筑波大学　情報学群　情報科学類・情報メディア創成学類

令和8年度　学群編入学試験

学力試験問題（専門科目）

[注意事項]

1. 試験開始の合図があるまで、問題の中を見てはいけません。
2. 解答用紙の定められた欄に、氏名、受験番号を記入しなさい。
3. 問題 1 から問題 4（数学、情報基礎）の計 4 問をすべて答えなさい。
4. 解答用紙は、4 問に対して、各問 1 枚の合計 4 枚を用いなさい。
5. 解答用紙上部の [　　] 欄に解答する問題番号を記入しなさい。
6. 解答用紙の裏面を使用する場合には、その旨を解答用紙の表面に示しなさい。
7. この問題冊子は全部で 11 ページ（表紙、白紙を除く）です。

問題１　数学（1）

関数

$$f(x,y) = \log(1 + x^2 + y^2)$$

について，以下の問いに答えなさい．

(1) 関数 $f(x,y)$ の１次偏導関数および２次偏導関数を求めなさい．

(2) 関数 $f(x,y)$ の極値の有無を調べ，極値がある場合はそれを与える x, y と $f(x,y)$ の値をそれぞれ求めなさい．

(3) 領域

$$D = \left\{(x,y) \in \mathbb{R}^2 \mid x^2 + y^2 \leq 1\right\}$$

における関数 $f(x,y)$ の重積分

$$I = \iint_D f(x,y)\, dxdy$$

を極座標変換を用いて求めなさい．

問題2　数学 (2)

3つの漸化式

$$\begin{cases} x_{n+1} = x_n + 2y_n + 2z_n \\ y_{n+1} = x_n - z_n \\ z_{n+1} = x_n + y_n + 2z_n \end{cases} \qquad (n = 0,\ 1,\ 2,\ \cdots)$$

で定まる数列 $\{x_n\}$, $\{y_n\}$, $\{z_n\}$ について以下の問いに答えなさい.

(1) $\begin{pmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{pmatrix} = A \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix}$ と表すとき, 行列 A を求めなさい.

(2) 行列 A の全ての固有値を求め, 各固有値に対応する固有ベクトルを一つずつ求めなさい.

(3) $\begin{pmatrix} x_0 \\ y_0 \\ z_0 \end{pmatrix} = \begin{pmatrix} 2 \\ -2 \\ 2 \end{pmatrix}$ とするとき, これを設問(2)で求めた固有ベクトルの一次結合で表しなさい.

(4) 設問(3)の結果を用いることにより, 数列 $\{x_n\}$, $\{y_n\}$, $\{z_n\}$ の一般項を求めなさい.

問題3　情報基礎 (1)

頂点数がそれぞれ M, N の 2 つの頂点列

$$\mathcal{A} = (a_1, a_2, \cdots, a_i, \cdots, a_M)$$

$$\mathcal{B} = (b_1, b_2, \cdots, b_j, \cdots, b_N)$$

について、各頂点 $a_i = (x_i, y_i)$ と $b_j = (x_j', y_j')$ (ただし $x_i < x_{i+1}$, $x_j' < x_{j+1}'$) のマンハッタン距離の総和

$$\mathcal{E}(\mathcal{A}, \mathcal{B}) = \sum_{(i,j) \in \mathcal{P}} \left(|x_i - x_j'| + |y_i - y_j'| \right)$$

が最小となるような、頂点間の対応 (i, j) の列 $\mathcal{P}$ を求めたい (図 1 参照)。ただし $\sum_{(i,j) \in \mathcal{P}}$ は $\mathcal{P}$ の各要素 (i, j) に関する総和を表す。$\mathcal{P}$ の条件として、$\mathcal{P}$ の最初の要素は $(1, 1)$ とし、以降の要素ではインデックス i, j の片方または両方が 1 ずつ増えていき、最後の要素は (M, N) となるものとする。また、$\mathcal{E}(\mathcal{A}, \mathcal{B})$ が最小となる $\mathcal{P}$ はただ 1 つ存在するものとする。

