ПРИКЛАДНАЯ ТЕОРИЯ ГРАФОВ

УДК 519.17 DOI 10.17223/20710410/58/7

ПРЯМОЙ МЕТОД ВЫЧИСЛЕНИЯ ЦИКЛОВ ЯЧЕЕК КАРТЫ БЛОКА ПРОСТОГО ПЛАНАРНОГО ГРАФА

Б. Н. Иванов

Дальневосточный федеральный университет, г. Владивосток, Россия

E-mail: ivanov.bn@dvfu.ru

Предлагаемый алгоритм вычисления циклов ячеек карты блока графа простого планарного является расширением классического алгоритма поиска в глубину циклов DFS-базиса. Ключевой идеей модификации алгоритма является стратегия правого обхода при прохождении графа в глубину. Начальной вершиной при правом обходе назначается вершина с минимальной координатой по оси OY. Выход из начальной вершины выполняется по ребру с минимальным полярным углом. Продолжение обхода из каждой следующей вершины осуществляется по ребру с минимальным полярным углом относительно ребра, по которому пришли в текущую вершину. Вводится двухуровневая структура вложенности циклов — основной и нулевой уровни вложенности. Все циклы базиса относятся к основному уровню. Каждый из циклов может иметь и нулевой уровень вложенности в другом основном для него цикле, если он вложен в него и не вложен ни в какой другой цикл из основного цикла. При правом обходе циклы нулевой вложенности являются смежными основному циклу и не имеют между собой общих вершин вне основного цикла. Указанные два свойства позволили в каждом цикле базиса последовательно выделить и исключить из него все циклы нулевой вложенности, применяя операцию симметрической разности. Показано, что оставшаяся часть базисного цикла является циклом ячейки карты блока. Сложность каждого шага алгоритма не превышает квадратичной относительно числа вершин планарного графа.

Ключевые слова: *циклы планарного графа, циклы блока графа, планарный граф.*

A DIRECT METHOD FOR CALCULATING CELL CYCLES OF A BLOCK MAP OF A SIMPLE PLANAR GRAPH

B. N. Ivanov

Far Eastern Federal University, Vladivostok, Russia

The proposed algorithm for calculating the cycles of the cells the simple planar graph block map is an extension of the classical depth-first search algorithm for cycles of the DFS-basis. The key idea of the modification of this algorithm is the strategy of right-hand traversal when passing the graph in depth. The vertex with the minimum coordinate on the OY axis is assigned as the starting vertex in the right-hand traversal. The exit from the initial vertex is performed along the edge with the minimum polar

angle. The continuation of the traversal from each next vertex is carried out along an edge with a minimum polar angle relative to the edge along which arrived at the current vertex. A two-level structure of nested cycles is introduced. This is the main level and the zero level of nesting. All cycles of the basis belong to the main level. Each of the cycles can additionally have a zero level of nesting in another main cycle for it, if it is nested in the main cycle and not nested in any other cycle from the main cycle. With the right-hand traversal, zero nesting cycles are adjacent to the main cycle and do not have common vertices outside the main cycle. These two properties allowed in each basis cycle sequentially select and exclude from it all its zero nesting cycles, using the symmetric difference operation. It is shown that the rest of the basic cycle is the cycle of the block map cell. The complexity of each step of the proposed algorithm does not exceed the quadratic complexity with respect to the number of vertices of the simple planar graph.

Keywords: *planar graph cycles, graph block cycles, planar graph.*

Введение

Рассмотрим задачу раскраски территорий государств на карте поверхности Земли. В равной степени вместо государств можно взять любые другие области на карте. Исходными данными в этой задаче выступают география поверхности Земли (береговая черта) и границы государств. Данной модели поставим в соответствие конечный *простой планарный* граф $G = (X, U)$, где U — рёбра графа (границы государств); X — вершины (точки пересечения границ). К такому графу всегда можно свести рассматриваемую карту поверхности Земли, включая фиктивные вершины на границах или береговой черте.

Будем пользоваться терминами и обозначениями, принятыми в теории графов [1, с. 11]. *Подграфом* графа $G = (X, U)$ называется такой граф $G' = (X', U')$, что $X' \subseteq X$, $U' \subseteq U$. *Остовным* подграфом графа $G = (X, U)$ называется такой граф $G = (X, U')$, что $U' \subseteq U$. *Простой* граф — это неориентированный граф без петель, в котором любая пара вершин соединена не более чем одним ребром. В таком графе каждое ребро однозначно представляется неупорядоченной парой вершин $u = (x, y)$, где $u \in U$, $x, y \in X$. В этом случае говорят, что ребро u *инцидентно* вершинам x и y или вершины x и y инцидентны ребру u. *Маршрутом* в простом графе называется конечная последовательность вершин

$$S = (x_0, x_1, \ldots, x_{n+1}), \tag{1}$$

где каждая пара соседних вершин x_i и x_{i+1} определяет ребро $u_i = (x_i, x_{i+1})$ в графе. Тогда маршрут (1) можно записать и как последовательность рёбер $S = (u_0, u_1, \ldots, u_n)$. Маршрут называется *замкнутым*, если $x_0 = x_{n+1}$. Маршрут называется *цепью*, если все его рёбра различные. Замкнутая цепь называется *циклом*. Цепь называется *простой*, если никакая вершина в ней не повторяется. Простая замкнутая цепь называется *простым циклом*.

Укладка на плоскости рёбер конечного *простого планарного графа* без самопересечений определяет разбиение такой плоскости, называемое *планарным подразбиением* или *картой* графа [2, с. 31]. Карта фиксирует на плоскости геометрию рёбер графа, что позволяет на этой плоскости ввести *прямоугольную систему координат* и определить в ней координаты каждой вершины графа и положение рёбер. Координаты вершин определяют их взаимное расположение. В частности, можно найти вершину

графа, координата которой вдоль оси OY минимальная. В этом случае имеет смысл говорить о вершинах и рёбрах, *локализованных внутри цикла*, *заключенных внутри цикла* или *вложенных в цикл*. Такие вершины и рёбра графа геометрически располагаются внутри цикла. Сама карта складывается из простых многоугольников, которые будем называть *ячейками* карты. Каждый цикл ячейки карты графа представляет собой границы отдельного государства.

Замечание 1. Представление рёбер на плоскости будем выполнять кусочно-линейными кривыми. Рассматриваемый алгоритм вычисления циклов предполагает наличие в графе вершины с минимальной координатой вдоль оси OY среди всех вершин и *узлов рёбер* графа. Если узел с минимальной координатой по оси OY среди всех узлов рёбер меньше соответствующей координаты любой вершины графа, то такой узел необходимо перевести в разряд вершин графа. Поэтому далее полагаем, что в графе существует *единственная* вершина с минимальной координатой по оси OY среди всех вершин и узлов рёбер графа. Если таких вершин несколько, то уменьшим незначительно координату по оси OY одной из них и получим граф с единственной такой вершиной.

