Вычислительная геометрия на плоскости

Е.В. Андреева, Ю.Е. Егоров,
Москва

Продолжение. Начало в № 39, 40/2002

Многоугольники

4.1. Определение вида треугольника.

Сначала рассмотрим, как для трех вещественных чисел a, b и c проверить, существует ли треугольник, длины сторон которого равны указанным числам. Как известно, для положительных a, b и c необходимым и достаточным условием существования треугольника является выполнение неравенств треугольника: $a + b > c$, $a + c > b$ и $b + c > a$. Оказывается, что проверки справедливости только этих неравенств достаточно в любом случае. Действительно, складывая первые два неравенства, получим $2a + b + c > b + c$, откуда $a > 0$. Из других пар неравенств имеем $b > 0$ и $c > 0$. Значит, проверку знака a, b и c выполнять не нужно!

Если треугольник задан координатами своих вершин, то вычислять длины его сторон и проверять неравенства треугольника не требуется. В этом случае треугольник не существует тогда и только тогда, когда данные точки лежат на одной прямой. Так ли это, можно проверить, вычислив значение косого произведения $[\overrightarrow{P_1P_2}, \overrightarrow{P_2P_3}]$. Если оно равно нулю, то соответствующие векторы коллинеарны, т.е. все три точки лежат на одной прямой.

Как выяснить по координатам вершин треугольника, остроугольный он, прямоугольный или тупоугольный? Для этого достаточно знать, каким является наибольший из его углов. Вид угла легко определяется по знаку скалярного произведения образующих его векторов: оно положительно для острого угла, равно нулю для прямого и отрицательно для тупого. Поэтому подсчитаем все три скалярных произведения, перемножим их между собой и по знаку произведения определим вид треугольника. Обратите внимание на то, что значения самих углов для ответа на поставленный вопрос подсчитывать не нужно.

4.2. Проверка выпуклости многоугольника.

Выпуклость многоугольника с вершинами P_1, P_2, ..., P_n, перечисленными в порядке его обхода, легко проверить, если вычислить знаки косых произведений $[\overrightarrow{P_iP_{i+1}}, \overrightarrow{P_{i+1}P_{i+2}}]$, $i = 1, ..., n$ (здесь P_{n+1} есть P_1, а P_{n+2} — P_2). У выпуклого многоугольника знаки всех указанных произведений либо неположительны, либо неотрицательны (то есть знаки ненулевых произведений совпадают). Если мы знаем направление обхода, то знак косых произведений для выпуклого многоугольника определен: при обходе по часовой стрелке все косые произведения неположительны, а против часовой стрелки — неотрицательны.

4.3. Вычисление площади простого многоугольника.

Под простым мы понимаем такой многоугольник, граница которого не имеет самокасаний и самопересечений. Пусть вершины P_1, P_2, ..., P_n простого многоугольника перечислены в порядке обхода его границы. Напомним, что для вычисления его ориентированной площади нужно сложить ориентированные площади треугольников OP_1P_2, OP_2P_3, ..., OP_nP_1, где O — произвольная точка плоскости. Модуль полученной величины и есть искомая площадь:

$$S = \frac{1}{2}\left|\Sigma_{i=1}^{n}\,[\overrightarrow{OP_i}, \overrightarrow{OP_{i+1}}]\right|, \qquad (16)$$

— где полагается $P_{n+1} = P_1$. В качестве точки O в некоторых задачах бывает разумно выбрать одну из вершин многоугольника. Если же в качестве точки O взять начало координат, то формула (16) запишется в другом виде, также удобном для программирования:

$$\begin{aligned} S = \frac{1}{2}|&(x_1y_2 - x_2y_1) + (x_2y_3 - x_3y_2) + ... \\ &+ (x_ny_1 - x_1y_n)| = \frac{1}{2}|x_1(y_2 - y_n) + \\ &+ x_2(y_3 - y_1) + x_3(y_4 - y_2) + ... + \\ &+ x_n(y_1 - y_{n-1})|, \end{aligned} \qquad (17)$$

— где (x_i, y_i) — координаты точки P_i.

4.4. Построение выпуклой оболочки для множества из N точек плоскости.

Выпуклой оболочкой некоторого заданного множества точек называется пересечение всех выпуклых множеств, содержащих заданное множество. Для конечного множества точек выпуклой оболочкой всегда будет выпуклый многоугольник, все вершины которого являются точками исходного множества.

Задача состоит в том, чтобы для заданного конечного множества точек найти вершины выпуклой оболочки этого множества. Будем перечислять вершины в порядке обхода против часовой стрелки. Для эффективного решения этой задачи существует несколько различных алгоритмов (см. [1, 4]). Приведем наиболее простую реализацию одного из них — алгоритма Джарвиса. Этот алгоритм иногда называют "заворачиванием подарка" (см. *рис.* 18).

