

令和9年度 豊橋技術科学大学第3年次入学者選抜学力検査問題

専 門 科 目 （ 3：情報・知能工学 ）

注 意 事 項

- 1 試験開始の合図まで、この問題冊子と解答用紙を開いてはいけません。
- 2 問題冊子の枚数は表紙、草稿用紙を含めて9枚です。
- 3 問題冊子とは別に解答用紙が5枚あります。
- 4 問題は3問あります。全問解答してください。
- 5 試験開始の合図の後すぐに、すべての解答用紙の所定の箇所に受験番号を記入してください。
- 6 解答は必ず各問題別の解答用紙の所定の欄に記入してください。裏面には記入しないでください。
- 7 落丁、乱丁、印刷不鮮明の箇所などがあれば、ただちに申し出てください。
- 8 問題冊子の余白は草稿用として使用しても構いません。
- 9 試験終了時刻まで退出してはいけません。
- 10 問題冊子は持ち帰ってください。

(草 稿 用 紙)

[1] $x^2 + xy + y^2 = 1$, $z = 0$ で表される楕円を Q とする。また, 点 $(0,0,1)$ を通り, 方向ベクトルが $\begin{pmatrix} 1 \\ -1 \\ -\sqrt{2} \end{pmatrix}$ である直線を l とし, 楕円 Q 上の点 p から直線 l に下ろした垂線と直線 l との交点を q とする。以下の問いに答えよ。

(1) 楕円 Q を $\mathbf{x}^T \mathbf{A} \mathbf{x} = 1$, $\mathbf{x} = \begin{pmatrix} x \\ y \end{pmatrix}$, $z = 0$ と表すとき, 行列 $\mathbf{A}$ を求めよ。ただし, 行列 $\mathbf{A}$ は対称行列とする。

(2) 行列 $\mathbf{A}$ の固有値 λ_1, λ_2 ($\lambda_1 \leq \lambda_2$) と対応する単位固有ベクトル $\mathbf{u} = \begin{pmatrix} u_x \\ u_y \end{pmatrix}$, $\mathbf{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}$ を求めよ。ただし, $u_x > 0$, $v_x > 0$ とする。

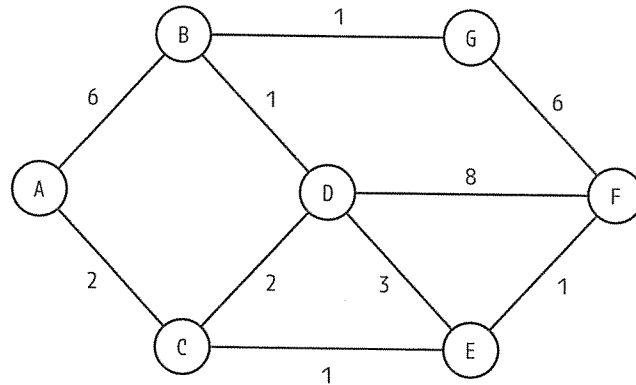
(3) 単位固有ベクトル $\mathbf{u}$, $\mathbf{v}$ を列ベクトルとする行列 $\mathbf{U} = \begin{pmatrix} u_x & v_x \\ u_y & v_y \end{pmatrix}$ を用いた行列 $\mathbf{A}$ の対角化により, 楕円 Q を Z 軸周りに回転した後の楕円 Q' の方程式を求めよ。

(4) 回転行列 $\mathbf{R}$ を $\mathbf{R} = \begin{pmatrix} u_x & u_y & 0 \\ v_x & v_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$ と定義し, 直線 l を回転行列 $\mathbf{R}$ で回転した後の直線を l' とする。点 $(1,3,-2)$ から直線 l' に下ろした垂線と直線 l' との交点の座標を求めよ。

(5) 線分 pq の長さを d とするとき, d^2 の最小値を求めよ。また, そのときの点 p の座標を求めよ。

(6) 点 $(0,0,k)$ を通り, 方向ベクトルが $\begin{pmatrix} 1 \\ 0 \\ -1 \end{pmatrix}$ である直線を m とし, 直線 m と点 $(\sqrt{2}, 0, 0)$ との距離を D とする。0以上1以下の実数を実数として選び, それを k とするとき, D の値が $\frac{1}{2}$ 以上1以下となる確率を求めよ。

- [2] 次の図は、地点A, B, C, D, E, F, G同士の繋がりを表したものである。このように、頂点と辺で関係を表した図を**グラフ**という。各辺に書かれた数は、その地点間の移動にかかる所要時間を表す。例えば地点Aから地点Cまで移動するのにかかる時間は2である。どの辺も双方向に通行できるものとする。このグラフについて、以下の問いに答えよ。