На рис. 1 показаны четыре ребра графа $(s,x),(x,a),(x,b),(x,c)$ с общей вершиной x и их узлы s_i,a_i,b_i,c_i кусочно-линейной интерполяции. Взаимное расположение кусочно-линейных рёбер $(x,a),(x,b),(x,c)$ относительно ребра (s,x) будем устанавливать по их первым отрезкам $(s_1,x),(x,a_1),(x,b_1),(x,c_1)$, примыкающим к вершине x (рис. 2). Для каждой пары отрезков $\{(s_1,x),(x,a_1)\}$, $\{(s_1,x),(x,b_1)\}$, $\{(s_1,x),(x,c_1)\}$ вычисляем величину угла, заключённого между ними, в направлении против часовой стрелки от отрезка (s_1,x) к другому отрезку соответствующей пары. Такие углы будем называть *полярными*. Сам отрезок (s_1,x) имеет нулевой полярный угол. Остальные такие углы могут принимать значения в диапазоне $(0,2\pi)$. Под *сортировкой рёбер* $(x,a),(x,b),(x,c)$ относительно ребра (s,x) будем понимать их расположение в порядке увеличения их полярных углов относительно ребра (s,x).

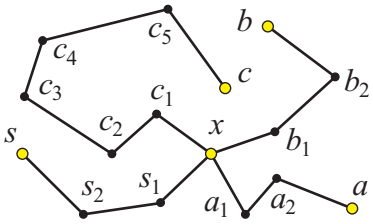

Рис. 1. Кусочно-линейные рёбра (s,x), (x,a), (x,b), (x,c) границ государств

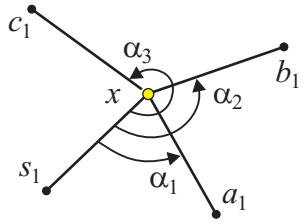

Рис. 2. Сортировка рёбер (x,a), (x,b), (x,c) относительно ребра (s,x) по полярным углам α_1, α_2, α_3

Рассмотрим другой случай вычисления полярных углов рёбер, инцидентных заданной вершине x, при условии, что координата вершины x вдоль оси OY меньше соответствующей координаты любой другой вершины карты графа. Перенесём начало системы координат в заданную вершину x. Тогда полярные углы всех рёбер, инцидентных вершине x, будем вычислять по рассмотренной ранее схеме, где роль ребра (x,s) выполняет ось OX. Углы могут принимать значения в диапазоне $(0,\pi)$.

Изучение циклической структуры графа является важной задачей общего анализа структуры графа. Первый практический алгоритм перечисления всех циклов графа опубликован в [3], его модифицированная версия — в [4]. Работы [3, 4] определили направление разработки алгоритмов генерации циклов графов. Показано, что в дан-

ной задаче метод поиска с возвращениями позволяет в среднем существенно понизить сложность полного перебора и перейти к практически значимым алгоритмам.

После этого был предложен ряд других алгоритмов [5, 6] порождения всех циклов графа, их обзор можно найти в работе [7]. Основу этих алгоритмов составляет *метод поиска в глубину* на графе [8]. Это адаптированный к структуре графа поиск с возвращениями.

Вычисление циклов графа с определёнными свойствами, как правило, выполняется на основе алгоритмов генерации всего множества циклов. Выделение циклов с заданными свойствами из общего множества может существенно увеличить сложность такого алгоритма.

Сегодня можно встретить и параллельные алгоритмы генерации циклов в графе [9, 10]. Авторы пытаются решать тестовые задачи большой размерности, используя параллельные вычисления. В этом случае для поиска циклов в работе [10] предлагается приближённый алгоритм как альтернатива полному перебору. Для информационных сетей актуальна задача сетевой надёжности передачи данных, и здесь первостепенное значение приобретает знание циклической структуры сети [11].

Вычисление циклов в простом планарном графе рассматривалось автором данной работы в [12], где в основу метода положены выделенные геометрические свойства циклов DFS-базиса. В настоящей работе предлагается прямой алгоритм вычисления циклов ячеек карты блока графа при обходе его в глубину.

1. Постановка задачи

Решается задача вычисления циклов ячеек карты блока конечного графа $G = (X, U)$ по данным множествам его вершин X и рёбер U. Вычисления проводятся в рамках классического алгоритма поиска в глубину циклов DFS-базиса. Новизна заключается в стратегии правого обхода при прохождении графа в глубину. Правый обход позволил для каждого цикла базиса последовательно выделить и исключить из него все циклы нулевой вложенности. Показано, что оставшаяся часть базисного цикла будет искомым циклом ячейки карты блока. Сложность предлагаемого алгоритма является квадратичной относительно числа вершин в графе.

2. Фундаментальное множество циклов карты графа

Полагаем, что $G = (X, U)$ — это конечный связный простой планарный граф.

Определение 1 (операции $\oplus$). Обозначим через $M = \{G_1, G_2, \ldots, G_{n_M}\}$ множество всех остовных подграфов $G_i = (X, U_i)$ графа $G = (X, U)$, где $U_i \subseteq U$. На множестве M определим бинарную операцию $\oplus$ так, что $G_i \oplus G_j = (X, U_i \oplus U_j)$ для всех $G_i, G_j \in M$, где символ $\oplus$ обозначает операцию симметрической разности: $U_i \oplus U_j = \{u : u \in U_i \cup U_j \wedge u \notin U_i \cap U_j\}$. Множество M с операцией $\oplus$ является абелевой группой. Единица группы — пустой подграф $O = (X, \varnothing)$. Обратным к G_k является тот же самый G_k, так как $G_k \oplus G_k = O$.

Определение 2 (смежных циклов). Простые циклы h_1 и h_2 будем называть *смежными*, если их пересечение $h_1 \cap h_2$ состоит из одного непустого непрерывного участка рёбер.

Из определения 1 имеем, что операция $\oplus$ применяется к остовным подграфам. Определим данную операцию $h_1 \oplus h_2$ для простых смежных циклов h_1 и h_2. Для этого рассмотрим определение цикла как остовного цикла.

Определение 3 (остовного цикла). Остовным циклом простого цикла h называется подграф $H = (X, [h])$, где $[h]$ — рёбра цикла h; X — вершины исходного графа

$G = (X, U)$. Подграф H тоже будем называть циклом, у него одна компонента связности, которая является простым циклом h, а вершины, не вошедшие в цикл h, являются изолированными. Каждый такой остовный цикл H будем представлять компонентой связности — простым циклом h.

Замечание 2. Далее под результатом операции $h_1 \oplus h_2$ простых смежных циклов h_1 и h_2 будем понимать простой цикл, представленный компонентой связности остовного цикла $H_1 \oplus H_2$, где $H_1 = (X, [h_1])$ и $H_2 = (X, [h_2])$.

