электронный журнал
МОЛОДЕЖНЫЙ
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО «Московский государственный технический университет им. Н.Э. Баумана»

Исследование и разработка алгоритма нахождения кратчайшего пути на графе

Студент,
кафедра «Системы автоматического управления»: В.В. Березницкий

Научный руководитель: Н.В. Лукьянова
к. т. н., доцент кафедры «Системы автоматического управления»

УДК 681.5

Введение. Благодаря своему широкому применению, теория о нахождении кратчайших путей в последнее время интенсивно развивается и используется в различных областях науки и техники, например для нахождения оптимального маршрута между двумя объектами на местности (кратчайший путь от дома до университета), для нахождения оптимального маршрута при перевозках, в системах автопилота, в системах коммутации информационного пакета в Internet и т.п.

Кратчайший путь можно определить с помощью некоторого математического аппарата, называемого графом.

Существуют три наиболее эффективных алгоритма нахождения кратчайшего пути:

- алгоритм Дейкстры (используется для нахождения оптимального маршрута между двумя вершинами);
- алгоритм Флойда (для нахождения оптимального маршрута между всеми парами вершин);
- алгоритм Беллмана — Форда (для нахождения кратчайшего пути от одной вершины графа до всех остальных).

Указанные алгоритмы легко выполняются при малом количестве вершин в графе. При увеличении их количества задача поиска кратчайшего пути усложняется. Поэтому решая частную задачу бывает рациональнее разработать собственный алгоритм.

Постановка задачи. Основной задачей данного проекта является разработка алгоритма поиска кратчайшего пути между двумя любыми вершинами графа, где граф — схема московского метрополитена, а вершины графа — станции метрополитена и его программная реализация

Программа должна работать так, чтобы пользователь вводил название начального и конечного пункта, а после обработки этих данных на экран выводился кратчайший путь между двумя заданными вершинами и его длина, то есть время, за которое пользователь может попасть из одной точки в другую. Необходимо предусмотреть различные варианты нахождения пути, чтобы программа выводила наиболее рациональные результаты.

Анализ алгоритмов нахождения кратчайшего пути на графах.

Граф или неориентированный граф G — это упорядоченная пара G: = (V,E), для которой выполнены следующие условия:

- V это непустое множество вершин или узлов,
- E это множество пар (в случае неориентированного графа — неупорядоченных) вершин, называемых рёбрами.

Алгори́тм Де́йкстры (Dijkstra's algorithm) [1,2] — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Алгоритм работает только для графов без рёбер отрицательного веса. Алгоритм широко применяется в программировании и технологиях, например, его использует протокол OSPF для устранения кольцевых маршрутов [1, 2] Алгоритм Дейкстры решает задачу о кратчайших путях из одной вершины для взвешенного ориентированного графа $G = (V, E)$ с исходной вершиной s, в котором веса всех рёбер неотрицательны ($\omega(u, v) \geq 0$ для всех $(u, v) \in E$).

Формальное объяснение. В процессе работы алгоритма Дейкстры поддерживается множество $S \subseteq V$, состоящее из вершин v, для которых $\delta(s, v)$ уже найдено. Алгоритм выбирает вершину $u \in V \setminus S$ с наименьшим $d[u]$, добавляет u к множеству S и производит релаксацию всех рёбер, выходящих из u, после чего цикл повторяется. Вершины, не лежащие в S , хранятся в очереди Q с приоритетами, определяемыми значениями функции d. Предполагается, что граф задан с помощью списков смежных вершин.

Неформальное объяснение. Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a. Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки. Работа алгоритма завершается, когда все вершины посещены.

