ИТОГИ НАУКИ И ТЕХНИКИ.
Современная математика и ее приложения.
Тематические обзоры.
Том 238 (2025). С. 59–68
DOI: 10.36535/2782-4438-2025-238-59-68

УДК 519.17

ЭФФЕКТИВНЫЙ АЛГОРИТМ ПОИСКА ФИНАЛЬНЫХ ВЕРШИН НА ОБОБЩЕННОМ ФУНКЦИОНАЛЬНОМ ГРАФЕ

© 2025 г. **О. В. ЗУБКОВ**

Аннотация. В работе введены в рассмотрение 2-исходящие графы, обобщающие функциональные графы и моделирующие дискретные динамические системы специального вида. Классифицированы вершины и дуги 2-исходящего графа, определены пути на этих графах и доказаны некоторые свойства этих путей. В итоге построен эффективный алгоритм, который с линейной сложностью строит финальные вершины для путей, начинающихся в каждой из вершин 2-исходящего графа и доказана его корректность.

***Ключевые слова*:** дискретная динамическая система, функциональный граф, сложность алгоритма.

AN EFFICIENT ALGORITHM FOR FINDING FINAL VERTICES IN A GENERALIZED FUNCTIONAL GRAPH

© 2025 **O. V. ZUBKOV**

Abstract. In the paper, 2-outgoing graphs are introduced into consideration, generalizing functional graphs and modeling discrete dynamic systems of a special type. The vertices and arcs of a 2-outgoing graph are classified, paths on these graphs are defined, and some properties of these paths are proved. As a result, an efficient algorithm is constructed that, with linear complexity, constructs final vertices for paths starting at each of the vertices of a 2-outgoing graph, and its correctness is proven.

***Keywords and phrases*:** discrete dynamic system, functional graph, algorithm complexity.

***AMS Subject Classification*:** 68W40

1. Введение. Рассмотрим конечное множество X, состоящее из n элементов и всюду определённую на нём функцию $f : X \to X$. Пара (X, f) определяет дискретную динамическую систему (сокращенно ДДС). Функцию f будем называть определяющей для ДДС.

Функциональным графом, или диаграммой состояний дискретной динамической системы называется ориентированный граф G_f, вершины которого соответствуют элементам множества X, а дуги — всем упорядоченным парам $x \to f(x)$ по всем $x \in X$. Далее будем отождествлять элементы множества X и вершины её функционального графа.

Функционирование дискретной динамической системы из начального состояния $\omega \in X$, называется бесконечная последовательность $\omega, f(\omega), f(f(\omega)) \ldots$. Очевидно, что функционирование ДДС из начального состояния ω может быть интерпретировано как некоторый путь на её функциональном графе, начинающийся (стартующий) в вершине ω.

Дискретные динамические системы имеют множество практических приложений и интерпретаций. В [3] рассмотрены линейные ДДС, определяющая функция для которых является линейным отображением на конечномерном векторном пространстве, в [1] — функционирование ДДС циркулянтного типа, в [2] — ДДС циркулянтного типа с дополнительными свойствами на вершинах.

В основополагающей работе [4] показано, что любая компонента слабой связности функционального графа состоит из единственного цикла (возможно петли), такого, что некоторые его вершины являются корнями деревьев, все дуги которых ориентированы к корню. Очевидно, что любой, достаточно длинный путь на функциональном конечном графе в итоге зацикливается в одном из его циклов.

В данной работе нас не будет интересовать характер и внутренние свойства множества X, будем рассматривать непосредственно сам функциональный граф G_f и некоторые алгоритмы эффективного построения путей на нём. Классической задачей алгоритмического программирования, связанной с функциональным графом, является следующая: для функционального графа с достаточно большим числом вершин n, для каждой его вершины v_i и заданного, так же достаточно большого числа k, эффективно определить в какой вершине закончится путь длины k, начинающийся в v_i. Под эффективным здесь понимаем алгоритм, имеющий асимптотику лучшую, нежели $O(nk)$, каковая достигается простым моделированием путей длины k из каждой вершины v_i. Эффективно данная задача решается путем построения двоичных подъемов из каждой вершины функционального графа G_f с общей совокупной сложностью алгоритма $O(n \log_2 k)$ для всех вершин.

Далее нас будут интересовать пути на функциональных графах и некоторых их обобщениях, обладающие свойством эйлеровости, т.е. содержащие в своем составе дуги не более чем по одному разу. Сразу оговорим, что в интересующей нас постановке мы не требуем, чтобы эти пути проходили по каждой дуге графа G_f или его некоторой компоненты. Единственное, что будет запрещено — проходить по уже пройденной дуге вторично при старте из любой вершины v_i. При этом (в изначальной постановке задачи), при переходе к старту из другой вершины v_j, все старые метки посещенности дуг стираются и необходимые дуги снова доступны для единоразового прохождения. Очевидно, что при старте из любой вершины v_i, на каком-либо шаге произойдет зацикливание пути, и этот процесс остановится в той вершине графа, которая является корнем дерева, содержащего вершину v_i. Эту вершину остановки v_{fin} будем называть финальной для пути из вершины v_i или просто финальной для v_i.

