Казанский (Приволжский) Федеральный университет

Высшая школа информационных технологий и информационных систем

Кафедра программная инженерии

Семестровая работа

по дисциплине «Информатика»

Тема: «SmoothSort – плавная сортировка»

**Выполнила:**

студентка 1 курса группы 11-709

Зырянова П.А.

**Проверила:**

ассистент кафедры программной инженерии

Шамсутдинова К.С.

Казань, 2018 г.

**1. Краткая историческая справка.**

**Плавная сортировка (SmoothSort)** была разработана [Эдсгером Вайбом 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 1981 году. Плавная сортировка является разновидностью сортировки выбором. Алгоритм является развитием идеи Heapsort — сортировки с помощью двоичной кучи (или пирамиды).

В связи с этим, напомню, как работает Heapsort. Он работает в 2 этапа:  
 1. Построить кучу, добавляя в нее по одному элементу из исходного массива  (сложность O(log n) на каждый, всего O(n log n));

2. Извлекать максимальный элемент, который лежит в корне построенной кучи, до тех пор, пока она не станет пуста. Но так как при удалении корня куча распадается на две независимых, требуется собрать их в одну за O(log n). В сумме для n элементов — также O(n log n).

Э. Дейкстра же в своём алгоритме предложил заменить одну двоичную кучу на упорядоченный набор из куч. Такие кучи носят название кучи Леонардо и строятся на основе специальных чисел Леонардо. Об этом более подробно в следующих пунктах отчёта.

**2. Основной принцип работы. Основные особенности.**

Числа Леонардо определяются рекуррентно следующим образом:

L(0)=1, L(1)=1, L(k)=L(k-1)+L(k-2)+1

Вот несколько таких чисел: 1, 3, 5, 9, 15, 25, 41…

k-ая куча Леонардо — это двоичное дерево с количеством вершин L(k), удовлетворяющее следующим условиям:

1. Число, записанное в корне кучи, не меньше чисел в поддеревьях;

2. Левым поддеревом является (k-1)-я куча Леонардо;

3. Правым поддеревом является (k-2)-я куча Леонардо.

Исключение составляют лишь кучи размера L(1) и L(0), которые состоят только из корня.

Для последовательности куч, в свою очередь, выполняется следующее свойство: значение корня каждой кучи должно быть не меньше значения корня кучи слева. Следствие этого — крайний правый узел в последовательности будет иметь наибольшее значение, и, что важно, частично отсортированный массив входных данных не будет нуждаться в перестановке элементов, для того чтобы стать правильной последовательностью куч.

Любое натуральное число можно разложить в сумму чисел Леонардо, а это значит, что вышеописанную структуру можно построить для любого количества чисел.

Алгоритм плавной сортировки, как и алгоритм пирамидальной, состоит из 2-х шагов:

***1. Формирование последовательности куч.*** В начале последовательность куч пуста. Последовательно добавляем элементы из исходного массива слева направо по следующим правилам:

* Если две последние кучи имеют размеры L(*x + 1*) и L(*x*) (двух последовательных чисел Леонардо), новый элемент становится корнем кучи большего размера, равного L(*x + 2*), объединяя 2 предыдущие кучи;
* Если размеры двух последних куч не равны двум последовательным числам Леонардо, новый элемент образует новую кучу размером 1. Этот размер полагается равным L(1).

Повторяем эту процедуру, пока не закончатся элементы во входном массиве.   
*2.* ***Формирование отсортированного массива.***Этот шаг осуществляется следующим образом:

* Крайняя правая куча (сформированная последней) считается «текущей» кучей;
* Пока слева от неё есть куча и значение её корня больше значения текущего корня, эти корни меняются местами (что не нарушит свойство для текущей кучи). Эта куча становится текущей.

***Замечание:*** На 1 шаге при объединении двух куч в одну нужно гарантировать сохранение свойства кучи, т.е. выполнять **“просейку вниз”** при необходимости. «Просейка вниз» выполняется следующим образом:

* Корень кучи сравнивается с обоими дочерними элементами. Если он меньше какого-либо из них, то он меняется с этим дочерним элементом. Если он меньше обоих своих дочерних элементов, то он меняется местами с большим из них;
* Указанная процедура повторяется сколь возможно вглубь кучи.

