УДК 681.324:621.325

А.В. Овчинников, В.В.Берковский

Харьковский университет Воздушных Сил имени Ивана Кожедуба, Харьков

АНАЛИЗ АЛГОРИТМОВ ПОИСКА ОПТИМАЛЬНЫХ ПУТЕЙ НА ГРАФАХ

В данной статье рассмотрены наиболее широко используемые алгоритмы поиска оптимальных путей на структурах, представленных графами. Показаны области знаний, в которых используются графовые задачи и проанализированы основныеалгоритмы поиска.

Ключевые слова: *граф, алгоритм поиска, сложность алгоритма.*

Введение

Графы широко используются во многих сферах науки и техники, в частности в следующих. Файловая система компьютера – иерархия файлов и папок во многих операционных системах имеет вид графа – дерева. Молекулы всех химических веществ можно изобразить в виде графа, где атомы являются вершинами, а связки между ними – ребрами. Карта автомобильных или любых других путей также является графом, причем каждая дорога может иметь определенное значение "веса" (например, плотность транспортного потока), тогда такой граф является взвешенным. Социальные сети также можно представить в виде графа, где каждый человек или социальная группа является вершиной, а связи между ними – ребрами. Генеалогические деревья являются примером бинарных деревьев, которые также являются отдельным случаем графа. Турнирные таблицы спортивных чемпионатов также могут быть изображены в виде графов. В биологии и экологии графы используются уже давно. Примерами могут быть цепи питания, экосистемы, генетические последовательности и карты, таксономическая иерархия живых организмов и тому подобное; В археологии и геологии графы используются в стратиграфии для изучения геологических пластов. Любой производственный процесс также может быть изображен с помощью графа.

Разработка программного обеспечения и компьютерные науки вообще является одной из тех отраслей, где графы применяются чаще всего. Сложность и большое количество модулей и протоколов в современных программных продуктах сильно усложняет понимание их работы, управления ею и ее оптимизацию, потому очень часто складываются графы программ, причем чаще всего это делается автоматически трансляторами или компиляторами. Графы также являются удобными для изображения структур данных, блок-схем, потоков данных, схем баз данных и баз знаний, конечных автоматов, схем компьютерных сетей и отдельных сайтов, схем вызовов подпрограмм и тому подобное [1 – 3].

Существуют три наиболее эффективных алгоритма нахождения кратчайшего пути на графах: алгоритм Дейкстры (используется для нахождения оптимального маршрута между двумя вершинами); алгоритм Флойда (для нахождения оптимального маршрута между всеми парами вершин); алгоритм Йена (для нахождения k-оптимальных маршрутов между двумя вершинами) [4 – 6]. Отмеченные алгоритмы легко выполняются при малом количестве вершин в графе. При увеличении их количества задача поиску кратчайшего пути усложняется, и потому без применения современной техники практически не обойтись. **Цель статьи** – провести сравнительный анализ жанных алгоритмов.

1. Алгоритм поиска в глубину (рис 1)

При решении многих заданий, которые касаются ориентированных графов, необходим эффективный метод систематического обхода вершин и дуг ориентированного графа. Таким методом является метод-поиск в глубину. Метод поиска в глубину является основой многих эффективных алгоритмов работы с графами. Предполагается, что есть ориентированный граф G, в котором сначала все вершины обозначены меткой "unvisited". Поиск в глубину начинается с выбора начальной вершины v ориентированного графа G, для этой вершины метка "unvisited" изменяется на метку "visited". Потом для каждой вершины, смежной с вершиной v и которая не была посещена раньше, рекурсивно применяется поиск в глубину. Когда все вершины, которых можно достичь из вершины, будут рассмотрены, поиск заканчивается. Если некоторые вершины остались не посещенными, то выбирается одна из них и алгоритм повторяется. Этот процесс длится до тех пор, пока не будут обойдены все вершины ориентированного графа G.

