УДК 519.5

©2011. **Е. А. Татаринов**

СЛОЖНОСТЬ ВОССТАНОВЛЕНИЯ ГРАФОВ, ЯВЛЯЮЩИХСЯ КВАЗИКОЛЬЦАМИ И КВАЗИДЕРЕВЬЯМИ

Анализируются модификации алгоритма восстановления графа агентом, перемещающимся по его ребрам, считывающим и изменяющим метки на элементах графа. Найдены и исследованы операции над графами. Результирующий граф этих операций восстанавливается с использованием числа камней, выражающееся через сумму числа камней, необходимых для восстановления исходных компонент.

Ключевые слова: *восстановление графов, агент, сложность алгоритма, блуждание по графу.*

1. Введение. В настоящее время интенсивно развивается одно из направлений математической кибернетики – теория дискретных динамических систем [1-3]. В общей схеме Глушкова-Летичевского дискретная система представляется как модель взаимодействия управляющей и управляемой систем (управляющего автомата и операционной среды) [4]. Взаимодействие этих систем зачастую представляется как процесс перемещения управляющего автомата по графу (лабиринту) управляемой системы [1], что привело к обширной и интенсивно развивающейся области исследования поведения автоматов в лабиринтах [1-3].

Общая проблема анализа графа включает в себя ряд частных проблем, важнейшими из которых являются: проблема самолокализации, т.е. определения вершины графа, в которой первоначально находится автомат, проблему контроля графа (карты), т.е. проверки изоморфизма исследуемого графа и заданного графа – эталона (карты), и проблему полного восстановления графа. При этом автомат перемещается по дугам графа от вершины к вершине и воспринимает некоторую локальную информацию о нем и может отмечать элементы графа специальными метками (красками и / или камнями).

В данной работе рассматривается задача распознавания конечного, неориентированного графа, без петель и кратных ребер при помощи агента, который перемещается по нему и изменяет метки на вершинах и ребрах графа при помощи красок и камней [5].

Ранее в работе [6] проводился анализ временной сложности восстановления графа, полученного из нескольких графов путем применения к ним операций сочленения. Данная работа посвящена анализу числа камней, используемых для восстановления такого графа. При этом предполагается, что агент выполняет модификации алгоритма (далее будем обозначать его A) и три его эвристики, предложенные в [7], (далее будем называть их A_1, A_2, A_3, соответственно), который является модификацией "Базового алгоритма" [8]. Найдены оценки числа камней, используемых для восстановления графов из классов квазикольца и квазидеревья.

2. Необходимые определения и понятия. Рассматриваются конечные, неори-

ентированные графы без петель и кратных ребер. Все такие графы объединим в класс K. Все неопределяемые понятия общеизвестны и их можно найти, например, в [9-11]. Пусть $G = (V, E)$ – граф, у которого V – множество вершин, E – множество ребер, $E \subseteq V \times V$, мощности n и m, соответственно. Тройку $((u, v), v)$ будем называть инцидентором ("точкой прикосновения") ребра (u, v) и вершины v. Множество таких троек обозначим I. Множество $L = V \cup E \cup I$ назовем множеством элементов графа G. Окрестностью $O(v)$ вершины v будем называть множество элементов графа, состоящее из вершины v, всех вершин u смежных с v, всех ребер (v, u) и всех инцидекторов $((v, u), v), ((v, u), u)$.

Последовательность $u_1, u_2, .., u_k$ попарно смежных вершин, где все вершины различны, будем называть путем длины k в графе G. При $u_1 = u_k$ этот путь называется циклом. Краской будем называть ресурс, который имеется в неограниченном количестве, а камнем – ограниченный ресурс, все камни попарно различны и перенумерованы. Камень устанавливается в вершине и может быть снят с нее, а краской метят вершины и инциденторы ребер. Краска в отличие от камня может быть только заменена другой краской. При этом краска, на вершине в которой установлен камень, становится невидимой до тех пор, пока камень не будет снят.