В качестве иллюстрирующего примера для этой задачи рассмотрим следующий. Пусть дана система, состоящая из идентичных простейших устройств с одним входом и одним выходом. Устройства работают с сигналом одного вида следующим образом: получив сигнал на свой вход, устройство передает его на свой выход и перестаёт функционировать. Ко входу любого такого устройства можно присоединить любое количество выходов других устройств. Для любого устройства A_i этой системы его выход подключен ко входу некоторого устройства A_j из этой же системы; возможна ситуация, когда $A_i = A_j$. Выбирается любое произвольное стартовое устройство и на его вход подается инициирующий сигнал. Далее этот сигнал передаётся по цепочке и в некоторый момент достигает финального устройства, после чего процесс заканчивается. По исходному устройству A_{start} требуется эффективно определить финальный для цепочки устройств A_{fin}, сигнал на вход которого ещё пришёл, но выходного сигнала уже не генерирует. Задача ставится в массовом варианте, т.е. в качестве стартового перебираются все устройства системы и ответ выводится в формате $A_1 \to A_{\text{fin}_1}$, $A_2 \to A_{\text{fin}_2} \dots A_n \to A_{\text{fin}_n}$.

Переходя к описанию алгоритма решения этой задачи в общем виде, заметим, что наивный алгоритм, моделирующий все пути из всех вершин графа снова будет иметь квадратичную сложность $O(n^2)$ и не будет считаться эффективным. Достаточно очевидно следующее эффективное решение методом динамического программирования: стартуем из произвольной вершины v_i, для которой ещё не известна её финальная. Проходим по пути из неё до тех пор, пока либо не зациклимся, либо не попадём в вершину, для которой уже известна финальная. Далее возвращаемся по только что пройденному пути и выставляем для каждой посещённой в этом пути вершины метку финальной для неё вершины. Если произошло зацикливание в вершине v_j, то возвращаемся по циклу и для каждой его вершины ставим финальной для неё саму эту вершину, а для только что пройденных вершин, не принадлежащих циклу — вершину v_j. Если же мы попали в вершину, для которой уже известна финальная и она равна v_{fin}, то возвратимся по пройденному из v_i пути и поставим для всех вершин этого пути в качестве финальной v_{fin}. Понятно, что при

этом мы пройдём по всем дугам два раза: первый — при прохождении в прямом направлении дуги при построении недостающей части пути, а второй — в обратном направлении при выставлении меток финальных вершин. Общая сложность такого алгоритма, очевидно, равна $O(n)$ для всех вершин совокупно.

2. 2-Исходящий граф. Постановка задачи. Усложним иллюстрирующий пример следующим достаточно естественным образом. Пусть каждое из устройств в системе имеет один вход и два пронумерованных числами 1 и 2 выхода. Ко входу любого такого устройства можно присоединить любое количество любых выходов других устройств. У любого устройства A_i системы, выход, помеченный 1, подключен ко входу некоторого устройства A_{j_1} этой системы, а выход, помеченный 2, подключен ко входу некоторого устройства A_{j_2} этой системы. Возможны ситуации, когда $A_{j_1} = A_{j_2}$, $Aj_1 = A_i$, $A_{j_2} = A_i$ или $A_{j_1} = A_{j_2} = A_i$. Любое устройство работает следующим образом: при получении сигнала на свой вход первый раз, оно передает его на свой выход, помеченный 1, и меняет состояние с 0 на 1. При получении на вход сигнала вторично, устройство передает его на свой выход, помеченный 2, и меняет состояние с 1 на 2, после чего перестает срабатывать. По исходному устройству A_{start} требуется эффективно определить финальный для цепочки устройств A_{fin}, сигнал на вход которого ещё пришёл, но выходного сигнала уже не генерирует. Задача ставится в массовом варианте, т.е. в качестве стартового перебираются все устройства системы и ответ выводится в формате $A_1 \to A_{\text{fin}_1}$, $A_2 \to A_{\text{fin}_2} \dots A_n \to A_{\text{fin}_n}$.

Заметим, что можно очевидным образом обобщить данный пример на систему устройств с любым другим количеством пронумерованных выходов $1, 2, \dots, k$. Данная постановка с двумя выходами в самом простом виде отображает возникающие идеи и задачи, но их легко перенести и на случай с k выходами.

Переходя к терминологии теории графов, будем рассматривать ориентированный граф $G(V, E)$, у каждой вершины которого степень полуисхода равна 2. Эти две исходящих дуги помечены числами 1 и 2. В дальнейшем будем называть граф такого вида 2-исходящим. Пусть число вершин графа G равно n; тогда число его дуг равно $2n$. В графе G допускаются петли и кратные дуги.

