УДК 519.17:519.6

ГЕНЕРАЦИЯ ЦИКЛОВ ЯЧЕЕК КАРТЫ ПРОСТОГО ПЛАНАРНОГО ГРАФА

Б. Н. Иванов[1]

Рассматривается конструктивный метод генерации циклов ячеек (chordless cycles) карты простого планарного графа. Циклы ячеек карты представляются как линейные комбинации циклов DFS-базиса. Искомые линейные комбинации строятся явно на основе выделенных свойств структуры вложенности циклов базиса и циклов ячеек карты графа. Карта планарного графа позволила отойти от общепринятого подхода генерации циклов и явно внести геометрию карты в структуру алгоритма. На множестве циклов базиса определяется отношение соседства, которое порождает корневые деревья структуры вложенности циклов. Ячейки карты графа являются результатом обхода данного множества корневых деревьев. Сложность предложенного алгоритма является кубической относительно числа вершин в графе. Рассматривается приложение алгоритма в рамках решения задач на планарном подразбиении.

Ключевые слова: генерация циклов, перечисление циклов, базис циклов, карта графа, вложенность циклов, фундаментальные циклы, chordless cycles.

1. Введение. Впервые предлагаемый метод был реализован как решение задачи раскраски различного рода территорий на поверхности Земли, в частности государств, в рамках действующей геоинформационной системы “Океан” [1], предназначенной для обеспечения безопасности полетов и мореплавания. Пример такой раскраски представлен на рис. 1a. Исходные данные в задаче раскраски территорий — это география поверхности Земли (береговая черта) и границы государств. Линии границ разбивают береговую черту на отрезки, которые дополняют границы государств. Данной модели поверхности Земли поставим в соответствие конечный простой планарный граф $G = (X, U)$, где U — ребра графа (границы государств) и X — вершины графа (точки пересечения границ). Простой граф является неориентированным графом, и любая пара его вершин соединена не более чем одним ребром. К такому графу $G = (X, U)$ можно всегда свести рассматриваемую модель поверхности Земли, включая фиктивные вершины на границах или береговой черте.

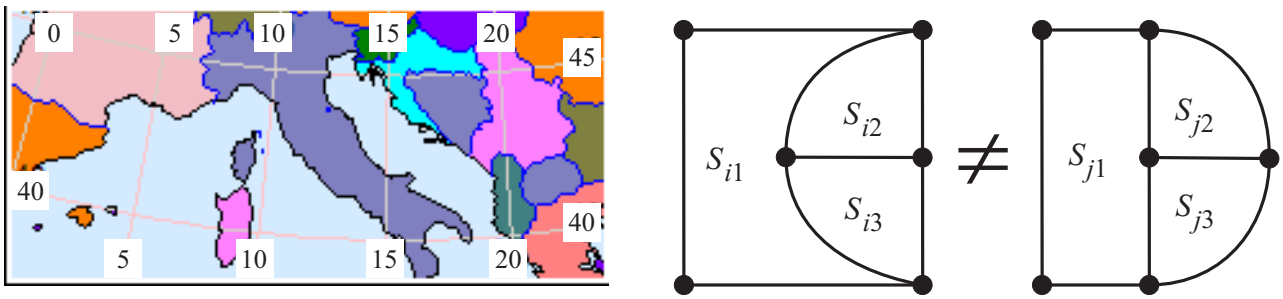

Рис. 1. Пример карты простого планарного графа

Укладка на плоскости ребер планарного графа без самопересечений определяет разбиение такой плоскости, называемое *планарным подразбиением* или *картой* графа [2]. Карта графа фиксирует геометрию его ребер. Так, на рис. 1b и 1c изображены два одинаковых планарных графа, однако карты указанных графов не равны. Сама карта складывается из простых многоугольников (подразбиений), которые будем называть *ячейками* карты. Для графа на рис. 1b ячейки карты обозначены через S_{i1}, S_{i2} и S_{i3}, а для графа на рис. 1c — через S_{j1}, S_{j2} и S_{j3}. Каждая сторона такой ячейки может принадлежать еще ровно одной такой же ячейке. Границы ячеек карты являются простыми циклами.

Замечание 1. Полагаем, что ячейки карты графа не могут включать другие ячейки. В нашей интерпретации это означает, что исключаются случаи расположения государства в государстве, а исходный

[1] Дальневосточный федеральный университет, Школа естественных наук, ул. Суханова, 8, 690091, г. Владивосток; доцент, e-mail: ibn8826@mail.ru

граф $G(X,U)$ связный. В общем случае граф $G(X,U)$ необходимо разложить на компоненты связности [3, с. 36] и уже для каждой компоненты отдельно решать задачу.

В качестве приложения предлагаемого алгоритма следует отметить его использование в задаче локализации точек на *планарном подразбиении* [2, с. 52]. Раскраска территорий — одна из задач такого класса. Рассмотренный в работе [2, с. 62] оптимальный алгоритм поиска на планарном подразбиении сложности $O(\log_2 n)$ предполагает, что в качестве предобработки будет решена задача генерации ячеек карты планарного подразбиения.

Отметим, что ячейки карты графа позволяют установить различные геометрические характеристики своих областей. Геометрическое представление ареала обитания популяций, колоний, хищников — это специальный вид планарного подразбиения, называемого диаграммой Вороного [2, с. 249; 4]. Диаграмма состоит из множества $S = \{s_1, s_2, \ldots, s_n\}$ локусов — областей точек, обладающих требуемыми свойствами. Общий вид диаграммы Вороного показан на рис. 2. Ячейки карты диаграммы определяют границы территорий обитания хищников. На основе диаграммы Вороного возможна триангуляция области. Так, если на множестве точек плоскости $S = \{s_1, s_2, \ldots, s_n\}$ построить диаграмму Вороного, то граф, двойственный диаграмме, является триангуляцией выпуклой оболочки точек множества S [5]. Ячейки карты диаграммы позволяют выполнить переход к триангуляции области.

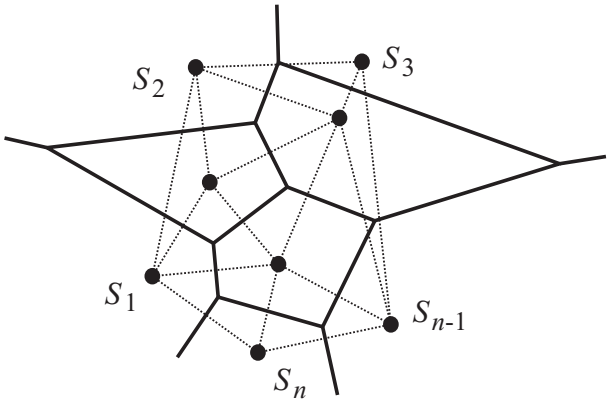

Рис. 2. Диаграмма Вороного

Сегодня значительное место в исследованиях различного рода сетей, включая социальные сети, технологические, биологические и др., занимает анализ их циклической структуры [6]. Для информационных сетей актуальна задача сетевой надежности передачи данных. Здесь первостепенное значение приобретает знание циклической структуры сети [7]. Циклическая структура характеризует способы подключения сети в целом.

Изучение циклических структур графа является важной составляющей общего анализа структуры графа. Циклы ячеек карты иначе называют “бесхордовыми циклами” графа (в переводе “chordless cycles”). С учетом контекста рассматриваемой задачи можно встретить и другое их название — “дыры” графа (в переводе “holes”).

