれ次のように定める．ここで，$+$ は $\mathbf{N}_0$ 上の通常の加法 (addition) を表す.

- $I(\mathrm{null}) = 0 \in \mathbf{N}_0$.
- $I(\mathrm{s}) : \mathbf{N}_0 \to \mathbf{N}_0$.
  ただし，任意の $n \in \mathbf{N}_0$ に対して，$I(\mathrm{s})(n) = n + 1$.
- $I(\mathrm{add}) : \mathbf{N}_0^3 \to \{\mathbf{true}, \mathbf{false}\}$.
  ただし，任意の $n, m, k \in \mathbf{N}_0$ に対して，$I(\mathrm{add})(n, m, k) = \mathbf{true}$ のとき，かつそのときに限り，$n + m = k$.

このとき，以下の論理式 F_1, F_2 を考える.

$$\begin{aligned} F_1 &: \forall x\, \mathrm{add}(\mathrm{null}, x, x) \\ F_2 &: \forall x \forall y \forall z (\mathrm{add}(x, y, z) \Rightarrow \mathrm{add}(\mathrm{s}(x), y, \mathrm{s}(z))) \end{aligned}$$

また，「$1 + 2 = n$ となる非負整数 n が存在する.」を適切に表す論理式を F_3 とする．以下の問いに答えよ.

(1) I のもとでの解釈がそれぞれ 1, 2 となる項 (term) t_1, t_2 を示せ.
(2) 論理式 F_1 は I のもとで真 (true) であるかどうかを示せ.
(3) 上記の記号を用いて，論理式 F_3 を表せ.
(4) 論理式 $F_1 \wedge F_2 \wedge \neg F_3$ を節集合 (set of clauses) として表せ.
(5) (4) の節集合に対する反駁木 (refutation tree) を求めよ．ただし，そのときに用いた変数への代入 (substitution) を明示せよ.

ハードウェア

[1] 下図のような 1 ビットのセレクタ（selector）回路について以下の問いに答えよ．なお出力 z は，入力 s が 0 のときには入力 x の値を，入力 s が 1 のときには入力 y の値を出力するものとする．また，この回路に対応する論理関数（logical function）を $f(s, x, y)$ とする．

(1) 論理関数 f の真理値表（truth table）を書け．

(2) (1)の結果を用い，論理関数 f の和積標準形（canonical product-of-sums form）を求めよ．

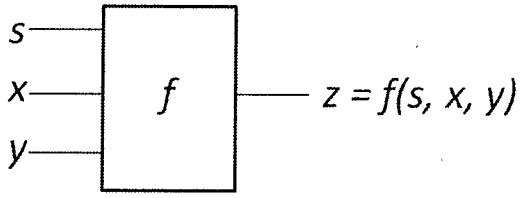
(3) 論理関数 f の否定に対するカルノー図（Karnaugh map）を書け．

(4) (3)の結果を用い，論理関数 f の否定に対する最簡積和形表現（minimum sum-of-product form）を求めよ．

(5) (4)の結果の否定を式変形することにより，論理関数 f の最簡和積形表現（minimum product-of-sum form）を求めよ．

(6) (5)の結果を用い，2 入力 NOR ゲートのみを使った 1 ビットセレクタ回路を設計せよ．

(7) (6)の結果を用い，2 入力 NOR ゲートの出力端子における単一縮退故障（single stuck-at fault）のみを想定するとして，冗長故障（redundant fault），等価故障（equivalent fault）があれば，すべて求めよ．なお，2 入力 NOR ゲートの出力端子には区別できるように名前をつけること．



[2] マイクロプロセッサ（microprocessor）のパイプライン処理（pipelining）について以下の問いに答えよ．

(1) 以下の 3 つのハザード（hazard）の意味をそれぞれ説明せよ．

(a) 構造ハザード（structural hazard）

(b) データハザード（data hazard）

(c) 制御ハザード（control hazard）

(2) 算術論理演算 (arithmetic-logical)，ロード (load)，ストア (store)，分岐 (branch) の 4 つの命令 (instruction) に対して，下表の 5 ステージ (IF, ID, EX, MEM, WB) からなるパイプライン処理に関する以下の問いに答えよ．なお，表中の「N/A」は何も実行されないステージであることを意味する．

