Оглавление

Лекция 9: Сортировки. Сжатие и защита информации.

8.11.2023

0.1 Сортировка Хоара (быстрая сортировка)

Идея: выберем элемент-разделитель. Разместим все меньшие разделителя элементы слева от него, большие — справа и запустим процесс для каждой из частей.

Пример.

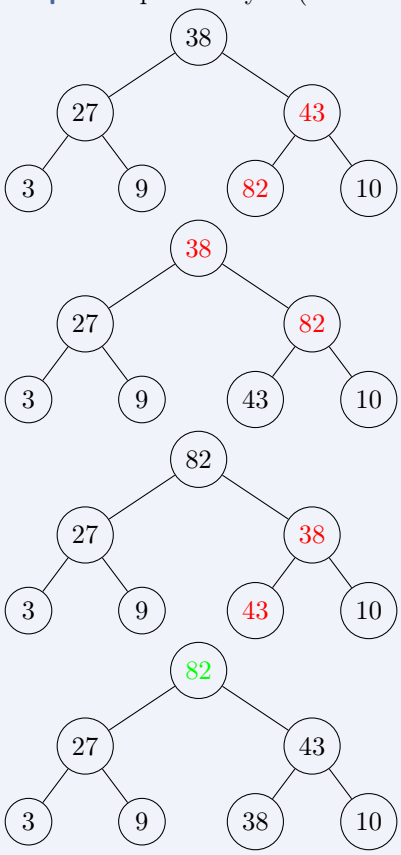
```
38 27 43 3 9 82 10 (pivot is 3)
3 27 38 43 9 82 10 (pivot is 38)
3 27 10 9 38 43 82 (pivot is 27)
3 10 9 27 38 43 82 (pivot is 10)
3 9 10 27 38 43 82 (pivot is 9)
3 9 10 27 38 43 82
```

0.2 Пирамидальная сортировка (иерархическая, heapsort)

1. **Построение кучи:** Преобразуем исходный массив в максимальную кучу. В максимальной куче значение каждого узла больше или равно значению его потомков.

2. **Сортировка:** Удаляем максимальный элемент (корень кучи), заменяем его последним элементом кучи, уменьшаем размер кучи на один и восстанавливаем свойство кучи. Повторяем, пока в куче не останется элементов.

Пример. Построение кучи (массив $a = [38, 27, 43, 3, 9, 82, 10]$):



Далее меняем местами 82 и 10, удаляем 82 из кучи и восстанавливаем свойство кучи.

0.3 Порозрядная сортировка (radix sort)

Если известно, что диапазон ключей ограничен, то сортировать можно значительно быстрее: сортируем последовательно по разрядам (например, по десятичным или двоичным), начиная с младшего. (Пример придумайте сами)

Глава 1

Сжатие и защита информации

1.1 Задача о префиксном коде.

Пусть $A = \{c_1, \ldots, c_n\}$ — алфавит, $p_1, \ldots, p_n$ — вероятности появления символов. Пусть задан код $\varphi : A \to \{0,1\}^*$
Фиксируем текст (большую строку) длины $N : a_1 a_2 \ldots a_N$. В этом тексте количество символов c_i – $N \cdot a_i$ (закон больших чисел). Тогда длина кодовой последовательности текста:

$$\sum_{i=1}^{n} N p_i \varphi(c_i)$$

Задача заключается в том, чтобы минимализировать длину кодовой последовательности такого текста, а т.к. N – фиксированная величина, не влияющая на коды, задача сводиться к тому, чтобы минимализировать сумму:

$$\sum_{i=1}^{n} p_i \varphi(c_i)$$

При этом для $p_i, \varphi(c_i)$ выполняются:

1. $\sum p_i = 1$
2. $0 < p_i < 1$
3. $\varphi(c_i) \in \mathbb{N}$
4. $\varphi(c_1), \ldots, \varphi(c_n)$ – длины кодов символов в префиксом коде, т.е. для этих длин выполняется неравенство Крафта.

Алгоритм решения такой задачи основывается на нескольких свойств длин кодовых последовательностей:

Лемма 1. Пусть p_i – вероятности символов и s_i – длины кодов символов для оптимального кода.

Если $p_1 \geqslant p_2 \geqslant ... \geqslant p_m$, то $s_1 \leqslant s_2 \leqslant ... \leqslant s_m$.

Доказательство. Мы просто покажем, что если найдутся две пары чисел (p_i, s_i) и (p_j, s_j), такие что $p_i < p_j$ и $s_i < s_j$, то значение суммы попарных произведений можно уменьшить, заменив эти две пары парами (p_i, s_j) и (p_j, s_i). Действительно, т.к. $(p_j - p_i)(s_j - s_i) > 0$, то, раскрывая скобки, получим после элементарных преобразований $p_i s_i + p_j s_j > p_i s_j + p_j s_i$.