Чтобы представить общее состояние вопроса перечисления циклов графа $G(X,U)$, необходимо пояснить структуру этого множества циклов. Хорошо известно, что множество циклов графа является линейным векторным пространством (см. раздел 2 настоящей статьи), размерность базиса которого определяется числом циклов $|U|-|X|+1$. Следовательно, произвольный цикл графа $G(X,U)$ можно получить линейной комбинацией циклов данного базиса. Существует достаточно много эффективных алгоритмов получения циклов базиса (см. раздел 3). Однако непосредственный поиск каких-либо циклов на основе базиса в практических алгоритмах встречается довольно редко. Связано это с экспоненциальной сложностью поиска, так как число линейных комбинаций на основе базиса может составить $2^{|U|-|X|+1}$.

Первый практический алгоритм перечисления всех циклов графа был опубликован в работе [8]. В основу алгоритма генерации циклов был положен базис пространства циклов графа (*фундаментальное множество циклов*). Поиск линейных комбинаций циклов базиса, соответствующих циклам графа, осуществлялся *методом поиска с возвращениями*. В общем случае сложность данного метода определяется структурой исходного графа и условиями поиска. В худшем случае придется перебрать все линейные комбинации, число которых, как указано выше, составляет $2^{|U|-|X|+1}$.

Модифицированная версия алгоритма [8] была опубликована в статье [9]. Работы [8, 9] определили направление разработки алгоритмов генерации циклов графов. Стало очевидным, что в данной 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 [10, 11] порождения *всех циклов графа*. Обзор этих алгоритмов и их относительных достоинств можно найти в [12]. Основу этих алгоритмов составляет *метод поиска в глубину* на графе [13]. Это адаптированный к структуре графа поиск с возвращениями. Оказалось, что генерацию всех циклов графа не обязательно проводить только на основе циклов базиса. Более того, перечисление всего множества циклов графа наиболее эффективно выполнить просто поиском их в глубину на графе [11]. Сложность генерации циклов в таких алгоритмах определяется величиной

$O\Big(\big(|X|+|U|\big)(\ell+1)\Big)$, где ℓ — число порожденных циклов.

В работе [14] предложен алгоритм генерации всех циклов сложности $O\Big(\big(|X|+|U|\big)(\ell+1)\Big)$ для планарных графов, для которых перечисление циклов выполняется уже на основе базиса пространства циклов и метода поиска в глубину. Таким образом, было показано, что в классе планарных графов перечисление всех циклов на основе циклов базиса может быть столь же эффективным, как и алгоритм непосредственной генерации циклов на основе поиска в глубину.

Перечисление циклов графа с выделенными свойствами, как правило, осуществляется на основе алгоритмов генерации всего множества циклов. Каждый такой алгоритм является уникальным. Проверка каких-либо условий может существенно увеличить сложность алгоритма. Здесь дополнительно накладываются эвристические характеристики метода поиска в глубину на графе. Поэтому при оценке сложности авторы прибегают к получению средних оценок. Для этого проводят многочисленные тестовые расчеты на случайных графах. Полученные временны́е характеристики сравнивают с подобными характеристиками других алгоритмов.

Исключением из этого правила может служить работа [15], в которой предлагается проводить генерацию произвольных циклов на основе "принципа исключения" относительно вершин графа. Из общего перечня вершин последовательно выделяются их подмножества, которые составляют циклы графа. Неявно формируется дерево поиска, что позволяет избежать полного перебора. Приводятся лишь экспериментальные данные, по которым можно судить о качестве алгоритма.

Сегодня можно встретить и параллельные алгоритмы генерации циклов в графе [16, 21]. В большей степени такие алгоритмы представляют интерес в теоретическом плане. Авторы пытаются решать тестовые задачи большой размерности с использованием параллельных вычислений. Основу таких алгоритмов, как правило, составляют уже известные практические алгоритмы. На взвешенных графах задача генерации циклов с определенными свойствами становится оптимизационной задачей. В этом случае для поиска циклов базиса в работе [17] предлагается приближенный алгоритм как альтернатива полному перебору.

В работе [6] рассматривается формализация экологических сетей неориентированными графами. Анализ циклов в таком графе позволяет лучше понять структуру экологических сетей. Для генерации циклов авторы предлагают свой алгоритм на основе поиска в глубину. Достоинством алгоритма можно считать возможность выделения циклов графа с заранее определенной длиной. Качество работы алгоритма проверяется на тестовых расчетах.

В настоящей статье предлагается прямой метод построения всех циклов ячеек карты простого планарного графа на основе циклов *DFS-базиса* (Depth First Search). Каждый цикл ячейки карты рассматривается как линейная комбинация циклов *DFS-базиса*. Предлагаемый подход является конструктивным методом и не обращается к перебору какого-либо множества циклов. Искомые линейные комбинации циклов *DFS-базиса* строятся явно на основе выделенных свойств структуры вложенности циклов базиса и ячеек карты графа. Карта планарного графа позволила отойти от общепринятого подхода генерации циклов и явно внести геометрию карты в структуру алгоритма. Сложность алгоритма является кубической относительно числа вершин в графе.

2. Фундаментальное множество циклов. Следуя [18, с. 382], введем определение *DFS-базиса* и свяжем его с циклами ячеек карты графа. Пусть $G=(X,U)$ — исходный граф. Обозначим через $M=\{G_1,G_2,\ldots,G_{n_M}\}$ множество всех подграфов $G_i=(X,U_i)$ графа G, где $U_i\subseteq U$. На множестве M определим бинарную операцию $\oplus$ так, что $\forall G_i,G_j\in M\ : G_i\oplus G_j=(X,U_i\oplus U_j)$, где символ $\oplus$ обозначает операцию симметрической разности, $U_i\oplus U_j=\big\{u\,|\,u\in U_i\cup U_j\ \wedge\ u\notin U_i\cap U_j\big\}$. Ясно, что множество M с операцией $\oplus$ является абелевой группой [19, с. 33]. Единица группы — пустой подграф $O=(X,\varnothing)$. Обратным к G_k является тот же самый G_k, так как $G_k\oplus G_k=O$.

Обозначим через $Z=\{Z_1,Z_2,\ldots,Z_{n_Z}\}$, где $Z_k=(X,z_k)$ — подграфы графа G, z_k — все простые циклы графа G. Пусть $L=\{\lambda_{i_1}Z_{k_1}\oplus\lambda_{i_2}Z_{k_2}\oplus\ldots\oplus\lambda_{i_S}Z_{k_S}\}$ — 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 коммутативной группой с операцией $\oplus$ и линейным векторным пространством *циклов* над полем $P=\big\langle\{0,1\},\oplus,\cdot\big\rangle$ [20, с. 61]. Уточним структуру, размерность и базис пространства L.

Пусть $T_0=(X,U_0)$ — произвольное остовное дерево исходного *связного* графа $G=(X,U)$. Для дерева выполняется соотношение $|U_0|=|X|-1$. Пусть $E=U\setminus U_0=\{e_1,e_2,\ldots,e_{n_F}\}$ — ребра, не вошедшие в T_0. Такие ребра называют обратными. Любое ребро $e_i\in E$ порождает в T_0 ровно один простой цикл C_i. Таких циклов C_i можно составить в количестве $n_F=|U\setminus U_0|=|U|-|X|+1$. Множество $F=\{C_1,C_2,\ldots,C_{n_F}\}$ называется *фундаментальным множеством циклов* и образует базис пространства L [20, с. 63].

Отметим, что для планарных графов выполняется соотношение $|U|=O\big(|X|\big)$ (формула Эйлера [3]).

Поэтому в вычислениях будем пользоваться оценкой $n_{_F} = O\big(|X|\big)$.

3. Поиск в глубину циклов DFS-базиса. На практике, как правило, для построения фундаментального множества циклов $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ прибегают к *методу поиска в глубину*. Подробное изложение данного алгоритма встречается во многих работах [18, 20, 22, 23]. Оригинальное его описание представлено в статье [24], сложность алгоритма кубическая $O\big(|X|^3\big)$ относительно числа вершин в графе. В настоящей статье будем следовать описанию алгоритма генерации циклов *DFS-базиса*, представленному в работе [18, с. 382].