Обозначим: $Z = \{Z_1, Z_2, \ldots, Z_{n_Z}\}$, где $Z_k = (X, [z_k])$ — остовные циклы; z_k — все простые циклы графа G. Пусть $L = \{\lambda_{i_1} Z_{k_1} \oplus \lambda_{i_2} Z_{k_2} \oplus \ldots \oplus \lambda_{i_Z} Z_{k_Z}\}$ — линейная оболочка циклов множества Z, где $\lambda_{i_r} \in \{0, 1\}$, $0 \cdot Z_{k_r} = O$ и $1 \cdot Z_{k_r} = Z_{k_r}$. Множество L является линейным векторным пространством циклов над полем $P = \langle \{0, 1\}, \oplus, \cdot \rangle$ [13, с. 61]. Уточним структуру, размерность и базис пространства L.

Пусть $T_0 = (X, U_0)$ — произвольное остовное дерево исходного графа $G = (X, U)$. Для дерева выполняется соотношение $|U_0| = |X| - 1$. Пусть $E = U \setminus U_0 = \{e_1, e_2, \ldots, e_{n_F}\}$ — рёбра, не вошедшие в T_0. Такие рёбра называются обратными. Любое обратное ребро $e_i \in E$ порождает в T_0 ровно один простой цикл C_i, где $e_i \in C_i$ и $e_i \notin C_j$, если $i \neq j$. Можно составить $n_F = |U \setminus U_0| = |U| - |X| + 1$ таких циклов C_i. Множество $F = \{C_1, C_2, \ldots, C_{n_F}\}$ называется *фундаментальным множеством циклов* и составляет базис пространства L [13, с. 63]. В простом планарном графе [13, с. 36] выполняется соотношение $|U| \leqslant 3|X| - 6$, что даёт основание пользоваться оценкой для числа циклов $n_F = |U| - |X| + 1 \leqslant 2|X| - 5$.

3. Блоковые свойства циклов DFS-базиса карты графа

Построение фундаментального множества циклов выполним *методом поиска в глубину со стеком*. Оригинальное описание метода представлено в [14]. Адаптированный вариант алгоритма для практического применения рассматривается в работе [15, с. 385]. При порождении фундаментального множества циклов методом поиска в глубину на простом связном графе строится дерево поиска, которое является остовным деревом графа и называется *DFS-деревом*. Каждое обратное ребро порождает один цикл относительно этого дерева. Всё множество таких циклов составляет *DFS-базис* циклов графа.

Предлагаемый алгоритм вычисления циклов ячеек карты графа ориентирован на свойства карты блока. Отметим несколько свойств циклов DFS-базиса карты блока, необходимых для обоснования алгоритма.

Теорема 1. Если пара циклов базиса C_1 и C_2 имеет хотя бы две общие вершины, то такие циклы являются смежными.

Доказательство. Пусть $e_1 \in C_1$ и $e_2 \in C_2$ — обратные рёбра циклов, $e_1 \notin C_2$ и $e_2 \notin C_1$. Пройдя по циклу C_1 в обе стороны от ребра e_1, найдём две вершины x_1 и x_2, принадлежащие также C_2 (рис. 3). Обозначим через $C_1(x_1, e_1, x_2)$, $C_2(x_1, e_2, x_2)$ участки циклов C_1 и C_2, содержащие соответственно рёбра e_1 и e_2. Их объединение $C_1(x_1, e_1, x_2) \cup C_2(x_1, e_2, x_2)$ есть простой цикл. Данная схема построения простого цикла из двух простых циклов при наличии у них не менее двух общих вершин рассматривается в [1, с. 110].

Обозначим через $R_1(x_1, x_2)$ и $R_2(x_1, x_2)$ участки циклов соответственно C_1 и C_2, не вошедших в простой цикл $C_1(x_1, e_1, x_2) \cup C_2(x_1, e_2, x_2)$. Если допустить, что $R_1(x_1, x_2) \neq$ $\neq R_2(x_1, x_2)$ (рис. 3), то вершины x_1 и x_2 будут связаны четырьмя маршрутами. Тогда при удалении всех обратных рёбер в графе получим остовное дерево T_0, в котором

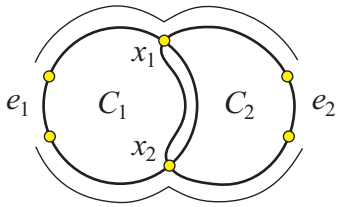

Рис. 3. Взаимное расположение циклов C_1 и C_2 базиса

вершины x_1 и x_2 останутся связанными двумя маршрутами $R_1(x_1, x_2) \neq R_2(x_1, x_2)$ (рис. 3), а значит, в T_0 будет цикл. Это противоречит тому, что T_0 есть дерево. Итак, имеем $R_1(x_1, x_2) = R_2(x_1, x_2)$, следовательно, C_1 и C_2 есть смежные циклы, а $C_1 \oplus C_2$ есть простой цикл, составленный из участков $C_1(x_1, e_1, x_2)$ и $C_2(x_1, e_2, x_2)$. ■

Определение 4 (нулевой вложенности). Пусть C_v и C_{v_j} — циклы базиса карты графа. Будем говорить, что цикл C_{v_j} имеет нулевую вложенность в цикле C_v, если цикл C_{v_j} вложен в C_v и не вложен ни в какой другой цикл из C_v.

Определение 5 (блока). В блоке графа все рёбра сильно циклически связаны [1, с. 109]. Это означает, что для любой пары рёбер u_1, u_2 блока существует такая последовательность простых циклов $H_1, H_2, \ldots, H_k$, что $u_1 \in H_1$, $u_2 \in H_k$ и любая пара соседних циклов H_i, H_{i+1} имеет по крайней мере одно общее ребро.

Замечание 3. Все простые циклы графа распределяются между блоками графа. Каждый простой цикл расположен целиком в своем блоке [1, с. 111]. Поэтому далее полагаем, что исходный граф $G = (X, U)$ есть невырожденный конечный блок карты графа (вырожденный блок состоит из одного ребра). В невырожденном блоке степень любой вершины не меньше двух.

Поиск в глубину на простом неориентированном графе осуществляется следующим образом. Когда посещаем вершину x, то далее идём по одному из рёбер (x, v), инцидентных вершине v. Если вершина v уже пройдена, то возвращаемся в x и выбираем другое ребро. Если вершина v не пройдена, то заходим в неё и применяем процесс прохождения рекурсивно уже с вершиной v.

Предлагаемая модификация алгоритма поиска в глубину [15, с. 385] касается выбора очередного ребра (x, v) при обходе карты блока. Выбор такого ребра выполняется в соответствии с двумя следующими правилами.

Правило 1 (начала обхода). Начальной вершиной поиска в глубину является вершина с минимальной координатой по оси OY (см. замечание 1). Инцидентные ей рёбра упорядочиваются по полярному углу против часовой стрелки. Выход из начальной вершины выполняется по ребру с минимальным значением полярного угла.

Правило 2 (правого обхода). Пусть в вершину x пришли по ребру (s, x). Упорядочим все рёбра, инцидентные вершине x, по полярному углу относительно ребра (s, x) против часовой стрелки (см. рис. 2). Ребру (s, x) назначается нулевой угол. Упорядочивание рёбер в каждой вершине выполняется один раз при первом её посещении. Для продолжения прохода по графу из вершины x будем выбирать ещё не пройденное ребро (x, v) с минимальным полярным углом.