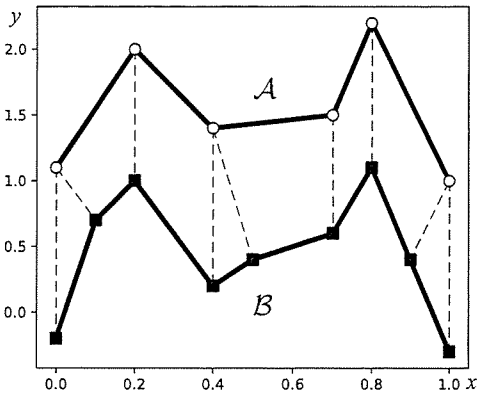

図 1: 適当な頂点列 $\mathcal{A}$ (白丸) と $\mathcal{B}$ (黒四角) に対する、頂点間の対応 (破線)

最小となる $\mathcal{E}(\mathcal{A}, \mathcal{B})$ およびそのときの $\mathcal{P}$ を、以下で説明するアルゴリズムによって計算する。プログラム 1 はこのアルゴリズムを実現する C 言語プログラムである。まず、2 つの 2 次元配列 D, T を用意する。D の i 行 j 列には対応 $(1, 1)$ から対応 (i, j) までのマンハッタン距離の総和の最小値を記録し、T の i 行 j 列には 1 つ前の対応からの遷移 (インデックス i だけ 1 増やす、j だけ 1 増やす、両方とも 1 増やす、の 3 通りのいずれか) を記録する。2 次元配列の i 行 j 列の要素を更新する際は、3 通りの遷移のうちマンハッタン距離の増分が最も小さいものを選び、そのときのマンハッタン距離の総和と遷移とを、それぞれ D, T に記録する。最終的に D の M 行 N 列に最小のマンハッタン距離の総和 $\mathcal{E}(\mathcal{A}, \mathcal{B})$ が記録され、T を M 行 N 列から逆順に辿ることで、対応の列 $\mathcal{P}$ が得られる。

次ページに続く

プログラム 1 について、以下の問いに答えなさい。なお、配列 Ax, Ay は頂点列 A の x, y 座標列を、 配列 Bx, By は頂点列 B の x, y 座標列を、また ¥n は改行を表すものとする。

(1) プログラム 1 の空欄（ア)〜(ケ）を適切に埋めて、プログラムを完成させなさい。ただし、空欄（ア)〜(ウ）については各行のコメントに従うこと。

(2) 上記のアルゴリズムの漸近的計算量 (オーダー) を、M および N を用いて表しなさい。

(3) プログラム 1 の (ア)〜(カ) を含む二重ループの計算が終わった直後の、2 次元配列 D の各要素の値を表にまとめなさい。ただし 0 行 0 列が左下となるようにすること。

(4) プログラム 1 の出力を書きなさい。

次ページに続く

プログラム 1

```
#include <stdio.h>
#include <math.h>
#define M 4
#define N 3

int main(void) {
    double Ax[M] = { 0.0, 0.3, 0.6, 1.0 };
    double Ay[M] = { 1.0, 1.5, 0.8, 1.8 };
    double Bx[N] = {  0.0, 0.4, 0.8 };
    double By[N] = { -0.3, 0.9, 0.1 };

    double D[M + 1][N + 1];
    unsigned char T[M + 1][N + 1];
    int Pi[M + N + 1];   /* 列 P の各要素 (i, j) の i を格納 */
    int Pj[M + N + 1];   /* 列 P の各要素 (i, j) の j を格納 */

    for (int i = 0; i <= M; i++)
        for (int j = 0; j <= N; j++)
            D[i][j] = 100.0;
    D[0][0] = 0.0;

    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            double d = fabs(Ax[i] - Bx[j]) + fabs(Ay[i] - By[j]);
            double a = [   (ア)   ]   /* i を 1 増やす場合 */
            double b = [   (イ)   ]   /* j を 1 増やす場合 */
            double c = [   (ウ)   ]   /* i と j を 1 増やす場合 */
            if (a <= b && a <= c) {
                D[i+1][j+1] = [   (エ)   ]
                T[i+1][j+1] = 1;
            } else if (b <= a && b <= c) {
                D[i+1][j+1] = [   (オ)   ]
                T[i+1][j+1] = 2;
            } else {
                D[i+1][j+1] = [   (カ)   ]
                T[i+1][j+1] = 0;
```