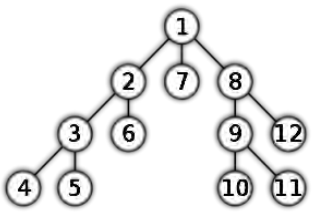

Рис. 1. Схема алгоритма поиска в глубину

Метод получил свое название – поиск в глубину, поскольку поиск непоселенных вершин идет в

направлении вглубь до тех пор, пока это возможно. Например, пусть x есть последней посещенной вершиной. Выбирают очередную дугу(ребро)(x,y), которая выходит из вершины x. Возможна следующая альтернатива: вершина y обозначена меткой "unvisited"; вершина y обозначена меткой "visited".

Если вершина y уже посещалась, то отыскивается другая вершина, смежная с вершиной x; иначе вершина y обозначается меткой "visited" и поиск начинается заново от вершины y. Пройдя все пути, которые начинаются в вершине y, возвращаются в вершину x, то есть в ту вершину, из которой впервые была достигнутая вершина y. Потом процесс повторяется, т.е. продолжается выбор нерассмотренных дуг, которые выходят из вершины x, и так до тех пор, пока не будут исчерпаны все эти дуги [3].

2. Алгоритм поиска в ширину (рис. 2)

Поиск в ширину – алгоритм поиска на графе. Если задан граф G = (V, E) и начальная вершина s, алгоритм поиска в ширину систематически обходит все досягаемые из s вершины. На первом шаге вершина s отражается, как пройденная, а в список добавляются все вершины, досягаемые из s без посещения промежуточных вершин. На каждом следующем шаге все текущие вершины списка отмечаются, как пройденные, а новый список формируется из вершин, которые являются еще не пройденными соседями текущих вершин списка. Для реализации списка вершин чаще всего используется очередь. Выполнение алгоритма продолжается до достижения искомой вершины или до тех пор, пока на определенном шаге в список не включается ни одна вершина. Второй случай означает, что все вершины, доступные из начальной, уже отмечены, как пройденные, а путь к целевой вершине не найден.

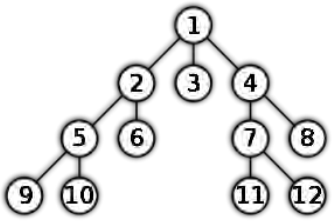

Рис. 2. Схема алгоритма поиска в ширину

Алгоритм имеет название поиска в ширину, поскольку "фронт" поиска (между пройденными и не пройденными вершинами) однообразно расширяется вдоль всей своей ширины, т.е. алгоритм проходит все вершины на расстоянии k перед тем как пройти вершины на расстоянии k+1[4].

Алгоритм поиска в ширину применяют на практике в волновом алгоритме при трассировке печатных плат и при поиске пути, который увеличивается (алгоритм Форда-Фалкерсона (алгоритм Едмондса-Карпа)).

Алгоритм Дейкстры (рис. 3)

Алгоритм Дейкстры (Dijkstra's algorithm) – алгоритм на графах, предложенный голландцем Е. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа ко всем другим. Алгоритм работает только для графов без ребер отрицательного веса. Широко применяется в программировании и технологиях, например, его использует протокол OSPF для устранения кольцевых маршрутов.

Формальное определение: дано: взвешенный ориентированный [5] граф G(V, E) без петель и дуг отрицательного веса. Найти кратчайшие пути от некоторой вершины a графа G ко всем другим вершинам этого графа. ***Неформальное определение***: каждой вершине из V сопоставляют метку - минимальное известное расстояние от этой вершины к a. Алгоритм работает пошагово - на каждом шагу он "посещает" одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

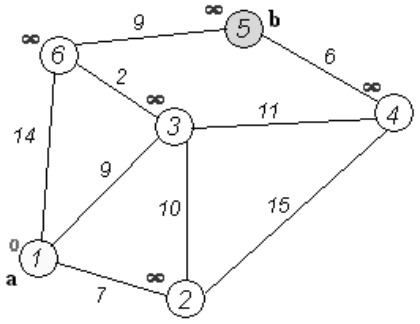

Рис. 3. Схема реализации поиска алгоритмом Дейкстри

Инициализация. Метка самой вершины a считается равной 0, метки других вершин - бесконечности. Это отображает тот факт, что расстояния от a к другим вершинам пока неизвестны. Все вершины графа обозначаются как непосещенные.

