Ведерников Николай Викторович,
Кротков Павел Андреевич,
Ульянцев Владимир Игоревич

ЗАДАЧА «ХВОСТ ГРАФА»

Этой статьей мы продолжаем цикл публикаций олимпиадных задач для школьников по информатике. Решение таких задач и изучение разборов поможет Вам повысить уровень практических навыков программирования и подготовиться к олимпиадам по информатике.

В этой статье рассматривается задача «Хвост графа», которая предлагалась на Второй индивидуальной интернет-олимпиаде по программированию в 2011–2012 учебном году. Материалы этой олимпиады можно найти на сайте http://neerc.ifmo.ru/school/io/ .

УСЛОВИЕ ЗАДАЧИ

Петя с детства любит воздушных змеев. Ему очень нравится наблюдать, как они летают. Особенно Пете нравится смотреть, как развевается хвост змея.

На одном из уроков математики, которую Петя тоже любит, учитель рассказывал про графы. Во время рассказа в качестве примера учитель на доске нарисовал неориентированный граф, похожий на воздушного змея. Петя сразу обратил внимание на хвост графа.

Хвостом графа назовем такую последовательность вершин, что первая вершина последовательности соединена ребром только со второй, вторая –только с первой и третьей, а все последующие вершины хвоста соединены только с соседними. Последняя вершина хвоста может быть связана либо только с предпоследней его вершиной, либо с предпоследней вершиной и вершиной, ко-

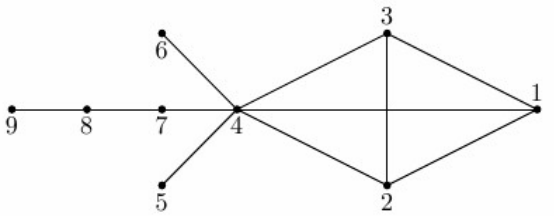

Рис. 1

торая не входит в наш хвост. Длиной хвоста графа назовем число вершин в нем. Так, приведенный на рисунке граф имеет один хвост длины три, состоящий из вершин с номерами девять, восемь и семь, и два хвоста длины один.

С тех пор, при виде графа Петя сразу же ищет у него самый длинный хвост. Помогите ему: напишите программу, решающую данную задачу для произвольного графа.

Формат входного файла

В первой строке входного файла заданы числа n и m ($1 \le n \le 100\,000$, $1 \le m \le 200\,000$) – количество вершин и ребер в графе соответственно. Следующие m строк содержат по два числа a_i и b_i – номера вершин, которые соединяет соответствующее ребро.

Каждая пара вершин соединена не более, чем одним ребром, и никакое ребро не соединяет вершину с ней же. Из любой вершины существует путь до любой другой вершины графа.

Формат выходного файла

В выходной файл требуется вывести одно целое число – длину самого длинного хвоста графа.

Примеры входных и выходных данных

graphtail.in	graphtail.out
9 11 1 2 1 3 1 4 2 3 2 4 3 4 4 5 4 6 4 7 7 8 8 9	3

РАЗБОР ЗАДАЧИ

В качестве первого шага к решению задачи рассмотрим вопрос о представлении графа в памяти компьютера. Один из самых известных и самых простых способов представления графов – матрица смежности, описание которой можно найти в [1]. Однако, при данных в условии задачи ограничениях, ее использование невозможно из-за объема памяти $O(n^2)$, который она требует. Поэтому при решении данной задачи будем хранить граф с помощью списков смежности [2].

Опишем этот способ хранения графов. Поскольку в данной задаче граф неориентированный, каждое его ребро превратим в два противоположно направленных ориентированных ребра. Далее, для каждого ребра в массиве `edge` мы будем хранить только один конец этого ребра, то есть вершину, в которую это ребро ведет. Кроме того, для каждого ребра в массиве `next` будем хранить номер следующего ребра, выходящего из этой же вершины.

Заметим теперь, что если для каждой вершины в массиве `first` хранить номер первого ребра в списке выходящих из нее, то мы сможем для каждой вершины перебрать все вершины, смежные с ней, за время, пропорциональное их числу. Также несложно добавить еще одно ребро в список: достаточно в `next` добавляемого ребра записать старое значение `first` для начала этого ребра, после чего нужно в `first` для начала добавляемого ребра записать его номер. Данная операция будет выполняться за $O(n)$.