Выберем в качестве стартовой вершины некоторую произвольную вершину v_{start} графа G. Определим путь из этой вершины следующим образом: пусть мы в данный момент находимся в некоторой текущей вершине v_i. Если мы ещё не проходили при построении текущего пути дугу, выходящую из v_i и помеченную 1, то мы переходим из v_i по этой дуге в следующую вершину пути v_{j1}. Если при построении текущего пути по дуге из v_i, помеченной 1, мы уже проходили, то переходим по дуге, помеченной 2, в следующую вершину пути v_{j2}. Если же мы проходили в текущем пути и по дуге 2, то считаем, что текущий путь закончен в данной вершине v_i. Эту вершину будем называть финальной для пути, начинающегося из v_{start} или просто финальной для v_{start}.

Если v_{start} совпадает с v_{fin}, путь называется циклом, а вершина — финальной для самой себя. В общем случае циклом на 2-исходящем графе будем называть часть пути, начинающуюся и заканчивающуюся в вершине v_{start}, независимо от финальной вершины этого пути.

Задача заключается в построении эффективного алгоритма, находящего для каждой вершины v_i графа G финальную вершину для пути, начинающегося в v_i. Очевидно, что самым простым методом решения этой задачи является последовательный выбор каждой вершины графа как стартовой и дальнейшее моделирование прохождения пути с нахождением финальной вершины. После прохождения очередного пути и нахождения финальной вершины, все пометки сбрасываются и для следующей вершины мы ищем финальную независимо от предыдущих попыток. Так как в процессе прохождения одного пути каждая дуга будет пройдена не более одного раза, то сложность данного алгоритма для всех вершин можно оценить как $|V| \cdot |E| = 2n^2 = O(n^2)$. Это дает квадратичное решение задачи, что не может считаться эффективным решением.

3. Максимальная циклическая компонента 2-исходящего графа.

Определение 1. Циклической (эйлеровой) компонентой ориентированного 2-исходящего графа G будем называть любой его подграф, для которого выполняется следующее свойство: для

любой вершины циклической компоненты верно, что она входит в него либо с одной входящей дугой исходного графа G и одной исходящей дугой, помеченной 1, либо эта вершина входит в него с двумя входящими дугами исходного графа G и двумя исходящими дугами, помеченными 1 и 2. Данное свойство циклической компоненты будем называть балансом входящих и исходящих дуг в каждой её вершине. Если дуга входит в циклическую компоненту, то в неё входят обе вершины, которые она соединяет.

Предложение 1. *Если мы стартуем из вершины v_{start}, находящейся в некоторой циклической компоненте 2-исходящего графа G, то при прохождении пути из неё мы пройдем по некоторым дугам этой компоненты и обязательно вернемся в вершину v_{start}. Если v_{start} входит в циклическую компоненту с одной входящей и одной исходящей дугой, то дальнейший путь из неё будет проходить по дуге, помеченной 2; если же v_{start} входит в циклическую компоненту с двумя входящими и исходящими дугами, то в процессе построения пути мы пройдем по обеим исходящим дугам и дважды вернемся в v_{start}, т.е. в этом случае вершина v_{start} будет финальной для самой себя.*

Доказательство. При прохождении пути из указанной вершины v_{start} мы для каждой промежуточной вершины, кроме стартовой, будем входить в неё по некоторой дуге из циклической компоненты и обязательно выходить по соответственно помеченной дуге в следующую вершину этой же компоненты. Таким образом, выйти из этой циклической компоненты мы сможем только вернувшись в стартовую вершину v_{start}. □

Далее разобьём все вершины графа G на следующие непересекающиеся подмножества:

(i) вершины без возврата — это такие вершины, в которые нельзя вернуться ни одного раза при старте из них;

(ii) вершины с одним возвратом — при старте из этих вершин по дуге, помеченной 1, путь возвращается в них один раз и затем, после повторного выхода из них по дуге, помеченной 2, путь повторно в них уже не возвращается;

(iii) вершины с двумя возвратами, путь из которых возвращается в них дважды, т.е. они являются финальными для самих себя.

Для каждой из вершин второго и третьего типов зафиксируем все дуги и вершины, находящиеся на циклах, пройденных при старте из этих вершин. Для вершин с одним возвратом цикл содержит одну входящую дугу и одну исходящую дугу, помеченную 1. Для вершин с двумя возвратами, цикл содержит две входящих и две исходящих дуги, помеченных 1 и 2.

Определение 2. Максимальную циклическую компоненту 2-исходящего графа G определим как объединение всех циклов для всех вершин второго и третьего типов.