Замечание 4. При поиске циклов DFS-базиса используется поиск в глубину со стеком, в котором хранятся пройденные вершины. Когда попадаем на обратное ребро (x, v), то найденный цикл будет состоять из этого ребра и рёбер, соединяющих вершины из верха стека до вершины v обратного ребра в стеке.

Определение 6 (правого обхода). Будем говорить, что цикл карты блока построен при «правом обходе», если он сформирован при обходе карты блока в соответствии с правилами 1 и 2.

Определение 7 (классификация рёбер). Обозначим через X_1 непустое подмножество множества вершин X простого графа $G(X)$, а через $X_2 = X \setminus X_1$ — его дополнение [1, с. 111]. Тогда

1) Ребро $u_1 = (x, y)$ и вершины $x, y \in X_1$ называются внутренними для X_1 и располагаются в подграфе $G(X_1)$.
2) Ребро $u_2 = (x, y)$, где $x \in X_1$, $y \in X_2$, называется соединяющим для X_1, а вершина x — соединяющей.
3) Ребро $u_3 = (x, y)$ и вершины $x, y \in X_2$ называются внешними для X_1.

Определение 8 (разделяющей вершины). В простом графе $G(X)$ вершина $a \in X$ называется разделяющей, если существует непустое подмножество $X_1 \subset X$, для которого $a \in X_1$ является единственной соединяющей вершиной. Тогда в любой вершине $b \neq a$ из X_1 все рёбра идут к вершинам из X_1.

Теорема 2 (о соединяющих вершинах цикла). Если множество вершин X_C, заключенных в простом цикле C карты графа, имеет соединяющие вершины, то эти вершины являются вершинами цикла C.

Доказательство. Исходный граф — это карта графа, а значит, пересечения рёбер цикла C с рёбрами графа являются вершинами цикла C. Следовательно, соединяющими вершинами множества X_C могут быть только вершины цикла C. ■

При «правом обходе» поиск в глубину начинается с вершины с минимальной координатой по оси OY (см. правило 1). Пройденные вершины сохраняются в стеке пройденных вершин и удаляются из стека при возврате по дереву поиска. Признаком начала возврата является обнаружение обратного ребра сформированного цикла. Пусть после обнаружения обратного ребра очередного сформированного цикла C_v вернулись по рёбрам этого цикла до первой его вершины a_0, которая имеет не пройденные инцидентные ей ребра. Данная вершина a_0 является началом формирования нового цикла C_0. Рассмотрим формирование данного цикла. Вершина a_0 имеет не пройденное инцидентное ей ребро, с которого можно начать строить простую цепь по карте блока. Степень любой вершины блока (невырожденного) не меньше 2. Поэтому при посещении какой-либо вершины первый раз (новой вершины v_i) из неё всегда можно продолжить простую цепь по ещё не пройденному ребру. Пусть такая последовательность из новых пройденных вершин $v_1 v_2 \dots v_n$ закончится обратным ребром в ранее пройденной вершине b_0. В стеке пройденных вершин будем иметь следующую последовательность вершин, составляющих простой цикл:

$$C_0 = (b_0 r_1 r_2 \dots r_m \, a_0 v_1 v_2 \dots v_n b_0). \tag{2}$$

Здесь a_0 — вершина начала формирования цикла; b_0 — вершина начала цикла; $r_1 r_2 \dots r_m$ — ранее пройденные вершины до вершины a_0; $v_1 v_2 \dots v_n$ — новые пройденные вершины после вершины a_0.

Определение 9 (нулевой вершины цикла). Вершину a_0 в цикле (2) будем называть *нулевой вершиной* цикла C_0 как вершину начала формирования данного цикла.

Теорема 3 (структура цикла при «правом обходе»). Вершины и рёбра цикла C_0, сформированного при «правом обходе» карты графа согласно формуле (2), удовлетворяют следующим свойствам:

1) все непройденные рёбра, инцидентные вершинам цикла C_0, кроме вершины b_0, локализованы внутри данного цикла;
2) вершина b_0 не совпадает ни с одной из новых вершин $v_1, v_2, \ldots, v_n$ цикла C_0;
3) если вершины a_0 и b_0 совпадают, то цикл C_0 является циклом, огибающим по периметру карту блока.

Доказательство.

1) При «правом обходе» формирования цикла C_0 в каждой его вершине, кроме вершины b_0, по построению (см. правила 1 и 2) непройденные инцидентные ей рёбра остаются внутри формируемого цикла.

2) Если допустить, что b_0 совпадает с одной из новых пройденных вершин $v_1, v_2, \ldots, v_n$, например $b_0 = v_k$, то вершина v_k будет единственной соединяющей для цикла $C_0 = (b_0 = v_k v_{k+1} \ldots v_n b_0 = v_k)$, так как при «правом обходе» в каждой новой вершине цикла, кроме b_0, инцидентные ей рёбра остаются внутри сформированного цикла. Следовательно, вершина $b_0 = v_k$ будет разделяющей для множества вершин, заключенных в цикле C_0. Это является противоречием, в блоке нет разделяющих вершин.

3) Пусть a_0 и b_0 совпадают, тогда цикл C_0 по формуле (2) запишется в виде $C_0 = (b_0 v_1 v_2 \ldots v_n b_0)$. В данном цикле только одна вершина b_0 не является новой. Как и в случае 2, такая вершина b_0 будет единственной соединяющей вершиной в цикле C_0. Если такой цикл C_0 охватывает не все вершины карты блока, то вершина b_0 является разделяющей. Это противоречие, в блоке нет разделяющих вершин. Следовательно, цикл C_0 является циклом, огибающим по периметру карту блока. ■

Теорема 4 (цикл по периметру блока). Первым циклом, сформированным при «правом обходе», будет цикл C_0, огибающий по периметру карту блока.

Доказательство. По правилу 1 начальной вершиной a_0 обхода карты блока является вершина с минимальной координатой по оси OY. Такой обход закончится в ранее пройденной вершине. Запишем данный маршрут обхода в виде $C_0 = (a_0, v_1, v_2, \ldots, v_n, b_0)$, где b_0 — ранее пройденная вершина, которая совпадает с одной из вершин $a_0, v_1, v_2, \ldots, v_n$. Покажем, что $b_0 = a_0$. Допустим, что $b_0 \neq a_0$, тогда $b_0 = v_k$, а маршрут обхода примет вид $C_0 = (a_0, v_1, v_2, \ldots, v_{k-1}, b_0, v_{k+1}, \ldots, v_n, b_0)$, где $B = (b_0, v_{k+1}, \ldots, v_n, b_0)$ — простой цикл. Вершина начала обхода a_0 не может быть локализована в цикле B, так как её координата по оси OY минимальная среди соответствующих координат всех вершин и узлов рёбер графа (см. правило 1 и замечание 1). Тогда вершина b_0 становится разделяющей вершиной блока, но в блоке нет разделяющих вершин. Следовательно, $b_0 = a_0$, а маршрут $C_0 = (a_0, v_1, v_2, \ldots, v_n, b_0)$ является простым циклом. Из теоремы 3 имеем, что такой цикл является циклом, огибающим по периметру карту блока. ■

Рассмотрим формирование вложенных циклов при «правом обходе». Пусть при возврате по дереву поиска после обнаружения обратного ребра цикла C_v вернулись по рёбрам этого цикла до первой его вершины a_1 (рис. 4), которая имеет непройденные инцидентные ей ребра. Вершина a_1 является нулевой вершиной формирования нового цикла C_{v_1}, который по формуле (2) запишется в следующем виде:

$$C_{v_1} = (b_1 r_1 r_2 \ldots r_m\, a_1, v_1, v_2, \ldots, v_n, b_1), \tag{3}$$

где a_1 — нулевая вершина цикла; b_1 — начальная вершина цикла; $r_1 r_2 \ldots r_m$ — ранее пройденные вершины до a_1; $v_1 v_2 \ldots v_n$ — новые пройденные вершины после a_1.