- (1) 地点Aから他の地点を経由せず直接行くことのできる地点を、アルファベット順に全て答えよ。
- (2) 地点Aから地点Fに行く際の経路として、次のa, b, cの3通りの候補を考える。それぞれの所要時間を求めよ。
 - a. $A \rightarrow B \rightarrow D \rightarrow F$
 - b. $A \rightarrow B \rightarrow G \rightarrow F$
 - c. $A \rightarrow B \rightarrow D \rightarrow E \rightarrow F$
- (3) 図のグラフにおいて、地点Aから地点Fへの経路を探索するプログラムを、C99規格に準拠したC言語で作成する。このプログラムでは、リスト1に示すコードによってグラフをGraph構造体で管理し、build_graph関数によって初期化する。コードでは、地点A, B, C, D, E, F, Gを、それぞれ整数0, 1, 2, 3, 4, 5, 6で表すものとする。図で与えられたグラフとなるように、リスト1の空欄a, b, c, d, eを埋めよ。
- (4) リスト2は、**深さ優先探索**によって出発地startから目的地goalまでの経路を探索するdfs関数とその利用例である。ここで、コードはリスト1から続いて書かれているものとする。このコードについて、次の問いに答えよ。
 - ア. このコードでは、探索した地点が順番にorder配列に格納される。dfs関数終了時のorder配列の値を答えよ。
 - イ. このコードでは、探索した経路に関する情報がparent配列に格納される。dfs関数終了時のparent配列の値を答えよ。また、この配列の値によって計算した地点Aから地点Fまでの経路を答えよ。

- (5) リスト3は、**ダイクストラ法**によって出発地startから目的地goalまでの経路を探索するdijkstra関数とその利用例である。ここで、コードはリスト1から続いて書かれているものとする。このコードについて、次の問いに答えよ。

ア. 次の表は、50～76行目のforループの各繰り返し終了時におけるdist配列の値を示したものである。空欄a, b, c, d, eに入る値を答えよ。

処理段階	dist [0]	dist [1]	dist [2]	dist [3]	dist [4]	dist [5]	dist [6]
ループ1回目	0	b	2	a	a	a	a
ループ2回目	0	b	2	4	3	a	a
ループ3回目	0	b	2	4	3	4	a
ループ4回目	0	c	2	4	3	4	a
ループ5回目	0	c	2	4	3	4	d
ループ6回目	0	c	2	4	3	4	e
ループ7回目	0	c	2	4	3	4	e

- イ. このコードでは、探索した経路に関する情報がparent配列に格納される。dijkstra関数終了時のparent配列の値を答えよ。また、この配列の値によって計算した地点Aから地点Fまでの経路を答えよ。

- (6) 深さ優先探索とダイクストラ法について、その違いの説明として最も適切なものを次のa, b, c, dの中から1つ選び、記号で答えよ。

- 深さ優先探索は通る辺の本数が最も少ない経路を求め、ダイクストラ法は調べる順序によって経路を決める。
- 深さ優先探索は最短所要時間の経路を見つけられない場合があり、ダイクストラ法は所要時間が最も短い経路を求める。
- どちらの方法も、求まる経路は同じである。
- 深さ優先探索だけが、必ず所要時間が最も小さい経路を求める。

```

1  #include <stdio.h>
2  #include <stdbool.h>
3
4  #define N 7 /* 頂点数 */
5  #define INF 9999 /* 無限大の重み (= 辺が存在しない) を表す */
6
7  typedef struct {
8      int weight[N][N];
9  } Graph;
10
11 void graph_init(Graph *g) {
12     for (int i = 0; i < N; i++)
13         for (int j = 0; j < N; j++)
14             g->weight[i][j] = a;
15 }
16
17 void graph_add_edge(Graph *g, int u, int v, int weight) {
18     g->weight[u][v] = b;
19     g->weight[v][u] = b;
20 }
21
22 void build_graph(Graph *g) {
23     graph_init(g);
24
25     graph_add_edge(g, 0, 1, c); /* A-B */
26     graph_add_edge(g, 0, 2, 2); /* A-C */
27     graph_add_edge(g, 1, 3, 1); /* B-D */
28     graph_add_edge(g, 1, d, 1); /* B-G */
29     graph_add_edge(g, 2, 3, 2); /* C-D */
30     graph_add_edge(g, e, 4, 1); /* C-E */
31     graph_add_edge(g, 3, 4, 3); /* D-E */
32     graph_add_edge(g, 3, 5, 8); /* D-F */
33     graph_add_edge(g, 4, 5, 1); /* E-F */
34     graph_add_edge(g, 5, 6, 6); /* F-G */
35 }
36

```

リスト1: グラフの管理方式

```

37 bool dfs_visit(const Graph *g, int u, int goal, bool visited[],
38               int parent[], int order[], int *count) {
39     visited[u] = true;
40     order[(*count)++] = u;
41
42     if (u == goal)
43         return true;
44
45     for (int v = 0; v < N; v++) {
46         if (g->weight[u][v] < INF && !visited[v]) {
47             parent[v] = u;
48
49             if (dfs_visit(g, v, goal, visited, parent,
50                         order, count))
51                 return true;
52         }
53     }
54
55     return false;
56 }
57
58 void dfs(const Graph *g, int start, int goal) {
59     bool visited[N];
60     int parent[N];
61     int order[N];
62     int count = 0;
63
64     for (int i = 0; i < N; i++) {
65         visited[i] = false;
66         parent[i] = -1;
67         order[i] = -1;
68     }
69
70     dfs_visit(g, start, goal, visited, parent, order, &count);
71 }
72
73 int main(void) {
74     Graph g;
75     build_graph(&g);
76     dfs(&g, 0, 5);    /* A -> F */
77
78     return 0;
79 }