Предложение 2. *Максимальная циклическая компонента 2-исходящего графа G удовлетворяет определению 1, т.е. в каждой её вершине соблюдается баланс входящих и исходящих дуг.*

Доказательство. Рассмотрим процесс построения максимальной циклической компоненты. Будем по очереди независимо стартовать из каждой вершины второго или третьего типа и добавлять новые посещенные дуги в некоторую общую компоненту. Первый добавленный в неё цикл, стартующий из некоторой вершины второго или третьего типа, очевидно соблюдает баланс входящих и исходящих дуг в каждой своей вершине. Далее берем любую другую вершину второго или третьего типа и независимо стартуем из неё. Каждый раз, когда текущий путь попадает в уже принадлежащую текущей общей циклической компоненте вершину v_i, он некоторое время проходит по её дугам, и, за счет предложения 1, если и покидает эту компоненту, то исключительно в вершине v_i. Таким образом, во всех пройденных в текущем добавленном цикле вершинах соблюдается баланс входящих и исходящих дуг. □

Очевидно, что по построению максимальная циклическая компонента является единственной для графа G и не зависит от порядка добавления в неё новых циклов. Она состоит из всех вершин второго типа с исходящими из них дугами, помеченными 1, и вершин третьего типа с двумя исходящими из них дугами, помеченными 1 и 2. Сложность её построения описанным выше способом

квадратичная. Эта компонента нужна для доказательства корректности всех последующих рассуждений. Далее при построении эффективного алгоритма, данная компонента будет пострена с линейной сложностью.

4. Классификация дуг 2-исходящего графа, прямой путь из вершины.

Определение 3. Для вершин первых двух типов определим прямую дугу как дугу, по которой путь из вершины выходит и уже не возвращается в неё. Для вершин первого типа (без возврата) прямой дугой является исходящая дуга, помеченная 1, для вершин второго типа (с одним возвратом) прямой дугой является исходящая дуга, помеченная 2. Дуги, по которым не проходит ни один из путей, стартующих в любой вершине будем называть несущественными.

После построения максимальной циклической компоненты в 2-исходящем графе G, все его дуги можно разбить на три непересекающихся подмножества:

(a) дуги, принадлежащие циклической компоненте;
(b) дуги, по которым путь, стартующий в некоторой вершине, выходит и уже не возвращается в неё (прямые дуги);
(c) остальные дуги, по которым не проходит ни один из путей, стартующих в любой вершине (несущественные дуги).

Теперь можно более детально описать путь из произвольной вершины v_{start}. Пройдя по некоторому набору (возможно пустому) дуг из максимальной циклической компоненты, путь либо закончится в вершине v_{start}, если она является финальной для самой себя, либо вернётся в неё согласно предложению 1 и выйдет из неё без возврата по дуге не из циклической компоненты (прямой дуге) в следующую вершину v_{next_1}. В вершине v_{next_1} повторится та же ситуация, с возможным переходом по прямой дуге в вершину v_{next_2}, и т. д. Рассмотрим цепочку вершин $v_{\text{start}}, v_{\text{next}_1}, v_{\text{next}_2} \dots v_{\text{fin}}$, состоящую из пройденных на основном пути дуг не из циклической компоненты (далее покажем, что это именно прямые дуги). Эту цепочку назовём прямым путём из вершины v_{start}. Прямой путь не может зациклиться (неважно, в вершине v_{start} или любой другой вершине v_{next_i}), так как в таком случае все дуги этого нового цикла должны попадать в циклическую компоненту, в силу того, что они сохраняют баланс в своих вершинах. Помимо этого, зацикливание прямого пути приводит к возможности возврата в вершины этого цикла после выхода из них по прямым дугам. Единственным вариантом завершения этой цепочки является её попадание в вершину с двумя возвратами v_{fin}, являющуюся финальной для самой себя.

Таким образом, доказано следующее утверждение.

Предложение 3. *Множество прямых дуг 2-исходящего графа образует ориентированный лес, корнями комопнент которого являются вершины с двумя возвратами, финальные сами для себя. Эти вершины будут финальными и для всех вершин ориентированного дерева, корнем которого они являются.*

Очевидно, что среди дуг прямого пути будут только прямые дуги. Если вершина прямого пути является невозвратной вершиной первого типа, то путь из неё сразу уходит по прямой дуге, помеченной 1, и более никогда в неё не возвращается, согласно предложению 3, т.е. дуга из этой вершины, помеченная 2, никогда не проходится (она является несущественной). Если вершина прямого пути является вершиной с одним возвратом (вершиной второго типа), то исходящая из неё дуга, помеченная 1, принадлежит максимальной циклической компоненте, а исходящая дуга, помеченная 2, является прямой дугой, по которой и проходит дальнейший прямой путь.

Полученное в предложении 3 описание прямых дуг 2-исходящего графа G позволяет построить и обосновать корректность эффективного алгоритма поиска финальных вершин сразу для всех вершин этого графа.