Метод поиска в глубину на простом связном графе $G = (X, U)$ строит дерево поиска T_0, которое является остовным деревом графа и называется *DFS-деревом*. Если исходный граф $G = (X, U)$ несвязный, то T_0 будет лесом остовных деревьев. Базис $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ пространства циклов относительно *DFS-дерева* называется *DFS-базисом*. Алгоритм строит остовное дерево T_0, и каждое обратное ребро порождает цикл относительно этого дерева. Для того чтобы следить за ребрами дерева, используется *поиск в глубину со стеком*, в котором хранятся все текущие вершины пройденного пути в данный момент. Обратное ребро приводит маршрут в вершину графа, которая пройдена ранее и находится в стеке. Такое ребро порождает цикл $C_i \in F$, который будет состоять из обратного ребра и ребер, соединяющих вершины графа в стеке сверху вниз до обнаруженной вершины обратного ребра.

Утверждение 1. *Определим два цикла смежными, если у них есть общие ребра. Стековая природа алгоритма формирования циклов DFS-базиса методом поиска в глубину позволяет утверждать, что любая пара смежных циклов базиса может иметь только одну непрерывную границу общих ребер.*

Данное свойство циклов *DFS-базиса* лежит в основе предлагаемого метода генерации циклов ячеек карты графа.

4. Корневые деревья структуры вложенности DFS-базиса. Исходный граф $G = (X, U)$ задачи является простым планарным графом. Ребра графа укладываются на плоскости без самопересечений, формируя карту графа. Такой граф представляется в виде структуры вложенных циклов. Введем на множестве циклов базиса $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ отношения с целью определения новых структур на этом множестве, которые обеспечат доступ к перечню циклов $C_1, C_2, \ldots, C_{n_{_F}}$ в соответствии с их структурой вложенности.

Определение 1. Определим на $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ бинарное отношение $\prec$ охватывающего соседства. Отношение $C_i \prec C_j$ выполняется, если $C_i \subset C_j$ и $\forall C_k \neq C_j \ : (C_i \subset C_k \Rightarrow C_j \subset C_k)$. Это означает, что C_j — наименьший охватывающий цикл для C_i.

Поставим в соответствие структуре вложенности циклов $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ ориентированный *граф циклов* $G_F = (X_F, U_F)$, где $X_F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ — множество вершин; U_F — множество ребер, ребра представляются парами вершин. Так, ориентированное ребро $u = (C_i, C_j)$ принадлежит U_F, если для циклов C_i и C_j выполняется отношение $C_i \prec C_j$. Направленное ребро (C_i, C_j) ориентировано от вложенного цикла C_i к наименьшему охватывающему его циклу C_j $(C_i \to C_j)$.

Определение 2. Определим на $F = \{C_1, C_2, \ldots, C_{n_{_F}}\}$ бинарное отношение $\sim$. Будем полагать, что $C_i \sim C_j$, если существует маршрут из C_i в C_j по ребрам неориентированного графа $G_F = (X_F, U_F)$. Это отношение является отношением эквивалентности и выполняет разложение множества F на не пересекающиеся классы эквивалентности $D_1, D_2, \ldots, D_{n_{_D}}$, для которых $F = D_1 \cup D_2 \cup \ldots \cup D_{n_{_D}}$, $D_i \cap D_j = \varnothing$, $i \neq j$. Классы эквивалентности D_k называются *компонентами связности* [3, с. 36].

В рассматриваемой интерпретации каждая компонента связности D_k будет представлена *корневым деревом циклов*. Корнем дерева будет внешний цикл компоненты D_k. Все ребра ориентированы в направлении от листьев к корню и показывают для каждого цикла (вершины) направление доступа к соседнему охватывающему его циклу (вершине). Соседней вершины нет только у корня дерева, так как внешний цикл компоненты D_k не имеет охватывающего цикла. В разделе 8 рассмотрен пример графа, его циклов, компонент связности и соответствующих им корневых деревьев циклов.

В предлагаемом алгоритме представление исходного графа $G_F = (X_F, U_F)$ формируется *ориентированным* реберным списком $Q_1, Q_2, \ldots, Q_{n_{_F}}$, где элемент Q_i определяется для цикла C_i и равен номеру наименьшего охватывающего его цикла. Следовательно, $Q_i = j$, если выполняется отношение $C_i \prec C_j$ (см. определение 1), а в графе определяется ребро $(C_i \to C_j)$ или $(i \to j)$. Для циклов C_i, которые не имеют охватывающих циклов, значение $Q_i = 0$. Такими вершинами являются корни деревьев. В описании алгоритмов для представления неориентированного графа циклов $G_F = (X_F, U_F)$ будет использоваться его структура смежности $\mathrm{Adj}[x]$ — множество вершин, смежных с данной вершиной $x \in X_F$ [23, с. 269]. Процедура преобразования реберного списка $Q_1, Q_2, \ldots, Q_{n_{_F}}$ в структуру смежности $\mathrm{Adj}[x]$ представлена в алгоритме 1.

Сложность алгоритма 1 определяется двумя вложенными циклами и составляет $O\big(|X|^2\big)$, так как для

планарного графа $|X_F| = O\big(|X|\big)$.

Алгоритм 1. Преобразование реберного списка в структуру смежности
Procedure *CreateAdj* (Q, Adj);
  for $x \in X_F$ **do begin**
    Adj$[x] = \varnothing$; {Начальный пустой список для x}
    if $Q_x \neq 0$ **then** Adj$[x] = \{Q_x\}$; {Ребро, выходящее из x}
    for $v \in X_F$ **do begin**
      if $Q_v = x$ **then** Adj$[x]$ = Adj$[x] \cup \{v\}$; {Ребра, входящие в x}
    end;
  end;
end.

5. Формирование циклов ячеек карты графа. Введем определения, которые позволят на формальном уровне подойти к описанию предлагаемого метода генерации циклов ячеек карты графа.

Определение 3. Два цикла C_i и C_j будем называть *смежными*, если у них есть общие ребра. Под операцией $C_i \cap C_j$ будем понимать пересечение циклов по их ребрам. Следовательно, если $C_i \cap C_j \neq \varnothing$, то циклы C_i и C_j смежные.

В утверждении 1 отмечено, что любая пара смежных циклов *DFS-базиса* может иметь только одну непрерывную границу общих ребер. Поэтому результатом симметрической разности $C_i \oplus C_j$ смежных циклов *DFS-базиса* будет ровно один цикл, который будем называть *слиянием циклов*. Примеры смежных циклов можно найти в разделе 8 на рис. 4.

Определение 4. Пусть циклы $C_{w_1}, C_{w_2}, \ldots, C_{w_k}$ попарно *не вложены* друг в друга. Для данной совокупности циклов определим неориентированный *граф смежности* $J_w = (X_w, U_w)$. Вершинами графа $w_i \in X_w$ являются циклы $C_{w_1}, C_{w_2}, \ldots, C_{w_k}$. Ребро $(w_i, w_j) \in U_w$, если циклы C_{w_i}, C_{w_j} — смежные. Пусть $J_w = (X_w, U_w)$ является деревом. Тогда циклы $C_{w_1}, C_{w_2}, \ldots, C_{w_k}$ будем называть *смежными в совокупности*.