Алгоритм Флойда — Уоршелла — динамический алгоритм для нахождения кратчайших расстояний между всеми вершинами взвешенного ориентированного графа. Разработан в 1962 году Робертом Флойдом и Стивеном Уоршеллом.[2, 3]

Описание алгоритма. Пусть вершины графа $G = (V, E)$, $|V| = n$ пронумерованы от 1 до n и введено обозначение d_{ij}^{k} для длины кратчайшего пути от i до j, который кроме самих вершин i, j проходит только через вершины $1 \ldots k$. Очевидно, что d_{ij}^{0} — длина (вес) ребра (i, j), если таковое существует (в противном случае его длина может быть обозначена как ∞).

Существует два варианта значения $d_{ij}^{k}, k \in (1, \ldots, n)$:

1. Кратчайший путь между i, j не проходит через вершину k, тогда $d_{ij}^{k} = d_{ij}^{k-1}$

2. Существует более короткий путь между i, j , проходящий через k, тогда он сначала идёт от i до k, а потом от k до j. В этом случае, очевидно, $d_{ij}^{k} = d_{ik}^{k-1} + d_{kj}^{k-1}$

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

Тогда рекуррентная формула для d_{ij}^{k} имеет вид:

d_{ij}^{0} — длина ребра (i, j);

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

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

Алгоритм Беллмана—Форда — алгоритм поиска кратчайшего пути во взвешенном графе. Алгоритм находит кратчайшие пути от одной вершины графа до всех

остальных. В отличие от алгоритма Дейкстры, алгоритм Беллмана—Форда допускает рёбра с отрицательным весом. Предложен независимо Ричардом Беллманом и Лестером Фордом [4]

Опишем алгоритм по шагам:

Шаг 0. Занумеруем все вершины из $G=(A,B)$, так что $A = \{1, 2, \ldots p\}$ и при этом номер «1» имеет именно та вершина, из которой будут найдены кратчайшие пути во все остальные вершины. Построим, далее матрицу

$M = (m_{ij})$, $i,j = 1, 2, \ldots p$, положив

$$m_{ij} = \begin{cases} f((i,\ j)),\ \textit{если}(i,\ j) \in B \\ \textit{клетка остается незаполненой, если}\ (i, j) \notin B \end{cases}$$

Шаг 1. Около первой строки матрицы M, слева от матрицы, поставим числовую пометку «0» и такую же пометку проставим над первым столбцом матрицы. Затем просмотрим помеченную строку слева направо и всякий раз, встречая клетку с числом, прибавим это число к пометке строки и сумму проставим над столбцом, в котором эта клетка находится. Затем отразим пометки столбцов относительно главной диагонали. Возникнут помеченные строки. С каждой из помеченных строк проделаем то же самое: просмотрим помеченную строку слева направо и всякий раз, встречая клетку с числом, прибавим это число к пометке строки и сумму поставим в качестве пометки над столбцом, в котором эта клетка стоит. При этом будем соблюдать принцип: «имеющуюся пометку не менять». Затем пометки столбцов отразим относительно главной диагонали и с помеченными строками вновь проделаем то же самое. И так далее, пока не окажутся помеченными все строки и все столбцы.

Шаг 2. Просмотрим строки таблицы в порядке возрастания их номеров. В каждой строке просматриваются клетки слева направо и всякий раз, когда встречается число, оно складывается с пометкой строки и сумма сравнивается с пометкой столбца, в котором найденное число расположено. Если сумма оказалась меньше, чем пометка столбца, то эта пометка столбца заменяется на упомянутую сумму. Если же сумма оказалась больше или равной пометке, то ничего не меняется. После такого просмотра всех строк новые пометки столбцов отражаются относительно главной диагонали и вся процедура повторяется. И так до тех пор пока не прекратятся какие бы то ни было изменения в пометках.