```

リスト2: 深さ優先探索による経路探索

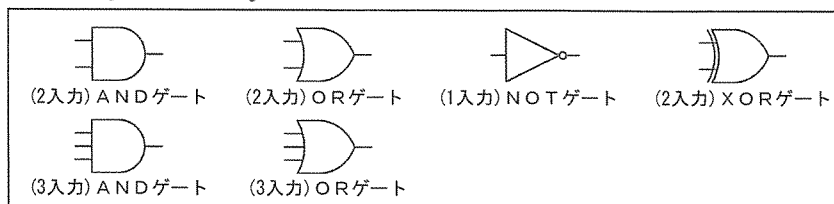
```

37 void dijkstra(const Graph *g, int start, int goal) {
38     bool used[N];
39     int dist[N];
40     int parent[N];
41
42     for (int i = 0; i < N; i++) {
43         used[i] = false;
44         dist[i] = INF;
45         parent[i] = -1;
46     }
47
48     dist[start] = 0;
49
50     for (int step = 0; step < N; step++) {
51         int u = -1;
52         int best = INF;
53
54         for (int v = 0; v < N; v++) {
55             if (!used[v] && dist[v] < best) {
56                 best = dist[v];
57                 u = v;
58             }
59         }
60
61         if (u == -1)
62             break;
63
64         used[u] = true;
65
66         for (int v = 0; v < N; v++) {
67             if (g->weight[u][v] < INF) {
68                 int w = g->weight[u][v];
69
70                 if (!used[v] && dist[u] + w < dist[v]) {
71                     dist[v] = dist[u] + w;
72                     parent[v] = u;
73                 }
74             }
75         }
76     }
77 }
78
79 int main(void) {
80     Graph g;
81     build_graph(&g);
82     dijkstra(&g, 0, 5); /* A -> F */
83
84     return 0;
85 }

```

リスト3: ダイクストラ法による経路探索

- [3] 2つの2ビット符号なし整数 $A = (A_1, A_0)$ および $B = (B_1, B_0)$ を考える。 A と B の値は、それぞれ10進数で0から3の整数値をとる。ここで A_1, B_1 はそれぞれ最上位ビット、 A_0, B_0 は最下位ビットである。 A と B を比較し、大きい方の値を出力する最大値選択回路 $\max(A, B)$ を設計する。なお、「 $\cdot$ 」は論理積、「 $+$ 」は論理和、「 $\bar{A}$ 」は変数 A の否定、「 $\oplus$ 」は排他的論理和を表すものとし、真理値表およびカルノー図においては真の値を1、偽の値を0で表すものとする。ANDゲート、ORゲート、NOTゲート、XORゲートはそれぞれ下図のように表現される。



これらを前提として、以下の問いに答えよ。

- (1) 4ビット入力 (A_1, A_0, B_1, B_0) に対して、比較結果を G とし、 $A > B$ のとき $G = 1$ 、それ以外するとき $G = 0$ として真理値表を作成する。解答用紙の真理値表の空欄に、10進数にて A と B の値を、また真偽値として1と0を用いて G の値をそれぞれ記入せよ。
- (2) G の標準積和形を求めよ。
- (3) G について、 A_1, A_0, B_1, B_0 を変数とする4変数カルノー図を作成せよ。また、 G の最小積和形を求めよ。ただし、最小積和形とは、複数個の論理積項が論理和で結ばれている論理式において、論理積と論理和の総数が最小になる形式である。
- (4) G を用いて、 A と B の最大値 $\max(A, B)$ を2ビット出力 (M_1, M_0) として出力する回路を設計する。ここで、 $G = 1$ ($A > B$) のとき A を選択し、 $G = 0$ ($A \leq B$) のとき B を選択する。
 - ア. M_1 および M_0 を、 G, A_1, A_0, B_1, B_0 を用いた論理式で表せ。また、入力には必ず G と $\bar{G}$ を用いること。
 - イ. 問(3)で求めた G の最小積和形の回路と問(4)アの最大値選択回路を実現することを考える。解答欄の回路図にANDゲートおよびORゲートを用いた回路を完成させよ。最大値選択回路は M_0 出力部のみ作図せよ。なお、3入力ゲートを用いてもよい。また、入力には必ず G と $\bar{G}$ を用いること。
- (5) $A = B$ のとき $E = 1$ 、それ以外するとき $E = 0$ とする。 E の最小積和形を求めよ。また、解答欄の回路図にXORゲート2つ、NOTゲート1つ、ORゲート1つの計4つのゲートを用いた E の回路を完成させよ。