УДК 681.3

Винничук С.Д.

ОПРЕДЕЛЕНИЕ КРАТЧАЙШИХ ПУТЕЙ ИЗ ОДНОЙ ВЕРШИНЫ С ПОМОЩЬЮ АЛГОРИТМА ПОИСКА В ШИРИНУ

Аннотация:

Предложен способ определения кратчайших путей из одной вершины взвешенного графа на основе использования алгоритма поиска в ширину. Приведены оценки вычислительной сложности алгоритма и затрат памяти.

Анотація:

Запропоновано спосіб знаходження найкоротших шляхів з однієї вершини зваженого графа на основі використання алгоритму пошуку в ширину. Наведені оцінки обчислювальної складності та затрат пам'яті.

Abstract:

A method for determining the shortest path from one vertex of a weighted graph using the search algorithm in width. Gives estimates of the computational complexity of the algorithm and the cost of memory.

Постановка задачи

В задаче поиска кратчайших путей полагаются известными множества вершин и ребер ориентированного или неориентированного графа $G(V,E)$ (V – множество вершин, E – множество ребер), а также вес ребер, где значение веса выражается действительным числом. Существует ряд задач поиска кратчайших путей из одной вершины, отличающихся своей постановкой, где такие отличия состоят в следующем:

- является ли граф ориентированным или неориентированным;
- является ли граф ациклическим или содержит циклы;
- принимают ли веса ребер только положительные значения или возможны и их отрицательные значения;
- принимают ли веса ребер только целочисленные значения;
- выражаются ли веса ребер малыми неотрицательными значениями.

К числу основных способов их решения относятся [1]:

- алгоритм Беллмана – Форда для общего случая, когда вес каждого из ребер может быть отрицательным, с трудоемкостью $O(V \cdot E)$, где дополнительно определяется наличие цикла с отрицательным весом;
- алгоритм поиска в ширину для случая ориентированных ациклических графов с трудоемкостью $O(V+E)$;
- алгоритм Дейкстры для произвольных графов с неотрицательными весами ребер, где трудоемкость определяется в зависимости от способа выборки помеченной вершины с минимальным весом и при использовании пирамид Фибоначчи может достигать величины порядка $O(V\lg V+E)$.

Существует и ряд других алгоритмов решения задачи поиска кратчайших путей, но для всех них трудоемкость оценивается величиной, превосходящей порядок $O(V+E)$, как в алгоритме поиска в ширину. Поэтому представляет интерес задача разработки и анализа

алгоритма поиска кратчайшего пути на основе алгоритма поиска в ширину, чему посвящено настоящее исследование.

Алгоритм А1 определения кратчайшего пути на основе поиска в ширину для положительных весов ребер

Пусть ориентированный (или неориентированный) взвешенный граф задается в виде списка смежности, кроме того, вместе с каждым ребром хранится его произвольный неотрицательный вес так, что этот вес при просмотре списка смежности можно определить за время $O(1)$. Пусть также в процессе работы алгоритма вершины будут окрашиваться в белый, серый и черный цвета, причем по ходу работы алгоритма в серый цвет могут окрашиваться как белые, так и черные вершины.

В основе работы алгоритма заложен базовый процесс ослабления (см. [1], процедура $RELAX(u,v,w)$) – проверка возможности улучшения найденного до этого кратчайшего пути к рассматриваемой вершине. Если $d(u)$ ($d(v)$) – расстояние до вершины u (v), а w – вес ребра (u,v), то ослабление (уменьшение длины пути (или уменьшение веса)) имеет место тогда, когда выполнено условие

$$d(u) > d(v) + w(u,v). \qquad (1)$$

Такую проверку предлагается реализовывать для узлов из организуемой очереди их просмотра с принципом работы – «первый пришел – первый ушел». Для реализации очереди будем использовать массивы $Q[V]$ и $Q1[V+1]$ длиной V и V+1 соответственно, где сама очередь реализуется с помощью массива $Q[V]$, а в массиве $Q1[V]$ отображается информация о состоянии вершины (белая – 0, серая – 1, черная – 2). Правило реализации очереди аналогичное описаному в методе поиска в ширину (см. [1]), но имеются и отличия, которые будут описаны ниже.

Опишем предлагаемый алгоритм **А1** последовательностью шагов.

0. На начальном шаге алгоритма все вершины окрашены в белый цвет, а начальная вершина – в серый. Голове очереди соответствует переменная head, а хвосту – tail. Начальная вершина помещается в начало очереди по работе с серыми вершинами, т.е. первоначально $tail$=0, а $head$=1. Для всех вершин определена величина веса, равная бесконечности, а для начальной вершины – 0.

1. Анализ узла $u=Q[tail]$. Если u = 0, то очередь исчерпана ($tail$ = $head$) и завершение работы алгоритма, а иначе переход к п.2.

2. Анализ вершин v смежных вершине u:

2.1. Если $Q1[v] = 0$, то: $Q[head]=v$; $Q1[head]=1$; $head = (head+1)(modV)$; $d(u) = d(v) + w(u,v)$.

