令和８年度

情報学部第３年次編入学試験問題

コンピュータ科学基礎

令和７年８月２１日（木）９：３０～１０：３０

注　意　事　項

１．試験開始の指示があるまで，この冊子を開いてはいけない。

２．定規（分度器付き定規は除く）・コンパスは，あらかじめ机上に置いておき，試験開始後に使用してよい。

３．試験開始の指示があったら，冊子の内容を確かめること。冊子は表紙１枚，問題紙６枚，解答用紙２枚（問題ごとに１枚ずつ），草稿用紙１枚である。不足，重複，印刷不鮮明の箇所があった場合には，直ちに申し出ること。

４．解答を書き込む前に，解答用紙と草稿用紙の所定の箇所に受験番号を記入すること。解答用紙と草稿用紙に氏名を記入してはいけない。

５．問題は，２題すべてに解答すること。

６．解答用紙は裏面も使用してよい。なお，裏面を使用する場合は，解答用紙の右下のチェックボックスにレ点を記入すること。

７．試験終了時刻まで退室してはいけない。

８．解答用紙，草稿用紙は持ち帰ってはいけない。その他は持ち帰ってよい。

[1]

以下の (1)–(2) を解け．答えには，対数関数を使ってよい．分数は既約にせよ．解の導出過程も書くこと．エントロピーの底は 2 とする．

(1) 確率変数 X は $P(X=0)=\dfrac{2}{5}$，$P(X=1)=\dfrac{2}{5}$，$P(X=2)=\dfrac{1}{5}$ を満たすとする．まず X を一度決めてから，その値に対応する確率で長さ 5 のビット列を生成する．各ビットは独立であり，

$$P(1 \mid X=0)=\frac{1}{4}, \qquad P(1 \mid X=1)=\frac{1}{2}, \qquad P(1 \mid X=2)=\frac{3}{4}$$

である．ビット列に含まれる 1 の個数を N とおく．

(a) $N=3$ となる確率を求めよ．
(b) $N \leq 4$ となる確率を求めよ．

(2) 番号 1 から 11 が記載されたコインが 1 枚ずつあり，そのうち 1 枚だけが軽く，残りの 10 枚の重さは同じである．最初に 4 枚・4 枚・3 枚に無作為に分けて 4 枚どうしを天秤に載せ，その結果を Y_1 とし，天秤の左皿が軽ければ $Y_1=-1$，つりあえば $Y_1=0$，右皿が軽ければ $Y_1=1$ と定義する．次に，$Y_1=\pm 1$ のときは軽い方の 4 枚から 2 枚を無作為に取り，右皿と左皿に 1 枚ずつ天秤に載せる．$Y_1=0$ のときは残り 3 枚から 2 枚を無作為に取り，同様に右皿と左皿に 1 枚ずつ載せる．その 2 回目の結果を Y_2 とする．

(a) 軽いコインの番号を一様分布に従う確率変数 X とする．天秤に載せる前の X のエントロピー $H(X)$ を求めよ．
(b) エントロピー $H(Y_1)$ と $H(Y_2)$ を求めよ．
(c) 結合エントロピー $H(Y_1, Y_2)$ と条件付きエントロピー $H(Y_1 \mid Y_2)$ を求めよ．
(d) 相互情報量 $I(Y_1; Y_2)$ を求めよ．

[2]

プログラム 1 は与えられた文字列と値の対を保持，参照する C 言語プログラムである．プログラム 1 に関する以下の問いに答えよ．

(1) プログラム 1 の次の行を実行したときに標準出力に表示される値をそれぞれ答えよ．
　① 127 行目，② 128 行目

(2) プログラム 1 の次の行を実行したときに関数 search が実行される回数をそれぞれ答えよ．
　① 127 行目，② 128 行目

(3) プログラム 1 の実行が終わるまでに次の関数が実行される回数を答えよ．
　① 関数 left_rotate，② 関数 right_rotate

なお，プログラム 1 で呼び出しているライブラリ関数の仕様は以下とする．

char* strdup(char *src)
引数 src が指す文字列を新しい領域にコピーし，コピーした文字列を指すポインタを返す．

int strcmp(char *str1, char *str2)
引数 str1 と引数 str2 が指す文字列が同一のものであれば，0 を返す．引数 str1 が指す文字列が引数 str2 が指す文字列よりも辞書順で前にあるときは，-1 を返す．引数 str1 が指す文字列が引数 str2 が指す文字列よりも辞書順で後にあるときは，1 を返す．

プログラム 1

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct Node {
    char *key;
    int value;
    struct Node* left;
    struct Node* right;
    int height;
} Node;

int height(Node* n) {
    if (n == NULL)
        return 0;
    return n->height;
}

int max(int a, int b) {
    if (a > b) return a;
    return b;
}

Node* new_node(char* key, int value) {
    Node *node;

    node = (Node*)malloc(sizeof(Node));
    node->key = strdup(key);
    node->value = value;
    node->left = NULL;
    node->right = NULL;
    node->height = 1;
    return node;
}

Node* right_rotate(Node* y) {
    Node *t, *x;
```

```
38          x = y->left;
39          t = x->right;
40
41          x->right = y;
42          y->left = t;
43
44          y->height = max(height(y->left), height(y->right)) + 1;
45          x->height = max(height(x->left), height(x->right)) + 1;
46
47          return x;
48      }
49
50      Node* left_rotate(Node* x) {
51          Node *t, *y;
52
53          y = x->right;
54          t = y->left;
55
56          y->left = x;
57          x->right = t;
58
59          x->height = max(height(x->left), height(x->right)) + 1;
60          y->height = max(height(y->left), height(y->right)) + 1;
61
62          return y;
63      }
64
65      Node* insert(Node* node, char* key, int value) {
66          int cmp, balance;
67          if (node == NULL) return (new_node(key, value));
68
69          cmp = strcmp(key, node->key);
70
71          if (cmp < 0) node->left = insert(node->left, key, value);
72          else if (cmp > 0) node->right = insert(node->right, key, value);
73          else {
74              node->value = value;
75              return node;
```

```
76        }
77
78        node->height = 1 + max(height(node->left), height(node->right));
79
80        if (node == NULL) balance = 0;
81        else balance = height(node->left) - height(node->right);
82
83        if ((balance > 1) && (strcmp(key, node->left->key) < 0)) return right_rotate(node);
84        if ((balance < -1) && (strcmp(key, node->right->key) > 0)) return left_rotate(node);
85
86        if ((balance > 1) && (strcmp(key, node->left->key) > 0)) {
87            node->left = left_rotate(node->left);
88            return right_rotate(node);
89        }
90
91        if ((balance < -1) && (strcmp(key, node->right->key) < 0)) {
92            node->right = right_rotate(node->right);
93            return left_rotate(node);
94        }
95
96        return node;
97    }
98
99    Node* search(Node* root, char* key) {
100       int cmp;
101
102       if (root == NULL) return NULL;
103
104       cmp = strcmp(key, root->key);
105
106       if (cmp == 0) return root;
107       else if (cmp < 0) return search(root->left, key);
108       else return search(root->right, key);
109   }
110
111   int get_value(Node* root, char *key){
112       Node *node;
113       node = search(root, key);
```

```
114         if (node == NULL) return -1;
115         return node->value;
116     }
117
118     void main() {
119         Node *root;
120
121         root = NULL;
122         root = insert(root, "orange", 200);
123         root = insert(root, "apple", 150);
124         root = insert(root, "banana", 100);
125         root = insert(root, "melon", 500);
126
127         printf("%d¥n", get_value(root, "orange"));
128         printf("%d¥n", get_value(root, "grapes"));
129     }
```