次ページに続く

```
            }
        }
    }

    int i = M, j = N, k = 0;
    while (i > 0 || j > 0) {
        Pi[k] = i - 1;
        Pj[k] = j - 1;
        k++;
        unsigned char t = T[i][j];
        if (t == 0) {
            (キ)
        } else if (t == 1) {
            (ク)
        } else {
            (ケ)
        }
    }

    printf("E = %.1f¥n", D[M][N]);
    for(int t = k - 1; t >= 0; t--)
        printf("A[%d] <-> B[%d]¥n", Pi[t], Pj[t]);
    return 0;
}
```

問題４　情報基礎（２）

GeoHash は，地図を格子状に分割し，与えられた緯度と経度が指し示す地点を含む 1 区画を短い文字列（以下ハッシュ値と呼ぶ）で表す手法である．例えば，筑波大学（緯度+36.111389，経度+140.103889）を含む区画を表す 5 桁のハッシュ値は「xn7tk」である．ハッシュ値の桁数が多いほど，地図上の小さな区画を表すことができる．図 1 に示すように，ハッシュ値「xn7tk」より 1 桁多い「xn7tkj」は，筑波大学を含むより狭い区画を表す．

（この部分は、著作権の都合により公開できません）

図 1 ハッシュ値「xn7tk」と「xn7tkj」が表す地図上の区画

（出典：国土地理院ウェブサイト（https://maps.gsi.go.jp/），地理院地図を加工して作成）

ハッシュ値は以下の手順で求める．

手順1.　経度の範囲の初期値を[−180,+180]，緯度の範囲の初期値を[−90, +90]とする．ビット列の生成は，経度→緯度の順に繰り返し行う．ビット列が目的の長さに達した場合，中止し，手順 5 へ移る．

手順2.　経度の範囲の中点を求め，与えられた経度が中点以上の場合は 1，中点より小さい場合は 0 とする．得られた値が 0 の場合は次に探索する経度の範囲の上限を中点とし，1 の場合は下限を中点とする．

手順3.　緯度の範囲の中点を求め，与えられた緯度が中点以上の場合は 1，中点より小さい場合は 0 とする．得られた値が 0 の場合は次に探索する緯度の範囲の上限を中点とし，1 の場合は下限を中点とする．

手順4.　求めるハッシュ値の桁数に必要な長さのビット列が得られるまで，手順 2 と手順 3 を繰り返し，求めたビットを左から「経度→緯度→経度→緯度…」の順番に並べる．図 2 は，手順 2 と手順 3 をそれぞれ 2 回繰り返した例

次ページに続く

である.

手順5.　手順4で得たビット列を5ビットごとに分割し，それぞれBase32方式で変換した文字を求め（文字とビット列の対応は表1に示す），連結した文字列をハッシュ値とする．例えば1110110100というビット列の場合，先頭5ビット11101からx，次の5ビット10100からnが得られ，「xn」がハッシュ値となる．