Утверждение 2. *Результатом слияния циклов $C_{w_1}, C_{w_2}, \ldots, C_{w_k}$, смежных в совокупности, будет ровно один цикл $J_w = C_{w_1} \oplus C_{w_2} \oplus \ldots \oplus C_{w_k}$, который будем обозначать через J_w по названию дерева смежности.*

Доказательство. Действительно, слияние циклов $C_{w_1} \oplus C_{w_2} \oplus \ldots \oplus C_{w_k}$ можно получить в результате обхода, например сверху–вниз, дерева смежности $J_w = (X_w, U_w)$, соответствующего указанным циклам. Положим в начале обхода цикл слияния J_w равным C_{w_k} — начало обхода дерева. Каждый раз, приходя в вершину C_{w_j}, будем выполнять слияние $J_w = J_w \oplus C_{w_j}$. Вершина C_{w_j} смежна ранее пройденной вершине $C_{w_i} \in J_w$. Так как слияние смежных циклов $C_{w_i} \oplus C_{w_j}$ дает один цикл, то результатом операции $J_w = J_w \oplus C_{w_j}$ будет тоже один цикл. Что и требовалось доказать.

Утверждение 3. *Пусть циклы DFS-базиса $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$ попарно не вложены друг в друга и определяют связный граф смежности $J_v = (X_v, U_v)$. Тогда граф смежности $J_v = (X_v, U_v)$ данного множества циклов будет деревом.*

Доказательство. Если утверждение верное и граф смежности $J_v = (X_v, U_v)$ является деревом, то структура циклов $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$ будет иметь форму “кактуса” (рис. 3a). “Кактус” собирается из веток (рис. 3b), ветки формируются из листьев, листья — циклы DFS-базиса $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$. Циклы DFS-базиса $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$ формируются из ребер, соединяющих вершины графа в стеке, при обходе его в глубину (см. раздел 3). Ветки “кактуса” формируются независимыми друг от друга. Новая ветка циклов может начать формироваться, если при обходе графа вершины циклов текущей ветки уйдут из стека. Поэтому в стеке в любой момент времени могут располагаться вершины циклов только одной ветки.

На рис. 3b и рис. 3c показана одна из веток циклов DFS-базиса, где A_i — вершина начала обхода цикла C_i; B_i — вершина стека, в которую приводит обратное ребро и которая порождает цикл C_i. На рис. 3c у каждого цикла C_i вершины начала A_i и конца B_i располагаются на одном предыдущем цикле ветки “кактуса”. Покажем, что если вершины A_i и B_i будут располагаться на разных циклах, то среди циклов $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$ будут вложенные друг в друга, что нарушит начальное условие их невложенности. На рис. 3c показан пример, когда A_5 и B_5 располагаются на разных циклах. В этот момент в стеке находятся части вершин циклов, выделенных на рис. 3c более толстой линией. Поэтому новый цикл $C_5 = \{B_5 A_2 A_3 A_4 A_5 B_5\}$ будет включать в себя циклы C_2 и C_4, что нарушает условие невложенности циклов DFS-базиса $C_{v_1}, C_{v_2}, \ldots, C_{v_k}$. Таким образом, граф смежности $J_v = (X_v, U_v)$ является деревом. Что и требовалось доказать.

Утверждение 4. *Пусть C_v является произвольным циклом DFS-базиса и структура вложенных в него циклов представляется совокупностью следующих деревьев смежности: $J_{t_{v1}}, J_{t_{v2}}, \ldots, J_{t_{vk_v}}$. Каждое дерево $J_{t_{vj}} = (X_{t_{vj}}, U_{t_{vj}})$ определяет свой цикл слияния $J_{t_{vj}}$, не смежный с другими. Тогда цикл $S_v = C_v \oplus J_{t_{v1}} \oplus J_{t_{v1}} \oplus \ldots \oplus J_{t_{vk_v}}$ будет определять одну ячейку карты графа.*

Доказательство. Операция $S_v = C_v \oplus J_{t_{v1}} \oplus J_{t_{v1}} \oplus \ldots \oplus J_{t_{vk_v}}$ удаляет из C_v вложенные в него все циклы базиса. Значит, S_v есть цикл ячейки карты графа. Предположение, что остаток S_v может оказаться объединением двух ячеек карты или более, нарушит условие полноты *DFS-базиса*. Так как по построению S_v не содержит циклов *DFS-базиса*, то нельзя составляющие этого цикла S_v представить линейной комбинацией циклов *DFS-базиса*. Что и требовалось доказать.

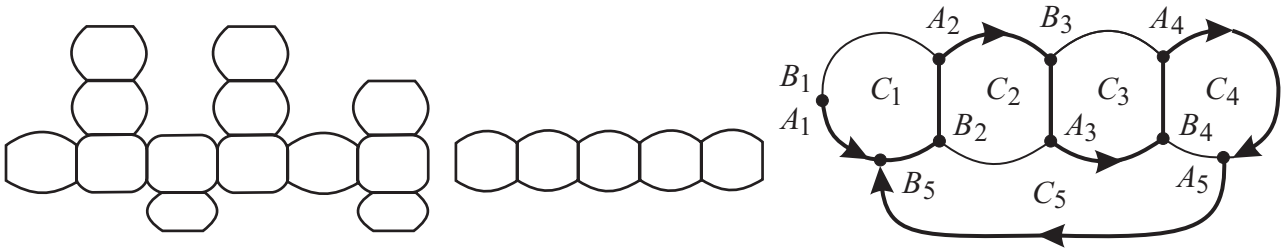

Рис. 3. Структура невложенных циклов DFS-базиса

Введем обозначение $S = \{S_1, S_2, \ldots, S_{n_F}\}$ для множества циклов ячеек карты графа $G = (X, U)$. В настоящей работе речь идет о генерации именно данного множества циклов.

Утверждение 5. *Циклы $S = \{S_1, S_2, \ldots, S_{n_F}\}$ составляют базис пространства L.*

Доказательство. Заметим, что циклы C_i *DFS-базиса* должны покрывать все ячейки S_j карты графа. В противном случае *DFS-базис* будет не полным. Каждый цикл C_i покрывает некоторую совокупность ячеек карты графа, что позволяет составить систему уравнений

$$\begin{cases} C_1 = a_{11}S_1 \oplus \ldots \oplus a_{1n_F}S_{n_F}, \\ C_2 = a_{21}S_1 \oplus \ldots \oplus a_{2n_F}S_{n_F}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ C_{n_F} = a_{n_F1}S_1 \oplus \ldots \oplus a_{n_Fn_F}S_{n_F}, \end{cases} \tag{1}$$

где $a_{ij} \in \{0, 1\}$; коэффициент $a_{ij} = 1$, если цикл C_i покрывает ячейку S_j карты, иначе $a_{ij} = 0$. Для примера графа в разделе 8 на рис. 4 система уравнений (1) имеет вид:

$$\begin{array}{lll} C_1 = S_1, & C_2 = S_2 \oplus S_3 \oplus S_4 \oplus S_5 \oplus S_6, & C_3 = S_3 \oplus S_4 \oplus S_5 \oplus S_6, \\ C_4 = S_4 \oplus S_5, & C_5 = S_5, & C_6 = S_6. \end{array}$$

Циклы базиса $F = \{C_1, C_2, \ldots, C_{n_F}\}$ выражаются через циклы множества $S = \{S_1, S_2, \ldots, S_{n_F}\}$. Размерность линейного пространства является *инвариантом*, следовательно $|S| \geqslant |F|$. Покажем, что $|S| = |F|$. Из утверждения 4 имеем, что каждая ячейка S_v карты однозначно определяется соответствующим циклом C_v *DFS-базиса*. Поэтому, если допустить, что ячеек карты больше, чем циклов *DFS-базиса*, то два или более циклов таких ячеек должны будут определяться одним и тем же циклом *DFS-базиса*, а значит, они совпадают. Итак, циклы ячеек карты графа $S = \{S_1, S_2, \ldots, S_{n_F}\}$ составляют базис пространства L. Что и требовалось доказать.

