平成27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. 問題は全てを解答してください．
3. この試験に使用する解答用紙は 4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4）までを記入してください．
   問題1の問1，問2に対し（その1）を，
   問題1の問3，問4に対し（その2）を，
   問題2の問1，問2に対し（その3）を，
   問題2の問3，問4に対し（その4）を使用してください．
4. 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3と4に記載の事項は，試験開始前に必ず記入してください.】
5. 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．
6. 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．
7. 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

マクローリン展開	Maclaurin expansion
ラジアン	radian
標準ライブラリ	standard library
関数	function
引数	argument
x の n 乗	the n-th power of x
階乗	factorial
再帰関数	recursive function
角度	angle
度	degree
変換する	convert
円周率	circular constant
定数	constant
項	term
絶対値	absolute value
総和	summation
計算	calculation
文字列	string
編集距離	edit distance
レーベンシュタイン距離	Levenshtein distance
挿入	insertion
削除	deletion
置換	substitution
操作回数	the number of edits
格子点	lattice point
配列	array
最小	minimum
戻り値	return value
エディットグラフ	edit graph
最短距離	shortest distance
計算する	calculate
二次元配列	two-dimensional array
整数	integer

アルゴリズムとプログラミング

問題 1 $\sin x$ をマクローリン展開すると式 (1) となる．なお，このときの x の単位はラジアンである．

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!} + \cdots \tag{1}$$

この式を用いれば，標準ライブラリの関数を使わずに，$\sin x$ の値を求めるプログラムを作成することができる．リスト 1 はそのように printf 以外の標準ライブラリ関数を使わずに作成したプログラムで，d が $0°$ から $180°$ までの範囲の $\sin d$ の値を $10°$ 刻みで表示するものである．

以下の問 1 ～ 4 に答えよ．なお，引数に大きな数値を与えたときのオーバーフローは考えないものとする．

問 1 空欄 (1) を埋め，x を n 乗した数値を返す関数 double power(double x, int n) を完成させよ．なお，リスト 1 のプログラムでは，この関数が呼び出されるときに $n < 0$ または $x = n = 0$ となることはないため，そのような場合の処理は省略してもよい．

問 2 空欄 (2) を埋め，n の階乗を返す関数 double fact(int n) を完成させよ．ただし，$n! = n \times (n-1)!$ であることを利用した再帰関数とすること．

問 3 空欄 (3) を埋め，角度 d の単位を度からラジアンへ変換する関数 double toradian(int d) を完成させよ．$180°$ がラジアンでは円周率 π に等しいことを利用し，π はリスト 1 で定義されている定数 PI を用いること．

問 4 空欄 (4) を埋め，関数 power, fact, toradian と式 (1) を用いて $\sin d$ の値を double 型で返す関数 double msin(int d) を完成させよ．引数 d の単位は度とする．なお，式 (1) の n の値が充分に大きな項は絶対値が非常に小さくなるため，総和の計算では無視することができる．そのため，ここでは式 (1) の総和の計算は $n = 0$ の項から順に行い，絶対値が 10^{-10} より小さい項が現れた時点で終了し，その項から後の項は総和の計算に用いないものとする．また 10^{-10} の値はリスト 1 で定義される定数 MIN を用いること．

リスト 1：d が $0°$ から $180°$ までの $\sin d$ の値を表示するプログラム

```
#include <stdio.h>
#define PI 3.1415926536
#define MIN 1.0e-10

double power(double x, int n);
double fact(int n);
double toradian(int d);
double msin(int d);

int main(void) {
    int d;

    for (d=0; d<=180; d+=10) {
        printf("%4d : %15.10f\n", d, msin(d));
    }
```

リスト1：d が 0° から 180° までの $\sin d$ の値を表示するプログラム（続き）

```
    return 0;
}

double power(double x, int n) {
    ┌──────────────────────────┐
    │           (1)            │
    └──────────────────────────┘
}

double fact(int n) {
    ┌──────────────────────────┐
    │           (2)            │
    └──────────────────────────┘
}

double toradian(int d) {
    │           (3)            │
}

double msin(int d) {
    ┌──────────────────────────┐
    │                          │
    │           (4)            │
    │                          │
    └──────────────────────────┘
}
```

問題 2　2 つの文字列の違いを測る数値として編集距離 (レーベンシュタイン距離) が存在する．編集距離とは，ある文字列に対して 1 文字の挿入や削除，置換の操作を繰り返し適用して，別の文字に変換するために必要な操作回数の最小値である．例えば，文字列 “apple” を文字列 “peach” に変換する場合，図 2-1 に示す操作によって変換が完了する．図 2-1 は最小の操作回数で変換する手順の 1 つを示しており，編集距離は 5 となる．この編集距離を計算するために，ここではエディットグラフと呼ばれるグラフの最短距離取得問題の考え方を利用する．変換前の文字列 (長さ L_1) を横軸に，変換後の文字列 (長さ L_2) を縦軸に配置すると，エディットグラフは図 2-2 のような，$(L_1+1)\times(L_2+1)$ である格子点の配列として表現される．変換前の文字列を開始点 $(0,0)$ とし，文字の削除は横に，文字の挿入は下に，置換もしくは編集なしの場合は右下に移動する．この移動を繰り返し，点 (L_1,L_2) まで到達すれば文字列の変換が完了することとなる．1 回の移動については操作内容に応じたコストがあり，文字の挿入，削除，置換はコストが 1，編集なしの場合はコストが 0 と設定される．各格子点の左下に表示されている数値は，その点に移動するまでに要するコストの最小値を表す．これらの格子点の中で，点 (L_1,L_2) におけるコスト値が 2 つの文字列の編集距離となる．図 2-2 のエディットグラフ中の太線は，図 2-1 に示す操作を行った時の経路を表している．