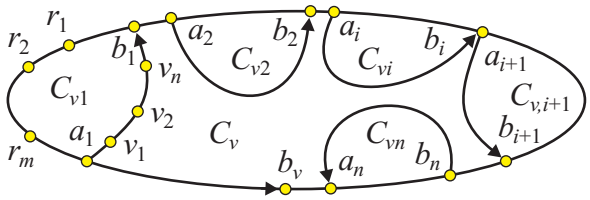

Рис. 4. Формирование цикла нулевой вложенности C_{v_1} в цикле C_v

Теорема 5 (первый вложенный цикл). Цикл C_{v_1}, представленный формулой (3), является циклом нулевой вложенности в цикле C_v и смежным ему.

Доказательство. Пусть обратное ребро цикла C_v заканчивается в вершине b_v. Из этой вершины начинается возврат (по часовой стрелке) по рёбрам данного цикла до первой вершины $a_1 \neq b_v$, у которой есть непройденные инцидентные ей рёбра (рис. 4). По теореме 3 при «правом обходе» все непройденные рёбра, инцидентные вершинам цикла C_v, кроме рёбер, инцидентных вершине b_v, остаются внутри этого цикла. Следовательно, все новые вершины v_i (см. формулу (3)) цикла C_{v_1} останутся внутри цикла C_v. Так как граф конечный, перечень новых вершин v_i цикла C_{v_1} завершится обратным ребром (v_n, b_1) в ранее пройденной вершине b_1, которая не совпадает ни с одной из новых вершин v_i (см. теорему 3). Так как вершина v_n локализована внутри цикла C_v, ранее пройденная вершина b_1 может быть только вершиной цикла C_v. Из вершин цикла C_v в стеке пройденных вершин остался участок вершин от a_1 до начальной вершины b_v . Поэтому b_1 может оказаться любой из них, исключая вершину a_1 (см. теорему 3).

Таким образом, циклы C_{v_1} и C_v имеют две общие вершины a_1 и b_1. По теореме 1 такие циклы являются смежными, а цикл C_{v_1} — вложенным в цикле C_v (рис. 4). Покажем, что цикл C_{v_1} является циклом нулевой вложенности. Если допустить, что существует вложенный цикл Q в цикле C_v, который охватывает цикл C_{v_1}, то нулевая вершина цикла Q должна размещаться на участке цикла C_v (против часовой стрелки) от вершины a_1 до конечной его вершины (рис. 4). На этом участке вершина a_1 была первой, которая имела непройденные инцидентные ей рёбра. Все такие рёбра остались вложенными в цикле C_{v_1} при «правом обходе» его формирования. Следовательно, цикл C_{v_1} не имеет охватывающих циклов и, значит, является циклом нулевой вложенности в цикле C_v. ■

Замечание 5. Из теоремы 5 следует, что цикл нулевой вложенности C_{v_1} в цикле C_v состоит из двух простых цепей (рис. 4). Первая цепь — это смежный участок указанных циклов от начальной вершины b_1 цикла C_{v_1} и до нулевой его вершины a_1. Вторая цепь — это участок цикла C_{v_1} из новых вершин.

При «правом обходе» после формирования любого вложенного цикла, например цикла C_{v_1}, мы переходим к формированию вложенных в него циклов, так как возвращаемся по рёбрам данного цикла, а все непройденные рёбра, инцидентные вершинам цикла C_{v_1}, остались локализованными внутри него (см. теорему 3).

Пусть мы вернулись по дереву поиска в цикл C_v, в котором был сформирован первый цикл C_{v_1} нулевой вложенности (рис. 4). На верху стека пройденных вершин имеем участок вершин цикла C_v от вершины b_1 до начальной вершины b_v, среди которых выполняется поиск нулевой вершины a_2 для построения следующего цикла C_{v_2}, который по теореме 5 будет циклом нулевой вложенности и смежным циклу C_v. По данной

схеме построим в цикле C_v все такие циклы нулевой вложенности

$$C_{v_1}, C_{v_2}, \ldots, C_{v_n}. \tag{4}$$

Каждый из таких циклов C_i определяется парой вершин a_i и b_i (см. рис. 4).

Теорема 6 (второй и следующие вложенные циклы). Сформированные при «правом обходе» циклы нулевой вложенности $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$ в цикле C_v (см. формулу (4)) не имеют общих вершин вне цикла C_v.

Доказательство. Допустим, что циклы C_{v_i} и C_{v_j} имеют общую вершину w, которая не принадлежит циклу C_v. Тогда нулевые вершины этих циклов a_i и a_j связаны маршрутом M по рёбрам циклов C_{v_i} и C_{v_j} через вершину w. Исходная карта блока является связным графом. После удаления обратных рёбер получим остовное дерево T_0 карты [13, с. 63]. Вершины a_i и a_j в этом дереве останутся связанными маршрутом M, так как обратные рёбра примыкают к вершинам b_i и b_j. Но эти же вершины останутся связанными в остовном дереве T_0 и маршрутом N по рёбрам цикла C_v. Маршруты M и N образуют цикл, что невозможно в дереве. Значит, предположение неверное и никакая пара циклов C_{v_i} и C_{v_j} не имеет общих вершин вне цикла C_v. Отметим, что вершины цикла C_v могут быть общими вершинами соседних циклов нулевой вложенности. Например, начальная вершина b_i цикла C_{v_i} может быть нулевой вершиной a_{i+1} цикла $C_{v_{i+1}}$ (см. рис. 4). Перечень циклов, представленных формулой (4), получен полным последовательным перебором по алгоритму поиска в глубину. Поэтому любой другой цикл нулевой вложенности в цикле C_v должен совпадать с одним из них. ■

Теорема 7. Пусть $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$ — все циклы нулевой вложенности в цикле C_v, полученные в результате «правого обхода» карты блока. Тогда цикл

$$S_v = (((C_v \oplus C_{v_1}) \oplus C_{v_2}) \oplus \ldots \oplus C_{v_n}) \tag{5}$$

является циклом одной ячейки карты блока, локализованной в цикле C_v.