Таким образом, структура восстанавливается необходимым нам (отсортированным) образом.

**3. Оценка временной сложности алгоритма**

***1 шаг. Построение последовательности куч.*** Последовательность куч получается при вставке элементов массива по очереди в эту самую последовательность. Получаем время работы O(logN) для каждого элемента, то есть, в сумме, O(NlogN).

***2 шаг. Получение отсортированного массива.*** Так как удаление максимального элемента из последовательности выполняется за O(logN), то время работы сортировки составляет O(NlogN). То есть в худшем случае алгоритм имеет асимптотику O(NlogN).

***Сложность в лучшем случае***

Однако если подать на вход плавной сортировке уже отсортированный массив, асимптотика будет составлять O(N). Дело в том, что:

* Операция добавления элемента последовательности на таком примере будет выполняться за константное время O(1) из-за того, что в конец будет добавляться максимальный элемент, и просеивание будет сразу останавливаться;
* Операция получения и удаления максимального элемента будет также выполняться за константное время O(1), потому что в силу построения в корнях куч-детей будут новые максимальные элементы и, следовательно, восстановление свойства последовательности закончится на просмотре корня соседней кучи.

В итоге на таком примере получается линейная асимптотика O(N). При уменьшении степени упорядоченности входных данных сложность будет плавно(smoothly) возрастать от O(N) до O(NlogN). Отсюда и название — SmoothSort – плавная сортировка.

**4. Таблица зависимости времени работы от размера входных данных.**

С помощью случайной генерации чисел были заготовлены 55 массивов случайной длины от 100 до 10000 элементов. Примерное время работы для некоторых случаев приведено в таблице.

|  |  |  |  |
| --- | --- | --- | --- |
| Размер массива | Время работы в миллисекундах | Размер массива | Время работы |
| 100 | 7,5 | 5300 | 10,4 |
| 500 | 7,6 | 5700 | 10,75 |
| 900 | 7,8 | 6100 | 10,9 |
| 1300 | 8,2 | 6500 | 11,2 |
| 1700 | 8,5 | 6900 | 11,45 |
| 2100 | 8,7 | 7300 | 11,6 |
| 2500 | 8,8 | 7700 | 11,9 |
| 2900 | 9 | 8100 | 12,4 |
| 3300 | 9,15 | 8500 | 12,75 |
| 3700 | 9,47 | 8900 | 12,9 |
| 4100 | 9,69 | 9300 | 13,2 |
| 4500 | 9,8 | 9700 | 13,5 |
| 4900 | 10 | 10000 | 14 |

**5. График зависимости времени работы от размера входных данных**

По полученной таблице построен примерный график зависимости времени работы в миллисекундах от размера входных данных.

**6. Выводы.**

### Таким образом, алгоритм плавной сортировки работает в 2 шага:

1. Добавляя по одному, строится множество куч Леонардо для всех элементов (не более O(logN) на каждый)
2. По очереди извлекаются максимальные элементы из построенной структуры (максимумом всегда будет корень самой правой кучи) и структура восстанавливается необходимым образом (не более O(logN) на каждый элемент)

***Преимущества алгоритма:***

* худшее время работы — O(NlogN);
* время работы в случае, когда подается отсортированный массив — O(N).

### *Недостаток алгоритма:*

* Требует O(logN) дополнительной памяти для хранения длин куч в последовательности. Однако с помощью некоторых модификаций расходы на дополнительную память можно сократить до O(1).

**7. Список использованной литературы.**

1) И снова про сортировки: выбираем лучший алгоритм. Статья на сайте <https://habrahabr.ru/post/133996>.

2) SmoothSort. Статься на сайте <http://neerc.ifmo.ru/wiki/index.php?title=Smoothsort>.

3) Плавная сортировка. Статья на сайте <https://ru.wikipedia.org/wiki>.

4) Плавная сортировка (SmoothSort). Статья на сайте <http://cppalgo.blogspot.ru/2010/10/smoothsort.html>.