Сегодня информационные технологии плотно вошли в нашу повседневную жизнь: дома, в школе, на транспорте, на предприятии. При разработке любого программного обеспечения высчитывается математическая сложность алгоритмов для примерной оценки скорости работы программ. Но действительно ли время работы зависит от математической сложности алгоритма? В этом надо разобраться.

Для этого возьмём три наиболее популярных алгоритма поиска кратчайших путей в полном взвешенном орграфе: алгоритм Дейкстры, алгоритм Форда-Беллмана и алгоритм Флойда-Уоршалла. Первоначально вспомним определения из теории графов [2].

*Граф* *G* – это совокупность двух множеств *V* и *E*: где *V* – множество (оно называется основным), а *E* – множество двухэлементных подмножеств множества *V*. Элементы основного множества *V* называются *вершинами*. Элементы множества *E* называются *ребрами*. Граф *G* называется *графом со взвешенными ребрами*, если задано отображение *Е* на множество действительных чисел: Действительные числа, характеризующие каждое ребро, называются *весами ребер*. Граф *Kn* с *n* вершинами называется *полным графом* или *кликой*, если каждую пару его вершин соединяет ребро. *Орграф* (ориентированный граф) *G* – это граф, у которого *E* - множество упорядоченных двухэлементных подмножеств множества *V*. *Матрица смежности* [1] орграфа *G=(V,E)* с *n* вершинами *V={v1,…,vn}* называется булева матрица *W* размера *n×n* с элементами

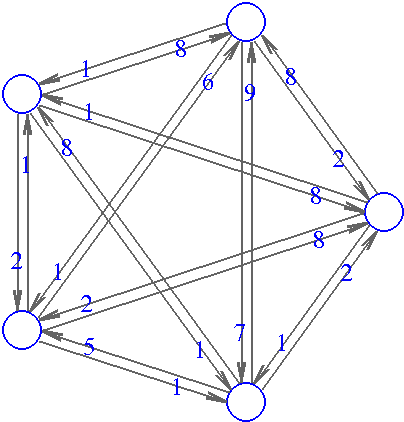


Рис.1 Взвешенный орграф.

(1)

Однако во взвешенных графах вместо 1 записывают значение веса соответствующего ребра. К примеру, матрица смежности графа на рисунке 1 следующая:

(2)

Разберемся, что собой представляет каждый из вышеназванных алгоритмов.

1. Алгоритм Дейкстры – это алгоритм поиска кратчайших расстояний от заданной вершины до всех остальных вершин взвешенного графа без рёбер отрицательного веса. Данный алгоритм был разработан в 1959 году нидерландским учёным Э́дсгером Ви́бе Де́йкстрой. В настоящее время алгоритм очень популярен, его можно использовать, например, при планировании путешествий. В информационных технологиях, кроме всего прочего, этот алгоритм используется в протоколе динамической маршрутизации OSPF для устранения кольцевых маршрутов.

Пример кода на языке Python 3.2 (W-матрица смежности, N-число вершин, start-номер стартовой вершины):

def Dijkstra(W,N,start):

INF = 10\*\*10

F = [INF]\*N

F[start] = 0

V = [True]\*N

while True:

min\_R = INF

min\_v = None

for i in range(N):

if V[i] and F[i] < min\_R:

min\_R = F[i]

min\_v = i

if min\_v is None:

break

for j in range(N):

if F[j] > F[min\_v]+W[min\_v][j]:

F[j] = F[min\_v]+W[min\_v][j]

V[min\_v] = False

return F

Алгоритм устроен следующим образом: каждой вершине графа в соответствие ставится определённое число *F[i]* – кратчайший из известных путей. Изначально значение для стартовой вершины принимается за 0, а значение для всех остальных вершин – бесконечность (расстояние до них неизвестно). Просматриваются все вершины, и находится вершина с наименьшим значением *F[i]*. Это будет реальное кратчайшее расстояние до данной вершины от стартовой т.к. если оно проходит через другие вершины, то расстояние до них должно быть меньше (рёбра имеют только положительные веса). Далее

просматриваются все рёбра из этой вершины и, если это возможно, меняется предполагаемое кратчайшее расстояние *F[j]* до следующих вершин. Данная вершина помечается как просмотренная *V[i]*. Далее просматриваются все непомеченные вершины по тому же алго- ритму, до тех пор, пока все вершины не станут помеченными. Функция возвращает список *F*.