Доказательство. Из теорем 5 и 6 имеем, что каждый из циклов $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$ является смежным циклу C_v и никакая пара циклов из $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$ не имеет общих вершин. Значит, результатом вычисления по формуле (5) будет цикл C_v, из которого последовательно удаляются все циклы $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$ нулевой вложенности. Цикл S_v не может быть пустым, так как в этом случае будем иметь линейно зависимые циклы C_v и $C_{v_1}, C_{v_2}, \ldots, C_{v_n}$, что противоречит их принадлежности DFS-базису циклов. Предположение, что остаток S_v может оказаться объединением двух или более ячеек карты, нарушит условие полноты циклов DFS-базиса. Значит, S_v есть цикл ячейки карты блока. Всё множество циклов ячеек карты блока $S = \{S_1, S_2, \ldots, S_{n_F}\}$ составляет базис циклов блока. Каждый цикл DFS-базиса C_j карты блока можно представить как линейную комбинацию $C_j = S_{j_1} \oplus S_{j_2} \oplus \ldots \oplus S_{j_k}$ циклов ячеек карты блока, которые охватывает данный цикл C_j. Размерность линейного пространства является инвариантом, следовательно, число циклов DFS-базиса и циклов ячеек карты совпадает. Отсюда заключаем, что все циклы S_v ячеек карты представляются по формуле (5), где каждый цикл C_v определяет свой цикл S_v ячейки карты. ■

4. Вычисление циклов ячеек карты графа

Предлагаемый алгоритм вычисления циклов ячеек карты блока реализует их расчёт по формуле (5). Справедливость данной формулы ограничена рамками карты блока. При разложении исходного графа на блоки можно воспользоваться алгоритмом,

представленным в [8]. Формирование циклов DFS-базиса и вычисление циклов ячеек карты блока по формуле (5) представлены в алгоритмах 1 и 2.

Алгоритм 1. Поиск в глубину циклов ячеек блока карты графа

1: $i = 0$; // *Счетчик циклов C_i DFS-базиса*
2: $j = 0$; // *Индекс верха стека циклов S_j и стеков вершин ev_j, ex_j обратных ребер*
3: $m = 0$; // *Число циклов S_j ячеек карты в архиве циклов T_m*
4: **Для всех** $v \in X$
5: $Mark[v] = 0$; // *Начальные метки вершин*
6: $v =$ `MinOY`(); // *Поиск вершины v с минимальной координатой по оси OY*
7: `Sort1`(v); // *Сортировка по полярному углу ребер, инцидентных вершине v*
8: $count = 0$; // *Счётчик меток пройденных вершин*
9: $nC = 1$; // *Число вершин в стеке C пройденных вершин*
10: $C[nC] = v$; // *Добавить вершину v в стек C пройденных вершин*
11: `Cycle`($v, 0$); // *Рекурсивный проход в глубину, 0 — фиктивная вершина*
12: $nC = nC - 1$; // *Удалить вершину из стека C пройденных вершин*
13: **ПРОЦЕДУРА** `Cycle`(x, y) // *Рекурсивный проход в глубину*
14: // *x — текущая вершина, y — вершина, из которой пришли в x*
15: $count = count + 1$; $Mark[x] = count$. // *Вершина x пройдена*
16: **Для всех** $v \in Adj[x]$
17: $nC = nC + 1$; $C[nC] = v$. // *Добавить вершину v в стек вершин C*
18: **Если** $Mark[v] = 0$, **то**
19: // *Вершина v не пройдена*
20: `Sort2`(v, x); // *Сортировка по полярному углу относительно ребра (x, v)*
21: `Cycle`(v, x); // *Рекурсивный проход в глубину по ребру (x, v)*
22: **иначе если** $Mark[v] < Mark[x] \wedge v \neq y$, **то**
23: // *(x, v) — обратное ребро*
24: $i = i + 1$; // *Номер нового цикла C_i DFS-базиса*
25: `CopyCycle`(C_i); // *Копировать C_i из стека вершин C*
26: **Если** $j > 0$, **то**
27: // *Удалить из цикла S_j, вложенный в него цикл C_i*
28: $S_j = S_j \oplus C_i$;
29: // *(x, v) — вершины обратного ребра цикла сохранить в стеках ev_j и ex_j*
30: $j = j + 1$; // *Индекс верха стеков S_j, ev_j и ex_j*
31: $ev_j = v$; // *v — первая вершина цикла*
32: $ex_j = x$; // *x — последняя вершина цикла*
33: $S_j = C_i$; // *Начать формирование нового цикла S_j ячейки карты*
34: **иначе если** $(x = ev_j) \wedge (v = ex_j)$, **то**
35: // *Вернуться по дереву поиска в начало цикла S_j ячейки карты*
36: // *Вершина x — начало цикла S_j, вершина v — конец цикла*
37: // *Цикл S_j ячейки карты сформирован, сохранить цикл в архиве T*
38: $m = m + 1$; // *Увеличить размер архива T циклов S_j ячеек карты*
39: $T_m = S_j$; // *Сохранить цикл S_j ячейки карты в архиве T*
40: $j = j - 1$; // *Сформированный цикл S_j удаляем из стека*
41: $nC = nC - 1$. // *Удалить вершину из стека C при возврате по дереву поиска*

Алгоритм 2. Копировать из стека C вершины цикла C_i

1: **ПРОЦЕДУРА** `CopyCycle`(C_i)
2: $k = nC$; // *Верх стека вершин* C
3: $w = C[k]$; // *Вершина* w *— начало цикла по обратному ребру*
4: $n = 1$; // *Индекс упаковки вершин цикла* C_i
5: **Повторять**
6: $k = k - 1$; $C_i[n] = C[k]$; $n = n + 1$;
7: **Пока** $C[k] = w$.

Карта блока исходного графа $G(X, U)$ задаётся структурой смежности $Adj[x]$, где $Adj[x]$ обозначает множество вершин, смежных с вершиной $x \in X$. Чтобы отличить пройденные вершины от непройденных, вводится вектор $Mark[x]$ меток вершин, которые постепенно нумеруются от 1 до $|X|$ по мере того, как попадаем в них.

Пройденные по рёбрам карты вершины сохраняются в стеке C пройденных вершин, например, $C[nC] = v$, где v — пройденная вершина, nC — индекс верха стека. Когда попадаем на обратное ребро (x, v), обнаруженный цикл C_i базиса будет состоять из этого ребра и рёбер, соединяющих вершины из верха стека до вершины v обратного ребра в стеке, где x — последняя и v — первая вершины цикла. Процедура `CopyCycle`(C_i) (алгоритм 2) копирует вершины цикла из стека C в цикл C_i базиса.

Циклы S_j ячеек карты блока во время их вычисления размещаются в стеке циклов ячеек карты, где j — индекс верха стека. Вычислению по формуле (5) подлежит текущий цикл S_j, после вычисления он копируется в архив T_m и удаляется из стека.

