Разбор задач Седьмой Интернет-олимпиады

Автор: Федор Царев

Введение

В базовой номинации Седьмой Интернет-олимпиады сезона 2006-2007 участникам было предложено для решения 8 задач. В олимпиаде приняло участие 108 команд, из них 103 решили хотя бы одну задачу.

Наиболее простой оказалась задача «G. Словарные квадраты» — ее решили 96 команд. Наиболее сложными — задачи «B. Цепочка слов» и «C. Шифр Решетка» — каждую из них решило по 11 команд.

Условия задач, результаты олимпиады, тесты и решения жюри можно найти на сайте интернет-олимпиад `http://neerc.ifmo.ru/school/io`.

Задача A. Автогонки

Если переформулировать задачу в терминах теории графов, то она будет звучать так: «Задан неориентированный связный граф. Проверить, существует ли в нем вершинно-простой цикл.».

Напомним, что связный неориентированный граф без циклов называется *деревом*. Известно, что необходимым и достаточным условием того, что граф с n вершинами (в терминологии данной задачи — в городе n перекрестков) — дерево, является то, что в нем $(n-1)$ ребро (дорога) [?].

Таким образом, задача свелась к определению количества ребер в графе. Для этого воспользуемся так называемой *матрицей смежности*. Пусть $a[i][j] = 1$, если перекрестки i и j соединены дорогой, и $a[i][j] = 0$ — в противном случае [?].

Соответственно, при чтении из входного файла дороги, описанной числами u и v, необходимо выполнить два присваивания: $a[u][v] = 1$ и $a[v][u] = 1$.

Количество дорог в городе будет равно количеству единиц в двумерном массиве a, деленному пополам — так как каждая дорога в массиве a отмечена два раза.

Задача B. Цепочка слов

Отсортируем все заданные слова по неубыванию длины и перенумеруем их числами от 1 до n в полученном порядке. Пусть $a[i][j] = 1$, если слово номер i является собственным префиксом слова номер j.

Далее, применим для решения задачи метод динамического программирования [?]. Пусть $l[i]$ — максимальная длина цепочки слов, заканчивающейся в слове номер i. Тогда для $l[i]$ справедливо следующее рекуррентное соотношение: $l[i] = \max_{1 \le j < i, a[j][i]=1}(l[j]) + 1$.

Ответом на задачу является максимум из $l[i]$.

Задача C. Шифр «Решетка»

Рассмотрим подквадрат размера $n/2 \times n/2$ заданного квадрата с левым верхним углом в левом верхнем угле «большого» квадрата.

Запишем в каждую клетку этого подквадрата число от 1 до 4. Эти числа будут иметь следующий смысл: 1 — соответствующая клетка будет накрыта вырезанной клеткой в тот момент, когда ключ повернут на 0 градусов, 2 — на 90 градусов, 3 — на 180 градусов, 3 — на 270 градусов.

Нетрудно видеть, что любому «правильному ключу» соответствует некоторая расстановка чисел от 1 до 4 в описанном подквадрате, и, наоборот, каждой расстановке чисел соответствует «правильный ключ». Таким образом, количество «правильных ключей» равно $4^{n^2/4}$.

Для того, чтобы вычислить это число по модулю m, можно применить, например, дихотомическое возведение в степень [?]. Кроме этого, необходимо быть внимательным и использовать тип

`long` (в Java), `long long` (в C++) или `int64` (в Delphi), так как промежуточные результаты вычислений могут превышать $2^{31} - 1$ и не «помещаться» в тип `int` (`longint`).

Задача D. Гирлянда

Задача решается методом перебора. Заметим, что количество различных способов раскраски лампочек гирлянды равно k^n. В худшем случае $k^n = 8^8 = 16777216$, что достаточно мало. Для того, чтобы проверить, удовлетворяет ли конкретная раскраска лампочек эстетическим взглядам заказчиков, необходимо для каждого провода проверить, что он соединяет лампочки разных цветов.

Наиболее простой способ организации перебора в этой задаче — использование рекурсии [?]. Приведенная ниже процедура (на языке *Delphi*) реализует этот подход.

```
procedure go(pos : longint);
var
  i : longint;
  good : boolean;
begin
  if (pos = n + 1) then begin
    good := true;
    for i := 1 to m do begin
      if (cur[e[i][1]] = cur[e[i][2]]) then begin
        good := false;
        break;
      end;
    end;
    if (good) then begin
      inc(ans);
    end;
    exit;
  end;
  for i := 1 to k do begin
    cur[pos] := i;
    go(pos + 1);
  end;
end;
```

Задача E. Игра «Жизнь»

Решение этой задачи состоит в моделировании процесса, описанного в условии задачи. При разумной реализации на переход от одного поколения к следующему будет требоваться порядка $8 \cdot n \cdot m$ операций. Так как $n, m \leq 100$, то $8 \cdot n \cdot m \leq 80000$. Общее время работы программы будет составлять порядка $8 \cdot k \cdot n \cdot m \leq 8000000$. Таким образом, такое решение легко укладывается в ограничение по времени.