Так как число возможных расположений конечно, поскольку равно $m!$, то начиная с любого расположения за конечное число шагов мы закончим процесс улучшений на расположении, которое дальше улучшить невозможно. На нём и достигается минимум. □

Лемма 2. Две самые длинные кодовые последовательности имеют одинаковую длину.

Доказательство. Предположим, что $s_{m-1} < s_m$. Тогда в силу префиксности кода, можно отбросить лишние символы из кода s_m. Мы получим снова префиксный код, но с меньшим значением целевой функции. Противоречие. □

Лемма 3. Рассмотрим задачу P', которая получается объединением двух самых редких символов в один символ. Минимальное значение целевой функции в задаче P' будет отличаться от значения в задаче P на $p'_{m-1} = p_{m-1} + p_m$, а оптимальный кодовый набор для P получается из решения задачи P' удлинением на 1 бит кодовых последовательностей для символов, которые были объединены.

Доказательство. Пересчитываем новую функцию:

$$\sigma(P) = \sum_{i=1}^{m-2} p_i s_i + (s'_{m-1} + 1)p_{m-1} + (s'_{m-1} + 1)p_m =$$

$$= \sum_{i=1}^{m-2} p_i s_i + s'_{m-1}(p_{m-1} + p_m) + p_{m-1} + p_m = \sigma(P') + p'_{m-1}.$$

Осталось доказать, что построенный код для P действительно оптимальный.

Предположим, что существует решение для P, такое что $\sigma^*(P) < \sigma(P)$. По лемме 2 длины кодов самых редких символов равны. Объединим эти символы в один, а код для него построим откидыванием последнего бита. В силу префиксности кода для P это можно сделать.

Таким образом, получили префиксный код для задачи P', и при этом $\sigma^*(P') < \sigma(P')$. Противоречие. □

1.2 Код Шеннона-Фано и алгоритм Хаффмана

Кодирование Шеннона — Фано — алгоритм префиксного неоднородного кодирования. Относится к вероятностным методам (т.е. мера на множестве событий, принимающая значения от 0 до 1) сжатия.

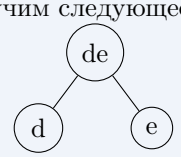
Часто встречающийся символ кодируется кодом меньшей длины (короткой двоичной последовательностью), редко встречающийся — кодом большей длины (длинной двоичной последовательностью).

Алгоритм (Хаффмана). Лемма 3 фактически даёт нам алгоритм для построения оптимального кода:

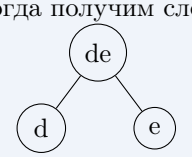
1. Составим список кодируемых символов вместе с их частотами. Каждый символ представим в виде одноэлементного дерева с соответствующим весом.
2. Из списка выберем два дерева с наименьшими весами.
3. Создадим новый корень и назначим ему в качестве детей эти два дерева. Вес сформированного дерева положим равным сумме весов дочерних. Таким образом, два дерева “склеились” в одно.
4. Добавим новое дерево в список.
5. Если в списке осталось ровно одно дерево, закончить процесс, иначе - перейти к п.2.

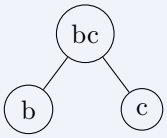
Пример. Пусть $A = \{a, b, c, d, e\}$, $p_a = 0.37, p_b = 0.22, p_c = 0.16, p_d = 0.14, p_e = 0.11$.

Объединим d и e в один символ de с вероятностью 0.25. Тогда получим следующее дерево:

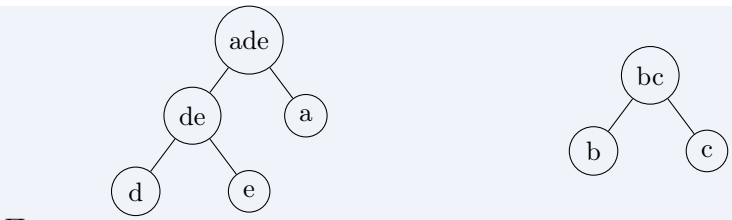


На следующем шаге объединим c и b в один символ bc с вероятностью 0.38. Тогда получим следующее дерево:

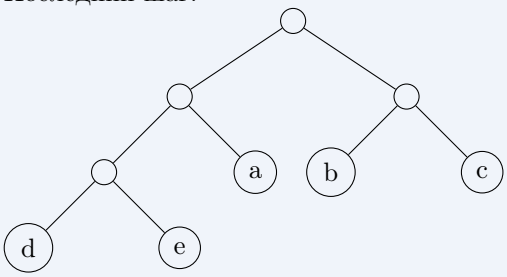




Следующий шаг:



Последний шаг:



Коды для символов: $a = 01, b = 10, c = 11, d = 000, e = 001$.