Утверждение 6. *Для построения циклов S_v границ ячеек карты графа необходимо перебрать все циклы C_v DFS-базиса, удаляя по формуле $S_v = C_v \oplus J_{t_{v1}} \oplus J_{t_{v1}} \oplus \ldots \oplus J_{t_{vk_v}}$ из каждого вложенные в него циклы.*

Доказательство. Данное утверждение следует из утверждений 4 и 5 и составляет правило генерации циклов ячеек карты графа. Что и требовалось доказать.

Замечание 2. Предложенный конструктивный метод перехода от *DFS-базиса* циклов $C_1, C_2, \ldots, C_{n_F}$ к базису циклов ячеек карты графа $S_1, S_2, \ldots, S_{n_F}$ позволяет в явном виде записать решение системы

линейных уравнений (1):

$$\begin{cases} S_1 = C_1 \oplus J_{t_{11}} \oplus \ldots \oplus J_{t_{1k_1}}, \\ S_2 = C_2 \oplus J_{t_{21}} \oplus \ldots \oplus J_{t_{2k_2}}, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ S_{n_F} = C_{n_F} \oplus J_{t_{n_F1}} \oplus \ldots \oplus J_{t_{n_Fk_{n_F}}}, \end{cases} \tag{2}$$

где $J_{t_{ij}}$ — циклы слияния, вложенные в цикл C_i.

6. Формальный алгоритм выделения циклов ячеек карты графа. Вынесем из общего алгоритма 3 реализацию процедуры слияния циклов по формуле $S_v = C_v \oplus J_{t_{v1}} \oplus J_{t_{v2}} \oplus \ldots \oplus J_{t_{vk_v}}$ согласно утверждению 6, где C_v — произвольный цикл *DFS-базиса* и $J_{t_{v_j}}$ — циклы слияния, вложенные в цикл C_v и смежные с ним (см. замечание 1). Циклы слияния $J_{t_{v_j}}$ представляются смежными в совокупности циклами *DFS-базиса* $J_{t_{v_j}} = C_{t_{v_j1}} \oplus C_{t_{v_j2}} \oplus \ldots \oplus C_{t_{v_jk}}$. Сама процедура слияния $\mathrm{Join}(W, S_v)$ представлена в алгоритме 2. Параметрами процедуры выступают множество указанных циклов $W = \{C_v, J_{t_{v1}}, J_{t_{v2}}, \ldots, J_{t_{vk_v}}\}$ и результирующий цикл слияния S_v. Начальное значение цикла полагаем равным пустому циклу $S_v = O$. Чтобы результатом операции слияния $S_v = S_v \oplus C_w$, где $C_w \in W$, на каждом шаге был ровно один цикл, необходимо, чтобы цикл C_w был смежным с текущим циклом слияния S_v. Согласно утверждению 2, это возможно, если перебор $C_w \in W$ выполнить одним из обходов дерева смежности $J_w = (X_w, U_w)$ исходной совокупности циклов W. В данном случае воспользуемся обходом сверху–вниз. Процедура обхода сверху–вниз заимствована из работы [13], в основе которой лежит обход графа в глубину. В начале процедуры формируется структура смежности $\mathrm{Adj}[x]$ представления дерева смежности $J_w = (X_w, U_w)$. Рекурсивная процедура $\mathrm{Depth}(x)$ выполняет обход сверху–вниз поддерева с корнем в вершине $x \in X_w$. Результатом обхода дерева $J_w = (X_w, U_w)$ будет цикл слияния S_v — ячейка карты графа.

Алгоритм 2. Слияние множества W смежных в совокупности циклов

```
Procedure Join(W, S_v);
  X_w = ∅;
  for i = 1 to |W| do X_w = X_w ∪ {i}; {Номера циклов множества W}
  for C_x ∈ W do begin {Формирование Adj[x] дерева J_w = (X_w, U_w)}
    Adj[x] = ∅; {Начальный пустой список Adj[x]}
    for C_y ∈ W do begin
      if (x ≠ y) ∧ (C_x ∩ C_y ≠ ∅) then begin
        Adj[x] = Adj[x] ∪ {y}; {Расширить список Adj[x]}
      end;
    end;
  end;
  z ∈ X_w; {Начало обхода дерева смежности J_w = (X_w, U_w)}
  S_v = C_z; {Начальный цикл слияния}
  Depth(z); {Обход поддерева с корнем в z}
end.

Procedure Depth(x); {Обход сверху–вниз поддерева с корнем в x}
  for y ∈ Adj[x] do begin {Перебор смежных вершин}
    S_v = S_v ⊕ C_y; {Слияние циклов}
    Depth(y); {Обход поддерева с корнем в y}
  end;
end.
```

Оценим сложность алгоритма 2 слияния. Сложность обхода дерева является линейной: $O(X)$. Сложность формирования структуры смежности $\mathrm{Adj}[x]$ определяется двумя вложенными циклами и равна $O\big(|X|^2\big)$, так как для множества циклов W выполняется условие $|W| = O\big(|X|\big)$. Общая сложность составит $O\big(|X|^2\big)$. Отметим, что операции проверки $C_x \cap C_v \neq \varnothing$ и слияния $S_w = S_w \oplus C_v$ являются циклическими, так как в практической реализации циклы представляются векторами. Поэтому, если положить сложность их равной α, то сложность алгоритма 2 составит $O\big(\alpha|X|^2\big)$. Следуя утверждению 6, расчет циклов S_v ячеек карты графа выполняется по формуле $S_v = C_v \oplus J_{t_{v1}} \oplus J_{t_{v2}} \oplus \ldots \oplus J_{t_{vk_v}}$. Последовательность вычисления циклов S_v имеет существенное значение. В рассматриваемой задаче перебор циклов C_v наиболее оптимально выполнить обходом снизу–вверх корневых деревьев *DFS-базиса* (см. раздел 4). Суть

этого обхода составляет процедура, заимствованная из работы [13]. В этом случае обработке цикла C_v будет предшествовать перебор вложенных в него циклов, что позволит предварительно сформировать все его циклы слияния $J_{t_{vj}}$. Данный подход реализован в алгоритме 3 формирования циклов ячеек карты графа.

Замечание 3. В алгоритме 3 граф циклов $G_F = (X_F, U_F)$ задается структурой смежности Adj[x], формирование ее рассматривалось в алгоритме 1. Корневая же структура деревьев графа G_F должна быть сохранена и в структуре смежности Adj[x]. При обходе графа G_F это будет выполнено, если в перечне X_F корни деревьев будут встречаться раньше своих вершин.

Чтобы отличить уже пройденные вершины, вводится вектор меток вершин Mark[x], значения которых равно 1 для пройденных вершин и 0 — для непройденных. Цикл C_x и вложенные в него циклы формируют множество W смежных в совокупности циклов. Слияние циклов множества W процедурой Join(W, S_x) дает цикл S_x ячейки карты графа. Все выделенные циклы S_x составят искомое множество $S = \{S_1, S_2, \ldots, S_{n_F}\}$ ячеек карты графа.

