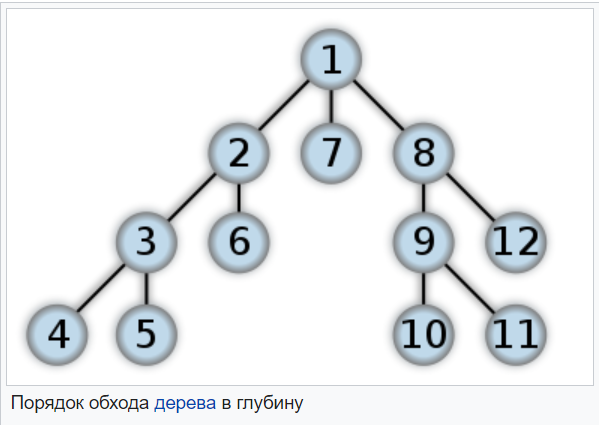
Отчёт

Описание тестируемых алгоритмов

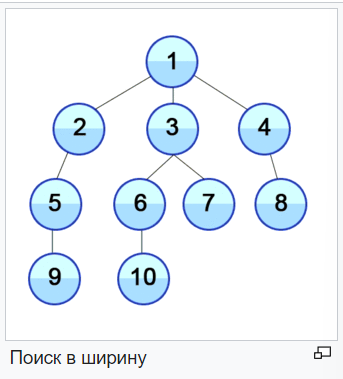
DFS

Один из методов обхода графа. Depth-first search – стратегия поиска в глубину: алгоритм устроен так, чтобы идти «вглубь» графа, насколько это возможно. Алгоритм поиска описывается рекурсивно: перебираем все исходящие из рассматриваемой вершины рёбра. Если ребро ведёт в вершину, которая не была рассмотрена ранее, то запускаем алгоритм от этой нерассмотренной вершины, а после возвращаемся и продолжаем перебирать рёбра. Возврат происходит в том случае, если в рассматриваемой вершине не осталось рёбер, которые ведут в нерассмотренную вершину. Если после завершения алгоритма не все вершины были рассмотрены, то необходимо запустить алгоритм от одной из нерассмотренных вершин.



BFS

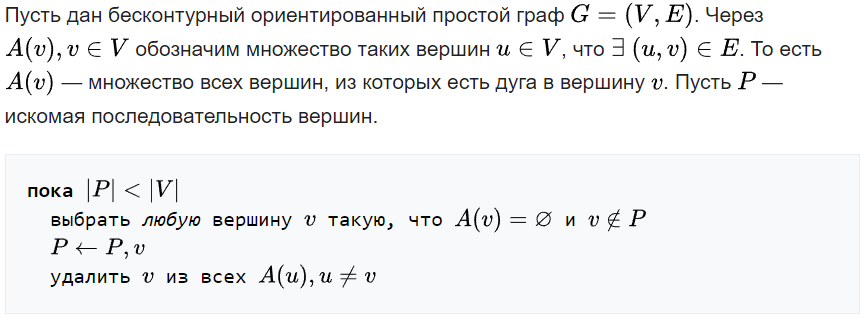
Один из методов обхода графа. Breadth-first search – стратегия поиска в ширину, в процессе обхода мы идём вширь, то есть перед тем как приступить к поиску вершин на расстоянии k + 1, выполняется обход вершин на расстоянии k.



Topological Sort by Kahn

Топологическая сортировка - способ нумерации вершин ориентированного ацикличного графа, при котором каждое ребро ведёт из вершины с меньшим номером в вершину с большим номером. Для одного графа может существовать несколько способов топологической сортировки (например, когда он несвязный), а может не существовать не одного (при наличии циклов).

Алгоритм Кана - один из первых алгоритмов, и наиболее приспособленный к исполнению вручную.



Top Sort on DFS

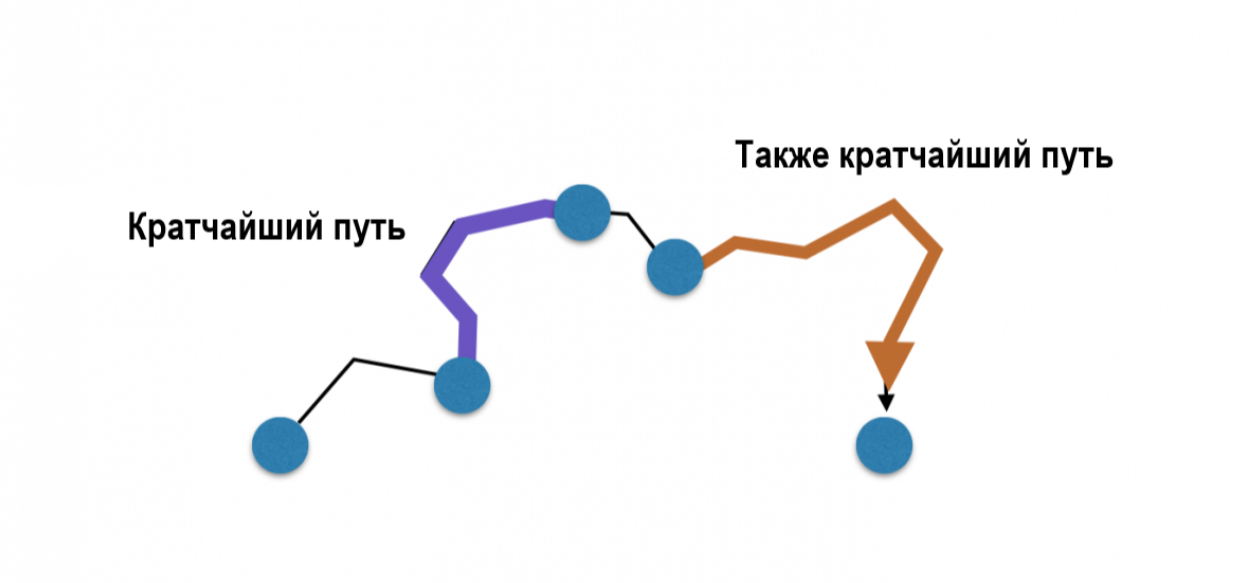
Заметим, что вершину, из которой не ведет ни одно ребро, можно всегда поставить последней. И такая вершина всегда есть в графе без циклов. Из этого сразу следует доказательство: просто будем класть в массив вершину, из которой ничего не ведет и убирать ее из графа. Массив потом надо будет развернуть.