Шаг алгоритма. Если все вершины посещены, алгоритм завершается. Если нет, то, из еще не посещенных вершин выбирается вершина u, которая имеет минимальную метку. Рассматриваются всевозможные маршруты, где u является предпоследним пунктом. Вершины, в которые ведут ребра из u, называют соседями этой вершины. Для каждого соседа вершины u, кроме обозначенных как посещенные, рассматривают новую длину пути, равную сумме значений текущей метки u и длины ребра, которое соединяет u с этим соседом. Если полученное значение длины меньшее значение метки соседа, заменяют значение метки полученным значениям длины. Рассмотрев всех соседей, обозначают вершину u как посещенную и повторяют шаг алгоритма.

Алгоритм. Обозначения : V – множество вершин графа; E – множество ребер графа; w[ij] – вес (длина) ребра ij a – вершина, расстояние от которой

находится; U – множество вершин, которые были посещены; d[u] – по окончанию работы алгоритма равняется расстоянию кратчайшего пути от a к вершине u; p[u] – по окончанию работы алгоритма содержит кратчайший путь от a к u.

Псевдокод: присваивают $d[a] \leftarrow 0, p[a] \leftarrow a$; для всех $u \in V$, отличающихся от a присваивают $d[u] \leftarrow \infty$, пока $\exists v \notin U$. Пусть $v \notin U$ – вершина с минимальным $d[v]$. Для всех $u \notin U$ таких, что $vu \in E$, если $d[u] > d[v] + w[vu]$ тогда изменяют $d[u] \leftarrow d[v] + w[vu]$ и $d[u] \leftarrow p[v], u$

Описание: В самой простой реализации для хранения чисел d[i] можно использовать массив чисел, а для хранения принадлежности элемента огромного количества U – массив булевых переменных.

В начале алгоритма расстояние для начальной вершины полагается равным нулю, а все другие расстояния заполняются большим положительным числом (больше, чем максимально возможный путь в графе). Массив флагов заполняется нулями.

После этого запускается основной цикл.

На каждом шаге цикла ищется вершина с минимальным расстоянием и флагом, равным нулю. Затем устанавливается флаг в 1 и проверяются все соседние с ней вершины. Если расстояние больше, чем сумма расстояния к текущей вершине и длине ребра, то уменьшают его. Цикл завершается, когда флаги всех вершин становятся равными 1, или, если у всех вершин флаги равны 0. Последний случай возможен тогда и только тогда, если граф G не связный.

Доказательство корректности :

Пусть l(v) – длина кратчайшего пути из вершины a в вершину v. Докажем по индукции, что в момент посещения любой вершины z, d(z)=l(z). База. Первой посещается вершина a.

В этот момент d(a)=l(a)=0. Шаг. Пусть выбрана для посещения вершина $Z \neq a$. Докажем, что в этот момент d(z)=l(z). Для начала отметим, что для любой вершины v, всегда выполняется

$d[v] \geq l[v]$ (алгоритм не может найти путь короче, чем кратчайший из всех существующих). Пусть P – кратчайший путь из a в z, y – первая не посещенная вершина на P, x - предыдущая к ней (следовательно, посещенная). Поскольку путь P кратчайший, его часть, которая ведет из a через x в y, тоже кратчайший, следовательно l(y)=l(x)+w(xy). По предположению индукции, в момент посещения вершины x выполнялось d(x)=l(x), следовательно, вершина y тогда получила метку не больше чемd(x)+w(xy)=l(x)+w(xy)=l(y). Следовательно, d(y)=l(y). С другой стороны, поскольку сейчас выбрана вершина z, ее метка минимальна среди не посещенных, то есть

$$d(z) \leq d(y) = l(y) \leq l(z).$$

Комбинируя это из d(z), получают d(z)=l(z), что и было нужно доказать. Поскольку алгоритм заканчивает работу, когда все вершины посещены, в этот момент d=l для всех вершин.