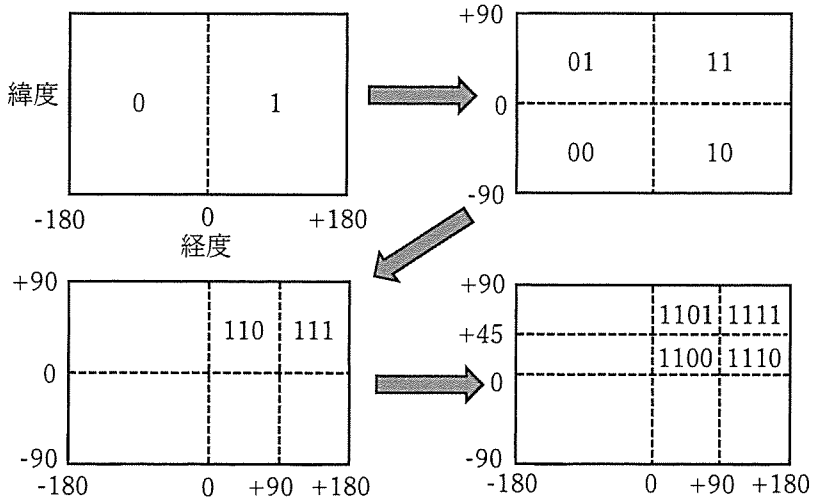

図2 手順2と手順3を2回繰り返したビット列の例

表1　Base32方式の文字とビット列の対応

文字	ビット列	文字	ビット列	文字	ビット列	文字	ビット列
0	00000	8	01000	h	10000	s	11000
1	00001	9	01001	j	10001	t	11001
2	00010	b	01010	k	10010	u	11010
3	00011	c	01011	m	10011	v	11011
4	00100	d	01100	n	10100	w	11100
5	00101	e	01101	p	10101	x	11101
6	00110	f	01110	q	10110	y	11110
7	00111	g	01111	r	10111	z	11111

（0〜9の数字とa,i,l,oを除く小文字アルファベット22文字を用いる）

次ページに続く

GeoHash のハッシュ値を求める C 言語プログラム（プログラム 1）について以下の問いに答えなさい.

(1) 手順 2 と手順 3 に従い, 緯度-33.9 と経度 18.4 を含む区画のビット列での表現を 3 ビットまで答えなさい.

(2) 桁数 1 のハッシュ値「x」が表す区画の範囲は南北 4,989,600m, 東西 4,050,000m である. ハッシュ値「xn7tk」の桁数を 1 つ増やし,「xn7tkj」とした場合, 表す区画の面積はおおよそ何分の 1 になるか, その理由とともに答えなさい.

(3) ハッシュ値がわかれば, 隣接する区画の同じ桁数のハッシュ値を演算によって求められる. 例えば, ハッシュ値「x」が表す区画の北に隣接する区画のハッシュ値は「z」, 東に隣接する区画のハッシュ値は「8」である. ハッシュ値「x」が表す区画の西に隣接する区画と南東に接する区画のハッシュ値をそれぞれ答えなさい.

(4) 関数 encode_geohash は, 与えられた緯度 (lat) と経度 (lon) を含む区画のハッシュ値（桁数は length）を求め, 変数 geohash に代入する. 空欄（ア）～（オ）を埋めて関数を完成させなさい.

(5) プログラム 1 を実行したときの出力を答えなさい.

次ページに続く

（プログラム１）

```
#include <stdio.h>
#define GEOHASH_MAX_LEN 10
const char BASE32[] = "0123456789bcdefghjkmnpqrstuvwxyz";
void encode_geohash(double lat,double lon,int length,char *geohash) {
    double lat_range[2] = {-90.0, 90.0};
    double lon_range[2] = {-180.0, 180.0};
    int bit = 0, even = 1, i;
    int geohash_char = 0, geohash_index = 0;
    for (i = 0; i < length * [ (ア) ]; i++) {
        if (even) {
            double mid = (lon_range[0] + lon_range[1]) / 2;
            if (lon >= mid) {
                geohash_char = [ (イ) ] ;
                lon_range[0] = mid;
            } else {
                geohash_char = [ (ウ) ] ;
                lon_range[1] = mid;
            }
        } else {
            double mid = (lat_range[0] + lat_range[1]) / 2;
            if (lat >= mid) {
                geohash_char = [ (イ) ] ;
                lat_range[0] = mid;
            } else {
                geohash_char = [ (ウ) ] ;
                lat_range[1] = mid;
            }
        }
        even = [ (エ) ] ;
        bit = bit + 1;
        if (bit == 5) {
            geohash[geohash_index] = BASE32[ [ (オ) ] ];
```

次ページに続く

```
            geohash_index = geohash_index + 1;
            bit = 0;
            geohash_char = 0;
        }
    }
    geohash[geohash_index] = '¥0';
}
int main() {
    double lat = -33.9;  // 緯度
    double lon = 18.4;   // 経度
    int    length = 1;   // 求めるハッシュ値の桁数
    char geohash[GEOHASH_MAX_LEN + 1]; //求めたハッシュ値の格納
    encode_geohash(lat, lon, length, geohash);
    printf("Geohash: %s¥n", geohash);
    return 0;
}
```