УДК 681.322

РЕАЛИЗАЦИЯ АДАПТИВНОГО АЛГОРИТМА ПОИСКА МИНИМАЛЬНЫХ ПУТЕЙ НА ОБЪЕКТАХ С СЕТЕВОЙ ОРГАНИЗАЦИЕЙ

к.т.н. Ю.Э. Парфенов, к.т.н. С.С. Багров, к.т.н. О.И. Богатов
(представил д.т.н., проф. Г.А. Поляков)

Рассмотрены вопросы вычислительной реализации адаптивного алгоритма поиска минимальных путей на объектах с сетевой организацией. Проведен анализ применимости различных структур данных для представления информации о графах этих объектов. Приведены схемы алгоритма. Получены оценки его вычислительной трудоемкости.

Постановка проблемы. Известно достаточно большое количество алгоритмов поиска минимальных путей на графах [1, 2]. Однако они имеют ряд недостатков, существенно ограничивающих возможность их использования.

Исходя из этого в работе [3] разработан алгоритм, являющийся адаптивным к различным структурам графов, в том числе имеющим транзитивно-рефлексивные замыкания. Практическое применение указанного алгоритма требует его эффективной реализации на вычислительных средствах.

Анализ последних достижений и публикаций. Реализация алгоритмов на вычислительных средствах как правило предполагает их предварительное преобразование, необходимое для рациональной организации вычислительного процесса с целью снижения их вычислительной трудоемкости. При этом важным является выбор структур данных, предназначенных для хранения обрабатываемой информации.

Существуют различные подходы к представлению графов в памяти компьютера. Так в работах [4, 5] выделены следующие варианты:

1. ***Представление с помощью матрицы смежности***. Одно из самых распространенных. Для графов с большим числом дуг (например, полносвязанных) это достаточно компактное представление. К недостаткам следует отнести: существенные затраты памяти при работе с графами, имеющими небольшое число дуг, вследствие значительной разреженности матрицы смежности; невозможность снижения трудоемкости алгоритмов в том случае, когда она пропорциональна числу дуг, а не числу вершин.

2. ***Представление с помощью матрицы инциденций.*** Применяется крайне редко в силу большой разреженности матрицы и практическо-

го отсутствия алгоритмов, работающих на такой структуре данных.

3. ***Представление с помощью списков смежности***. Является главной альтернативой представлению с помощью матрицы смежности. В случае ориентированного графа список смежности для некоторой вершины *v* есть список концов дуг, исходящих из этой вершины. Граф представляется с помощью n списков смежности (n – количество вершин графа). Если число дуг мало по сравнению с сильносвязанным графом, то этот способ представления весьма эффективен. Однако, для задания взвешенных графов этот способ неудобен, так как возникает дополнительная задача хранения весов дуг и установления соответствующих связей между дугами и их весами.

4. ***Представление с помощью списка дуг***. Применяется в тех случаях, когда необходимо иметь отдельную, независимую нумерацию дуг. При этом каждой дуге сопоставляется тройка <u, x, y>, где u – дуга, x – ее начало, y – ее конец. Этот способ представления легко обобщается на случай взвешенных графов, однако не обеспечивает приемлемой скорости поиска информации.

Цель статьи. Таким образом, целью статьи является разработка эффективной вычислительной реализации алгоритма, предложенного в [3].

Изложение основного материала. Исходный алгоритм [3] является адаптивным по отношению к различным структурам графов. Из этого следует, что объем памяти для хранения информации о графе может изменяться в широких пределах. В этом случае традиционные подходы к представлению графов в памяти компьютера, рассмотренные выше, являются неэффективными, так как они либо требуют выделения на этапе компиляции программы значительного объема памяти, либо имеют другие ограничения.

В данном случае целесообразно использовать динамические структуры данных, которые также обеспечивали бы эффективное хранение и поиск информации. Исходя из приведенных соображений в предложенной реализации алгоритма использованы ассоциативные контейнеры (АК) «ключ-значение» и динамические массивы. Для хранения промежуточных результатов применяются статические структуры данных (простые переменные и массивы).