Каждый новый сформированный цикл C_i базиса будет циклом нулевой вложенности (если это не огибающий цикл блока) в текущем вычисляемом цикле S_j ячейки карты, из которого его исключаем ($S_j = S_j \oplus C_i$) по формуле (5).

С другой стороны, новый цикл C_i базиса определяет и новый цикл S_{j+1} ячейки карты блока (см. теорему 7). Начальное значение устанавливается равным $S_{j+1} = C_i$. Данный цикл размещается на верху стека ячеек карты и далее уже в этом цикле S_{j+1} выполняется поиск циклов базиса нулевой вложенности.

Определим условие, которое фиксирует, что все циклы базиса нулевой вложенности в цикл S_j (начальное значение $S_j = C_i$) исключены из S_j по формуле (5). Формирование цикла C_i базиса заканчивается обратным ребром (x, v), где v — первая и x — последняя вершины цикла. Вершины v и x обратных рёбер циклов C_i сохраняются в своих стеках $ev_j = v$ и $ex_j = x$, где ev_j — стек первых вершин циклов; ex_j — стек последних вершин циклов; j — индекс верха стеков. Индекс j является индексом верха и для стека циклов S_j ячеек.

При поиске циклов нулевой вложенности в цикле C_i мы возвращаемся от последней вершины x по ребрам этого цикла к его начальной вершине v, найденной по обратному ребру (x, v) цикла. Пусть мы вернулись в начальную вершину цикла. Текущая рассматриваемая вершина в алгоритме 1 обозначается через x, а исследуемое ребро — через (x, v). Пусть далее вершина x является начальной вершиной цикла C_i. Все рёбра (x, v), инцидентные вершине x и локализованные в границах цикла C_i, будут пройдены как обратные для вложенных циклов. Для продолжения обхода из вершины x выполняется перебор инцидентных ей рёбер (x, v). Инцидентные рёбра вершин сортированы по полярному углу (см. правила 1 и 2). Поэтому перебор начнётся с рёбер (x, v), инцидентных вершине x и локализованных в границах цикла C_i. Для выделения об-

ратного ребра выполняется проверка условия $(x = ev_j) \wedge (v = ex_j)$, которое является признаком окончания вычисления текущего цикла ячейки S_j карты блока, так как все циклы нулевой вложенности в цикле C_i выделены.

Описание алгоритма 1. Начальная вершина $v = \texttt{MinOY}()$ прохода по графу — это вершина с минимальной координатой по оси OY, что гарантирует принадлежность вершины v огибающему циклу карты блока (см. теорему 4). В соответствии с правилом 1 выполняется сортировка $\texttt{Sort1}(v)$ рёбер, инцидентных вершине v, в порядке возрастания их полярных углов. В этом порядке располагаются соответствующие рёбрам вершины в структуре смежности $Adj[v]$ для вершины v. Поэтому перебор рёбер, инцидентных вершине v, выполняется против часовой стрелки в порядке роста их полярных углов. Выбранная вершина v первой помещается в стек пройденных вершин ($nC = 1$; $C[nC] = v$). Далее выполняется первый вызов рекурсивной процедуры $\texttt{Cycle}(v, 0)$ обхода графа в глубину с двумя параметрами: v — текущая выбранная вершина для просмотра; 0 — фиктивная вершина при первом вызове.

ПРОЦЕДУРА $\texttt{Cycle}(x, y)$ — заголовок описания рекурсивной процедуры обхода карты блока в глубину, где x — текущая обрабатываемая вершина; y — вершина, из которой пришли в вершину x по ребру (y, x). Далее выполним разбор ключевых операторов данной процедуры.

Разбор цикла: Для всех $v \in Adj[x]$. В данном цикле из текущей вершины x пытаемся выполнить проход в глубину по ребру (x, v). Для вершины v возможны три случая.

С л у ч а й 1: «**Если** $Mark[v] = 0$**, то**». Имеем вершину v, которая не пройдена. Рекурсивно выполняем проход в глубину $\texttt{Cycle}(v, x)$ с новой текущей вершиной v, где (x, v) — ребро, по которому пришли в v. Предварительно, в соответствии с правилом 2, выполняется сортировка $\texttt{Sort2}(v, x)$ рёбер, инцидентных вершине v, в порядке возрастания полярных углов относительно ребра (x, v) против часовой стрелки. В соответствующем порядке располагаются вершины в структуре смежности $Adj[v]$.

С л у ч а й 2: «**иначе Если** $Mark[v] < Mark[x] \wedge v \neq y$**, то**». Условие фиксирует, что вершина v была пройдена и мы приходим в неё по обратному ребру (x, v) цикла, так как $v \neq y$. Вершины v и x обратного ребра найденного цикла сохраняем в соответствующих стеках $ev_j = v$ и $ex_j = x$, где v — вершина начала цикла; x — последняя вершина цикла. Сформированный цикл из пройденных вершин в стеке C копируем в очередной цикл C_i базиса.

Новый цикл базиса C_i является циклом нулевой вложенности в текущем вычисляемом цикле S_j ячейки карты блока (см. теорему 5), из которого его исключаем ($S_j = S_j \oplus C_i$) по формуле (5).

Новый цикл C_i базиса определяет и новый цикл S_{j+1} ячейки карты блока (см. теорему 7). Начальное значение устанавливается равным $S_{j+1} = C_i$. Данный цикл размещается на верху стека циклов ячеек карты и далее уже в этом цикле S_{j+1} выполняется поиск циклов базиса нулевой вложенности.

С л у ч а й 3: «**иначе Если** $(x = ev_j) \wedge (v = ex_j)$**, то**». Здесь (x, v) — ребро прохода по карте блока из вершины x в вершину v. Ребро (ex_j, ev_j) — обратное ребро текущего вычисленного цикла S_j ячейки карты, где ev_j — вершина начала цикла; ex_j — вершина конца цикла. Выражение $(x = ev_j) \wedge (v = ex_j)$ означает, что в результате прохода по графу мы вернулись в вершину x и, продолжив перебор смежных ей вершин, выбрали ребро (x, v), которое является обратным (ex_j, ev_j) для цикла S_j. Отсюда заключаем, что цикл S_j ячейки карты сформирован. Цикл копируется в архив ($T_m = S_j$) и удаляется из стека циклов ячеек карты посредством операции $j = j - 1$. Это значит, что

возвращаемся к формированию оставшегося на верху стека цикла S_j ячейки карты блока.

Сложность алгоритма 1 оценим относительно числа вершин в графе. Пусть $G = (X, U)$ — исходная карта блока. В простом планарном графе выполняется соотношение $|U| \leqslant 3|X| - 6$ (см. п. 2). Пусть $n = |X|$, тогда $|U| \leqslant 3|X| - 6 = 3n - 6 = O(n)$. Оценим сложность в худшем случае, когда $|U| = O(n)$.

Сложность начальной сортировки $\texttt{Sort1}(v)$ вершин, смежных вершине v, можно считать равной $O(n \log_2 n)$. Сложность поиска $v = \texttt{MinOY}()$ является линейной — $O(n)$.