Приведем одну из возможных реализаций этого подхода (на языке *Delphi*, приводится только та часть программы, которая осуществляет переход к следующему поколению):

```
    for i := 1 to n do begin
      for j := 1 to m do begin
        cnt := 0;
        for di := -1 to 1 do begin
          for dj := -1 to 1 do begin
            if (di = 0) and (dj = 0) then begin
              continue;
```

```
          end;
          ni := (i + di + n - 1) mod n + 1;
          nj := (j + dj + m - 1) mod m + 1;
          if (curr[ni][nj] = '*') then begin
            inc(cnt);
          end;
        end;
      end;
      if (curr[i][j] = '.') then begin
        if (cnt = 3) then begin
          next[i][j] := '*'
        end else begin
          next[i][j] := '.'
        end;
      end else begin
        if (cnt = 2) or (cnt = 3) then begin
          next[i][j] := '*';
        end else begin
          next[i][j] := '.';
        end;
      end;
    end;
  end;
  for i := 1 to n do begin
    for j := 1 to m do begin
      curr[i][j] := next[i][j]
    end;
  end;
```

Задача F. Оптимизация на окружности

Так как $A, B > 0$, то ясно, что искомый максимум достигается при $x, y > 0$. Таким образом, $y = \sqrt{R^2 - x^2}$.

Осталось максимизировать функцию $A\,x + B\,\sqrt{R^2 - x^2}$. Для этого найдем ее производную и приравняем ее нулю:

$$(A\,x + B\,\sqrt{R^2 - x^2})' = A - B\frac{x}{\sqrt{R^2 - x^2}} = 0$$

Решив это уравнение, получаем:

$$x = \frac{A\,R}{\sqrt{A^2 + B^2}}$$

$$y = \frac{B\,R}{\sqrt{A^2 + B^2}}$$

Задача G. Словарные квадраты

Прочитаем элементы заданного во входном файле квадрата в массив A. Нетрудно видеть, что решение задачи после этого сводится к проверке следующего свойства: $\forall i, j$ $(1 \le i, j \le 4)$ $A[i][j] = A[j][i]$.

Это можно сделать, например, с помощью следующей программы (на языке Delphi):

```
for i := 1 to 4 do begin
  for j := 1 to 4 do begin
    if (a[i][j] <> a[j][i]) then
    begin
      writeln('SOMETHING OTHER');
      halt(0);
    end;
  end;
end;
writeln('WORD SQUARE');
```

Задача H. Диалоги по *UCM*

Решение задачи состоит в аккуратной реализации того, что описано в ее условии.

Ниже приведена одна из возможных реализаций решения (на языке *Delphi*):

```
var
  name : array[1..2] of string;

function secondToken(s : string) : string;
var
  l : longint;
  i : longint;
  res : string;
begin
  l := length(s);
  res := '';
  i := 1;
  while (s[i] <> ' ') do begin
    res := res + s[i];
    inc(i);
  end;
  res := '';
  inc(i);
  while (s[i] <> ' ') do begin
    res := res + s[i];
    inc(i);
  end;
  result := res;
end;

var
  st : string;
  replica : string;
  cur : longint;
  l : longint;

begin
  reset(input, 'ucm.in');
  rewrite(output, 'ucm.out');
  name[1] := 'Fedya';
  readln(st);
```

```
  name[2] := secondToken(st);
  cur := 1;
  while (true) do begin
    readln(st);

    replica := copy(st, 11, length(st) - 10);

    if (replica = name[2] + ' signed off') then begin
      break;
    end;

    if not (replica[length(replica)] in ['.', '!', '?']) then begin
      replica := replica + '.';
    end;

    if (replica[length(replica)] = '.') then begin
      replica := copy(replica, 1, length(replica) - 1) + ',';
    end;

    writeln('"' + replica + '" --- skazal ' + name[cur] + '.');
    cur := 3 - cur;
  end;

end.
```

Список литературы

[1] Свободная многоязычная энциклопедия http://ru.wikipedia.org/wiki

[2] Асанов М.О., Баранский В.А., Расин В.В. Дискретная математика: Графы, матроиды, алгоритмы.