Схема алгоритма представлена на рис. 1, 2 и состоит из двух частей – "прямого" и обратного "хода". "***Прямой ход***" алгоритма (рис. 1) предназначен для определения числовых характеристик всех вершин графа в соответствии с этапами 1 – 6 алгоритма [3]. Блок 1 алгоритма – чтение информации о структуре графа и запись ее в ассоциативный контейнер "ключ-значение" *inputInfo*. Информация в *inputInfo* хранится в виде пар "ключ-значение" (табл. 1, в которой "ключ"(j) – номер

Таблица 1

Формат хранения информации в АК *inputInfo*

"ключ"	"значение"
j	j a, W_{ja}; … j i, W_{ji}; …
…	…

вершины графа, имеющей хотя бы одну входящую дугу; "значение" – информация о структуре входящих связей вершины с номером j; W_{ji} – числовое значение дуги, входящей в вершину j из вершины i). Будем называть такие

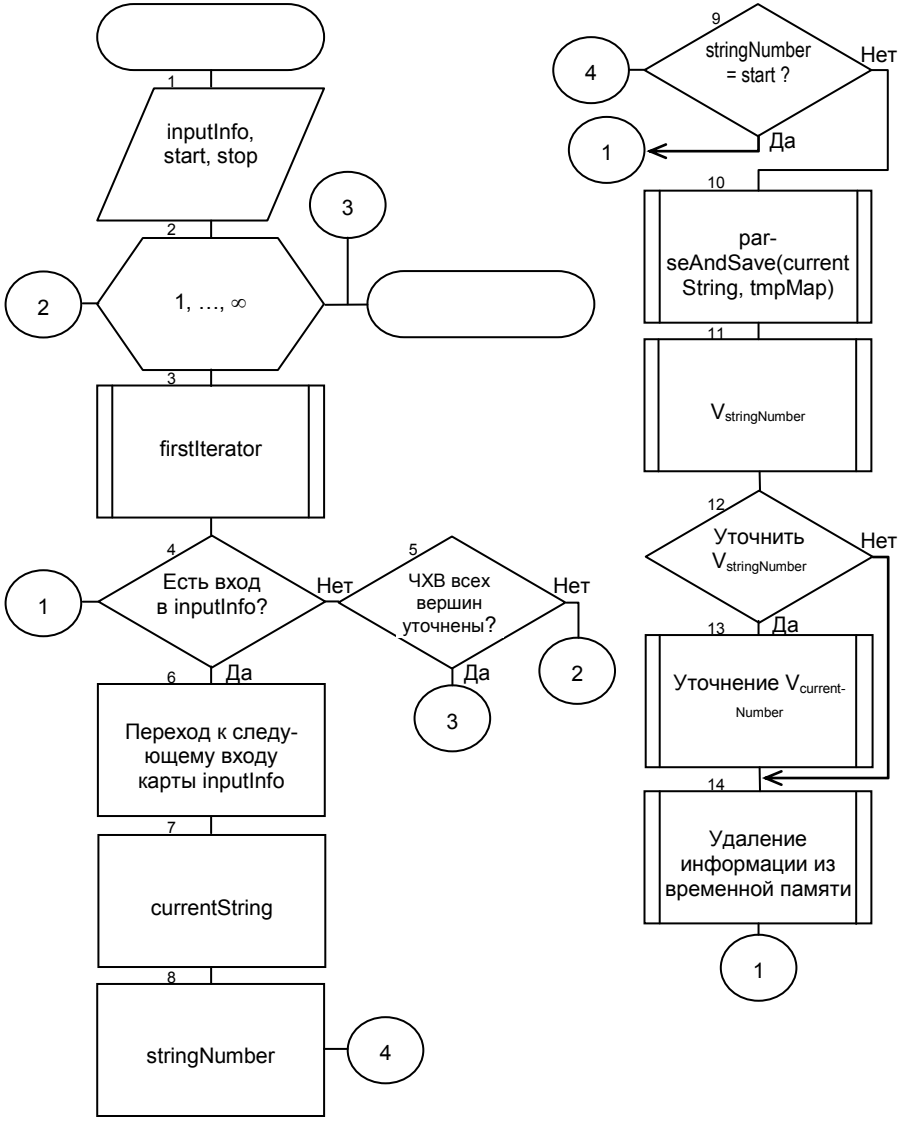

Рис. 1. "Прямой ход" алгоритма

пары "входами". Использование АК позволяет хранить графы с любым количеством вершин и любой структурой связей, а также осуществлять ее эффек-

тивный поиск.