Сложность выполнения процедуры $\texttt{Cycle}(x, y)$ определяется сложностью выполнения цикла «**Для всех** $v \in Adj[x]$». Каждое ребро (x, v) просматривается дважды. Пусть n_i — число вершин, смежных вершине x_i, тогда $\sum\limits_{i=1}^{n} n_i = 2|U| = O(n)$. Таким образом, цикл выполняется $O(n)$ раз.

Для каждой вершины один раз в цикле выполняется сортировка $\texttt{Sort2}(v, x)$ смежных ей вершин. Полагаем, что сортировка n_i вершин имеет сложность $C_i n_i \log_2 n_i$, где C_i — положительная константа. Суммарная сложность сортировки составит $\sum\limits_{i=1}^{n} C_i n_i \log_2 n_i \leqslant C \sum\limits_{i=1}^{n} n_i \log_2 n_i \leqslant C \log_2 n \sum\limits_{i=1}^{n} n_i \leqslant C(\log_2 n) O(n) = O(n \log_2 n)$, где C — константа, максимальная из всех C_i.

Сложности операций копирования $\texttt{CopyCycle}(C_i)$ и слияния $S_j = S_j \oplus C_i$ являются линейными относительно длин циклов. Длины циклов не превышают числа рёбер $|U| = O(n)$. Суммарная сложность операций копирования не превышает величины $\sum\limits_{i=1}^{n} n_i |U| = |U| \sum\limits_{i=1}^{n} n_i \leqslant 2|U||U| = 2O(n)O(n) = O(n^2)$. Отсюда заключаем, что сложность алгоритма 1 составляет $O(n^2)$.

Заключение

Рассмотренный алгоритм вычисления циклов ячеек карты блока графа является расширением классического алгоритма поиска в глубину циклов DFS-базиса. Ключевой идеей модификации классического алгоритма является предложенная стратегия «правого обхода» при прохождении графа в глубину. Правый обход при поиске в глубину позволил для каждого цикла DFS-базиса последовательно выделить и исключить из него все циклы нулевой вложенности. Показано, что оставшаяся часть базисного цикла является циклом ячейки карты блока. Сложность каждого шага алгоритма не превышает квадратичной относительно числа вершин в графе.

ЛИТЕРАТУРА

1. *Оре О.* Теория графов. М.: Наука, 1980. 336 с.
2. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. М.: Мир, 1989. 478 с.
3. *Welch J.* A mechanical analysis of the cyclic structure of undirected linear graphs // J. Assoc. Comput. Mech. 1966. V. 13. P. 205–210.
4. *Gibbs N. W.* A cycle generation algorithm for finite undirected linear graphs // J. Assoc. Comput. Mech. 1969. V. 16. P. 564–568.
5. *Tarjan R.* Enumeration of the elementary circuits of a directed graph // SIAM J. Comput. 1973. V. 2. P. 211–216.
6. *Jonson D. B.* Finding all the elementary circuits of a directed graph // SIAM J. Comput. 1975. V. 4. P. 77–84.
7. *Mateti P. and Deo N.* On algorithms for enumerating all circuits of a graph // SIAM J. Comput. 1976. V. 5. No. 1. P. 90–99.

8. *Tarjan R.* Depth-first search linear graph algorithms // SIAM J. Comput. 1972. V. 1. P. 146–160.
9. *Mahdi F., Safar M., and Mahdi K.* Detecting cycles in graphs using parallel capabilities of GPU // Digital Inform. Commun. Techn. Appl. (DISTAP). 2011. V. 167. P. 193–205.
10. *Kavitha T., Mehlhorn K., and Michail D.* New approximation algorithms for minimum cycle bases of graphs // J. Algorithmica. 2011. V. 59. No. 4. P. 471–488.
11. *Pfaltz J. L.* Chordless cycles in networks // IEEE 29th Intern. Conf. ICDEW. 2013. P. 223–228.
12. *Иванов Б. Н.* Генерация циклов ячеек карты простого планарного графа // Вычислительные методы и программирование. 2014. № 15. С. 304–316.
13. *Алексеев В. Е., Таланов В. А.* Графы. Модели вычислений. Структуры. Н. Новгород: Изд-во ННГУ, 2005. 307 с.
14. *Paton K.* An algorithm for finding a fundamental set of cycles of a graph // Comm. ACM. 1969. V. 12. No. 9. P. 514–518.
15. *Рейнголд Э., Нивергельд Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980. 476 с.

REFERENCES

1. *Ore O.* Theory of Graphs. AMS Press, Providence, 1962. 336 p.
2. *Preparatat F. P. and Shamos M. I.* Computational Geometry: An Introduction. N.Y., Springer Verlag, 1989. 478 p.
3. *Welch J.* A mechanical analysis of the cyclic structure of undirected linear graphs. J. Assoc. Comput. Mech., 1966, vol. 13, pp. 205–210.
4. *Gibbs N. W.* A cycle generation algorithm for finite undirected linear graphs. J. Assoc. Comput. Mech., 1969, vol. 16, pp. 564–568.
5. *Tarjan R.* Enumeration of the elementary circuits of a directed graph. SIAM J. Comput., 1973, vol. 2, pp. 211–216.
6. *Jonson D. B.* Finding all the elementary circuits of a directed graph. SIAM J. Comput., 1975, vol. 4. pp. 77–84.
7. *Mateti P. and Deo N.* On Algorithms for enumerating all circuits of a graph. SIAM J. Comput., 1976, vol. 5. no. 1, pp. 90–99.
8. *Tarjan R.* Depth-first search linear graph algorithms. SIAM J. Comput., 1972, vol. 1, pp. 146–160.
9. *Mahdi F., Safar M., and Mahdi K.* Detecting cycles in graphs using parallel capabilities of GPU. Digital Inform. Commun. Techn. Appl. (DISTAP), 2011, vol. 167, pp. 193–205.
10. *Kavitha T., Mehlhorn K., and Michail D.* New approximation algorithms for minimum cycle bases of graphs. Algorithmica, 2011, vol. 59, no. 4, pp. 471–488.
11. *Pfaltz J. L.* Chordless cycles in networks. IEEE 29th Intern. Conf. ICDEW, 2013, pp. 223–228.
12. *Ivanov B. N.* Generatsiya tsiklov yacheek karty prostogo planarnogo grafa [Generation of cycles of map cells for a simple planar graph]. Vychislitel'nye Metody i Programmirovanie, 2014, no. 15, pp. 304–316. (in Russian)
13. *Alekseev B. E. and Talanov V. A.* Grafy. Modeli vychisleniy. Struktury [Graphs. Computational Models. Structures]. N. Novgorod, Lobachevsky State University, 2005. 307 p. (in Russian)
14. *Paton K.* An algorithm for finding a fundamental set of cycles of a graph. Comm. ACM, 1969, vol. 12, no. 9, pp. 514–518.
15. *Reingold E. M., Nievergelt J., and Deo N.* Combinatorial Algorithms. Theory and Practice. Prentice-Hall, Englewood Cliff, 1977. 476 p.