5. Эффективный алгоритм поиска финальных вершин на 2-исходящем графе. Выберем произвольную вершину v_{start}, для которой ещё не найдена финальная вершина. Пойдем из неё по определённому выше пути, отмечая все посещённые на данный момент вершины как актуальные для текущего пути. Если в какой-то момент мы придем в актуальную вершину v_i (т.е.

обнаружим промежуточный цикл), то все вершины этого цикла, кроме самой v_i (той, из которой мы в этот цикл вошли), удаляем из списка актуальных, все дуги, по которым проходит этот цикл, объявляем принадлежащими циклической компоненте, после чего выходим из v_i по дуге, помеченной 2. Вершина v_i считается принадлежащей и циклической компоненте (по исходящей дуге 1) и множеству актуальных вершин по исходящей дуге 2.

Следует отметить, что здесь промежуточно возникающий цикл не удовлетворяет основному определению цикла, требующему, чтобы он зацикливался в стартовой вершине (что необходимо для гарантии поддержания баланса в вершинах циклической компоненты). Однако жёсткое требование переноса любого первого встретившегося такого промежуточного цикла в циклическую компоненту будет гарантировать, что у любой её вершины будет добавляться сначала исходящая дуга, помеченная 1, а уже затем дуга, помеченная 2, но никак не наоборот, так как мы каждый раз выходим из любой вершины по наименьшей по номеру не пройденной ещё дуге, что гарантирует поддержание баланса. Из этого очевидно, что все вершины, оказавшиеся на таком промежуточном цикле и дуги, по которым проходит этот цикл, принадлежат максимальной циклической компоненте. Эта посещенная часть максимальной циклической компоненты может дополняться новыми вершинами и дугами после прохождения и зацикливания в других актуальных вершинах или в этой же вершине v_i при вторичном зацикливании.

Уже при первом запуске, при вторичном прохождении уже посещенной вершины v_i, содержащейся в текущей циклической компоненте, используем ключевое соображение, которое подробнее будет раскрыто чуть позже. Его суть в том, что мы не проходим по уже пройденным дугам ключевой компоненты, а уходим из этой вершины по наименьшей ещё непройденной дуге.

Очевидно, что в любой момент цепочка актуальных на данный момент вершин образует простую цепь. В какой-то момент из текущей актуальной вершины v_{fin} обе исходящих дуги будут уже пройдены и путь в ней закончится; в этот момент все дуги текущего пути, не попавшие в циклическую компонету, будут прямыми дугами, образующими прямой путь из v_{start} в v_{fin}. Вершина v_{fin} является финальной для v_{start}. Помимо этого, она является финальной для всех актуальных на данный момент вершин (в том числе и для себя) согласно предложению 3. Для завершения данной попытки, вернёмся по текущему прямому пути от финальной вершины до стартовой и для всех вершин прямого пути снимем метку актуальности и поставим финальной для них v_{fin}. Метки принадлежности циклической компоненте на дугах, принадлежащих текущей циклической компоненте, не снимаем.

Далее до тех пор, пока для каких-то вершин графа G не найдена финальная вершина, выбираем произвольную такую вершину, объявляем её актуальной вершиной v_{start} и стартуем из неё в поисках финальной согласно предыдущему описанию. Ключевым моментом является то, что мы не проходим по уже пройденным ранее в других попытках (или в текущей попытке) дугам текущей циклической компоненты. Это значит, что придя в текущую актуальную вершину, мы выходим из неё по ещё не пройденной из неё ни в какой более ранней попытке дуге с самой маленькой пометкой. Корректность этого можно обосновать при помощи предложения 1. Действительно, если текущая актуальная вершина v_i уже проходилась в более ранних попытках и теперь находится в циклической компоненте, то это значит, что реальный путь будет некоторое время идти по дугам этой компоненты и выйдет из неё только в вершине v_i. Повторно проходить эту циклическую часть пути нет необходимости. При этом из вершины, не являющейся для себя финальной, мы будем в итоге уходить (уже не возвращаясь) по её прямой дуге. Если же вершина является финальной для самой себя, то в этом случае мы снова вернёмся в неё, завершим путь и поместим новый полученный цикл в циклическую компоненту.

Помимо этого, мы не проходим вторично и по прямым дугам при попадании в вершину, у которой уже известна финальная для неё, а сразу выставляем метку финальной вершины для всех актуальных на данный момент вершин прямого пути, согласно предложению 3.

Далее опишем возникающие при работе алгоритма случаи более подробно.

1. Если мы пришли в ещё не актуальную вершину v_i и из v_i мы ещё не проходили по дуге 1, значит, эта вершина ещё не была ни разу посещена ни в одной попытке, объявляем её актуальной и выходим из неё по дуге 1.