Для каждого камня существует счетчик количества непомеченных ребер, которые есть в окрестности текущей вершины. Счетчик камня уменьшается на единицу, когда агент "видит" в окрестности текущей вершины вершину, помеченную этим камнем. После обнуления счетчика камня этот камень станет свободным, агент перейдет в вершину, снимет камень, вершину пометит краской b и возвращается обратно. Агент использует для уставновки в вершине свободный на данный момент камень наименьшим номером. С камнем ассоциируется M-номер [11], который неявно был присвоен вершине в момент установки в ней данного камня. Такая ассоциация сохраняется до тех пор, пока камень не станет свободным.

Пару (G, v), где G – граф, а v – его вершина, назовем правильной, если удаление вершины v и всех инцидентных ей ребер не нарушает связности полученного графа G. То есть v не является точкой сочленения в смысле [12]. В противном случае пару назовем неправильной.

Сочленение двух графов [6] осуществляется отождествлением двух их вершин. В графах G_1 и G_2 выбираются соответственно вершины v_1, v_2 и отождествляются в одну вершину v. Если в результате отождествления образовались кратные ребра или петли, кратные ребра заменяются одним, а петли – удаляются из результирующего графа. Вершину v назовем вершиной сочленения. Сочленние графов G_1 и G_2 будем обозначать $G_1 \bigodot G_2$. Подграфы $G_1 \bigodot G_2$ изоморфные графам G_1 и G_2 будут компонентами сочленения.

Пусть задан класс $K' \subseteq K$, рассмотрим класс T^k графов, полученных из G_1, .., $G_k \in K'$ последовательным применением сочленения к $G_1, .., G_k$:

$H_1 = G_1 \bigodot G_2$;

$H_2 = H_1 \bigodot G_3$;

$G = H_{k-1} \bigodot G_k$.

Граф G назовем квазидеревом, если он не содержит простых циклов, которые

содержат одновременно вершины из G_i и G_j, $i \neq j$.

Пусть задан класс $K' \subseteq K$, рассмотрим класс S^k графов, полученных из $G_1, .., G_k \in K'$ последовательным применение операции сочленения к $G_1, .., G_k$ и отождествлением двух вершин компонент G_1 и G_k:

$H_1 = G_1 \bigodot G_2$;

$H_2 = H_1 \bigodot G_3$, при этом в графе H_1 вершина для отождествления выбирается из комопненты G_2;

$H_{k-1} = H_{k-2} \bigodot G_k$, при этом в графе H_{k-2} вершина для отождествления выбирается из комопненты G_{k-1};

Результирующий граф G получается отождествлением $G_1 \bigodot G_k$. Будем называть квазикольцом.

Связный ациклический граф, степень вершин которого не более двух, будем называть линией. Ясно, что степень один могут иметь только две его вершины, которые будем называть концами линии.

Будем говорить, что граф принадлежит классу M_c^p, если при его восстановлении агент использует p – камней и c – красок.

3. Постановка задачи. Пусть задан класс K графов: конечных, неориентированных, без петель и кратных ребер, – все элементы которых окрашены краской w. Задан агент,который может перемещаться по исследуемому графу из вершины по ребру, соединяющему их, и изменять метки на элементах графа так, как это было описано в пункте 2. Агент помещается в произвольную вершину априори неизвестного ему графа $G \in K$.

Агенту требуется восставновить граф G, т. е. построить граф H, изоморфный графу G, с точностью до меток на элементах графов.

4. Стратегия восстановления графа с использованием камней. Алгоритм A восстановления графа при помощи камней подробно описан в [7]. Алгоритм A основан на методе обхода графа в глубину [9, 10]. Этот метод модифицирован с учетом того, что агенту граф изначально неизвестен, агент может воспринимать локальную информацию о графе и агент строит на вершинах графа нумерацию в порядке посещения вершин (M – нумерация)[11]. Агент реализует нумерацию неявно, при помощи набора попарно различных камней.

В процессе обхода графа в глубину агент строит неявное дерево поиска в глубину, относительно которого ребра графа разбиваются на два множества: древесные и обратные. Древесные ребра проходятся два раза – в прямом (процедура *ВПЕРЕД*(v)) и обратном направлении (процедура *НАЗАД*(v)). При первом прохождении по древесному ребру оно восстанавливается в графе G и один из его инцидентов метиться краской r. После перехода по ребру агент устанавливает в новой непомеченной вершине свободный камень и восстанавливает все обратные ребра текущей вершины. Для восстановления обратных ребер агенту не требуется переходить по ним, поскольку, находясь в текущей вершине, помеченной некоторым камнем, он увидит камень, установленный в вершине, с которой смежно восстанавливаемое обратное ребро. Зная эти два камня, агент может однозначно определить ассоциируемые с ними неявные M-номера и восстановить обратное ребро. При установке в вершине

камня она добавляется в помеченный путь, при пометке ее краской b она удаляется из помеченного пути.