Математическая сложность алгоритма Дейкстры: *O(N2)* т.к. цикл *while* выполнится *N* раз по количеству вершин, а внутри него два цикла *for*, которые выполняются ровно *N* раз. Значит максимальная вложенность циклов – 2, и каждый цикл выполняется ровно *N* раз, т.е. *N\*N* *=* *N2*.

1. Алгоритм Форда-Беллмана – это динамический алгоритм поиска кратчайших расстояний от заданной вершины до всех остальных верши для любых взвешенных графов, включая графы с рёбрами отрицательного веса. Этот алгоритм был предложен в 1969 году независимо друг от друга двумя американскими математиками Ричардом Беллманом и Лестером Фордом.

Пример реализации на языке Python 3.2 (W - матрица смежности, N – число вершин, start – номер стартовой вершины):

def F\_B(W,N,start):

INF = 10\*\*10

F = [INF]\*N

F[start] = 0

for k in range(N-1):

for i in range(N):

for x in range(N):

if F[i] > F[x]+W[x][i]:

F[i] = F[x]+W[x][i]

return F

Алгоритм динамический и устроен следующим образом: каждой вершине в соответствие ставится число – минимальное известное расстояние до неё *F[i]*, изначально до всех вершин бесконечность, а до стартовой- *0*. Динамика идёт по параметру *k* максимальное число вершин, через которые может идти путь из стартовой вершины в заданную. Рекуррентное соотношение представлено формулой:

(3)

Поясню эту формулу: кратчайшее расстояние до вершины *i* через *k* вершин – это кратчайшее расстояние из двух вариантов: минимальное расстояние до какой-либо другой вершины *x* из путей, проходящих через *k-1* вершину, плюс расстояние от вершины *x* до вершины *i* или минимальное расстояние до *i* через *k-1* вершину. Очевидно, что можно не сохранять всю двухмерную матрицу т.к. по индексу *k* используется только *k-1* значение, следовательно, можно изменять значения известного кратчайшего расстояния в одном списке (если значение до его использования было изменено, то оно меньше либо равно предыдущему значению, а нужно минимальное). Тогда рекуррентное соотношение:

. (4)

Сколько же раз надо запускать этот алгоритм? Для этого надо вспомнить с чего мы начинали. С параметра *k* - через сколько вершин можно пройти по пути от одной вершины до другой? *N-2* вершины, но первый запуск всё равно, что по пути пройти через *0* вершин, значит надо запустить *N-1* раз. Функция возвращает список *F*.

Математическая сложность алгоритма Форда-Беллмана для графов, представленных матрицей смежности: *O(N3)*, т.к. первый цикл выполняется *N-1* раз второй *N* раз и третий *N* раз, т.е. сложность алгоритма *O((N-1)\*N\*N)* или *O(N3)*.

1. Алгоритм Флойда-Уоршалла - это динамический алгоритм поиска кратчайших путей во взвешенном графе между любыми двумя вершинами. Предложен в 1962 году американским учёным Робертом Флойдом и Стивеном Уоршаллом.

Алгоритм динамический и устроен следующим образом каждой паре вершин *x,y* ставится в соответствие число *F[x][y]* – кратчайшее из известных расстояний между ними, причём расстояние от *x* до *y* не равно расстоянию от *y* до *x* т.к. граф взвешенный. Изначально расстояние между любыми двумя вершинами равно весу ребра между этими вершинами. Динамика идёт по параметру *k* – минимальный номер вершины, через которые пролегает путь из одной вершины в другую. Рекуррентное соотношение:

. (5)

Поясню формулу: минимальное расстояние от вершины *x* до вершины *y*, через вершины, номера которых меньше либо равны *k* – это минимум из расстояния от *x* до *y*через вершины, номера которых меньше либо равны *k-1* и расстояния от *x* до *k* плюс расстояние от *k* до *y*, через вершины, номера которых меньше либо равны *k-1*. Как и в алгоритме Форда-Беллмана не обязательно учитывать параметр *k*, следовательно, рекуррентное соотношение:

(6)

Очевидно, что запускать алгоритм надо столько раз, сколько вершин в графе.

Пример реализации на языке Python3.2 (W-матрица смежности, N-количество вершин):

import copy

def F\_U(W,N):

F = copy.deepcopy(W)

for k in range(N):

for x in range(N):

for y in range(N):

F[x][y] = min(F[x][y],F[x][k]+F[k][y])

return F

Алгоритм возвращает двумерную матрицу *F* – матрицу минимальных расстояний между любыми двумя вершинами графа. Сложность алгоритм Флойда-Уоршалла: *O(N3)*, т.к. тройная вложенность циклов каждый из который выполняются по *N* раз.