2.2. Если $Q1[v] = 2$ и выполнено условие (1), то: $Q[head]=v$; $Q1[head]=1$; $head = (head+1)(modV)$; $d(u) = d(v) + w(u,v)$.

2.3. Если $Q1[v] = 1$ и выполнено условие (1), то: $d(u) = d(v) + w(u,v)$.

2.4. Если $Q1[v] > 0$, но не выполнено условие (1), то: вес вершины u не меняется как и параметры очереди.

По завершению цикла: $Q[tail]=0$; $Q1[tail]=2$; $tail = (tail+1)(modV)$.

3. Переход к п.1.

Работа алгоритма **А1** иллюстрируется на рис. 1, где отображается изменение веса вершин, а также информация об изменении данных массива Q по результатам обработки соответствующего узла. При этом: на рис. 1 а) представлен результат обработки узла 1; на рис. 1 б) – узла 3; на рис. 1 в) – узла 5; на рис. 1 г) – узла 2; на рис. 1 д) – узла 4; на рис. 1

е) – повторно узла номер 5. Следует отметить также, что при обработке узла 3 улучшается вес для узла 5 с сохранением цвета вершины, а при обработке узла 4 при улучшении веса для узла 5 меняется ее цвет с черного на серый.

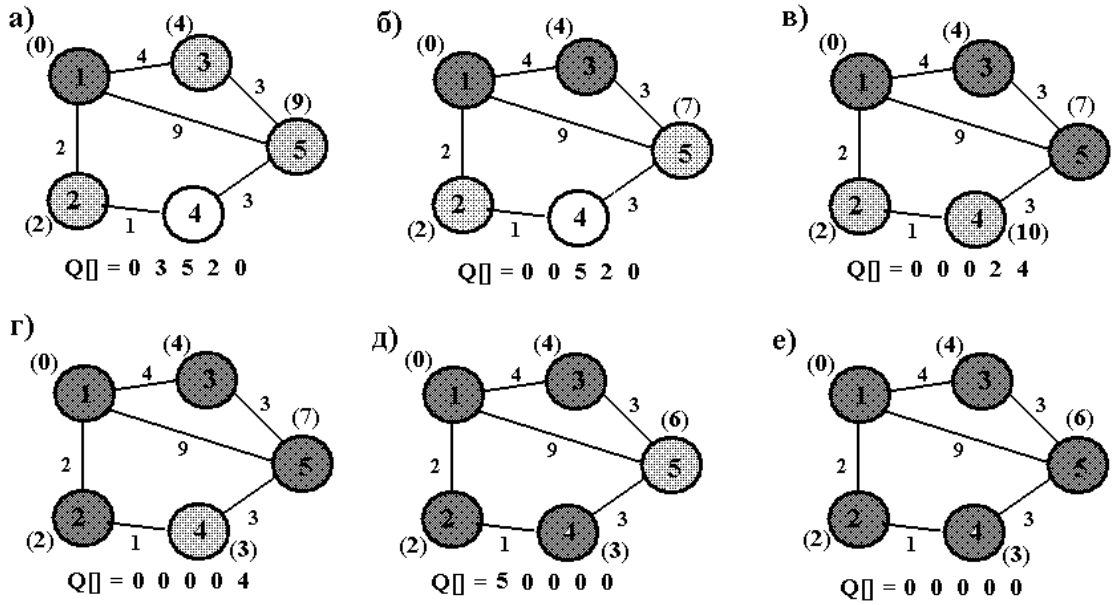

Рис. 1. Пример работы алгоритма **A1**

Анализ алгоритма A1

Алгоритм A1 обеспечивает определение кратчайших путей от заданной вершины до всех остальных. Такое утверждение вытекает из следующих положений.

Для случая положительных весов доказано (см. [1]), что алгоритм Дейкстры обеспечивает определение кратчайших путей и существует корневое дерево кратчайших путей с корнем в начальной вершине. Пусть h – высота дерева.

Представим алгоритм **A1** последовательностью укрупненных шагов, которые строятся по следующему правилу (по аналогии с алгоритмом поиска в ширину):

- на начальном укрупненном шаге k=0 анализируется начальная вершина;

- на каждом k+1 – ом шаге анализируются все серые вершины, определенные на k – ом шаге.

Возвращаясь к остовному дереву кратчайших путей, найденному с помощью алгоритма Дейкстры, заметим, что вершины k+1 – го шага являются смежными вершинам k – го шага, которые в случае алгоритма Дейкстры получаются из анализа одной вершины (с наименьшим весом), а в алгоритме **A1** кроме одной этой вершины анализируются и все остальные серые вершины. Кроме того, серые вершины k+1 – го шага получаются только в случае уменьшения их веса при анализе вершин k – го шага. Но при конечном числе ветвей получается конечное число вариантов путей от исходной вершины к текущей, а, значит, и конечное число раз возможно уменьшение веса произвольной из вершин. Следовательно, алгоритм A1 определяет кратчайшие пути из одной вершины.