Алгоритм останавливается, когда помеченный путь становиться пустым, а все вершины помечены краской b.

Данный алгоритм можно модифицировать с целью уменьшения количества камней, используемых при восстановлении графа или же сократить время анализа окрестности вершины.

Первая модификация, реализуется алгоритмом A_1.Она заключается в том, что агент анализирует вершину, в которой собирается установить камень. Выделяется два вида вершин, для которых нет необходимости использовать камни: 1) вершины из которых не видно непомеченных вершин; 2) вершины, из которых видно только одну непомеченную вершину.

Вторая модификация реализуется алгоритмом A_2. Она заключается в том, что для увеличения области восприятия агента, т. е. метятся камнями все вершины из окрестности текущей вершины. При этом часть камней будет освобождаться, так как за один шаг будет исследовано большее количество вершин.

Третья модификация реализуется алгоритмом A_3. Она заключается в том, что камни, установленные в вершинах со степенью, большей чем D, не снимаются. D – заранее известная константа. При этом агент анализирует только вершины со степенью, меньше либо равной D.

5. Восстановление квазидеревьев и квазиколец, полученных при помощи сочленения. Сформулируем результаты для алгоритма из [7], аналогичные результатам, полученным в [6]. А именно, покажем, что при восстановлении графов из классов T^k и S^k число камней, есть сумма числа камней используемых для восстановления его компонент.

Пусть G получен сочленением двух правильных пар. Покажем, что при восстановлении G агент либо сначала восстанавливает одну компоненту, а затем другую, либо восстанавливает часть первой компоненты, переходит в другую компоненту и не возвращается в первую до тех пор, пока вторая не будет полностью восстановлена.

Лемма 1. *Пусть изначально агент помещается в вершину v графа $G \in K$ такую, что (G, v) – правильна пара. Тогда при выполнении агентом алгоритма $A, A_1 - A_3$, агент перейдет в вершину v после того, как восстановит все вершины графа G.*

Доказательство. Доказательство от противного. Предположим, что агент восстановил часть графа G и вернулся в вершину v, при выполнении процедуры НАЗАД(v). При этом из вершины v агент может попасть в ранее не посещенные вершины. Так как (G, v) – правильная пара, то в силу связности $(G - v)$ из v существует путь в некоторую вершину u, которая смежна, хотя бы с одной из ранее посещенных вершин x, которая отлична от v. Это противоречит тому, что агент придерживается стратегии обхода графа в глубину. Поскольку при такой стратегии агент из вершины x перешел бы в вершину u, а не наоборот. Полученное противоречие доказывает лемму. □

Теперь покажем, что число камней, используемых для восстановления графа G, будет равно сумме числа камней, используемых для восстановления каждой из его компонент.

Лемма 2. *Пусть* (G_1, v_1) *и* (G_2, v_2) *– правильные пары и* $G_1 \in M_2^{p_1}$, $G_2 \in M_2^{p_2}$, *тогда* G, *полученный сочленением вершин* v_1 *и* v_2 *этих двух графов, принадлежит классу* M_2^p, *где* $p = p_1 + p_2$.

Доказательство. Рассмотрим всевозможные варианты начала выполнения агентом восстановления G. Пусть u_1 вершина сочленения G_1 и G_2. Существует три различных начальных положения агента:

1) агент попал в вершину u_1;

2) агент попал в вершину $v_1 \in V(G_1) \setminus \{u_1\}$;

3) агент попал в вершину $v_2 \in V(G_2) \setminus \{u_1\}$.

Рассмотрим эти случаи.

