平成26年度 茨城大学大学院 理工学研究科
博士前期課程 入学試験問題
(第2次募集)

情報工学専攻

アルゴリズムとプログラミング

注意事項

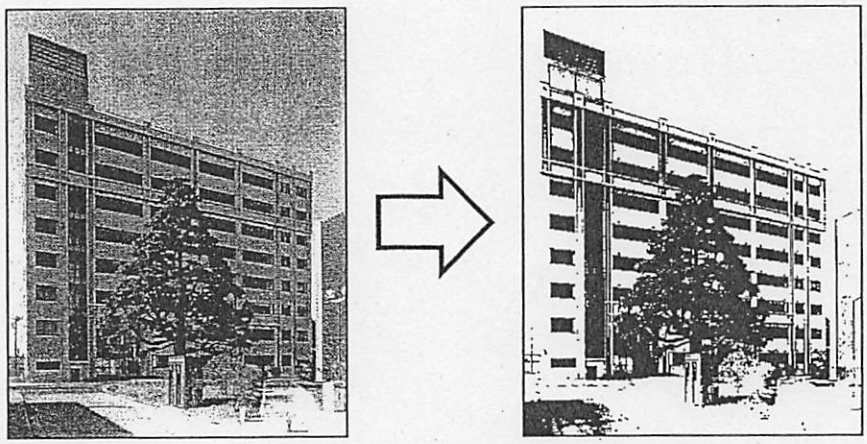
1. 試験開始の合図があるまで，この問題冊子の中を見てはいけません.
2. この試験に使用する解答用紙は 6枚です．枚数を確認して，解答用紙の表題の右側にある（ ）内に,（その1）から（その6）までを記入してください.
   問題1の問1に対し（その1）を,
   問題1の問2に対し（その2）を,
   問題1の問3，問4，問5に対し（その3）を,
   問題2の問1，問2に対し（その4）を,
   問題2の問3に対し（その5）を,
   問題2の問4に対し（その6）を使用してください.
3. 解答用紙の全てのページに，解答科目名と受験番号を記入してください.
   受験番号に記入漏れや記入間違いがあると，試験は無効になりますので，手元の受験票で確認のうえ注意深く記入してください.
   【上記の2と3に記載の事項は，試験開始前に必ず記入してください.】
4. 試験中に問題冊子の印刷不鮮明，ページの落丁・乱丁等に気が付いた場合は，手を上げて試験監督に知らせてください.
5. 携帯電話の電源は必ず切ってカバン等に入れ，身に着けないでください．また，携帯電話を時計として使用することはできません.
6. 試験中に辞書類・電卓を使用することはできません.

英語対応表

画像処理	image processing
グレースケール画像	grayscale image
バイナリ画像	binary image
変換する	convert
二値化	binarization
しきい値	threshold value
画素	pixel
濃淡値	gray value
大津の手法	Otsu's method
範囲	range
整数値	integer value
平均値	average value
生起確率	occurrence probability
クラス間分散	inter-class variance
一意	unique
逆ポーランド記法	reverse Polish notation
後置記法	postfix notation
四則演算	four arithmetic operations
スタック	stack
非負整数	non-negative integer
データ構造	data structure
関数	function
構造体	structure
演算数	operand
演算子	operator
中置記法	infix notation
文字列	character string
加算	addition
減算	subtraction
乗算	multiplication
除算	division
返却値，戻り値	return value
入力ファイル	input file
標準出力	standard output
ファイル名	file name

アルゴリズムとプログラミング

問題 1　画像処理において，下図のように，グレースケール画像を白黒のバイナリ画像に変換する操作が必要とされることがある．この操作は，二値化と呼ばれている．



二値化には様々な方法が存在するが，何らかの手法でしきい値を適切に定め，濃淡値がしきい値以下の画素は黒（濃淡値が最小）に変換し，濃淡値がしきい値より大きい画素は白（濃淡値が最大）に変換するという方法が用いられることが多い．