Алгоритм 3. Формирование множества S циклов ячеек карты графа
$S = \varnothing$; {Начальное пустое множество ячеек карты графа}
for $v \in X_F$ **do** Mark[v] = 0; {Метки не пройденных вершин}
for $v \in X_F$ **do if** Mark[v] = 0 **then begin** {Поиск начала обхода}
  Root(v); {Обход корневого дерева с корнем в вершине v}
end;

Procedure Root(x); {Обход дерева снизу–вверх с корнем в x}
  Mark[x] = 1; {Вершина пройдена}
  $W = \varnothing$; {Начальное пустое множество циклов для C_x}
  for $v \in$ Adj[x] **do begin** {Перебор смежных вершин}
    if Mark[v] = 0 **then begin** {Поиск не пройденной вершины}
      Root(v); {Обход поддерева с корнем в v}
      $W = W \cup \{C_v\}$; {Добавить вложенный цикл C_v}
    end;
  end;
  $W = W \cup \{C_x\}$; {Добавить цикл C_x — корень дерева}
  Join(W, S_x); {S_x — результат слияния циклов множества W}
  $S = S \cup \{S_x\}$; {Добавить цикл ячейки S_x к карте S графа}
end.

Сложность алгоритма 3 — это сложность обхода дерева, которая является линейной относительно $|X|$. Однако для каждой вершины выполняется процедура слияния Join(W, S_x), сложность которой $O\big(\alpha|X|^2\big)$. Следовательно, сложность предложенного алгоритма генерации циклов ячеек карты простого планарного графа является кубической функцией $O\big(\alpha|X|^3\big)$.

7. Определение данных корневой структуры графа циклов. Рассмотрим формирование начальных данных для расчета циклов ячеек карты графа по алгоритму 3. Для представления графа циклов $G_F = (X_F, U_F)$ необходимо сформировать его реберный список (см. раздел 4) и выполнить разложение множества циклов $X_F = \{C_1, C_2, \ldots, C_{n_F}\}$ на компоненты связности $X_F = D_1 \cup D_2 \cup \ldots \cup D_{n_D}$ корневых деревьев.

В соответствии с замечанием 3 требуется перенумерация вершин множества X_F, которая должна поддерживать корневую структуру деревьев графа G_F, заданного структурой смежности Adj[x]. В нашем случае это будет выполняться, если в перечне вершин X_F корни деревьев будут встречаться раньше своих вершин.

Данными для практической реализации обозначенных задач являются циклы $F = \{C_1, C_2, \ldots, C_{n_F}\}$ *DFS-базиса* исходного графа $G = (X, U)$. Как сказано выше в разделе 3, циклы базиса вычисляются по алгоритму, представленному в работе [18, с. 382]. Геометрически циклы C_i являются замкнутыми векторами точек на плоскости без самопересечений. Укладка ребер на плоскости определяет карту графа.

Для описания алгоритма 4 формирования начальных данных введем дополнительные обозначения. В векторе $\mu_1, \mu_2, \ldots, \mu_{n_F}$ будем хранить площади, заключенные в циклах C_i. Выражение $\mu_i \cap \mu_j$ обозначает площадь, заключенную в области пересечения циклов C_i и C_j. Для вычисления μ_i и пересечения $\mu_i \cap \mu_j$ воспользуемся эффективной практической процедурой из работы [25].

Разложение графа циклов $G_F = (X_F, U_F)$ на компоненты связности будем фиксировать в векторе

$D_1, D_2, \ldots, D_{n_F}$. Значение элемента D_i — это номер компоненты связности, которой принадлежит цикл C_i. Количество компонент связности равно числу различных значений среди элементов $D_1, D_2, \ldots, D_{n_F}$.

Алгоритм 4. Формирование корневой структуры графа циклов G_F

```
for i = 1 to n_F do begin {Начальная установка данных}
  D_i = 0; {Номера компонент связности размещения циклов}
  μ_i = μ(C_i); {Вычисление площадей циклов}
  R_i = 0; {Значения счетчиков вложенности циклов}
  Q_i = 0; {Номера наименьших охватывающих циклов}
  μQ_i = +∞; {Начальные площади наименьших охватывающих циклов}
end;
count = 0; {Число компонент связности}

for k = 1 to n_F do begin {Перебор циклов C_k}
  if D_k = 0 then begin {Начать новую компоненту связности}
    count = count + 1; {Увеличить номер новой компоненты связности}
    D_k = count; {Отнести цикл C_k к новой компоненте D_k с номером count}
  end;
  for j = k + 1 to n_F do begin {Перебор циклов C_j}
    if μ_k ∩ μ_j ≠ 0 then begin {Вложенность циклов C_k и C_j}
      if D_k ≠ D_j then begin {Слияние компонент связности D_k = D_k ∪ D_j}
        w = D_j; {Переназначить циклы компоненты D_j}
        if w ≠ 0 then for i = 1 to n_F do if D_i = w then D_i = D_k;
        D_j = D_k;
      end;
      if μ_k < μ_j then begin {Цикл C_k вложен в C_j}
        R_k = R_k + 1; {Счетчики вложенности циклов C_k ⊂ C_j}
        if μQ_k > μ_j then begin {Уточнить соседний сверху цикл для C_k}
          μQ_k = μ_j; {Новая площадь цикла сверху}
          Q_k = j; {Новый цикл сверху для C_k}
        end;
      end
      else begin {Цикл C_j вложен в C_k}
        R_j = R_j + 1; {Счетчики вложенности циклов C_k ⊃ C_j}
        if μQ_j > μ_k then begin {Уточнить соседний сверху цикл для C_j}
          μQ_j = μ_k; {Новая площадь цикла сверху}
          Q_j = k; {Новый цикл сверху для C_j}
        end;
      end;
    end;
  end;
end.
```

В рамках каждой компоненты связности вычисляются: (i) номера минимальных охватывающих циклов $Q_1, Q_2, \ldots, Q_{n_F}$, т.е. формируется реберный список графа циклов $G_F = (X_F, U_F)$, и (ii) уровни вложенности циклов $R_1, R_2, \ldots, R_{n_F}$. Значение $Q_i = j$ определяет в графе ориентированное ребро $C_i \to C_j$ или $i \to j$. Значение R_i показывает уровень вложенности цикла C_i в соответствующем корневом поддереве (компоненте связности). Если $R_i = 0$, то цикл C_i является корнем поддерева.

Вычисленный реберный список $Q_1, Q_2, \ldots, Q_{n_F}$ задания графа $G_F = (X_F, U_F)$ определяет корневую структуру его компонент связности (поддеревьев). Сортировка (перенумерация) множества вершин X_F графа в порядке возрастания их уровней вложенности $R_1, R_2, \ldots, R_{n_F}$ гарантирует, что корни деревьев в списке X_F будут встречаться раньше своих вершин.

Сложность формирования реберного списка определяется тремя вложенными циклами размерности $n_F = O\big(|X|\big)$ и равна $O\big(|X|^3\big)$. Сложность алгоритма 3 формирования циклов ячеек карты графа составляет $O\big(\alpha|X|^3\big)$, а значит, это и общая сложность предложенного алгоритма генерации циклов ячеек карты графа.