Вершина, из которой мы вышли в DFS-е первой - как раз та, из которой ничего не выходит. Ведь если из нее есть ребро, то только в уже посещенную, но мы не могли выйти из нее, ведь мы берем самую первую из которой мы вышли. Значит, это одна из вершин, в которые мы вошли, и не вышли, и это цикл.

Из этого следует, что достаточно просто брать вершины в порядке выхода из DFS, то есть в конце DFS, например, просто класть эту вершину в конец массива с ответом.

Алгоритм Дейкстры

Алгоритм Дейкстры работает на том основании, что любой подпуть B -> D кратчайшего пути A -> D между вершинами A и D также является кратчайшим путем между вершинами B и D.



Подробное описание реализации алгоритмов – см. комментарии к коду

Анализ результатов тестирования

После многократных тестовых запусков было выявлено, что лучшие данные для этих алгоритмов - множество графов, состоящих из одной вершины.

Время работы получилось таким:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | DFS time | BFS time | Topological sort by Kahn time | Topological sort on DFS time |
| Number of dots |
| 100 | 0.000001 | 0.000001 | 0.000026 | 0.000035 |
| 300 | 0.000001 | 0.000001 | 0.000073 | 0.000097 |
| 500 | 0.000001 | 0.000002 | 0.000117 | 0.000172 |
| 800 | 0.000001 | 0.000002 | 0.000195 | 0.000266 |
| 1200 | 0.000001 | 0.000004 | 0.000283 | 0.000386 |

Предполагаем, что сейчас работа DFS быстрее BFS, так как в нашем варианте реализации BFS создаётся список, размер которого равен количеству вершин, а в DFS используется словарь.

Алгоритм Кана в сначала проходит по всем точкам и находит корни, в данном случае все точки - корни дерева. То есть последовательно проходит по всем точкам, потом выводит их.

Топологическая сортировка, которая использует DFS, из случайной вершины начинает спускаться вниз, и так как графы состоят из одной вершины, работа алгоритма завершается, из-за этих вызовов топологическая сортировка на DFS работает медленнее в данном случае.

Далее худшими данными будет граф, в котором каждая вершина соединена с каждой другой.

Время получилось таким

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | DFS time | BFS time | Topological sort by Kahn time | Topological sort on DFS time |
| Number of dots |
| 100 | 0.000363 | 0.000565 | 0.001123 | 0.000343 |
| 300 | 0.003117 | 0.004995 | 0.009109 | 0.003074 |
| 500 | 0.009410 | 0.013582 | 0.026856 | 0.009316 |
| 800 | 0.025796 | 0.033652 | 0.070872 | 0.024975 |
| 1200 | 0.058072 | 0.074392 | 0.162351 | 0.057561 |

Так как в алгоритме Кана мы проходим по каждой вершине, алгоритм начинает работать в разы медленнее, чем остальные алгоритмы.

Интересная и необъяснимая деталь: топологическая сортировка на DFS работает быстрее, чем сам DFS, хотя должно быть наоборот, так как в топ сорте есть проверка всех вершин на то, были они посещены или нет.

Алгоритм BFS так же работает медленнее чем DFS. Это происходит по той же причине.

Возможно, влияет работа deque.

Так же мы рассмотрели случайно сгенерированный граф

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| function | DFS time | BFS time | Topological sort by Kahn time | Topological sort on DFS time |
| Number of dots |
| 100 | 0.000153 | 0.000214 | 0.000397 | 0.000140 |
| 300 | 0.001209 | 0.001866 | 0.003253 | 0.001224 |
| 500 | 0.003746 | 0.005125 | 0.009280 | 0.003739 |
| 800 | 0.012600 | 0.015861 | 0.027084 | 0.012670 |
| 1200 | 0.032061 | 0.036615 | 0.062623 | 0.033357 |

Тут у нас уже топологическая сортировка на DFS и сам DFS чередуются по скорости.

По тем же самым причинам BFS и алгоритм Кана работают медленнее.