В блоке 1 также осуществляется ввод номеров начальной (*start*) и конечной (*stop*) вершин графа, между которыми необходимо найти один или не-

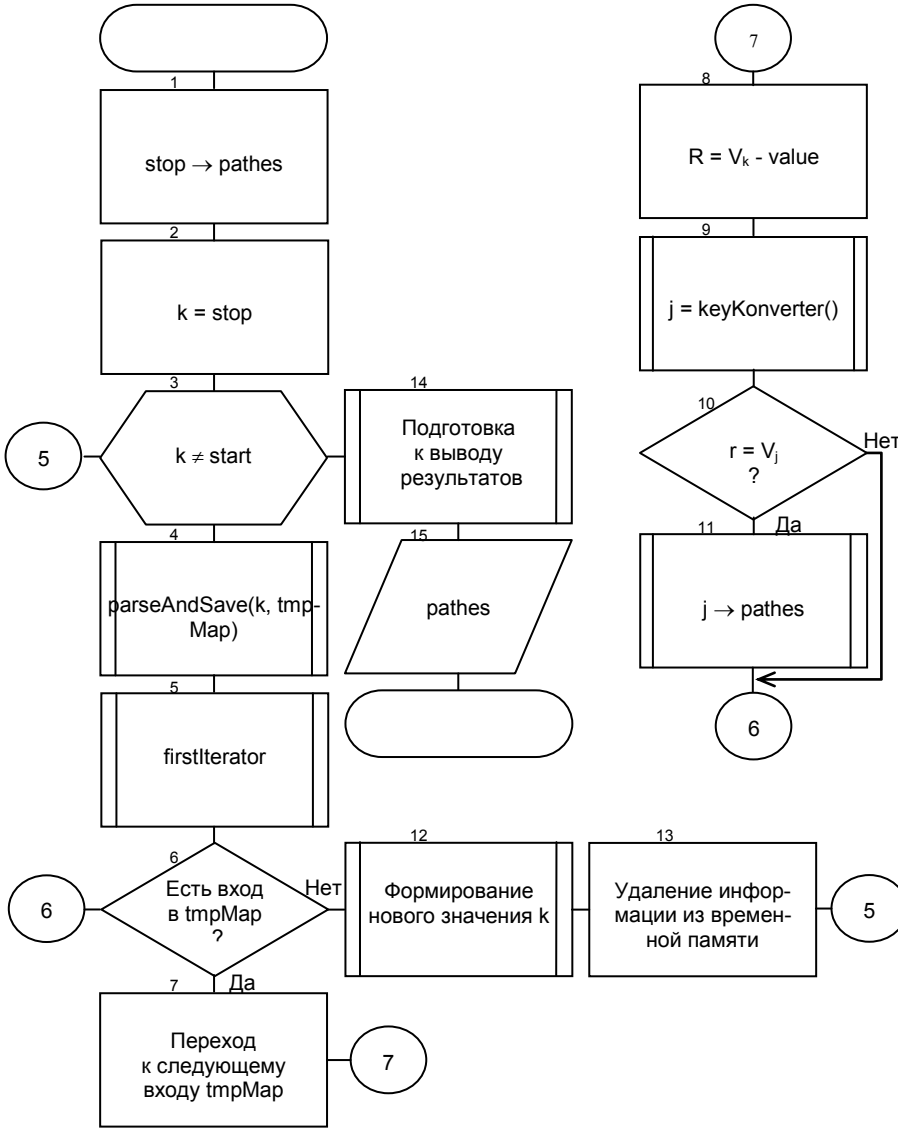

Рис. 2. "Обратный ход" алгоритма

сколько минимальных путей (этап 2 алгоритма [3]). Блок 2 – начало цикла

выборки информации (табл. 1) из *inputInfo*. Блок 3 необходим для инициализации итератора, предназначенного для организации последовательного перебора "входов" *inputInfo*. Блоки 4, 6 – организация последовательного перебора "входов" *inputInfo* с помощью итератора *firstIterator*. Блоки 7, 8 – извлечение "значения" (*currentString*) и "ключа" (*stringNumber*) текущего "входа" *inputInfo* соответственно. Блок 9 – если текущий "вход" *inputInfo* соответствует начальной точке искомого пути, то переходим к следующему "входу" (блок 4), иначе – на блок 10. Блок 10 предназначен для синтаксического разбора "значения" текущего входа *inputInfo* и последующей записи его во временный ассоциативный контейнер *tmpMap*; *tmpMap* является вспомогательной структурой данных, предназначенной для организации вычисления приближенной числовой характеристики вершины графа с номером *currentString*. Формат хранения информации в *tmpMap* представлен в табл. 2.