1. Агент метит вершину u_1 камнем номер 1 и добавляет ее в помеченный путь. После чего он начинает восстанавливать компоненту G_1, используя камни, учтенные при подсчете p_1. После этого агент, либо попадет в вершину $v_1' \in V(G_1) \setminus \{u_1\}$, либо в вершину $v_2' \in V(G_2) \setminus \{u_1\}$. Предположим, что он попал в v_1' (v_2'). Далее агент будет восстанавливать вершины графа G и ребра G_1 (G_2). Так как (G_1, v_1) $((G_2, v_2))$ правильная пара, то по лемме 1 агент не уйдет из компоненты G_1 (G_2), пока полностью ее не восстановит. При этом он будет использовать камни, которые были учтены при подсчете p_1 (p_2). Следовательно, агенту потребуется $p_1 + p_2$ камней.

2. Агент пометит вершину v_1 камнем номер 1 и добавит в помеченный путь. После чего он начинает восстанавливать вершины и ребра компоненты G_1. В этом случае существует два различных варианта дальнейшего выполнения алгоритма.

2.1. Агент восстановит полностью компоненту G_1, используя при этом p_1 камней, и только после этого попадет в вершину сочленения u_1, пометит ее камнем, который уже был учтен при подсчете p_2. После чего он восстановит все вершины и ребра компоненты с G_2, используя камни, учтенные при подсчете p_2. Следовательно, агенту потребуется $p_1 + p_2$ камня.

2.2. Агент восстановит часть компоненты G_1, используя при этом p_1 камней. После чего он попадет в вершину сочленения u_1, пометит ее камнем i. После чего агент переходит в компоненту G_2. Так как (G_2, v_2) правильная пара, то по лемме 1 он полностью его восстанавливает, используя p_2 камней. При этом камень номер i уже был учтен при подсчете p_1. Даже если этот камень будет не свободным при восстановлении компоненты G_2, то это не увеличит число камней p_2, которые агент использует для ее восстановления. После восстановления компоненты G_2 агент возвращается в u_1 и восстанавливает оставшуюся часть G_1, используя p_1 камней. Следовательно, агенту потребуется $p_1 + p_2$ камней.

3. Этот случай полностью аналогичен случаю 2. Воспользуемся рассуждениями из пункта 2. □

Леммы 1, 2 обобщим на случай сочленения k правильных пар.

Теорема 1. *Если граф* G *получен последовательным отождествлением в одну*

вершин $v_1,..,v_k$ *правильных пар* $(G_1,v_1),..,(G_k,v_k)$, *где* $G_i \in M_2^{p_i}$, $i = 1,..,k$, *то* $G \in M_2^p$, *где* $p = \sum_{i=1}^{k} p_i$. *При этом порядок восстановления компонент не имеет значения.*

Доказательство. Пусть v вершина сочленения всех правильных пар. Если агент помещен в вершину v, то он, согласно лемме 1, перейдет в эту вершину v только после того, как полностью восстановит одну из компонент. Следовательно, агент начнет последовательно восстанавливать компоненты G_i, используя p_i камней, где $i = 1,..,k$. Тогда согласно лемме 2 $G \in M_2^p$, где $p = \sum_{i=1}^{k} p_i$ вне зависимости от порядка восстановления компонент.

Если же агент будет помещен в компоненту G_j, в вершину отличную от вершины v, то возможны два варианта выполнения агентом восстановления графа: 1) агент полностью восстановит компоненту G_j и переходит в вершину сочленения v; 2) агент восстанавливает часть компоненты G_j, переходит в вершину сочленения v, а из нее в вершину v_i компоненты G_i и $i \neq j$.

1. В этом случае агент из вершины сочленения v переходит в вершину v_i компоненты G_i и $i \neq j$. Поскольку все компоненты правильные, то согласно лемме 1 агент не перейдет в вершину сочленения v при выполнении процедуры НАЗАД(v) до тех по пор, пока он не восстановит все ребра и вершины компоненты G_j. После чего он при выполнении процедуры НАЗАД(v) переходит в вершину сочленения v, а из нее в следующую не восстановленную компоненту. Следовательно, агент последовательно восстановит компоненты G_i, где $i = 1,..,k$ и $i \neq j$, а $G \in M_2^p$, где $p = \sum_{i=1}^{k} p_i$ вне зависимости от порядка восстановления компонент.