Сложность алгоритма. Сложность алгоритма Дейкстры зависит от способа нахождения вершины v, а также способа хранения множества не посещенных вершин и способа возобновления меток. Пусть n- количество вершин, а m – количество ребер в графе G.В самом простом случае, когда для поиска вершины с минимальным d[v] просматривается все множество вершин, а для хранения величин d – массив, время работы алгоритма есть O(n2 + m). Основной цикл выполняется порядка n раз, в каждом из них на нахождение минимума тратится порядка n операций, плюс количество релаксаций (изменений меток), которое не превосходит количества ребер в исходном графе.

Для разреженных графов (то есть таких, для которых m много меньше n^2) не посещенные вершины можно хранить в двоичной куче, а в качестве ключа использовать значение d[i], тогда время определения вершины из $\overline{U}$ будет logn, при том, что время модификации d[i] вырастает в logn. Поскольку цикл выполняется порядка n раз, а количество релаксаций не больше m, скорость работы такой реализации O(nlogn+mlogn)

Если для хранения не посещенных вершин использовать фибоначчиеву кучу, для которой удаление происходит в среднем за O(logn), а уменьшения значения в среднем за O(1), то время работы алгоритма составит O(nlogn + m).

4. Алгоритм Флойда-Уоршелла (рис. 4)

Алгоритм Флойда–Уоршелла – динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом.

Алгоритм. Пусть вершины графа G пронумерованы от 1 до n и введено обозначение d^k_{ij} для длины кратчайшего пути от i к j, который кроме самих вершин i,jпроходит только через вершины 1…k.Очевидно, что d^0_{ij}– длина (вес) ребра (i,j) если такое существует (а если нет, то его длина может быть обозначена как ∞) [6].Существует два варианта значения: $d^k_{ij}, k \in (1,\ldots,n)$. Кратчайший путь между i ијне проходит через вершину k, тогда $d^k_{ij} = d^{k-1}_{ij}$

Существует более короткий путь между i и j, который проходит через k, тогда он сначала идет от i к k, а затем от k к j. В этом случае, очевидно,

$$d^k_{ij} = d^{k-1}_{ik} + d^{k-1}_{kj}.$$

Таким образом, для нахождения значения функции достаточно выбрать минимум из двух обозначенных значений. Тогда рекуррентная формула

для d^k_{ij}имеет вид:d^0_{ij}– длина ребра (i,j)

$d^k_{ij}=\min(d^k_{ij},d^{k-1}_{ik}+d^{k-1}_{kj})$.

Алгоритм Флойда–Уоршелла последовательно вычисляет все значения $d^k_{ij}\ \forall_i$, j для k от 1 до n. Полученные значения d^n_{ij} являются длинами кратчайших путей между вершинами i и j.

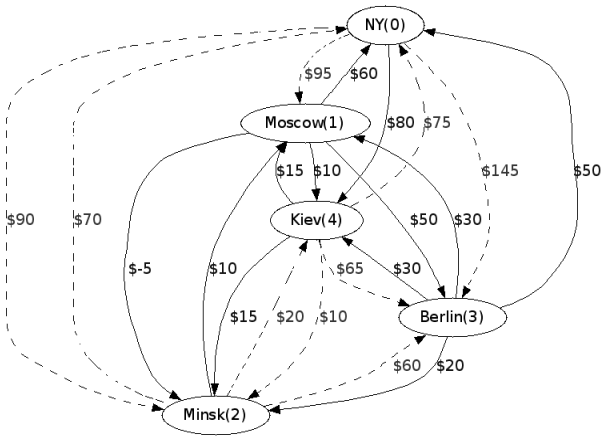

Рис. 4. Схема реализации поиска алгоритмом Флойда-Уоршелла

Псевдокод: На каждом шаге алгоритм генерирует двумерную матрицу W, $w_{ij}=dn_{ij}$. Матрица W содержит длины кратчайших путей между всеми вершинами графа. Перед работой алгоритма матрица W заполняется длинами ребер графа.

