

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

**«Дальневосточный федеральный университет»**

Институт математики и компьютерных технологий

**Департамент информационных и компьютерных систем**

**Алгоритмы и структуры данных**

**ОТЧЁТ**

на тему **«Алгоритм Дейкстры на Фибоначчиевой куче»**

|  |
| --- |
| Выполнила:  студент гр. Б9121-09.03.03пикд  \_\_\_\_\_\_\_\_\_\_ Коканов А.В. |
| Проверила:  \_\_\_\_\_\_\_\_\_\_\_ Кленин А.С.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (оценка) |

г. Владивосток

2022 г.

**Введение**

Для полного понимая алгоритма и его преимуществ среди других его аналогов необходимо ознакомиться с некоторыми терминами:

**Граф** — это топологическая модель, которая состоит из множества вершин и множества соединяющих их рёбер. При этом значение имеет только сам факт, какая вершина с какой соединена. Граф называется связным, если в нём существует путь между любыми двумя вершинами.

**Дерево** - структура данных в информатике, эмулирующая древовидную структуру и являющаяся связным графом, в котором нет ни одного пути, где начальная и конечная вершина совпадают (цикл). **Рангом дерева** называется количество дочерних деревьев корня этого дерева.

**Двусвязный список** - структура данных в информатике, состоящая из элементов, содержащих в себе значение и ссылки на следующий и предыдущий элемент списка.

**Фибоначчиева куча** — это набор деревьев, каждое из которых имеет не менее F(n + 2) элементов, где F(i) – i -ое число фибоначчи, а n – ранг дерева.Корни деревьев объединены в циклический двусвязный список. Основными операциями фибонччиевой кучи являются: вставка нового элемента, получение значения минимального элемента, удаление минимального элемента, уменьшение значения элемента, соединение двух куч.

Фибоначчиева куча является одним из самых оптимальных структур данных для использования алгоритма Дейкстры. Данный алгоритм является алгоритмом на графах и позволяет находить кратчайшие пути от одной из вершин графа до всех остальных. Благодаря своему строению и особенности выполнения некоторых операций фибоначчиева куча выполняет данный алгоритм значительно быстрее, чем другие подобные структуры данных.

Данная структура данных была разработана [Майклом Фредман](https://vies.wiki/wiki/ru/Michael_Fredman)ом и [Робертом Тарьян](https://vies.wiki/wiki/ru/Robert_Tarjan)ом  в 1984 году и опубликована ими в научном журнале в 1987 году. Была создана при работе по улучшению асимптотической сложности [алгоритма Дейкстры](https://neerc.ifmo.ru/wiki/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B).

**Описание алгоритма**

Фибоначчиева куча хранит в себе указатель на минимальный узел и, в некоторых случаях, размер.

Сам узел хранит в себе следующие данные:

1. Ключ – значение, с помощью которого можно идентифицировать узел.
2. Значение элемента
3. Указатель на родительский узел
4. указатель на один из дочерних узлов
5. указатель на левый узел
6. указатель на правый узел
7. Ранг дерева, корнем которого является данный узел
8. Метка о том, был ли удален в процессе изменения ключа ребенок этой вершины

**Описание операций Фибоначчиевой кучи**

**Вставка элемента**

Вставка элемента, как и большинство операций, происходит довольно просто. Необходимо создать новый элемент и связать его с корневым списком кучи. Получается так, что при создании нового элемента мы просто создаём фибонначиево дерево с одной вершиной или же с нулевым рангом. Также после этого необходимо сравнить значения нового элемента и минимума, и если новый элемент меньше, то заменить минимум.

**Получение минимума**

Так как мы храним указатель на узел с минимальным значением, нам необходимо просто вернуть значение, которое лежит в данном узле.

**Слияние двух куч**

Слияние двух куч представляет собой, по сути, добавление корня первой кучи в корневой список второй кучи. Таким образом, одна из куч становится подкучей второй. Естественно, после этой операции также необходимо сравнить значения минимумов двух куч и, в случае необходимости, изменить указатель на минимум.

**Уменьшение элемента**

При уменьшении элемента нам необходимо проверить, не стал ли наш элемент меньше своего предка. Если такое произойдёт, то у нас нарушится главное свойство: значение корня всегда не больше значения всех детей. Так что в данной ситуации нам необходимо выполнить некоторые преобразования с нашей кучей. В каждой вершине мы храним пометку о том, был ли у неё удалён потомок. И если мы знаем, что у вершины был удалён один потомок или ни одного вовсе, мы её не трогаем. Однако, если мы нам необходимо удалить потомка у вершины, которая уже помечена, то мы и эту вершины отправляем в корневой список. А также снимаем пометки со всех деревь Данная операция называется каскадное вырезание.