Таблица 2

Формат хранения информации в АК *tmpMap*

"ключ"	"значение"
ja	W_{ja}
jb	W_{jb}
…	…
ji	W_{ji}
…	…

Блок 11 – определение числовой характеристики вершины с номером *currentString* в соответствии с соотношением (2) работы [3]. Блоки 12, 13 предназначены для уточнения приближенной числовой характеристики вершины *currentString* в соответствии с этапом 6 алгоритма [3]. Блок 14 – стирание текущей информации из временной памяти, в частности из *tmpMap*. В блоке 5 если числовые характеристики (ЧХВ) всех вершин уточнены, то прямой ход алгоритма заканчивается, в противном случае – цикл (блок 2) повторяется (этап 6 алгоритма [3]).

"***Обратный ход***" алгоритма (рис. 2) предназначен для реализации этапа 7 исходного алгоритма [3]. В блоке 1 конечная точка пути (*stop*) записывается в динамический массив *pathes*, что позволяет хранить там любое количество путей и снизить затраты памяти. В блоке 2 счетчику цикла k присваивается номер конечной вершины (stop), так как определение минимальных путей идет в обратной последовательности. Блок 3 – начало цикла определения вершин, принадлежащих минимальному пути (путям). Блок 4 – синтаксический разбор "значения" из *inputInfo*, соответствующего ключу k и запись его в *tmpMap*. Формат хранения данных аналогичен табл. 2. В блоке 8 выполняются подготовительные операции для вычисления соотношения (3) [3]. Блок 9 предназначен для определения номера вершины, которая является возможным продолжением минимального пути. Это делается путем преобразования ключа текущего входа *tmpMap*. Блоки 10, 11 предназначены для определения вершины – продолжения минимального пути и отнесения ее к одному из имеющихся путей. Блок 12 определяет значение "ключа", по которому обращаться в *inputInfo*. Блок 13 аналогичен блоку 14 "прямого хода" алгоритма. Блок 14 – реверси-

рование найденных путей. Блок 15 – вывод найденных путей.

Предложенный алгоритм доведен до программной реализации. Вычислительные эксперименты показали, что по сравнению с использованием традиционного представления графов в памяти компьютера (в виде матрицы смежности) объем необходимой памяти уменьшается на 15 – 25% в зависимости от типа структуры и количества вершин графа. Трудоемкость алгоритма для графов с древовидной структурой пропорциональна количеству вершин, для графов с сильносвязанной структурой – квадрату количества вершин.

Выводы. 1. Разработана вычислительная реализация адаптивного алгоритма поиска минимальных путей на объектах с сетевой организацией [3], позволяющая снизить затраты памяти компьютера на 15 – 25% по сравнению с традиционным подходом.

2. Предложенный алгоритм может быть использован в качестве основного бизнес-компонента программных систем исследования объектов любой природы, структура которых представима в виде графа, в частности автоматизированных систем управления.

ЛИТЕРАТУРА

1. *Бацамут В.Н. Анализ известных подходов к решению задачи поиска транзитивно-рефлексивных замыканий // Системи обробки інформації. – Х.: НАНУ, ПАНМ, ХВУ. – 2000. – Вип. 3(9). – С. 61 – 68.*
2. *Ермольев Ю.М., Мельник И.М. Экстремальные задачи на графах. – К.: Наук. думка, 1968. – 174 с.*
3. *Багров С.С., Парфенов Ю.Э., Богатов О.И. Адаптивный алгоритм поиска минимальных путей на объектах с сетевой организацией // Системи обробки інформації. – Х.: НАНУ, ПАНМ, ХВУ. – 2002. – Вип. 6(22). – С. 9 – 12.*
4. *Евстигнеев В.А. Применение теории графов в программировании. – М.: Наука, 1985. – 351 с.*
5. *Кристофидес Н. Теория графов: алгоритмический подход. – М.: Мир, 1978. – 432 с.*

Поступила 27.08.2003

***ПАРФЕНОВ Юрий Эдуардович**, канд. техн. наук, нач. лаборатории научного центра при ХВУ. В 1993 году окончил ВИРТА ПВО. Область научных интересов – математическое моделирование, проектирование информационных систем.*

***БАГРОВ Сергей Сергеевич,** канд. техн. наук, ст. научн. сотрудник научного центра при ХВУ. В 1992 году окончил ВИРТА ПВО. Область научных интересов – моделирование, системный анализ.*

***БОГАТОВ Олег Игоревич,** канд. техн. наук, ст. научн. сотр., начальник научно-исследовательского отдела научного центра при ХВУ. В 1983 году окончил Киевское ВИРТУ. Область научных интересов – параллельная обработка информации, САПР.*