Сложность алгоритма. Три вложенных цикла содержат операцию, которая выполняется за константное время. $\sum O(1) = O(n3)nnn$, т.е. алгоритм имеет кубическую сложность, при этом простым расширением можно получить также информацию о кратчайших путях - кроме расстояния между двумя узлами ,записывая в матрицу идентификатор первого узла в пути.

Вариации алгоритма. Построение матрицы смежности. Алгоритм Флойда-Уоршелла может быть использован для нахождения замыкания отношения E по транзитивности. Для этого в качестве W[0] используется бинарная матрица смежности графа,(w0ij)nxn=1 оператор min замещается дизъюнкцией, добавление замещается конъюнкцией:

После выполнения алгоритма матрица W является матрицей досягаемости.

Использование битовых масок при реализации алгоритма позволяет существенно ускорить алгоритм. При этом сложность алгоритма уменьшается к O(n3/k), где k – длина битовой маски (в модели вычислений RAM). На практике, еще большего ускорения можно достичь, используя такие специализированные наборы микропроцессорных команд, таких как SSE.

Выводы

В статье рассмотрены алгоритмы поиска оптимальных путей на графах, позволяющие моделировать процесс функционирования алгоритмов Дейкстры и Флойда по поиску кратчайшего пути в неориентируемом и ориентированном графах, также были рассмотрены алгоритмы для обхода дерева, структуры подобной дереву, или графу такие как: алгоритм поиска в глубину и поиска в ширину

Список литературы

1. Введение в .NET [Электронный ресурс]. – Режим доступа: ttp://mindberg.blogspot.com/2006/08/net.html [Введение в .NET - интернет блог] – Назв. с экрана..

2. Спекторський І.Я. Дискретна математика / І.Я. Спекторський. – 2004. – 220 с.

3. Алгоритмы на графах – Интернет университет информационных технологий [Электронный ресурс]. – Режим доступа: http://www.intuit.ru/department/algorithms /algocombi/16/1.html.

4. Thomas H. Cormen. CliffordStein "22.2", IntroductiontoAlgorithms / Thomas H. Cormen, E. Charles Leiserson, RonaldL. Rivest. – 2001 (англ.). – 531 с.

5. Ананий В. Глава 9. Жадные методы: Алгоритм Дейкстры / В. Ананий // Алгоритмы: введение в разработку и анализ. – Introduction to The Design and Analysis of Aigorithms. — М.: "Вильямс", 2006. — С. 189—195.

6. Ананий В. Алгоритмы: введение в разработку и анализ / В. Ананий. — М.: Вильямс, 2006. – 548 с.

7. Пауэрс Л. Microsoft Visual Studio 2008 Unleashed by Lars Powers and Mike Snell. / Л. Пауэрс, М. Снелл — С.Пб.: "БХВ-Петербург", 2008. – 1200 с.

Поступила в редколлегию 1.02.2012

Рецензент: канд. техн. наук доц. И.В. Ильина, Харьковский университет Воздушных Сил им. И. Кожедуба, Харьков.

АНАЛІЗ АЛГОРИТМІВ ПОШУКУ ОПТИМАЛЬНИХ ШЛЯХІВ НА ГРАФАХ

А.В. Овчинников, В.В. Берковський

Розглянуті найбільш широко використовувані алгоритми пошуку оптимальних шляхів на структурах, представлених графами. Показані галузі знань, в яких використовуються завдання на графах і проаналізовані основні алгоритми пошуку.

Ключові слова: *граф, алгоритм пошуку, складність алгоритму.*

ANALYSIS OF SEARCH OF OPTIMAL PATHS ENGINES ON COLUMNS

A.V. Ovchinnikov, V.V.Berkovsky

In this article the widely used search of optimal paths engines are considered most on structures, presented by columns. The areas of knowledge, in which count tasks are used and an analysis of the basic algorithms search, are shown.

Keywords: *count, search engine, complication of algorithm.*