8. Численный пример. Рассмотрим на примере простого планарного графа, представленного на рис. 4, последовательность шагов вычисления циклов S_1, S_2, S_3, S_4, S_5, S_6 ячеек карты графа. Установлен-

ная на рис. 4a нумерация циклов S_i определяется нумерацией циклов C_1, C_2, C_3, C_4, C_5, C_6 *DFS-базиса*. Пусть исходный граф задается следующей структурой смежности Adj$[x]$:

x	1	2	3	4	5	6	7	8	9	10	11	12
Adj$[x]$	$2,4,11$	$3,1,6,12$	$4,2,5$	$1,3$	$3,6$	$5,7,2$	$6,8,12$	$7,9$	$8,10,12$	$9,11$	$1,10,12$	$2,7,9,11$

Согласно разделу 3, вычисляем фундаментальное множество циклов *DFS-базиса*. Циклы нумеруются по мере их генерации:

$$C_1=\{1,2,3,4,1\}, \quad C_2=\{1,2,3,5,6,7,8,9,10,11,1\}, \quad C_3=\{2,3,5,6,7,8,9,10,11,12,2\},$$
$$C_4=\{7,8,9,10,11,12,7\}, \quad C_5=\{9,10,11,12,9\}, \quad C_6=\{2,3,5,6,2\}.$$

Структура вложенности данного множества циклов такова: $C_5 \subset C_4$; $C_4 \subset C_3$; $C_6 \subset C_3$; $C_3 \subset C_2$; C_1. На рис. 4b показана геометрия их вложенности. На рис. 4c представлен граф циклов G_F и его разложение на компоненты связности $G_F = D_1 \cup D_2$. Каждая компонента связности D_i представляется своим корневым деревом. Реберный список представления корневой структуры графа G_F:

$$Q_1=0, \quad Q_2=0, \quad Q_3=2, \quad Q_4=3, \quad Q_5=4, \quad Q_6=3.$$

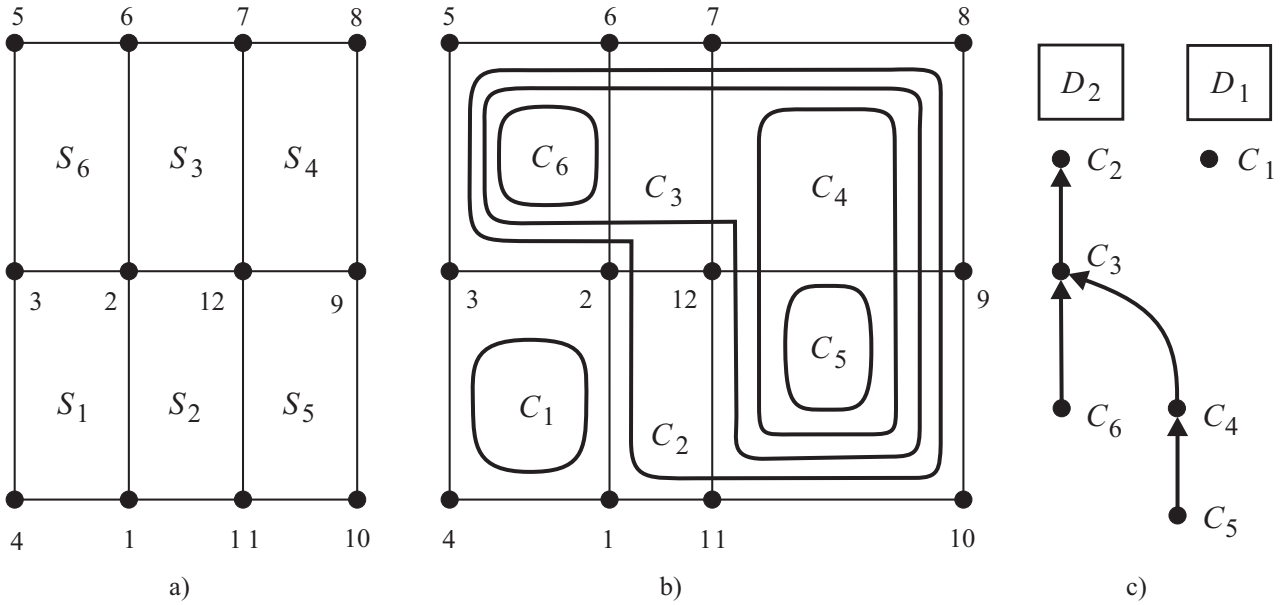

Рис. 4. Пример простого планарного графа

Запишем для данного графа систему уравнений (1):

$$C_1=S_1, \quad C_2=S_2\oplus S_3\oplus S_4\oplus S_5\oplus S_6, \quad C_3=S_3\oplus S_4\oplus S_5\oplus S_6,$$
$$C_4=S_4\oplus S_5, \quad C_5=S_5, \quad C_6=S_6.$$

Для каждого цикла C_k строим вложенные в него циклы слияния J_{k_i}: $J_1 = O$ — нет вложенных циклов; $J_2 = C_3$ — один вложенный цикл; $J_3 = J_{31} \oplus J_{32} = C_4 \oplus C_6$ — два вложенных несмежных цикла; $J_4 = C_5$ — один вложенный цикл; $J_5 = O$ — нет вложенных циклов; $J_6 = O$ — нет вложенных циклов.

Формирование циклов ячеек карты графа по формулам (2):

$$S_1=C_1\oplus J_1=C_1\oplus O=C_1; \quad S_2=C_2\oplus J_2=C_2\oplus C_3; \quad S_3=C_3\oplus J_3=C_3\oplus C_4\oplus C_6;$$
$$S_4=C_4\oplus J_4=C_4\oplus C_5; \quad S_5=C_5\oplus J_5=C_5\oplus O=C_5; \quad S_6=C_6\oplus J_6=C_6\oplus O=C_6.$$

9. Заключение. В настоящей статье предложен конструктивный метод генерации циклов ячеек карты простого планарного графа. Карта планарного графа позволила отойти от общепринятого подхода генерации циклов и внести геометрию карты в структуру алгоритма. Именно геометрия циклов *DFS-базиса* дала возможность определить на этом множестве отношение соседства, которое и привело к корневому

дереву структуры вложенности циклов. Генерация циклов ячеек карты графа стала результатом обхода указанного корневого дерева.

Формализованная часть алгоритма и его структуры данных рассмотрены на уровне, близком к практическому их использованию. Важным приложением рассмотренного алгоритма может служить использование его в задачах локализации данных на планарном подразбиении. Обзор такого рода задач можно найти в работе [2]. Раскраска территорий (см. введение) одна из задач такого класса.

Сложность алгоритма генерации циклов ячеек карты графа составляет $O\big(\alpha|X|^3\big)$, что равносильно сложности генерации циклов *DFS-базиса*. Данное обстоятельство позволяет надеяться на использование базиса построенных циклов ячеек карты графа наравне с циклами *DFS-базиса*.