しきい値の定め方の中で，よく利用されている「大津の手法」を以下に示す．

大津の手法

画素の集合を $P = \{1, 2, \ldots, n\}$ とし，各画素 i は 0 から m の範囲の整数値である濃淡値 $g(i)$ を持つものとする．
しきい値を t とすると，P は次のような 2 つのクラス C_1 と C_2 に分割される．

$$C_1 = \{i \in P \mid g(i) \leq t\},\quad C_2 = \{i \in P \mid g(i) > t\}$$

C_1 と C_2 に含まれる画素の濃淡値の平均値 μ_1, μ_2，及び，C_1 と C_2 の生起確率 ω_1, ω_2 は次のように求まる．なお，$|C_1|$，$|C_2|$，$|P|$ は，それぞれ C_1，C_2，P の要素数である．

$$\mu_1 = \frac{\sum_{i \in C_1} g(i)}{|C_1|},\quad \mu_2 = \frac{\sum_{i \in C_2} g(i)}{|C_2|},\quad \omega_1 = \frac{|C_1|}{|P|},\quad \omega_2 = \frac{|C_2|}{|P|}$$

このとき，C_1 と C_2 のクラス間分散 σ^2 を次のように定義する．

$$\sigma^2 = \omega_1 \cdot \omega_2 \cdot (\mu_1 - \mu_2)^2$$

t の値を 0 から m まで変化させながら，C_1 と C_2 がいずれも空でないならば，クラス間分散 σ^2 の値を求め，σ^2 の値が最大になる t を出力する．

大津の手法でしきい値を計算する C 言語の関数を作成することを考える．ただし，濃淡値は 0 から 255 の範囲の整数であり，画素の濃淡値は `unsigned char` 型の配列で渡されるものとする．

リスト 1 に，画素の濃淡値の列が入力されたときに，大津の手法でしきい値を求め，二値化後の画素の濃淡値を表示する C 言語のプログラムの一部を示す．その下に，このプログラムの実行例を示す．入力は，整数を空白で区切って並べ，整数以外の記号列で入力の終わりを表す．

リスト 1 ：大津の手法によって二値化を行うプログラム

```
#include <stdio.h>
#define G_MAX 255    /* 濃淡値の最大値 */

int otsu_method(int n, unsigned char *g);
void binarization(int n, unsigned char *g, int t);

int main() {
    int i, t, n = 0;
    unsigned char g[10000];

    while (scanf("%d", &i) == 1) g[n++] = (unsigned char) i;
    t = otsu_method(n, g);
    printf("t=%d\n", t);
    binarization(n, g, t);
    for (i = 0; i < n; i++) printf("%d ", g[i]);
    printf("\n");
    return 0;
}

int otsu_method(int n, unsigned char *g) {  問1  }
void binarization(int n, unsigned char *g, int t) {  問2  }
```

●実行例

```
> a.out
20 25 25 30 40 60 60 60 .
t=40
0 0 0 0 0 255 255 255
>
```

リスト 1 のプログラムについて，以下の問 1 〜問 5 に答えよ．

問 1 大津の手法でしきい値を計算する関数 `int otsu_method(int n, unsigned char *g)` の中身を記述せよ．なお，クラス間分散の値が最大となるしきい値が一意でない場合には，その中で最小のものを返すこと．

問 2 しきい値 t が与えられたときに，画素の濃淡値の入った `unsigned char` 型の配列を書き換え，二値化を行う関数 `void binarization(int n, unsigned char *g, int t)` の中身を記述せよ．

問 3 4 つの画素の濃淡値の列 $(10, 20, 30, 40)$ が与えられ，しきい値を $t = 30$ としたとき，クラス間分散 σ^2 の値を計算せよ．

問4 “10 20 30 40 .” が入力されたときのプログラムの出力を書け.

問5 “10 15 20 25 30 55 60 .” が入力されたときのプログラムの出力を書け.