2. В этом случае агент из вершины сочленения v переходит в вершину v_i компоненты G_i и $i \neq j$. Поскольку все компоненты правильные, то согласно лемме 1 агент не перейдет в вершину сочленения v при выполнении процедуры НАЗАД(v) до тех по пор, пока он не восстановит все ребра и вершины компоненты G_j. После чего он при выполнении процедуры НАЗАД(v) переходит в вершину сочленения v. Далее агент либо возвращается в компоненту G_j, либо переходит в следующую не восстановленную компоненту. В первом случае агент восстановит оставшуюся часть компоненты G_j (по лемме 1 в силу того, что G_j правильная компонента). Во втором случае, агент восстановит полностью еще одну компоненту. Следовательно, агент восстановит часть компоненты G_j, затем некоторое число компонент G_i, после чего восстановит оставшуюся часть компоненты G_j, а затем – оставшиеся компоненты G_i, где $i = 1,..,k$ и $i \neq j$, а $G \in M_2^p$, где $p = \sum_{i=1}^{k} p_i$ вне зависимости от порядка восстановления компонент. □

Следствие 1. *Граф* G, *полученный сочленением двух графов* G_1 *и* G_2, *где* $G_1 \in M_2^{p_1}$ *и* $G_2 \in M_2^{p_2}$, *принадлежит классу* M_2^p, *где* $p = p_1 + p_2$ *при выполнении агентом алгоритмов* A, A_1 – A_3.

Любой граф G_i $i = 1, 2$, можно представить в виде сочленения правильных пар $\left(v_1, G_i^1\right),.., \left(v_{k_i}, G_i^{k_i}\right)$, где G_i^j $j = 1,..,k_i$ подграфы графа G_i. Тогда G будет сочленением правильных пар компонент G_1 и G_2. Из теоремы 1 получаем справедливость следствия.

Следствие 2. *Пусть $G \in T^k$ получен последовательным сочленением $G_1, ..., G_k$, где $G_j \in M_2^{p_j}$, $j = 1, .., k$. Тогда $G \in M_2^p$ где $p = \sum_{i=1}^k p_i$. При этом p_j определяются выполняемым агентом алгоритмом A, A_1 – A_3.*

Докажем следствие 2 методом математической индукции.

При $k = 1$. В этом случае $G = G_1$, соответственно $p = p_1$ и следствие справедливо.

Пусть при $k = j$ следствие справедливо, тогда $p = \sum_{i=1}^k p_i$.

Покажем, что при $k = j + 1$ выполняется $p = \sum_{i=1}^k p_i$. Поскольку в этом случае результирующий граф получается путем сочленения графа G',полученного j последовательными сочленениями графов $G_1, .., G_j$, и графа G_{j+1}, то для них справедливо следствие 1, тогда $p = \sum_{i=1}^k p_i$.

Доказанные выше утверждения показывают, что число камней используемых при восстановлении графа с агентом, выполняющим алгоритмы A, A_1 – A_3, определяется формулой $p = \sum_{j=1}^k p_j$, где p_j – число камней, используемых для восстановления компоненты G_j.

Другим классом графов, у которых число камней есть функцией числа камней компонент, является класс квазикольца.

При восстановлении графа $G \in S^k$, для простоты рассуждений, будем предполагать, что агент начинает восстановление из вершины компоненты G_1.

Теорема 2. *Пусть $G \in S^k$ получен последовательным сочленением $G_1, ..., G_k$, где $G_j \in M_2^{p_j}$, $j = 1, .., k$. тогда $G \in M_2^p$ где $p = \sum_{i=1}^k p_i + p_1$. При этом p_j определяются выполняемым агентом алгоритмом A, A_1 – A_3,*

Доказательство. Для графа $G' \in T^k$ выполняется следствие 2, тогда $G' \in M_2^p$, где $p = \sum_{j=1}^k p_j$. Однако, в силу того, что G_1 и G_k сочленяются, то агент может пройти по всем G_j, где $j = 1, .., k$, так что все $\sum_{j=1}^k p_j$ будут заняты, и после того, как он перейдет из вершин компоненты G_k в вершину сочленения u_{k-1}, метит камнем номер i, который был учтен при подсчете p_k, после этого агент переходит в вершину компоненты G_1. Поскольку камень номер i может остаться не свободным, то для восстановления оставшейся части вершин компоненты G_1 агенту потребуется дополнительно не менее p_1 камней, так как из вершины сочленения u_{k-1} агент может попасть в компоненты G_1, в которой агент до этого еще не был. Следовательно, всего потребуется не более $\sum_{j=1}^k p_j + p_1$ камней. □

Доказанные выше утверждения показывают, что число камней, используемых при восстановлении графа агентом, выполняющим A, A_1, A_2, определяется формулой $\sum_{i=1}^k p_i + p_1$.