2. Если мы пришли в ещё не актуальную вершину v_i, для которой ещё не известна её финальная вершина и из v_i мы уже проходили по дуге 1, но не проходили из неё по дуге 2, то просто объявляем её актуальной и выходим из неё по дуге 2. Корректность этого следует из предложения 1.
3. Если мы пришли в ещё не актуальную вершину v_i, для которой уже известна финальная для неё v_{fin}, то эта v_{fin} будет финальной и для всех актуальных на данный момент вершин. Пройдем по ним, снимем метки актуальности и поставим для них финальной v_{fin}, закончим текущую попытку.
4. Если мы пришли в актуальную вершину v_i, то мы обнаружили очередной промежуточный цикл, все вершины этого цикла (кроме v_i) удаляем из списка актуальных и добавляем дуги и вершины этого цикла к циклической компоненте. Так как вершина v_i актуальная, значит мы уже выходили из неё. Если пройдена только выходящая из неё дуга 1, а в данный момент мы пришли в v_i мы по некоторой дуге e_{k2}, то вершина v_i попадает в циклическую компоненту с входящей дугой e_{k2} и выходящей из неё дугой 1, а так же v_i входит в текущий прямой путь с входящей дугой e_{k1}, по которой мы первый раз пришли в v_i и выходящей из неё дугой 2. Если же из v_i мы проходили по обеим выходящим дугам, то v_i является финальной для себя и множества всех актуальных на данный момент вершин. Возвращаемся по прямому пути, выставляем актуальным вершинам финальной вершину v_i, снимаем метки актуальности, заканчиваем текущую попытку.

При выполнении данного алгоритма мы пройдем по каждой дуге не более двух раз. Один раз в прямом направлении при построении пути. Второй раз в обратном направлении: для дуг из циклической компоненты — при удалении меток актуальности у вершин и переносе дуг из текущего прямого пути в циклическую компоненту во время обработки очередного промежуточного цикла; для прямых дуг — при установке метки финиша для текущих актуальных вершин во время обработки прямого пути. Общая сложность этого алгоритма равна $O(n)$ суммарно для всех вершин.

6. Пример. Проиллюстрируем все сказанное выше следующим примером. На рис. 1 представлен 2-исходящий граф G, состоящий из 16 вершин. Для каждой вершины укажем путь, получающийся при старте из неё и классифицируем все вершины по типам (без возврата, с одним возвратом, с двумя возвратами).

Вершина 1 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$1 \to 6 \to 1 \to 2 \to 4 \to 3 \to 7 \to 8 \to 4 \to 7 \to 2 \to 8 \to 12 \to 11 \to 16 \to 15 \to 11 \to 7.$$

Вершина 2 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$2 \to 4 \to 3 \to 7 \to 8 \to 4 \to 7 \to 2 \to 8 \to 12 \to 11 \to 16 \to 15 \to 11 \to 7.$$

Вершина 3 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$3 \to 7 \to 8 \to 4 \to 3 \to 2 \to 4 \to 7 \to 2 \to 8 \to 12 \to 11 \to 16 \to 15 \to 11 \to 7.$$

Вершина 4 — это вершина третьего типа с двумя возвратами, финальная сама для себя:

$$4 \to 3 \to 7 \to 8 \to 4 \to 7 \to 2 \to 4.$$

Вершина 5 — это вершина первого типа без возврата и финальной вершиной 6:

$$5 \to 6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 13 \to 9 \to 6.$$

Вершина 6 — это вершина третьего типа с двумя возвратами, финальная сама для себя:

$$6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 13 \to 9 \to 6.$$

Вершина 7 — это вершина третьего типа с двумя возвратами, финальная сама для себя:

$$7 \to 8 \to 4 \to 3 \to 7 \to 2 \to 4 \to 7.$$

Вершина 8 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$8 \to 4 \to 3 \to 7 \to 8 \to 12 \to 11 \to 16 \to 15 \to 11 \to 7 \to 2 \to 4 \to 7.$$

