# アルゴリズムとデータ構造 第 3 回課題 模範解答・解説

井上 康介

問題：以下のプログラムは，ユーザから入力された 100 個の点の座標について，それらの点の中からいずれか 3 つを選択して作ることのできる 3 角形のうち，最大の面積を有するものについて，その 3 角形を構成する**頂点の番号**および**面積**を出力するものである．

(1) 空欄（薄く塗られた部分）では何を行うべきか考え，行うべき処理を C 言語で記せ．なお，新たに変数を使う場合は変数の宣言を追加せよ．

(2) このプログラムの効率性を評価したい．面積を求める関数 SquareMeasure()が呼び出される回数ができるだけ少ない方が効率がよいものと考える．自分の考えたやり方では，呼出回数は何回になるかを理由をつけて示せ．

考え方：前回とほとんど同様の問題であり，今回は「全ての 2 点の組み合わせ」でなく「全ての 3 点の組み合わせ」を網羅しなくてはならない点が異なる．3 点を網羅する場合，三重ループを使い，1 点目 i のループは 0 番から NUM-3 番まで，2 点目 j は i+1 から NUM-2 番まで，3 点目 k は j+1 から NUM-1 番 (最後) まで回せばよい．

面積については，授業中に説明したとおり，3 角形を構成する 2 つの 3 次元ベクトル（$z$ 成分は $0$）の外積（$x$, $y$ 成分は 0）の $z$ 成分の絶対値の半分が 3 角形の面積である．

繰り返し回数については，前回の課題と同様であり，NUM 点から 3 点を選ぶ組み合わせの数となる．

解答：

(1) 解答コード例は以下の通り．ただし，このコードでは 100 個の点の座標を乱数で発生させている．

```
#include <stdio.h>
#include <stdlib.h>   // 乱数発生関数 drand48() のため
#include <math.h>     // 絶対値関数 fabs() のため

#define NUM 100       // 点の数を指定

// 関数 SquareMeasure() のプロトタイプ宣言
double SquareMeasure(double *, double *, double *);


int main(int argc, char *argv[])
{
  int       i, j, k;       // 汎用変数
  double    p[NUM][2];     // 点の座標の配列（座標型）
  double    s;             // 面積の一時保存先
  double    max;           // 「今のところの最大面積」
  int       points[3];     // 見つかった三角形の点番号


  srand48(0); // 乱数の種を初期化

  // NUM 個の点座標を乱数により設定
  for (i = 0; i < NUM; i++) {
    p[i][0] = 10.0 * drand48() - 5.0;
    p[i][1] = 10.0 * drand48() - 5.0;
  }

  max = 0.0;  // max の初期化（最初は小さい数）


  // メインの三重ループ
  for (i = 0; i < NUM - 2; i++)
    for (j = i + 1; j < NUM - 1; j++)
      for (k = j + 1; k < NUM; k++) {
        // s に面積を代入
        s = SquareMeasure(p[i],p[j],p[k]);

        // max より更に大きい面積の三角形が見つかった場合
        if (s > max) {
          max = s;          // max を更新
          points[0] = i;    // 点番号の保存
          points[1] = j;
          points[2] = k;
        }
      }

  // 結果の表示
  printf("Largest triangle: p[%d]-p[%d]-p[%d]",
         points[0], points[1], points[2]);
  printf("¥nSquare Measure = %f¥n", max);

  return 0;
}

// 三頂点の座標から三角形の面積を求める関数
double SquareMeasure(double *p0, double *p1,
                     double *p2)
{
  // 外積の z 成分の絶対値の 0.5 倍を返す
  return(0.5 * fabs((p1[0] - p0[0]) * (p2[1] - p0[1]) -
                    (p1[1] - p0[1]) * (p2[0] - p0[0])));
}
```

(2) 100 点から 3 点をとる組み合わせなので，

$$_{100}\mathrm{C}_3 = \frac{100 \cdot 99 \cdot 98}{3 \cdot 2 \cdot 1} = 161700$$

となる．より一般的に，$n$ 個の点の場合は，

$$_{n}\mathrm{C}_3 = n(n-1)(n-2)/6$$

となる．

※ このソースコードでは，点の座標をユーザ入力する代わりに，乱数（ランダムな数）で決めている．drand48() は，C 言語で [0, 1) の範囲の一様乱数を発生させるための関数であり，まず srand48()で乱数の初期条件（「乱数の種 (seed)」という）を決めた後，drand48()を呼び出すたびに 0.0 以上 1.0 未満の一様乱数（double 型）を返す．なお，整数の乱数を発生させる関数 rand()および srand()もある．詳細については，「drand48」で Google 検索したりすると情報が得られる (今の時代，どの科目に対しても分からないことをネット検索するという手段が強力である)．

※ 実はさらに計算量を減らす工夫の余地がある．0.5 倍の計算をどの時点で行うか，および fabs() を使う代わりに「もしも負なら -1.0 倍する」という処理の仕方をするなど．各自で考えてみよ．