6. Восстановление квазидеревьев и квазиколец, полученных соединением компонент мостом. Рассмотрим сочленение двух графов G_1 и G_2, когда их вершины v_1 и v_2 не отождествляются в одну, а соединяются ребром. Такое ребро является мостом в смысле [12]. Мы будем рассматривать соединение двух графов не только одним ребром (мостом),но и соединение двух графов линией. В этом случае будем говорить, что графы соединены мостом длины l, где l – количество вершин в линии. Ясно, что при соединении графов G_1 и G_2 мостом длины l происходит

сочленение линии длины l с графами G_1 и G_2. При этом один конец линии v отождествляется с вершиной v_1 графа G_1, а второй конец линии u отождествляется с вершиной v_2 графа G_2. При этом, если один из соединяемых графов был получен соединением двух других графов при помощи моста, то при его сочленении с мостом может быть выбрана любая вершина. Т.е. конец моста может сочленяться с вершиной, которая принадлежала мосту при его построении. Следовательно, если при построении квазидеревьев и квазиколец заменить операцию сочленения на операцию соединения через мост некоторой длины l, то получим квазикольца и квазидеревья, для которых будут справедливы аналогичные леммы следствия и теоремы. Сформулируем результаты, аналогичные следствиям 1 и 2 и теоремам 1 и 2, для случая, когда операция сочленения заменяется операцией соединения графов мостом длины l.

Лемма 3. *При выполнении агентом алгоритмов A, A_2 на графе G вида линия агент использует не более трех различных камней и две различные краски, т.е. $G \in M_2^3$.*

Доказательство. Рассмотрим худший случай, когда агент попадает в вершину, не являющуюся концом линии. Тогда, согласно алгоритму, в ней он установит камень номер 1, и перейдет в соседнюю вершину, и установит в ней камень номер 2, после чего перейдет в соседнюю вершину и пометит ее камнем номер 3, и при этом камень номер 2 станет свободным, а вершину агент пометит краской b. После перехода в соседнюю, не помеченную вершину, агент пометит ее камнем 2, при этом камень номер 3 станет свободным, а вершину агент пометит b. И так далее, чередуя использование камней номер 2 и номер 3, агент дойдет до конца линии, а потом вернется в начальную вершину, и так же, чередуя использование камней номер 2 и 3, дойдет до другого конца линии и тем самым восстановит граф с использованием только трех красок. □

Лемма 4. *При выполнении агентом алгоритма A_1 на графе G агент использует один камень и две различные краски, т.е. $G \in M_2^1$.*

Доказательство. Воспользуемся рассуждениями леммы 3, заметив, что вершины, которые агент метит камнями номер 2 и номер 3, будут вершинами, из которых будет видно только одну не помеченную вершину. Следовательно, агенту не потребуются камни для их пометки, а только один камень для пометки начальной вершины. □

Лемма 5. *Пусть агент выполняет алгоритмы A, A_1, A_2, на графе G, который является деревом, у которого i вершин имеют степень больше двух. Тогда $G \in M_2^p$, где $p = i + 3$ для A, A_2 и $p = p + 1$ для A_1.*

Доказательство. Для простоты рассуждений будем предполагать, что такое дерево получается из графа вида линия путем его сочленения с i графами l_j вида линия $j = 1, .., i$. Наихудшим случаем является случай, когда длины этих линий больше трех и они сочленяются путем отождествления их вершин с разными вершинами исходного графа, поскольку, при этом установленные в них камни не освобождаются сразу после установки. Согласно лемме 3 при выполнении агентом A агент использу-

