Оценка времени работы алгоритмов:

1. **Нахождение диаметра сети (без циклов)**

Рассмотрим сеть (граф) G(V,E), V – множество вершин, E – множество ребер (через списки смежности). В программе 2 раза используется алгоритм BFS (это удобно, т.к отсутствуют циклы) – оценим его сложность.

Т.к. каждая вершина может быть помещена в очередь только один раз, а операции внесения и удаления из очереди имеют сложность O(1), для всех вершин оценка дает O(|V|). Затем для каждой вершины необходимо подсчитать имеющиеся у нее ребра, считая все вершины в графе, это количество должно быть не более 2|E| раз (т.к. ребро будет учитываться для каждой вершины). Получаем, что оценка для этих операций – O(|E|). Таким образом, общее время работы алгоритма BFS: O(|V| + |E|).

Все остальные операции (кроме BFS) занимают время O(1).

Таким образом, сложность этого алгоритма составляет **O(|V| + |E|)**.

1. **Нахождение диаметра сети (с циклами)**

Рассмотрим сеть (граф) G(V,E), где V – множество вершин, E – множество ребер,  в котором вершины пронумерованы от 1 до N. Граф представим через модифицированную матрицу смежности.

Итак, оценим сложность алгоритма. Алгоритм Флойда-Уоршелла выполняется за O(N^3) времени, т.к. содержит в себе три вложенных друг в друга цикла. При поиске максимальных значений прохождение двумерного массива размером NxN, учитывая то, что внешний цикл включает N итераций, и что внутренний так же выполняется N раз, определяет сложность O(N^2).

Таким образом, общая сложность алгоритма составляет **O(N^3)**.

1. **Какая сложность у оптимального алгоритма нахождения диаметра произвольной сети?**

Алгоритм: для всех вершин N неориентированного невзвешенного графа необходимо найти длины кратчайших путей, а после выбрать максимальное из найденных расстояний.

Наиболее эффективный алгоритм выполняется за O()) (ранее показатель был равен 2.376). Он оценивается поиском длин кратчайшего пути для всех пар неориентированного невзвешенного графа с N вершинами, где – время, необходимое для умножения двух матриц целых чисел размером NxN.

Пусть G - неориентированный невзвешенный граф с N вершинами, где A(NxN)– его матрица смежности со значениями ячеек 0 или 1, D(NxN) – матрица расстояний (– число ребер на кратчайшем пути, соединяющем вершины i и j).

Вычислим матрицу расстояний D графа G. Вызываем distance\_matrix(A).

Псевдокод:

def distance\_matrix(adjacency\_matrix):

Z = A\*A

B[N][N], where

If i≠j and (:

else:

D[N][N]

if

D = 2B - A

return D

T = distance\_matrix(B)

X = T\*A

return D, where

:

1

Пояснение: B – матрица смежности для графа G’, где G’ – простой неориентированный n-вершинный граф, полученный из G путем соединения каждых 2-х вершин i,j ребром, когда ꓱ путь длины 1 или 2 между i,j в G. Тогда – длина кратчайшего пути, соединяющего i,j в G’.

Приведем оценку временных параметров:

Пусть f(N, c) – время работы алгоритма distance\_matrix, где N – кол-во вершин графа G, а δ – его диаметр. Заметим, что G’ имеет диаметр [δ /2].

Тогда

, где – время умножения двух матриц NxN.

Значит, для :

При поиске максимальных значений прохождение двумерного массива D размером NxN, учитывая то, что внешний цикл включает N итераций, и что внутренний так же выполняется N раз, определяет сложность O(N^2).

Зная, что , зная M(N), а также уточнив, что практическую значимость данный алгоритм даст только на очень больших сетях, можем сказать, что сложность алгоритма составляет .

Дополнительная информация:

Откуда взялись истоки алгоритма? Дело в том, что существует очень тесная связь между вычислением кратчайших путей в графе и вычислительной мощностью квадратной матрицы. Так, например, можно увидеть это, сравнив алгоритм возведения в квадрат матрицы A размера nxn (слева) с внутренним циклом алгоритма кратчайшего пути (справа).

Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст

Автоматически созданное описание

Единственное различие между этими двумя алгоритмами состоит в том, что второй алгоритм использует сложение вместо умножения и минимизацию вместо сложения.

Таким образом, одним из лучших является простой рандомизированный алгоритм, открытый в 1991 Цви Галилом и Одедом Маргалитом и дополнительно упрощенный в 1992 Раймундом Зайделем, который вычисляет расстояния кратчайших путей для всех пар в невзвешенных неориентированных графах за ожидаемое время O(M(V)log V), где M(n)— это время, необходимое для умножения двух целочисленных матриц размера nxn. В настоящее время этот подход активно разрабатывается, посредством улучшения, времени матричного умножения. Например, это значение улучшилось в 2014 году с до , что можно прочитать в соответствующей статье - http://theory.stanford.edu/~virgi/matrixmult-f.pdf.