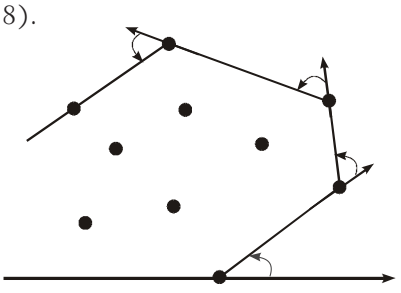

Рис. 18

Перечисление точек искомой границы выпуклого многоугольника начнем с правой нижней точки P_1, которая заведомо принадлежит границе выпуклой оболочки. Обозначим ее координаты (x_1, y_1). Следующей при указанном порядке обхода будет точка $P_2(x_2, y_2)$. Она, очевидно, обладает тем свойством, что все остальные точки лежат "слева" от вектора $\overrightarrow{P_1P_2}$, т.е. ориентированный угол между векторами $\overrightarrow{P_1P_2}$ и $\overrightarrow{P_1M}$ неотрицателен для любой другой точки M нашего множества. Для кандидата на роль точки P_2 проверяем выполнение условия $[\overrightarrow{P_1P_2}, \overrightarrow{P_1M}]$? 0 со всеми точками M. Если точек, удовлетворяющих этому условию, несколько, то вершиной искомого многоугольника станет та из них, для которой длина вектора $\overrightarrow{P_1P_2} = (x_2 - x_1, y_2 - y_1)$ максимальна.

Будем поступать так же и дальше. Допустим, уже найдена i-я вершина $P_i(x_i, y_i)$ выпуклой оболочки. Для следующей точки $P_{i+1}(x_{i+1}, y_{i+1})$ косые произведения $[\overrightarrow{P_iP_{i+1}}, \overrightarrow{P_iM}]$ неотрицательны для всех точек M. Если таких точек несколько, то выбираем ту, для которой вектор $\overrightarrow{P_iP_{i+1}}$ имеет наибольшую длину. Непосредственно поиск такой точки можно осуществлять так. Сначала мы можем считать следующей, $(i + 1)$-й, любую точку. Затем вычисляем значение $[\overrightarrow{P_iP_{i+1}}, \overrightarrow{P_iM}]$, рассматривая в качестве M все остальные точки. Если для одной из них указанное выражение меньше нуля, считаем следующей ее и продолжаем проверку остальных точек (аналогично алгоритму поиска минимального элемента в массиве). Если же значение выражения равно нулю, то сравниваем квадраты длин векторов. В результате за $O(N)$ операций очередная вершина выпуклой оболочки будет найдена. Продолжая эту процедуру, мы рано или поздно вернемся к точке P_1. Это будет означать, что выпуклая оболочка построена.

При решении данной задачи в случае изначально целочисленных координат мы полностью можем избежать применения вещественной арифметики, а следовательно, избавиться от потери точности вычислений. В противном случае в решение могут быть включены "лишние" точки, близкие к границе выпуклой оболочки, или не учтены некоторые из "нужных" точек. Сложность данного алгоритма составит $O(kN)$, где k — количество точек в выпуклой оболочке, в худшем случае равное N.

Программа решения данной задачи алгоритмом Джарвиса была приведена в [6]. Ее можно несколько упростить, поэтому приведем уточненный вариант реализации данного алгоритма. Множество исходных точек находится в массиве a, все точки, принадлежащие выпуклой оболочке, будем записывать в массив b.

```
type vv = record
            x, y: longint;
          end;
var a, b: array[1..100] of vv;
  min, m, i, j, k, n: integer;
function vect(a1,a2,b1,b2: vv): longint;
{косое произведение векторов a1a2 и b1b2}
begin
  vect := (a2.x - a1.x)*(b2.y - b1.y) -
          (b2.x - b1.x)*(a2.y - a1.y)
end;
function dist2(a1,a2: vv): longint;
{квадрат длины вектора a1a2}
begin
  dist2 := sqr(a2.x - a1.x) + sqr(a2.y - a1.y)
end;

begin {Main}
  readln(n);{количество точек}
  for i := 1 to n do
    read(a[i].x, a[i].y);
 {ищем правую нижнюю точку}
  m := 1;
  for i := 2 to n do
    if a[i].y < a[m].y then m := i else
    if (a[i].y = a[m].y) and
       (a[i].x > a[m].x) then m := i;
 {запишем ее в массив выпуклой оболочки b и
  переставим на первое место в массиве a}
  b[1] := a[m];
  a[m] := a[1];
  a[1] := b[1];
  k := 1;
  min := 2;
  repeat
   {ищем очередную вершину выпуклой оболочки}
    for j := 2 to n do
     if (vect(b[k],a[min],b[k],a[j]) < 0) or
       ((vect(b[k],a[min],b[k],a[j]) = 0) and
      (dist2(b[k],a[min]) < dist2(b[k],a[j])))
      then min := j;
    k := k + 1;
    {записана очередная вершина}
    b[k] := a[min];
    min := 1;
 until (b[k].x = b[1].x) and (b[k].y = b[1].y);
 {пока ломаная не замкнется}
  for j := 1 to k - 1 do {печать результата}
    writeln(b[j].x,' ',b[j].y)
end.
```

Существует другой алгоритм решения этой задачи (алгоритм Грэхема) с вычислительной сложностью $O(N\log N)$, основанный на предварительной сортировке точек исходного множества по значению угла в полярной системе координат с центром в одной из точек выпуклой оболочки. То есть наиболее трудоемкой задачей оказывается именно сортировка исходных точек. Сортировку точек можно производить по знаку косого произведения $[\overrightarrow{P_1P_i}, \overrightarrow{P_1P_{i+1}}]$, где P_1 — любая вершина выпуклой оболочки (например, все та же правая нижняя точка). В отсортированном массиве точек все указанные произведения должны быть неотрицательны. Точки с равными углами $([\overrightarrow{P_1P_i}, \overrightarrow{P_1P_{i+1}}] = 0)$ располагаются в порядке увеличения длин соответствующих векторов $\overrightarrow{P_1P_i}$ (*рис.* 19).