В листинге 1 приведен фрагмент программы, считывающий данную во входном файле информацию и преобразующий ее в описанную структуру данных.

Приведенный фрагмент кода, кроме описанной выше информации, сохраняет также число ребер, выходящих из каждой вершины. Это число для каждой вершины называется ее *степенью* и после завершения выполнения приведенного фрагмента программы хранится в массиве `deg`.

Будем решать задачу, исходя из данного в условии определения хвоста графа. По определению, первая вершина хвоста всегда имеет степень один, значит, перебрав все такие вершины, мы сможем найти и все хвосты. Научимся определять длину хвоста, зная его начало.

Возьмем найденный конец хвоста (вершину со степенью один). Перейдем к един-

Листинг 1. Реализация ввода данных и преобразования их в описанную структуру данных

```
var
  deg, first : array [1..maxN] of integer;
  next, edge : array [1..2 * maxM] of integer;

procedure addedge(a, b : integer);
begin
  k := k + 1;
  deg[a] := deg[a] + 1;
  next[k] := first[a];
  first[a] := k;
  edge[k] := b;
end;

read(n, m);
k := 0;
for i := 1 to n do begin
  first[i] := 0;
  deg[i] := 0;
end;
for i := 1 to m do begin
  read(a, b);
  addedge(a, b);
  addedge(b, a);
end;
```

Листинг 2. Реализация описанного алгоритма

```
ans := 0;
for i := 1 to n do begin
  if (deg[i] = 1) then begin
    len := 1;
    curVertex := edge[first[i]];
    prevVertex := i;
    while (deg[curVertex] = 2) do begin
      len := len + 1;
      tmp := curVertex;
      if (edge[first[curVertex]] <> prevVertex) then begin
        curVertex := edge[first[curVertex]]
      end else begin
        curVertex := edge[next[first[curVertex]]];
      end;
      prevVertex := tmp;
    end;
    if (deg[curVertex] = 1) then begin
      len := len + 1;
    end;
    if (len > ans) then begin
      ans := len;
    end;
  end;
end;
```

ственной вершине, смежной с ним. Если степень этой вершины больше двух – принадлежать хвосту она не может, и хвост кончился. Если же степень этой вершины два – она принадлежит хвосту, и тогда можно перейти к той из двух смежных с ней вершин, которую мы еще не рассмотрели. Будем переходить так до тех пор, пока мы не встретим вершину со степенью, отличной от двух, – эта ситуация будет означать конец хвоста.

Фрагмент программы, выполняющий описанные действия, приведен в листинге 2. В приведенном фрагменте программы на каждой итерации цикла `while` предыдущая пройденная вершина хранится в переменной `prev`, а подсчитываемая длина хвоста – в переменной `len`. При этом важно не забыть, что если степень вершины – конца хвоста равна одному, то эту вершину также надо добавить в хвост.

Важным критерием качества решения, о котором еще не было сказано, является асимптотическое время его работы. Заметим, что каждое ребро принадлежит не более чем одному хвосту, а значит, будет рассмотрено приведенным алгоритмом не более одного раза. Кроме того, приведенный алгоритм по одному разу проверит степень каждой вершины на равенство единице. Значит, суммарное время работы алгоритма составляет $O(n+m)$.

Литература

1. *Асанов М., Баранский В., Расин В.* Дискретная математика: графы, матроиды, алгоритмы: Учебное пособие. 2-е изд., испр. и доп. СПб.: Издательство «Лань», 2010.

2. *Скиена С.* Алгоритмы. Руководство по разработке. 2-е изд. Пер. с англ. СПб.: БХВ-Петербург, 2011.

Ведерников Николай Викторович,
студент второго курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике,

Кротков Павел Андреевич,
студент второго курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике,

Ульянцев Владимир Игоревич,
студент пятого курса кафедры «Компьютерные технологии» НИУ ИТМО, член жюри Интернет-олимпиад по информатике, член жюри ВКОШП.