問題2

逆ポーランド記法 (後置記法ともいう) で記述された四則演算の式をスタックを用いて計算するプログラムをC言語で作成することを考える．ただし，ここでは簡略化のため，式に記述できる数値は0〜9の1桁の非負整数のみとする．以下の問1〜問4に答えよ．

問1 スタックとは，後入れ先出し (LIFO: Last In First Out) 型のデータ構造である．リスト2にその実現例の一部を示す．この例では，関数 `Stack *create_stack()` でスタックを作成し，関数 `void push(Stack *stack, int data)` でスタックの先頭へ整数を追加する．`Stack` はスタックを管理するための構造体として定義されている．

これらの2つの関数に対応するように，スタックの先頭から整数を取り出すための関数 `int pop(Stack *stack)` を記述せよ．なお，スタックが空の場合には，0を返すものとする．

問2 逆ポーランド記法とは，演算数の後に演算子を配置する式の記述法であり，括弧を用いずに演算の順序を明確に表現できる特徴を持つ．例えば，一般的な中置記法による式「1+2×3」は，逆ポーランド記法では「1 2 3 × +」となる．

逆ポーランド記法による式の計算は，式を先頭から走査し，演算子が現れる度にその直前の2つの演算数を演算することによって行う．先述の「1 2 3 × +」の場合には，「×」演算子による乗算が「2」「3」に対して行われて「6」となり，次に「+」演算子による加算が「1」と「6」に対して行われる．

中置記法による式「2 × 4 + 6 ÷ 3 − 5」を逆ポーランド記法で記述せよ．

問3 問1のスタックを使って，逆ポーランド記法で記述された式（文字列で与えられるものとする）を計算する関数 `int calc(char *exp)` を記述せよ．

なお，乗算を表す演算子は「*」，除算を表す演算子は「/」で代用するものとし，0で除算した場合などのエラー処理は考えなくてもよい．計算の手順は以下のとおりとする．

1. 文字列の先頭から1文字取り出す．
2. その文字が数字ならば，整数に変換してスタックへ追加する．
3. その文字が「+」ならば，スタックから整数を2つ取り出し，2つの数を加算した結果をスタックへ追加する．
4. その文字が「-」ならば，スタックから整数を2つ取り出し，先に取り出した数を後から取り出した数から減算して，結果をスタックへ追加する．
5. その文字が「*」ならば，スタックから整数を2つ取り出し，2つの数を乗算した結果をスタックへ追加する．
6. その文字が「/」ならば，スタックから整数を2つ取り出し，先に取り出した数で後から取り出した数を除算して，結果をスタックへ追加する．除算の結果の小数点以下は切り捨てとする．
7. その文字が「=」ならば，スタックから整数を1つ取り出して，これを関数の返却値（戻り値）として関数を終了する．
8. 文字列から次の文字を取り出して，2. 以降を繰り返す．

問4 問1，問3の関数を用い，逆ポーランド記法で記述された文字列を入力ファイルから1行ずつ読み込み，計算結果を標準出力に表示する `main` 関数を記述せよ．なお，入力ファイルの指定は，プログラム実行時の引数としてファイル名を記述することによって行うものとする．

入力ファイルの例（ファイル名「test.txt」）と test.txt に対する実行例をそれぞれ図 1，図 2 に示す．

リスト 2：スタックを実現するプログラムの例

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

/* スタックの要素 */
typedef struct cell {
    int data ; /* データ */
    struct cell *next; /* 次の要素へのポインタ */
} Cell;

/* スタック */
typedef struct stack {
    Cell *top; /* 先頭要素へのポインタ */
} Stack;

/* スタックを作成する */
Stack *create_stack() {
    Stack *new_stack;

    new_stack = (Stack *) malloc((size_t) sizeof(Stack));
    new_stack->top = NULL;

    return new_stack;
}

/* スタックの先頭にデータを追加する */
void push(Stack *stack, int data) {
    Cell *new_cell;

    new_cell = (Cell *) malloc((size_t) sizeof(Cell));
    new_cell->data = data;
    if (stack->top != NULL) {
        new_cell->next = stack->top;
    } else {
        new_cell->next = NULL;
    }
    stack->top = new_cell;
}

/* スタックの先頭からデータを取り出す */
int pop(Stack *stack) {  問1  }
```

```
123*+=
93/1-=
```

図１：入力ファイルの例（test.txt）

```
> a.out test.txt
7
2
>
```

図２：test.txt に対する実行例