Шаг 3. Теперь по пометкам можно построить кратчайшие пути из первой вершины во все остальные. Фиксируем произвольную вершину k $(k = 2,3,\ldots p)$ и опишем кратчайший путь из первой вершины в вершину k. Во-первых, длина этого кратчайшего пути равна пометке λ_k, стоящей над столбцом номер k. Во-вторых, предпоследняя вершина в кратчайшем пути из первой вершины в вершину k находится так: в столбце номер k отыскиваем число, сумма которого с пометкой строки, в которой оно расположено, равна λ_k. Пусть номер строки, в которой найденное число оказалось, равен l. Тогда предпоследней вершиной в кратчайшем пути из 1 в k будет вершина l. Вершину, которая предшествует вершине l, надо отыскивать как предпоследнюю в кратчайшем пути из 1 в l и так далее.

Приведем пример. Пусть исходный взвешенный граф дает следующую матрицу M:

	10	13					
10						11	
13			17	10			
		17			16	12	
		10			11		15
			16	11		15	10
	11		12		15		
				15	10		

Начнем расставлять пометки:

	0	10	13	30	23	46	21	38
0		10	13					
10	10						11	
13	13			17	10			
30			17			16	12	
23			10			11		15
46				16	11		15	10
21		11		12		15		
38					15	10		

Проведем уменьшение пометок:

		0	10	13	30	23	34	21	38
		0	10	13	30	23	46	21	38
0	0		10	13					
10	10	10						11	
13	13	13			17	10			
30	30			17			16	12	
23	23			10			11		15
34	46				16	11		15	10
21	21		11		12		15		
38	38					15	10		

Укажем кратчайшие пути:
1→2: длина 10; путь (от конца к началу) 2←1;
1→3: длина 13; путь (от конца к началу) 3←1;
1→4: длина 30; путь (от конца к началу) 4← 3←1;
1→5: длина 23; путь (от конца к началу) 5←3←1;
1→6: длина 34; путь (от конца к началу) 6←7←2←1;
1→7: длина 21; путь (от конца к началу) 7←2←1;
1→8: длина 38; путь (от конца к началу) 8←5←3←1.

Итоги анализа. Проведя анализ трех алгоритмов, можно сделать выводы о целесообразности использовании данных алгоритмов к поставленной задаче.

- ***Алгоритм Дейкстры***

Этот алгоритм является самым простым алгоритмом из исследуемых. Он хорошо выполняется в графах с небольшим количеством вершин, схема метрополитена не является таковым. Учитывая структуру схемы московского метрополитена, программа, написанная с использованием алгоритма Дейкстры будет выполняться медленно, а распределение памяти не будет рациональным.

- ***Алгоритм Флойда-Уоршелла***

Второй исследуемый алгоритм — алгоритм Флойда-Уоршелла, содержит три вложенных цикла, то есть имеет кубическую сложность, количество вершин графа (станций метрополитена) — 177, поэтому при использовании этого алгоритма будет использоваться большой объем памяти, что является не рациональным решением.

- ***Алгоритм Беллмана-Форда***

Алгоритм находит кратчайшие пути от одной вершины графа до всех остальных. Условие задачи — нахождение кратчайшего пути между двумя станциями метро является несколько другой задачей. Алгоритм использует полный перебор всех вершин графа, что приведет к большой потери времени и займет больший объем памяти.

В заключение отметим общие недостатки рассматриваемых методов:

Все 3 метода требуют полного перебора всех вершин графа. Требуется большой объем памяти и время расчета.

Поскольку схема метрополитена представляет собой разветвленный граф, то с целью минимизации времени расчета, был разработан алгоритм, осуществляющий перебор вершин лишь нескольких подграфов, которые определяются условиями взаимодействия ветвей графа (линий метрополитена).