	算術論理演算命令	ロード命令	ストア命令	分岐命令
IF	命令読込み			
ID	命令解釈およびレジスタ読込み			
EX	演算	アドレス算出		比較
MEM	N/A	データ読込み	データ書込み	PC 更新
WB	レジスタ書込み		N/A	N/A

(a) 算術論理演算命令が 40%，ロード命令が 25%，ストア命令が 10%，分岐命令が 25%の割合となるプログラムを考える．このプログラムを実行する際，パイプライン処理を行わないマルチサイクル方式と比べて，上記 5 ステージからなるパイプライン処理を行う方式の速度性能比を求めよ．ただし，プログラムの命令数は十分に多いものとし，パイプライン処理においてストール (stall) は一切発生しないとする．

(b) (a) と同一のプログラムにおいて，全分岐命令のうち 75%の分岐実行ではパイプラインの 1 クロックの遅延が起こるとする．このとき，パイプライン処理を行わないマルチサイクル方式と比べて，上記 5 ステージからなるパイプライン処理を行う方式の速度性能比を求めよ．

(3) 遅延スロット (delayed slot) が有効であるとき，以下のコードを遅延スロットにのみ着目して最適化せよ．ただし分岐命令には必ず 1 命令分の遅延スロットがあるとする．

```
    addi $s4, $s4, 1        # $s4 = $s4 + 1
    beq $s2, $s3, L2        # $s2 と$s3 の値が等しければ L2 に分岐
    nop                     # no operation
L1: lw $s1,100($s2)         # $s1 にメモリ番地[$s2+100]の値を読み込む
    addi $s1,$s1,1          # $s1 = $s1 + 1
    sw $s1,104($s2)         # $s1 の値をメモリ番地[$s2+104]に書き込む
    addi $s2,$s2,8          # $s2 = $s2 + 8
    beq $s2,$s3,L1          # $s2 と$s3 の値が等しければ L1 に分岐
    nop                     # no operation
L2: add $s4, $s3, $s2       # $s4 = $s3 + $s2
    sw $s4, 104($s2)        # $s4 の値をメモリ番地[$s2+104]に格納する
```

ソフトウェア

[1] レジスタ割当て（register allocation）は，コンパイラ最適化（compiler optimization）の技法のひとつである．その目標は，プログラムの実行速度を最大化するために，なるべく多くのプログラム変数をレジスタに割り当てることである．古典的なレジスタ割当ては以下の2ステップにより実現される．

1．データフロー解析（data flow analysis）における生存変数解析（live variable analysis）を用い，どの変数が同時に生存する可能性があるか求める．解析結果から，変数を頂点（vertex）とし，同時に生存する可能性を持つ変数どうし間を辺（edge）とする干渉グラフ（interference graph）を生成する．

2．1．で生成された干渉グラフに対し，辺で直接結ばれた頂点が同じ色にならないという条件のもと全ての頂点をk色（kはレジスタの数）で彩色するというグラフ彩色問題（graph coloring problem）を解く．解となる頂点の色の割当てがレジスタ割当てとなる．また，一般にグラフ彩色問題を解くためには発見的手法（heuristic）を用いる．