ет три различных камня для восстановления исходной линии. Согласно лемме 4 при выполнении агентом эвристики A_1 агент использует один камень для восстановления исходной линии l. Согласно лемме 4 при выполнении агентом эвристики A_2 агент использует три различных камня для восстановления исходной линии. Во всех трех случаях камни используются повторно. Повторное использование камней обуславливается тем, что они либо освобождаются, либо вершины относятся к вершинам специального вида (эвристика A_1). В полученном графе будет i вершин, которые не будут принадлежать к вершинам особого вида, камни, установленные в них, не будут сразу освобождаться. Для восстановления вершин линий l_j $j = 1, .., i$ агент будет использовать камни, которые использовались при восстановлении исходной линии l, и на момент начала восстановления добавленной линии l_j $j = 1, .., i$ стали свободны. Таким образом, в i вершинах появятся камни, которые не освободятся сразу же после установки. Для A, A_1 камни не будут освобождаться, поскольку степень вершины будет больше двух. То есть, после перехода в следующую вершину в окрестности предыдущей остается одна не посещенная вершина и камень не освободится. В случае выполнения эвристики A_2 при пометке 1− окрестности вершины степени больше двух в силу тех же причин один камень останется не свободным. Следовательно, агенту потребуется i дополнительных камней для всех трех случаев. □

Теорема 3. *Пусть $G \in T^k$, а $G_j \in M_2^{p_j}$, где $j = 1, .., k$ и вместо операции сочленения используется операция соединения графов при помощи моста. Тогда $G \in M_2^p$, при выполнении агентом алгоритмов A, A_2 $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 4$, а при выполнении алгоритма A_1 – $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 2$.*

Доказательство. В данном случае граф G получен сочленением компонент G_j $j = 1, .., k$ и $k-1$ мостов, которые могут образовывать деревья. В худшем случае будет образовано одно дерево с $k-2$ вершинами степени больше двух или же будет образовано большее количество деревьев, но количество вершин в них степени больше двух не превзойдет $k-2$. Следовательно, по лемме 5 для восстановления такого или каждого из таких деревьев с i вершинами степени больше двух потребует дополнительно камней $i+3$ при выполнении агентом A, A_2 и $i+3$ камня при выполнении агентом A_2. При этом дополнительные камни, которые используются для восстановления вершин степени меньше трех будут использоваться повторно. При этом, если агент изначально помещается в вершину, принадлежащую одному из мостов, то в начале восстановления один камень останется не свободным, а остальные камни, будут повторно использоваться при восстановлении вершин степень меньше трех. Так как, по следствию 2, верхняя оценка числа камней, используемых агентом, будет равна сумме верхних оценок всех компонент. Тогда, всего агенту потребуется при выполнении алгоритмов A, A_2 $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 3 + 1$ камней, а при выполнении алгоритмов A_1 – $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 1 + 1$ камней. □

Теорема 4. *При выполнении агентом алгоритмов A, A_1, A_2 на $G \in S^k$, где $G_j \in M_2^{p_j}$, $G \in M_2^p$, где $j = 1, .., k$ и вместо операции сочленения используется операция соединения графов при помощи моста. Тогда $G \in M_2^p$, при выполнении*

агентом A, A_2 *–* $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 4$*, а при* A_1 *–* $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 2$*.*

Доказательство. В данном случае граф G получен сочленением компонент G_j $j = 1, .., k$ и $k-1$ мостов, которые могут образовывать деревья. В худшем случае будет образовано одно дерево с $k-2$ вершинами степени больше двух или же будет образовано большее количество деревьев, но количество вершин в них степени больше двух не превзойдет $k-2$. Следовательно, по лемме 5 для восстановления такого или каждого из таких деревьев с i вершинами степени больше двух потребуется дополнительно камней $i+3$ при выполнении агентом A, A_2, а для A_1 – $i+1$ камней, при этом дополнительные камни, которые используются для восстановления вершин степени меньше трех будут использоваться повторно. При этом, если агент изначально помещается в вершину, принадлежащую одному из мостов, то в начале восстановления один камень останется не свободным, а остальные камни, будут повторно использоваться при восстановлении вершин степень меньше трех. Так как, по теореме 2 верхняя оценка числа камней, используемых агентом, будет равна сумме верхних оценок всех компонент и дополнительно числа камней, используемых для восстановления начальной компоненты. Тогда, всего агенту потребуется при выполнении алгоритмов A, A_2 $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 3 + 1$ камней, а при выполнении алгоритма A_1 – $p = \sum_{j=1}^{k} p_j + p_1 + (k-2) + 1 + 1$ камней. □

Доказанные выше утверждения обобщают результаты, полученные для операции сочленения.