Оценим трудоемкость алгоритма **A1**. В нем допускается возможность многократного анализа одной и той же вершины, т.е. по суммарному количеству анализов вершин он, как правило, превосходит алгоритм Дейкстры. Но в нем не требуется сортировки вершин по возрастанию весов. Поэтому возможны случаи, когда алгоритм **A1** оказывается эффективнее алгоритма Дейкстры. В частности, для случаев графов, не содержащих циклов. А также для произвольных неориентированных графов, для которых

остовное дерево кратчайших путей, построенное с помощью алгоритма Дейкстры, совпадает с остовным деревом, построенным с помощью алгоритма поиска в ширину.

При оценке трудоемкости алгоритма **A1** требуется определить суммарное количество анализов узлов T. Нижняя оценка такого числа это собственно число узлов V. Тогда трудоемкость оценивается величиной порядка $O(V+E)$.

При получении верхней оценки будем исходить из следующих положений:

- остовное дерево кратчайших путей имеет максимальную высоту h_{max};
- на каждом укрупненном $k-$ом шаге число серых вершин N_k не превышает V-k;
- для всех V узлов неориентированного графа существует только $2 \cdot E$ смежных к ним.

Следовательно, общее суммарное количество анализов узлов (шаг 2 алгоритма A1) число T может быть оценено сверху

$$T = \sum_{k=1}^{h_{\max}} N_k < \sum_{k=1}^{V-1} (V-k) < V^2/2 .$$

Поэтому в предельном наиболее сложном случае с учетом того, что для всех V узлов неориентированного графа существует только $2 \cdot E$ смежных к ним, общая трудоемкость грубо может быть оценена величиной порядка $O(V \cdot E)$, т.е. не превосходит по трудоемкости алгоритм Беллмана-Форда.

Алгоритм A2 для произвольных весов ребер

Если существуют кратчайшие пути на графе, то существует и дерево кратчайших путей. При этом веса ребер могут быть как положительными, так и отрицательными. Построение же такого дерева возможно в том числе с использованием алгоритма поиска в ширину, аналогично тому, как это предлагается в алгоритме **A1**. Но для случая, когда ребра могут иметь отрицательный вес, необходимо определять факт наличия циклов с отрицательным весом. А поскольку в случае неориентированных графов при наличии всего одной ветви с отрицательным весом образуется такой цикл, то задача поиска кратчайших путей с произвольными весами ребер имеет смысл только для ориентированных графов.

Для определения факта наличия циклов с отрицательным весом можно предложить следующие признаки:

1. серые вершины появляются даже после анализа $V^2/2$ узлов алгоритмом **A1** ;
2. вес одной из вершин оказался ниже суммы весов всех ребер с отрицательным весом - величины W.

Для реализации первого из признаков достаточно в алгоритме **A1** добавить счетчик числа проанализированных узлов и предусмотреть выход из расчета как только он превысит значение $V^2/2$.

Для реализации второго из признаков следует определить число W и сравнивать его каждый раз с улучшенным значением веса вершины на шагах 2.2 и 2.3 алгоритма **A1**. В этом случае на основании списка предшествования возможно определение одного из циклов с отрицательным весом.

Усовершенствованный алгоритм **A1** обозначим как алгоритм **A2**.

Следует также отметить, что для реализации алгоритма **A1** и его усовершенствований для случая отрицательных весов ребер (алгоритм **A2**) требуется выделение памяти для хранения информации списка смежности для узлов графа и синхронной информации о весах ребер, а также объема памяти порядка числа узлов для

массивов $Q[V]$, $Q1[V+1]$ и $d[V+1]$. Следовательно, суммарный объем памяти, необходимый для реализации алгоритмов **A1** и **A2,** линейно зависит от числа узлов и ребер.

Выводы

Предложены простые в реализации алгоритмы поиска кратчайших путей из одной вершины, позволяющие определять их как для неотрицательных весов ребер (произвольные графы) (алгоритм **A1**), так и для ориентированных графов с произвольными весами ребер (алгоритм **A2**).

Показано, что существуют случаи, когда по трудоемкости алгоритм **A1** оказывается более эффективным чем алгоритм Дейкстры, хотя в предельном случае его трудоемкость может быть оценена величиной порядка $O(V \cdot E)$.

Алгоритм **A2** в случае наличия ребер с отрицательным весом в ориентированном графе позволяет определять факт наличия циклов с отрицательным весом.

Показано также, что объем памяти, необходимый для реализации алгоритмов **A1** и **A2** линейно зависит от числа узлов и числа ребер графа.

Литература:

1. Кормен Т. Алгоритмы: построение и анализ / [Кормен Т., Лейзерсон Ч., Риверст Р., Штайн К.]. – [2-е изд.]. – М. : Издательский дом “Вильямс”, 2011. – 1296 с.