**Удаление минимума**

Удаление минимума является самой долгой операцией, так как именно при её выполнении мы приводим нашу кучу в порядок, а именно «прореживаем» деревья. Сначала мы отрезаем наш минимум от корневого списка, а всех его детей отправляем в корневой список. После этого мы проходимся по всем корням и соединяем все пары деревьев с равным рангом. То есть, если у нас есть два дерева ранга k, то мы их сливаем и получаем одно дерево ранга k + 1. Параллельно с этим мы ищем минимум среди корней. После выполнения данной операции у нас не будет два дерева с одинаковым рангом и также будет минимальное количество деревьев равное logn, где n – количество вершин.

**Алгоритм Дейкстры**

Алгори́тм Де́йкстры  — [алгоритм](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC) на [графах](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), изобретённый нидерландским учёным [Эдсгером Дейкстрой](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B0,_%D0%AD%D0%B4%D1%81%D0%B3%D0%B5%D1%80_%D0%92%D0%B8%D0%B1%D0%B5) в [1959 году](https://ru.wikipedia.org/wiki/1959_%D0%B3%D0%BE%D0%B4). Находит кратчайшие пути от одной из вершин графа до всех остальных. Алгоритм работает только для графов без [рёбер](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B1%D1%80%D0%BE_(%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2)) отрицательного [веса](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#%D0%92).

Неформальное объяснение

Каждой вершине из V сопоставим метку — минимальное известное расстояние от этой вершины до a.

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

Работа алгоритма завершается, когда все вершины посещены.

Инициализация.

Метка самой вершины a полагается равной 0, метки остальных вершин — бесконечности.

Это отражает то, что расстояния от a до других вершин пока неизвестны.

Все вершины графа помечаются как не посещённые.

Шаг алгоритма.

Если все вершины посещены, алгоритм завершается.

В противном случае, из ещё не посещённых вершин выбирается вершина u, имеющая минимальную метку.

Мы рассматриваем всевозможные маршруты, в которых u является предпоследним пунктом. Вершины, в которые ведут рёбра из u, назовём соседями этой вершины. Для каждого соседа вершины u, кроме отмеченных как посещённые, рассмотрим новую длину пути, равную сумме значений текущей метки u и длины ребра, соединяющего u с этим соседом.

Если полученное значение длины меньше значения метки соседа, заменим значение метки полученным значением длины. Рассмотрев всех соседей, пометим вершину u как посещённую и повторим [шаг алгоритма](https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B#%D0%A8%D0%B0%D0%B3).

Список литературы

1. <https://wiki5.ru/wiki/Fibonacci_heap#Implementation_of_operations>
2. <https://neerc.ifmo.ru/wiki/index.php?title=%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8%D0%B5%D0%B2%D0%B0_%D0%BA%D1%83%D1%87%D0%B0>
3. <https://ru.wikipedia.org/wiki/%D0%A4%D0%B8%D0%B1%D0%BE%D0%BD%D0%B0%D1%87%D1%87%D0%B8%D0%B5%D0%B2%D0%B0_%D0%BA%D1%83%D1%87%D0%B0>
4. <http://cppalgo.blogspot.com/2011/11/fibonacci-heap.html>
5. <https://ru.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D1%8B>
6. <https://habr.com/ru/post/111361/>
7. <http://acm.math.spbu.ru/~sk1/mm/au-download/conspect/conspect.pdf>
8. <https://prog-cpp.ru/deikstra/>
9. <https://www.cs.princeton.edu/~wayne/teaching/fibonacci-heap.pdf>
10. <https://vies.wiki/wiki/ru/Fibonacci_heap>
11. <https://youtu.be/CeAjTL-Fshs>
12. <http://acm.math.spbu.ru/~sk1/mm/au-download/14f-conspect/2014-10-21-Fib.pdf>
13. <https://wiki5.ru/wiki/Potential_method>
14. <https://kbaile03.github.io/projects/fibo_dijk/fibo_dijk.html>
15. <http://www-m3.ma.tum.de/foswiki/pub/MN0506/WebHome/dijkstra.pdf>
16. <http://staff.ustc.edu.cn/~csli/graduate/algorithms/book6/chap21.htm>
17. <https://infopedia.su/17xb2c6.html>
18. <https://e-maxx.ru/algo/dijkstra>
19. <https://www.programiz.com/dsa/fibonacci-heap>
20. <https://www.geeksforgeeks.org/fibonacci-heap-insertion-and-union/>
21. <https://ask-dev.ru/info/101216/what-is-the-intuition-behind-the-fibonacci-heap-data-structure>
22. <https://maryrosecook.com/blog/post/the-fibonacci-heap-ruins-my-life>
23. <https://cse.sc.edu/~mgv/csce580f08/gradPres/boccanfusoMcKenzie080910.ppt>
24. <https://www.cs.auckland.ac.nz/software/AlgAnim/dijkstra.html#dijkstra_anim>
25. <https://web.stanford.edu/class/archive/cs/cs166/cs166.1146/lectures/07/Small07.pdf>
26. <https://www.javatpoint.com/fibonacci-heap>
27. <https://www.boost.org/doc/libs/1_49_0/doc/html/boost/heap/fibonacci_heap.html>