СПИСОК ЛИТЕРАТУРЫ

1. *Иванов Б.Н.* Решение задачи расчета оптимальных маршрутов судов в рамках геоинформационной системы “ОКЕАН” // Вычислительные методы и программирование. 2012. **13**. 226–234.
2. *Препарата Ф., Шеймос М.* Вычислительная геометрия: Введение. М.: Мир, 1989.
3. *Оре О.* Теория графов. М.: Наука, 1980.
4. *Роджерс К.А.* Укладки и покрытия. М.: Мир, 1968.
5. *Делоне Б.Н.* О пустом шаре // Известия АН СССР. 1934. VII серия, № 6. 793–800.
6. *Sokhn N., Baltensperger R., Bersier L.F., Hennebert J., Ultes-Nitsche U.* Identification of chordless cycles in ecological networks // Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Vol. 126. Berlin: Springer, 2013. 316–324.
7. *Pfaltz J.L.* Chordless cycles in networks // IEEE 29th International Conference on Data Engineering Workshops (ICDEW). New York: IEEE Press, 2013. 223–228.
8. *Welch J.* A mechanical analysis of the cyclic structure of undirected linear graphs // J. Assoc. Comput. Mech. 1966. **13**. 205–210.
9. *Gibbs N.W.* A cycle generation algorithm for finite undirected linear graphs // J. Assoc. Comput. Mech. 1969. **16**. 564–568.
10. *Tarjan R.* Enumeration of the elementary circuits of a directed graph // SIAM J. Comput. 1973. **2**, N 3. 211–216.
11. *Jonson D.B.* Finding all the elementary circuits of a directed graph // SIAM J. Comput. 1975. **4**, N 1. 77–84.
12. *Mateti P., Deo N.* On algorithms for enumerating all circuits of a graph // SIAM J. Comput. 1976. **5**, N 1. 90–99.
13. *Tarjan R.E.* Depth-first search linear graph algorithms // SIAM J. Comput. 1972. **1**. 146–160.
14. *Syslo M.M.* An efficient cycle vector space algorithm for listing all cycles of a planar graph // SIAM J. Comput. 1981. **10**, N 4. 797–808.
15. *Wild M.* Generating all cycles, chordless cycles, and Hamiltonian cycles with the principle of exclusion // Journal of Discrete Algorithms. 2008. **6**. 93–102.
16. *Mahdi F., Safar M., Mahdi K.* Detecting cycles in graphs using parallel capabilities of GPU // Digital Information and Communication Technology and Its Applications. Vol. 167. Berlin: Springer, 2011. 193—205.
17. *Kavitha T., Mehlhorn K., Michail D.* New approximation algorithms for minimum cycle bases of graphs // Algorithmica. 2011. **59**, N 4. 471–488.
18. *Рейнголд Э., Нивергельд Ю., Део Н.* Комбинаторные алгоритмы. Теория и практика. М.: Мир, 1980.
19. *Курош А.Г.* Лекции по общей алгебре. М.: Наука, 1973.
20. *Алексеев В.Е., Таланов В.А.* Графы. Модели вычислений. Структуры. Нижний Новгород: Изд-во ННГУ, 2005.
21. *Dogrusoz U., Krishnamoorthy M.S.* Enumerating all cycles of a planar graph // Parallel Algorithms Appl. 1996. **10**, N 1/2. 21–36.
22. *Deo N., Prabhu G.M., Krishnamoorthy M.S.* Algorithms for generating fundamental cycles in a graph // ACM Trans. Math. Software. 1982. **8**, N 1. 26–42.
23. *Иванов Б.Н.* Дискретная математика. Алгоритмы и программы. М.: Известия, 2011.
24. *Paton K.* An algorithm for finding a fundamental set of cycles of a graph // Comm. ACM. 1969. **12**, N 9. 514–518.
25. *Иванов Б.Н.* Структуры вложенности поля изолиний в задаче градиентного заполнения // Вычислительные методы и программирование. 2006. **7**, № 1. 155–165.

Поступила в редакцию
05.04.2014

Generation of Cycles of Map Cells for a Simple Planar Graph

B. N. Ivanov [1]

[1] *Far Eastern Federal University, School of Natural Sciences; ulitsa Sukhanova 8, Vladivostok, 690950, Russia; Associate Professor, e-mail: ibn8826@mail.ru*

Received April 5, 2014

Abstract: A constructive method for generating the cycles of map cells of a simple planar graph is considered. These cycles are represented as a linear combination of DFS-basis cycles. The sought linear combinations are constructed explicitly on the basis of the allocated properties of the nesting structure of DFS-basis cycles and the map cells cycles of the graph. The map of a planar graph allows one to avoid the traditional approach used to generate such cycles and allows one to take into account the geometry of the map in the algorithm. The relation of neighborhood is defined on the set of the cycles, which induces the rooted tree of the nested cycle structure. The cells of the graph map are the result of traversal of the rooted tree. The complexity of the proposed algorithm is cubic relative to the number of the graph vertices. The application of this algorithm to solving the problems of planar subdivision is discussed.

Keywords: generation of cycles, enumeration of cycles, basis of cycles, graph map, nesting of cycles, fundamental cycles, chordless cycles.

References

1. B. N. Ivanov, "Solution of the Optimal Ship Route Problem in the Framework of the OKEAN Geoinformation System," Vychisl. Metody Programm. **13**, 226–234 (2012).

2. F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction* (Springer, Heidelberg, 1985; Mir, Moscow, 1989).

3. O. Ore, *Theory of Graphs* (AMS Press, Providence, 1962; Nauka, Moscow, 1980).

4. C. A. Rogers, *Packing and Covering* (Cambridge Univ. Press, Cambridge, 1964; Mir, Moscow, 1968).

5. B. N. Delaunay, "On an Empty Sphere," Izv. Akad. Nauk SSSR, Ser. VII, No. 6, 793–800 (1934).

6. N. Sokhn, R. Baltensperger, L.-F. Bersier, et al., "Identification of Chordless Cycles in Ecological Networks," in *Lecture Notes of the Institute for Computer Sciences: Social Informatics and Telecommunications Engineering* (Springer, Berlin, 2013), Vol. 126, pp. 316–324.

7. J. L. Pfaltz, "Chordless Cycles in Networks," in *Proc. IEEE 29th Int. Conf. on Data Engineering Workshops* (IEEE Press, New York, 2013), pp. 223–228.

8. J. Welch, "A Mechanical Analysis of the Cyclic Structure of Undirected Linear Graphs," J. Assoc. Comput. Mech. **13**, 205–210 (1966).

9. N. W. Gibbs, "A Cycle Generation Algorithm for Finite Undirected Linear Graphs," J. Assoc. Comput. Mech. **16**, 564–568 (1969).

10. R. Tarjan, "Enumeration of the Elementary Circuits of a Directed Graph," SIAM J. Comput. **2** (3), 211–216 (1973).

11. D. B. Jonson, "Finding All the Elementary Circuits of a Directed Graph," SIAM J. Comput. **4** (1), 77–84 (1975).

12. P. Mateti and N. Deo, "On Algorithms for Enumerating All Circuits of a Graph," SIAM J. Comput. **5** (1), 90–99 (1976).

13. R. Tarjan, "Depth-First Search Linear Graph Algorithms," SIAM J. Comput. **1** (2), 146–160 (1972).

14. M. M. Syslo, "An Efficient Cycle Vector Space Algorithm for Listing All Cycles of a Planar Graph," SIAM J. Comput. **10** (4), 797–808 (1981).

15. M. Wild, "Generating All Cycles, Chordless Cycles, and Hamiltonian Cycles with the Principle of Exclusion," J. Discrete Algorithms **6** (1), 93–102 (2008).

16. F. Mahdi, M. Safar, and K. Mahdi, "Detecting Cycles in Graphs Using Parallel Capabilities of GPU," in *Digital Information and Communication Technology and Its Applications* (Springer, Berlin, 2011), Vol. 167, pp. 193–205.

17. T. Kavitha, K. Mehlhorn, and D. Michail, "New Approximation Algorithms for Minimum Cycle Bases of Graphs," Algorithmica **59** (4), 471–488 (2011).

18. E. M. Reingold, J. Nievergelt, and N. Deo, *Combinatorial Algorithms. Theory and Practice* (Prentice-Hall, Englewood Cliffs, 1977; Mir, Moscow, 1980).

19. A. G. Kurosh, *Lectures on General Algebra* (Fizmatlit, Moscow, 1960; Chelsea, New York, 1963).

20. B. E. Alekseev and V. A. Talanov, *Graphs. Computational Models. Structures* (Lobachevsky State Univ. of Nizhni Novgorod, Nizhni Novgorod, 2005) [in Russian].

21. U. Dogrusoz and M. S. Krishnamoorthy, "Enumerating All Cycles of a Planar Graph," Parallel Algorithms Appl. **10** (1/2), 21–36 (1996).

22. N. Deo, G. M. Prabhu, and M. S. Krishnamoorthy, "Algorithms for Generating Fundamental Cycles in a Graph," ACM Trans. Math. Softw. **8** (1), 26–42 (1982).

23. B. N. Ivanov, *Discrete Mathematics. Algorithms and Software* (Izvestiya, Moscow, 2011) [in Russian].

24. K. Paton, "An Algorithm for Finding a Fundamental Set of Cycles of a Graph," Commun. ACM **12** (9), 514–518 (1969).

25. B. N. Ivanov, "Enclosure Structures of Equal Level Line Fields in the Gradient Filling Problem," Vychisl. Metody Programm. **7** (2), 30–40 (2006).