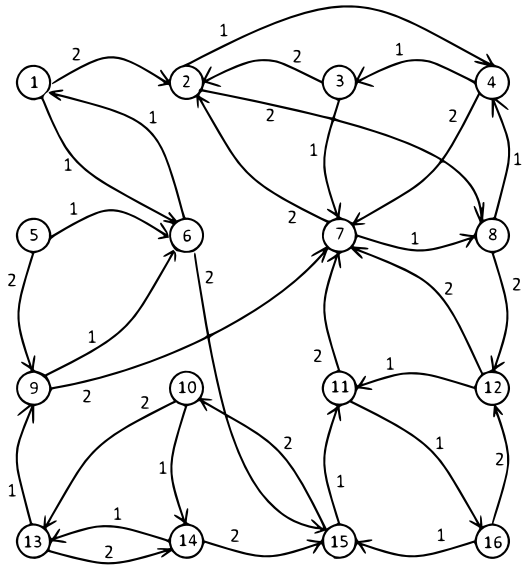

Рис. 1

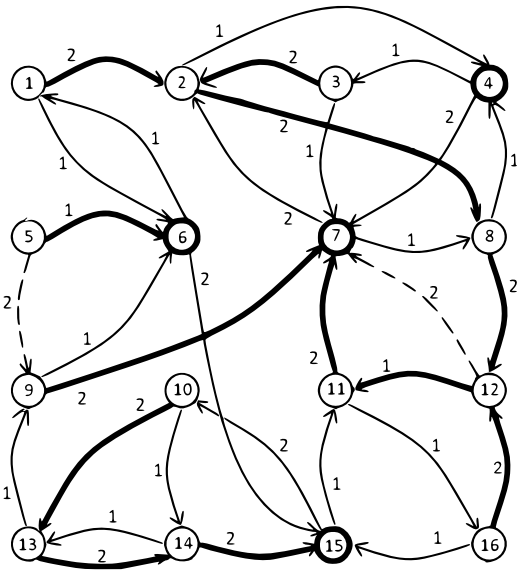

Рис. 2

Вершина 9 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$9 \to 6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 13 \to 9 \to 7 \to 8 \to 4 \to 3 \to 7 \to 2 \to 4 \to 7.$$

Вершина 10 — это вершина второго типа с одним возвратом и финальной вершиной 15:

$$10 \to 14 \to 13 \to 9 \to 6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 13 \to 14 \to 15.$$

Вершина 11 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$11 \to 16 \to 15 \to 11 \to 7 \to 8 \to 4 \to 3 \to 7 \to 2 \to 4 \to 7.$$

Вершина 12 — это вершина первого типа без возврата и финальной вершиной 7:

$$12 \to 11 \to 16 \to 15 \to 11 \to 7 \to 8 \to 4 \to 3 \to 7 \to 2 \to 4 \to 7.$$

Вершина 13 — это вершина второго типа с одним возвратом и финальной вершиной 15:

$$13 \to 9 \to 6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 13 \to 14 \to 15.$$

Вершина 14 — это вершина второго типа с одним возвратом и финальной вершиной 15:

$$14 \to 13 \to 9 \to 6 \to 1 \to 6 \to 15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 15.$$

Вершина 15 — это вершина третьего типа с двумя возвратами, финальная сама для себя:

$$15 \to 11 \to 16 \to 15 \to 10 \to 14 \to 13 \to 9 \to 6 \to 1 \to 6 \to 15.$$

Вершина 16 — это вершина второго типа с одним возвратом и финальной вершиной 7:

$$16 \to 15 \to 11 \to 16 \to 12 \to 11 \to 7 \to 8 \to 4 \to 3 \to 7 \to 2 \to 4 \to 7.$$

Данные пути позволяют построить максимальную циклическую компоненту и разбить множество дуг графа G на три класса (дуги, принадлежащие циклической компоненте, прямые дуги и несущественные дуги). На рис. 2 все эти дуги отмечены следующим образом: прямые дуги — жирным, дуги из циклической компоненты — обычным, несущественные дуги — пунктиром. Видно, что прямые дуги образуют деревья с корнями — вершинами третьего типа, финальными для самих себя. Все вершины внутри одного такого дерева имеют одну и ту же финальную вершину, являющуюся корнем этого дерева. Сами эти корни так же выделены жирным.

В заключение опишем порядок работы эффективного алгоритма для этого примера. Порядок выбора стартовых вершин не существенен, будем их перебирать по возрастанию номеров.

Выбираем стартовой вершину 1, далее путь выглядит следующим образом:

$$1 \to 6 \to 1.$$

Встретили промежуточный цикл в вершине 1, поместили дуги $1 \to 6 \to 1$ в циклическую компоненту, прямой путь состоит из вершины 1:

$$1 \to 2 \to 4 \to 3 \to 7 \to 8 \to 4.$$

Встретили промежуточный цикл в вершине 4, поместили дуги

$$4 \to 3 \to 7 \to 8 \to 4$$

в циклическую компоненту; прямой путь состоит из вершин и дуг $1 \to 2 \to 4$.

Из вершины 4 по дуге, помеченной 2, переходим в вершину 7. Она уже содержится в циклической компоненте, поэтому уходим из неё по дуге 2 в вершину 2, и снова получаем промежуточный цикл в этой вершине. Помещаем вершины и дуги $2 \to 4 \to 7 \to 2$ в циклическую компоненту; прямой путь имеет вид $1 \to 2$.