Описание алгоритма

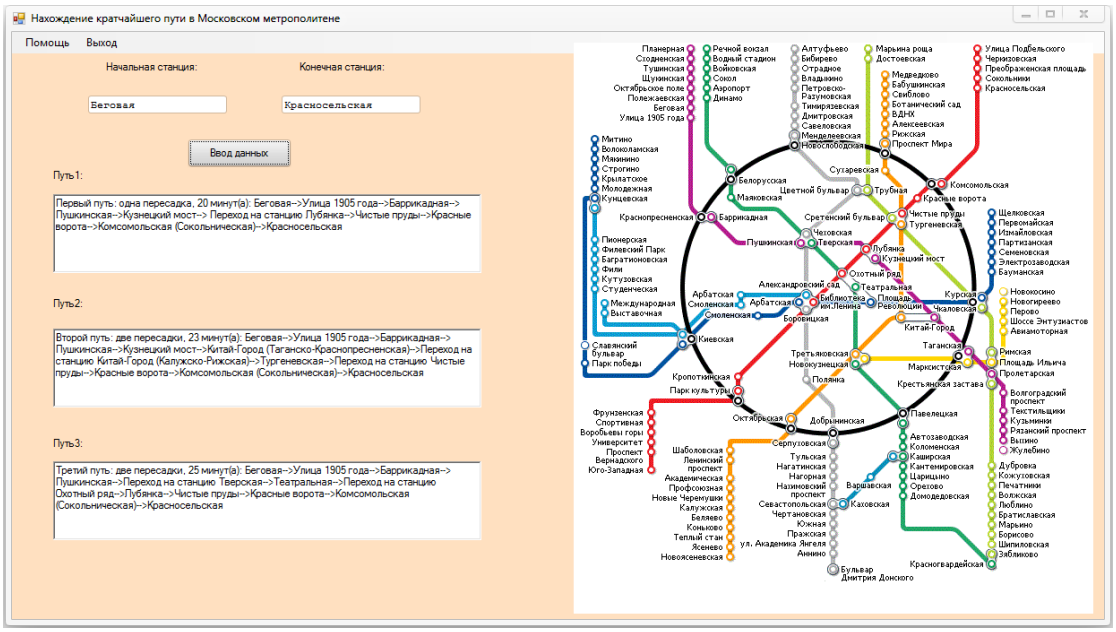
1. Ввод начальных данных (весовые функции, функции взаимодействия линий, метки вершин графа)
2. Проверка принадлежности начальной и конечной станций к той или иной линии
3. Нахождение взаимодействия отмеченных линий (ветвей графа):

- *Программа выбирает из всех возможных путей несколько наиболее коротких, учитывая, что пользователю может быть удобно ехать как с одной, так и с двумя пересадками.*
- *При расчете пути программа рассматривает случаи движения как через кольцо, так непосредственно через все возможные ветки, в том числе, если ветки начальной и конечной станции пересекаются.*
- *Программа выбирает наиболее близкую пересадку на кольцо, если ветка начала или конца пути взаимодействует с кольцом дважды.*
- *Если текущий путь идет через кольцо, программа также выбирает, в какую сторону нужно двигаться, чтобы рационально распределить время.*

4. Выбор оптимальных взаимодействий (определение подграфов)
5. Расчет путей в выбранных подграфах
6. Выбор кратчайших путей

По разработанному алгоритму была написана программа в среде Microsoft Visual Studio 2005 на языке C++.

Пример работы программы



Литература

1. E. W. Dijkstra. A note on two problems in connexion with graphs. // Numerische Mathematik. V. 1 (1959), P. 269-271
2. *Ананий В. Левитин* Глава 8. Динамическое программирование: Алгоритм Флойда поиска кратчайших путей между всеми парами вершин // Глава 9. Жадные методы: Алгоритм Дейкстры // Алгоритмы: введение в разработку и анализ = Introduction to The Design and Analysis of Aigorithms. — М.: Вильямс , 2006. — С. 189—195, С. 349 — 353.

3. *Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн* Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е изд. — М.: Вильямс, 2006. — С. 1296.
4. http://comp-science.narod.ru/KPG/BelmanFord.htm
5. *Герберт Шилдт.* Полный справочник по C++ = C++: The Complete Reference. — 4-е изд. — М.: Вильямс , 2006. — 800 с.