Испытания по производительности и памяти проводились на оборудовании

**Процессор** AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.10 GHz

**Оперативная память** 8,00 ГБ

**Тип системы** 64-разрядная операционная система

**Операционная система** Windows 11

Были проанализированы 4 алгоритма

Во всех последующих выкладках по оценке сложности будут использоваться переменные:

* n – количество вершин
* m – количество ребер

Алгоритм Дейкстры

Входные данные: граф, начальная вершина

Выходные данные: массив, в котором находятся минимальные расстояния от начальной вершины до всех остальных

Описание алгоритма

Каждой вершине из *n* сопоставим метку — минимальное известное расстояние от этой

вершины до первоначальной.

Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.

Алгоритм завершается, когда все вершины посещены.

Сложность

* Время работы: O(n\*\*2+m)
* Затрачиваемая память: O(n\*\*2), т. к. храним матрицу размера n\*n

Алгоритм Флойда

Входные данные: граф

Выходные данные: матрица n\*n, в которой находятся минимальные расстояния между двумя любыми вершинами.

Описание алгоритма

Каждой вершине из *n* сопоставим метку — минимальное известное расстояние от этой

вершины до первоначальной.

Алгоритм работает пошагово — на каждом шаге он «посещает» одну вершину и пытается уменьшать метки.

Алгоритм завершается, когда все вершины посещены.

Сложность

* Время работы: O(n\*\*3), т. к. всегда проходим 3 вложенных цикла длинной n
* Затрачиваемая память: O(n\*\*2), т. к. храним матрицу размера n\*n

Алгоритм поиска в ширину (BFS)

Входные данные: граф, начальную и конечную вершину

Выходные данные: последовательность вершин, по которым можно пройти из начальной вершины до конечной, причем путь будет кратчайшим

Описание алгоритма

В очередь помещается начальная вершина, затем на каждом шаге из очереди

извлекается, добавленная раньше всех вершина, из нее просматриваются все рёбра,

если ребро ведет в вершину, которая еще не добавлялась в очередь, то она добавляется

в очередь.

Как только метод доходит до конечной вершины, он восстанавливает путь и

заканчивает работу.

Сложность

* Время работы: алгоритм посетит не более n вершин, и пройдет по ребру не

более одного раза, поэтому О(n + m)

* Затрачиваемая память: максимальное число вершин, одновременно

хранящихся в очереди — n, поэтому O(n)

Алгоритм поиска в глубину

Входные данные: граф, начальная и конечная вершина

Выходные данные: массив - последовательность вершин, по которым можно пройти из начальной вершины до конечной

Описание алгоритма

Работает рекурсивно. Из вершины, в которой сейчас находится, пытается перейти в

другие смежные вершины, в которых еще не был. Также ведется запись предков для

вершин, чтобы в конце восстановить путь.

Когда алгоритм доходит до конечной вершины, он восстанавливает путь и прекращается.

Сложность

* Время работы: алгоритм посетит не более n вершин, и пройдет по ребру не

более одного раза, поэтому О(n + m)

* Затрачиваемая память: глубина рекурсии не превышает общего числа вызовов

функции dfs — числа вершин, поэтому O(n)

Сравнение

Тест 1:

Протестируем алгоритмы на случайном неориентированном графе, который содержит 500 вершин.

Тест 2:

Протестируем алгоритмы на случайном ориентированном графе, который содержит 500 вершин.

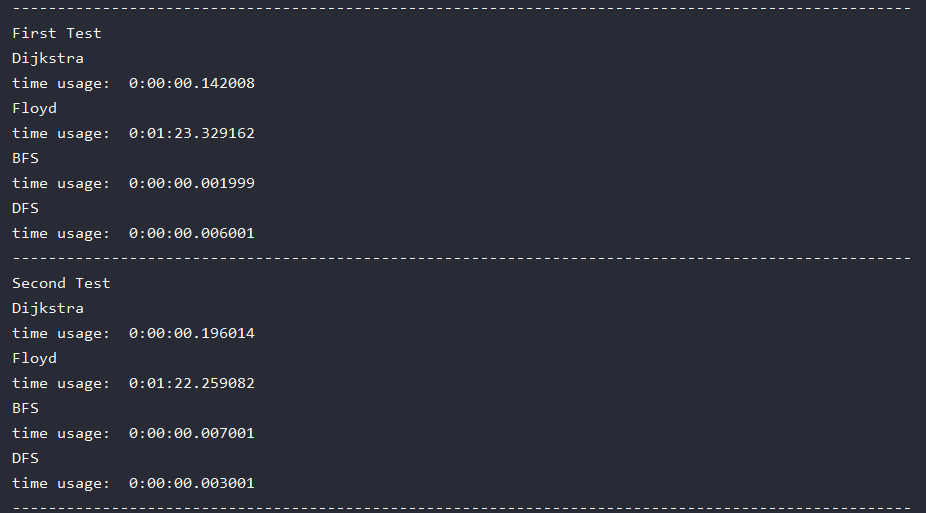
Тест 3:

Протестируем алгоритмы на графе, который является цепью и содержит 500 вершин.

Тест 4:

Протестируем алгоритмы на полном графе, который содержит 100 вершин.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Тест 1** | **Тест 2** | **Тест 3** | **Тест 4** |
| **Алгоритм** | **Время, с** | **Время, с** | **Время, с** | **Время, с** |
| Флойд | 83.329162 | 82.259082 | 48.912629 | 55.923123 |
| Дейкстра | 0.142008 | 0.196014 | 0.128010 | 0.005999 |
| DFS | 0.006001 | 0.003001 | 0 | 0.001000 |
| BFS | 0.001999 | 0.007001 | 0.002000 | 0.001000 |



Изображение выглядит как текст

Автоматически созданное описание