Из вершины 2 уходим по дуге, помеченной 2, в вершину 8; далее текущий путь имеет вид

$$1 \to 2 \to 8 \to 12 \to 11 \to 7.$$

Из вершины 7 обе исходящие дуги уже были пройдены, т.е. мы пришли в финальную вершину. Прямой путь на данный момент имеет вид

$$1 \to 2 \to 8 \to 12 \to 11 \to 7.$$

Возвращаемся по нему и для всех его вершин выставляем финальной вершину 7. Первая попытка закончена.

Вершина 2 уже получила свою финальную, переходим к вершине 3. Она находится на циклической компоненте, поэтому выходим из неё по дуге, помеченной 2. Попадаем в вершину 2, для которой уже известна финальная вершина 7, поэтому возвращаемся в вершину 3 и выставляем и ей вершину 7 финальной. Вторая попытка закончилась.

Вершина 4 находится на циклической компоненте, и из неё уже обе исходящие дуги пройдены. Это означает, что она является финальной для самой себя. Третья попытка закончилась.

Выбираем стартовой вершину 5. Из неё попадаем в вершину 6, которая уже находится в циклической компоненте, выходим из вершины 6 по дуге, помеченной 2. В итоге путь следующий:

$$5 \to 6 \to 15 \to 11 \to 16 \to 15.$$

Получили промежуточный цикл в вершине 15; помещаем дуги $15 \to 11 \to 16 \to 15$ в циклическую компоненту; прямой путь имеет вид $5 \to 6 \to 15$.

Далее движемся из вершины 15 по дуге, помеченной 2:

$$5 \to 6 \to 15 \to 10 \to 14 \to 13 \to 9 \to 6.$$

Получили промежуточный цикл в вершине 6; помещаем дуги $6 \to 15 \to 10 \to 14 \to 13 \to 9 \to 6$ в циклическую компоненту; прямой путь имеет вид $5 \to 6$. Из вершины 6 все дуги уже пройдены, поэтому она является финальной для себя и вершины 5. Четвёртая попытка закончена.

Вершины 6, 7 и 8 уже имеют свои финальные, их пропускаем.

Вершина 9 уже находится на циклической компоненте, выходим из неё по дуге, помеченной 2, в вершину 7. Вершина 7 имеет финальной саму себя, поэтому и у вершины 9 финальной будет вершина 7. Пятая попытка закончена.

Выбираем стартовой вершину 10. Она уже находится на циклической компоненте, выходим из неё по дуге, помеченной 2, в вершину 13. Та снова находится в циклической компоненте, выходим из 13 по дуге, помеченной 2, в вершину 14. Эта вершина также в циклической компоненте, выходим из неё по дуге, помеченной 2, в вершину 15. Из вершины 15 все исходящие дуги уже пройдены, она финальная для себя и текущих актуальных вершин 14, 13 и 10. Шестая попытка закончена.

Для вершин 11, 12, 13, 14 и 15 уже известны их финальные.

Стартуем из вершины 16. Она находится в циклической компоненте, поэтому выходим из неё по дуге, помеченной 2, в вершину 12. У этой вершины уже есть финальная, и это вершина 7. Выставляем вершине 16 в качестве финальной вершину 7, седьмая попытка закончена.

Для каждой вершины получена финальная для неё, работа алгоритма закончена со следующим результатом: вершина 7 является финальной для вершин 1, 2, 3, 7 8, 11, 12, 16; вершина 15 является финальной для вершин 10, 13, 14, 15; вершина 6 является финальной для вершин 5 и 6; вершина 4 является финальной для самой себя.

СПИСОК ЛИТЕРАТУРЫ

1. *Быков И. С.* Функционирование дискретной динамической системы циркулянтного типа с пороговыми функциями в вершинах// Прикл. дискр. мат. — 2014. — *26*, № 4. — С. 84–95.
2. *Евдокимов А. А., Пережогин А. Л.* Дискретные динамические системы циркулянтного типа с линейными функциями в вершинах сети// Дискр. анал. исслед. опер. — 2011. — *18*, № 3. — С. 39–48.
3. *Парфиненко А. С., Пережогин А. Л.* Функциональный граф линейной дискретной динамической системы с двумя доминирующими вершинами// Дискр. анал. исслед. опер. — 2018. — *25*, № 4. — С. 81–96.
4. *Harary F.* The number of functional digraphs// Math. Ann. — 1959. — *139*. — P. 203–210.

ДЕКЛАРАЦИЯ АВТОРА

Конфликт интересов. Автор заявляет об отсутствии конфликта интересов.

Финансирование. Автор заявляет об отсутствии финансовой поддержки от каких-либо организаций или частных лиц.

Финансовые интересы. Автор заявляет об отсутствии подлежащих раскрытию финансовых или нефинансовых интересов, связанных с публикуемым материалом.

Зубков Олег Владимирович (Zubkov Oleg Vladimirovich)
Иркутский государственный университет, Иркутск
(Irkutsk State University, Irkutsk, Russia)
E-mail: oleg.zubkov@mail.ru