**Сравнение времени работы алгоритмов Джонсона с использованием алгоритма Дейкстры на различных кучах**

*1.Сравнение времени работы алгоритмов:*

Рассматриваются следующие алгоритмы:

* Алгоритм Джонсона с использованием алгоритма Дейкстры на левосторонней куче.
* Алгоритм Джонсона с использованием алгоритма Дейкстры на d-куче (d = 2 и d=3).
* Алгоритм Джонсона с использованием алгоритма Дейкстры на самоорганизующейся куче.

Введем обозначения:

n – число вершин.

M – число ребер.

Min\_weight – нижняя граница весов.

Max\_weight – верхняя граница весов.

Рассматриваем только ориентированные графы.

Проведем эксперименты, запустив алгоритмы при следующих значениях параметров графа:

n = 1, … 500 c шагом 50, Min\_weight = 1, Max\_weight = 106. Количество ребер: a) m = 10\*n (граф близкий к разреженному) и b) m = n\*(n - 1) (полный граф)

Графики работы алгоритмов (зависимость времени работы алгоритма от числа вершин). По вертикали – время работы в секундах, по горизонтали – число вершин

В случае a) (Разреженный граф):

В случае b)(Полный граф):

Оценки алгоритмов Джонсона с использованием различных куч:

Создание дополнительного графа: O(n2)

Алгоритм Беллмана-Форда: O(m \* n)

В цикле по всем вершинам графа используем алгоритм Дейкстры: O(D), где параметр D зависит от используемой в алгоритме Дейкстры кучи:

1. Если используется d-куча, то сложность O((m + n) \* logn) (т.к. используем в алгоритме Дейкстры на d-куче т.к. производится n вызовов функции УДАЛИТЬ\_МИНИМУМ и максимум m вызовов функции ВСПЛЫТИЕ, а эти функции выполняются за O(logn))
2. Если используется левосторонняя куча, то сложность O(n \* m + n \* logn), т.к. мы производим n вызовов функции УДАЛИТЬ\_МИНИМУМ (выполняется за O(logn)) и максимум m вызовов функции УМЕНЬШИТЬ\_КЛЮЧ (обычно выполняется за O(logn), но в худшем случае будет O(n) при условии, если левосторонняя куча будет представлена в виде пути)
3. Если используется самоорганизующаяся куча, то сложность в худшем случае O(n\*(m+n)), а амортизационная O((m + n) \* logn), т.к. мы производим n вызовов функции УДАЛИТЬ\_МИНИМУМ (выполняется за O(logn) или за O(n) в худшем случае) и максимум m вызовов функции УМЕНЬШИТЬ\_КЛЮЧ (обычно выполняется за O(logn), но в худшем случае будет O(n)).

В итоге получаем следующие оценки алгоритмов Джонсона:

* Джонсон на d-куче O((n + m) \* n \* logn)
* Джонсон на левосторонней куче O((logn + m) \* n2)
* Джонсон на самоорганизующейся куче в худшем случае O(n2 \* (m + n)), а амортизационная O((n + m) \* logn)

*2.Выводы:*

* Оба графика показали, что алгоритм Джонсона с использованием алгоритма Дейкстры на d-куче работает эффективно при d = 2 – оптимальное значение. При d = 3 результаты оказались хуже (особенно в случае b) ) Это легко обосновать: основными используемыми операциями в алгоритме Дейкстры, используемом алгоритмом Джонсона, являются УДАЛИТЬ\_МИНИМУМ и УМЕНЬШИТЬ\_КЛЮЧ, временная сложность которых O(d \* log(d)n) из-за того, что они используют операции ВСПЛЫТИЕ и ПОГРУЖЕНИЕ, а эти операции в свою очередь используют операцию поиска минимального потомка. В d-куче при d = 3 нам нужно проходиться по 3ем потомкам узла, чтобы определить среди них минимального потомка, когда как при d = 2 нам требуется проверить только 2 потомка (но при использовании d-кучи c d = 2 высота кучи будет больше). С учетом того, что поиск минимального потомка вызывается довольно часто, проход по всем потомкам узла является решающим моментом и при d = 3 это не является оптимальным (особенно в случае, когда ищем кратчайшие пути в полном графе, и недопустимость использования такого значения d в этом случае показывается на графике в случае b) ). Интересным моментом является тот факт, что у d-кучи при d=3 высота меньше, чем при d = 2, но это не дает выигрыша в скорости исполнения алгоритма поиска кратчайших путей.
* Графики показывают, что алгоритм Джонсона с использованием алгоритма Дейкстры на d-куче следует применять к графам, близким к разреженным. Также можно применять этот алгоритм и к полным графам, но при d = 2.
* Графики также показывают, что объединяемые приоритетные очереди (представляются в нашем случае в виде левосторонней или самоорганизующейся кучи) следует применять к графам, близким к полным. Причем лучшие результаты (по отношению к полным графам) показал алгоритм Джонсона, использующий алгоритм Дейкстры на самоорганизующейся куче, обогнав при этом алгоритм Джонсона, использующий алгоритм Дейкстры на d-куче (d = 2). При больших значениях n, алгоритм Джонсона, использующий алгоритм Дейкстры на левосторонней куче, показывает результаты хуже, чем при использовании самоорганизующейся кучи (это объясняется тем, что в левосторонней куче мы должны поддерживать постоянно свойство левизны, т.е. после каждой операции обновлять ранги узлов, а в самоорганизующейся куче этого не требуется). Из графиков видно, что левостороннюю кучу приемлемо использовать при малых значениях n (в рамках нашей задачи), но с этой задачей вполне хорошо справляются d-куча (d = 2) и самоорганизующаяся.

**Итоговые выводы:**

Левосторонняя куча в алгоритме поиска кратчайших путей может использоваться по усмотрению (как для разреженного, так и для полного), но при малых значениях n.

d-куча (d = 2) может использоваться во всех случаях.

Самоорганизующаяся куча может использоваться в полных графах, и, исходя из графиков, эффективнее использовать ее, чем d-кучу (d = 2).