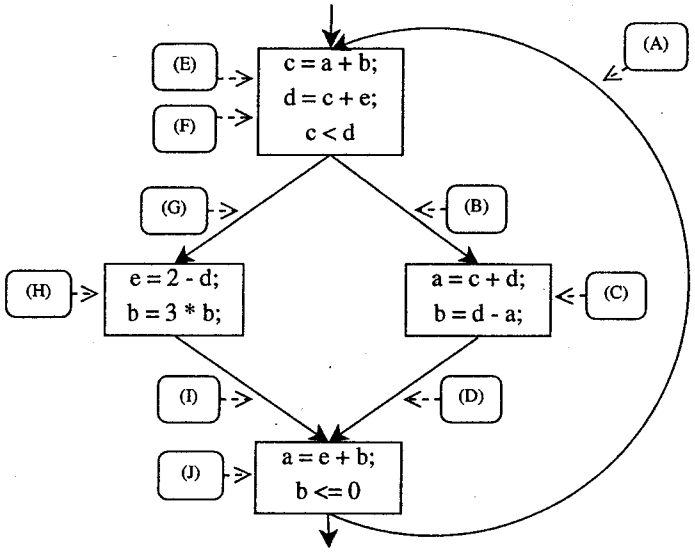


図 1: コントロールフローグラフ（control flow graph）

(1) 図1のコントロールフローグラフに対して生存変数解析を行った時，(A)–(J) の箇所に入る解析の結果を示せ．また，それから生成される干渉グラフを示せ．ただし，このコード実行後に使用される変数はaだけであると仮定せよ．

(2) レジスタ数kを4とし，図1のコントロールフローグラフに対する生存変数解析から得られた干渉グラフをリスト1の発見的手法に与えた時，レジスタ割当てが成功するかどうか述べよ．また，成功する場合は，得られるレジスタ割当てを示せ．ただし，(1) と同様に，このコード実行後に使用される変数はaだけであるとする．

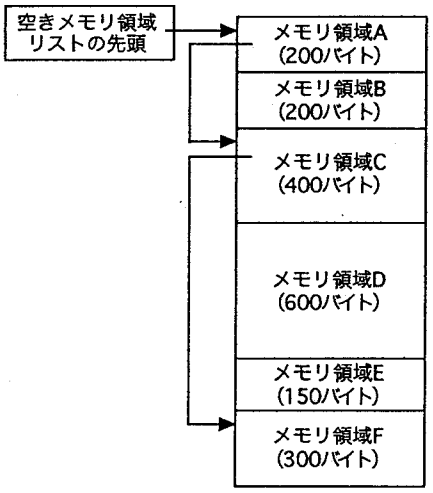
(3) リスト1の手法は，彩色不可能と判定しない限り（つまり，6行目に到達しない限り），17行目に到達する．また，その時，出力されるグラフは入力として与えられたグラフのk彩色になる．その理由を述べよ．

(4) 実際のレジスタ割当てでは，彩色不可能と判定された場合でも，何らかの方法でレジスタ割当てを行う．与えられたレジスタ数で彩色不可能と判定された場合，どのようにすればよいか述べよ．

```
 0 : C ← {1, ..., k} //配色可能な色の集合
 1 : G_init ← 入力として与えられたグラフ
 2 : G ← G_init
 3 : L ← 空リスト
 4 : while G が空でない do
 5 :   if 次数（degree）が k 未満の頂点が G にない then
 6 :     exit(“彩色不可能”)
 7 :   else
 8 :     次数が k 未満の頂点 N を選び G から削除し，N に接する辺も全て削除する
 9 :     N を L の先頭に追加する
10 :   fi
11 : od
12 : while L が空でない do
13 :   L から先頭の頂点 N を取り出し G に加える
14 :   G 内の頂点同士で G_init にある辺を全て G に加える
15 :   G 内で N と直接結ばれたどの頂点とも異なる色の中で最小の色を C から選び，N の色とする
16 : od
17 : G を出力する
```

リスト 1: グラフ彩色の発見的手法

［2］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（allocation）とその解放（release）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（free memory area）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．



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（management area）については無視してよい．

（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，どのメモリ領域から割り当てられるか．割当てアルゴリズムがファーストフィット（first-fit）の場合と，ベストフィット（best-fit）の場合のそれぞれについて答えよ．

（2）この図の状態で，メモリ領域Bを解放した場合に，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せよ．

（3）メモリ領域の割当てアルゴリズムの効率（efficiency）は，2つの観点で比較することができる．2つの観点が何であるか説明し，その2つの観点でファーストフィットとベストフィットを比較せよ．

（4）メモリコンパクション（memory compaction）は，使用中のメモリ領域が連続するように配置し直す技法であり，メモリ領域の動的な割当て・解放を繰り返すことによって生じる問題を解決する．メモリコンパクションを行うことによって得られる利点を2つ挙げて説明せよ．