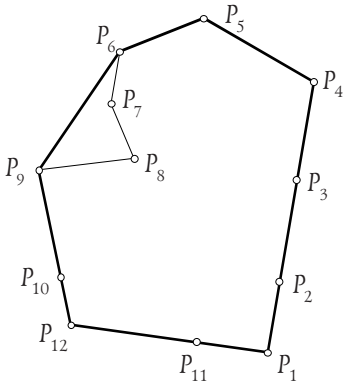

Рис. 19. Вершины выпуклой оболочки множества точек $\{P_i\}$: P_1, P_4, P_5, P_6, P_9, P_{12}. Номера всех точек соответствуют сортировке по полярному углу

Далее просмотр Грэхема использует стек, в котором хранятся точки, являющиеся кандидатами в выпуклую оболочку. Сначала в стек помещается первая из отсортированных точек. Затем — соседняя с ней вершина выпуклой оболочки. Если на первом из лучей точек несколько, то это точка этого луча P_i, наиболее удаленная от P_1. Третьей — точка P_{i+1}. Пусть в вершине стека находится точка P_k. Рассмотрим следующую в порядке увеличения полярного угла точку исходного множества P_i. Пока участок ломаной $P_{k-1}P_kP_i$ не является выпуклым (см. задачу 4.2), из стека удаляется очередная точка P_k. Затем P_i помещается в стек. В момент окончания просмотра всех точек в стеке будут находиться в точности все вершины выпуклой оболочки. Так как любая точка добавляется в стек ровно один раз, то и удаляется она из него не более одного раза, поэтому время просмотра составляет $O(N)$.

Приведем реализацию именно такого просмотра. Для наглядности сортировку проведем пузырьковым алгоритмом. В качестве стека используется массив b. Описания переменных и функций совпадают с приведенными выше.

```
begin
  readln(n);
  for i := 1 to n do
    read(a[i].x, a[i].y);
 {ищем правую нижнюю точку}
  m := 1;
  for i := 2 to n do
    if a[i].y < a[m].y then m := i else
    if (a[i].y = a[m].y) and
       (a[i].x > a[m].x) then m := i;
 {запишем ее в массив выпуклой оболочки b и
  переставим на первое место в массиве a}
  b[1] := a[m];
  a[m] := a[1];
  a[1] := b[1];
 {остальные точки сортируем пузырьком
  по значению полярного угла}
  for i := n downto 3 do
    for j := 2 to i - 1 do
     if (vect(a[1],a[j],a[1],a[j + 1]) < 0) or
        ((vect(a[1],a[j],a[1],a[j + 1]) = 0 ) and
       (dist2(a[1],a[j]) > dist2(a[1],a[j + 1])))
        then
      begin {b[n] - вспомогательная переменная}
        b[n] := a[j];
        a[j] := a[j + 1];
        a[j + 1] := b[n]
      end;
 {ищем вторую вершину выпуклой оболочки}
  i := 2;
  while vect(a[1],a[i + 1],a[1],a[2]) = 0 do
    i := i + 1;
  b[2] := a[i];
  b[3] := a[i + 1];
  k := 3;
  for i := i + 2 to n do
   begin
    {проверка выпуклости}
    while vect(b[k - 1],b[k],b[k],a[i]) <= 0 do
      k := k - 1;{удаляем точку из стека}
    k := k + 1;
    b[k] := a[i] {добавляем точку в стек}
   end;
  for j := 1 to k do {печать решения}
    writeln(b[j].x,' ',b[j].y)
end.
```

На этом мы заканчиваем обзор решения элементарных задач вычислительной геометрии на плоскости. На них опирается решение более сложных, в частности олимпиадных, задач — см., например, [4—6]. Надеемся, что внимательному читателю данной статьи решение последних окажется теперь по силам.

Литература

1. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы. Построение и анализ. М.: МЦНМО, 2000.

2. *Шикин Е.В., Боресков А.В., Зайцев А.А.* Начала компьютерной графики. М.: Диалог-МИФИ, 1993.

3. *Курант Р., Роббинс Г.* Что такое математика. М.: ОГИЗ, государственное изд-во технико-теоретической литературы, 1947.

4. *Окулов С.М.* 100 задач по информатике. Киров: Изд-во ВГПУ, 2000.

5. *Станкевич А.С.* Решение задач I Всероссийской командной олимпиады по программированию. Информатика № 12/2001.

6. *Андреева Е.В.* Геометрические задачи на олимпиадах по информатике. Информатика № 14/2002.