7. Выводы. Найдены оценки числа камней, используемых для восстановления графов из классов квазикольца и квазидеревья. Для квазидеревьев число камней используемых агентом, выражается через сумму числа камней, используемых для восстановления компонент. Для квазиколец число камней, используемых агентом, – через сумму числа камней, используемых для восстановления компонент и дополнительного числа камней, используемых для восстановления компоненты, из которой агент начал восстановления графа.

Результаты, полученные для классов T^k S^k, обобщаются для случая, когда сочленение заменяется соединением двух графов мостом. Для операции соединения двух графов мостом получены результаты, аналогичные результатам для операции сочленения.

1. *Килибарда Г., Кудрявцев В.Б., Ушчумлич Щ.*Коллективы автоматов в лабиринтах // Дискретная математика. – 2003. – Т. 15, вып. 3. – С. 3–40.
2. *Килибарда Г., Кудрявцев В.Б., Ушчумлич Щ.* Независимые системы автоматов в лабиринтах // Дискретная математика. – 2003. – Т. 15, вып. 2. – С. 3–39.
3. *Килибарда Г., Кудрявцев В.Б., Ушчумлич Щ.* О поведении автоматов в лабиринтах // Дискретная математика. – 1992. – Т. 4, вып. 3. – С. 3–28.
4. *Кудрявцев В.Б., Алешин С.В., Подколзин А.С.* Введение в теорию автоматов. – М: Наука, 1985, – 320 с.
5. *Dudek G., Jenkin M.* Computational principles of mobile robotic // Cambridge Univ. Press. – 2000. – 280 p.
6. *Татаринов Е.А.* Базовый алгоритм восстановления графа // Труды ИПММ НАН Украины. – 2010. – Т. 21. – С. 492-497.
7. *Татаринов Е.А.* Восстановление графа при помощи камней // Збірник наукових праць "Пи-

тання прикладної математики і математичного моделювання". – Дніпропетровськ: Вид. Дніпр. нац. ун-та. – 2011. – С. 232-255.
8. *Грунский И.С., Татаринов Е.А.* Распознавание конечного графа блуждающим по нему агентом // Вестник Донецкого университета. Серия А. Естественные науки. – 2009. – Вып. 1. – С. 492-497.
9. *Ахо А., Хопкрофт Дж., Ульман Дж.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 536 с.
10. *Кормен Т., Лейзерсон Ч., Ривест Р.* Алгоритмы: построение и анализ. – М.: МЦНМО, 2001. – 960 с.
11. *Касьянов В.Н., Евстигнеев В.А.* Графы в программировании: обработка, визуализация и применение. – СПБ.: БХВ – Петербург, 2003. – 1104 с.
12. *Харари Ф.* Теория графов. – М.: Мир, 1973. – 300 с.

E. A. Tatarinov

Basic algorithm for reconstructing a finite graph.

The modifications of the reconstruction a graph algorithm by agent moving moving through his edges, read and modify marks on the elements of the graph are analyzed. Found and research operations on graphs. The resulting graph of these operations is reconstructed with the use of stone, which is the sum of the number of stones needed to reconstruct the original components.

Keywords: *graph reconstraction, agent, complexity of the algorithm, move on graph.*

Є. О. Татаринов

Складність відновлення графів, що є квазікільцями і квазідеревами.

Аналізуються модифікації алгоритму відновлення графа агентом, що переміщається по його ребрах, що зчитує і змінює мітки на елементах графа. Знайдено та досліджено операції над графами. Результуючий граф цих операцій відновлюється з використанням числа каменів, що виражається через суму числа каменів, необхідних для відновлення вихідних компонент.

Ключові слова: *відновлення графів, агент, складнітъ алгоритму, блукання по графу.*

Ин-т прикл. математики и механики НАН Украины, Донецк
MDgerelo@yandex.ru

Получено 02.12.11