apple	→	pple	(1 文字目の a を削除，操作回数 1)
	→	pele	(2 文字目を e に置換，操作回数 2)
	→	peae	(3 文字目を a に置換，操作回数 3)
	→	peac	(4 文字目を c に置換，操作回数 4)
	→	peach	(5 文字目に h を挿入，操作回数 5)

図 2-1: 文字列を変換するための操作手順の例

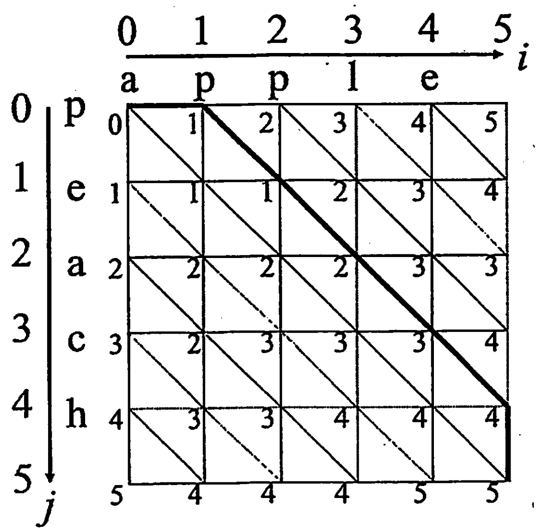

図 2-2: 文字列 “apple” を “peach” に変換するエディットグラフ

上記のような，2 つの文字列間の編集距離を返す関数 `EditDistance` を作成する．この関数 `EditDistance` の引数は変換前の文字列 str1 と変換後の文字列 str2 とし，戻り値は 2 つの文字

列間の編集距離とする．

文字列 str1 の文字数を len1，文字列 str2 の文字数を len2 として，各格子点までの最小コストを求め，その値を二次元配列 D に格納する．D[i][j] に格納するコストを求める処理は，大きく以下の 3 つの手順に分けることができる．

1. $0 \leq i \leq$ len1 を満たす各整数 i について，D[i][0] は i となる．
2. $0 \leq j \leq$ len2 を満たす各整数 j について，D[0][j] は j となる．
3. $1 \leq i \leq$ len1，$1 \leq j \leq$ len2 を満たすすべての整数 i，j の組について，コスト D[i][j] を以下のように求める．
   - str1[i-1] と str2[j-1] が同じ文字であれば編集する必要はないので置換コスト scost を 0，異なる文字であれば scost を 1 とする．
   - D[i-1][j-1] に scost を加えた値，D[i][j-1] に挿入コスト 1 を加えた値，D[i-1][j] に削除コスト 1 を加えた値の中から関数 `min` を利用して最小値を求め，D[i][j] に代入する．

これらの手順を実行することで，D[len1][len2] には文字列 str1 から str2 に変換する最小コスト，すなわち編集距離が格納される．

上記の処理により，2 つの文字列 “apple” と “peach” の編集距離を計算するプログラムをリスト 2 に示す．以下の問 1 〜 4 に答えよ．

問 1　3 つの整数値 a, b, c の最小値を返す関数 `int min(int a, int b, int c)` を作成するために，リスト 2 の空欄 (1) に適切なプログラムを記述せよ．

問 2　リスト 2 の空欄 (2) に適切なプログラムを記述せよ．

問 3　リスト 2 の空欄 (3) に適切なプログラムを記述せよ．

問 4　2 つの文字列 “peace” と “people” の編集距離を求めよ．

リスト2：2つの文字列間の編集距離を計算するプログラム

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

#define MAXSIZE 100

int min(int a, int b, int c)
{
    [ (1) ]
}

int EditDistance(char str1[], char str2[])
{
    int i, j;
    int scost;                  // 置換時のコストを格納する変数
    int D[MAXSIZE][MAXSIZE]; // エディットグラフを表現する配列
    int len1 = strlen(str1); // 文字列 str1 の長さ
    int len2 = strlen(str2); // 文字列 str2 の長さ

    // 元の文字列をすべて削除するためのコストを格納する
    for(i=0; i<=len1; i++)
        D[i][0] = i;
    // 空の文字列に変換後の文字列を挿入するためのコストを格納する
    [ (2) ]

    for(i=1; i<=len1; i++)
        for(j=1; j<=len2; j++){
            [ (3) ]
        }
    return D[len1][len2];
}

int main(void)
{
    int d;
    char str1[MAXSIZE] = "apple";
    char str2[MAXSIZE] = "peach";
    d = EditDistance(str1, str2);
    printf("%d\n", d);
    return 0;
}
```