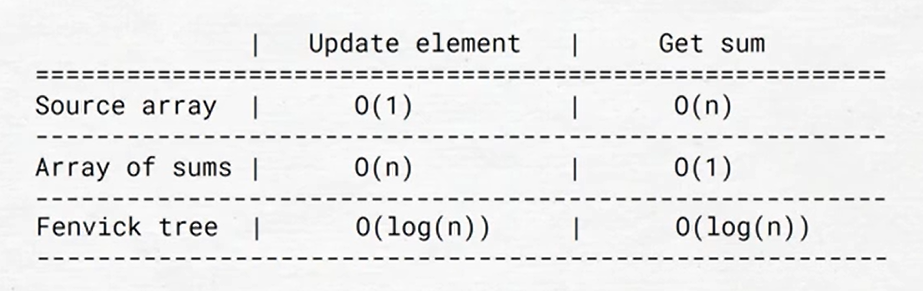
**Дерево Фенвика**

**Дерево Фенвика** (Бинарное индексированное дерево) - это структура данных, которая позволяет для массива чисел находить сумму подряд идущих элементов массива и обновлять значения за логарифмическую сложность и при этом не использовать дополнительную память.

Впервые описано Питером Фенвиком в 1994 году.

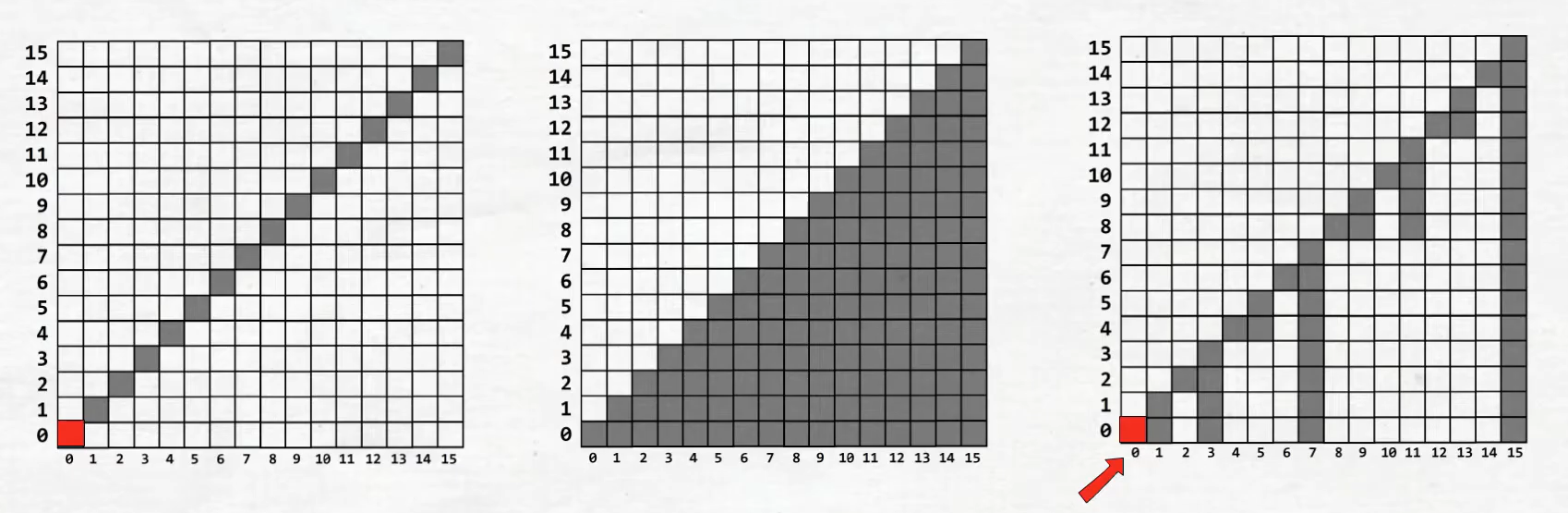
**Основной принцип.**

Создадим массив и запишем в каждую ячейку сумму от нуля до этого элемента. Теперь чтобы найти сумму нужно всего лишь обратиться к нужному индексу (если с начала массива суммирование), либо вычитанием одной суммы из другой (например, сумму от 7 до 9 равна arr[9]-arr[6]). Однако, при обновлении значения, в худшем случае нам придется обновлять все суммы, то есть все ячейки массива, а это линейная сложность.



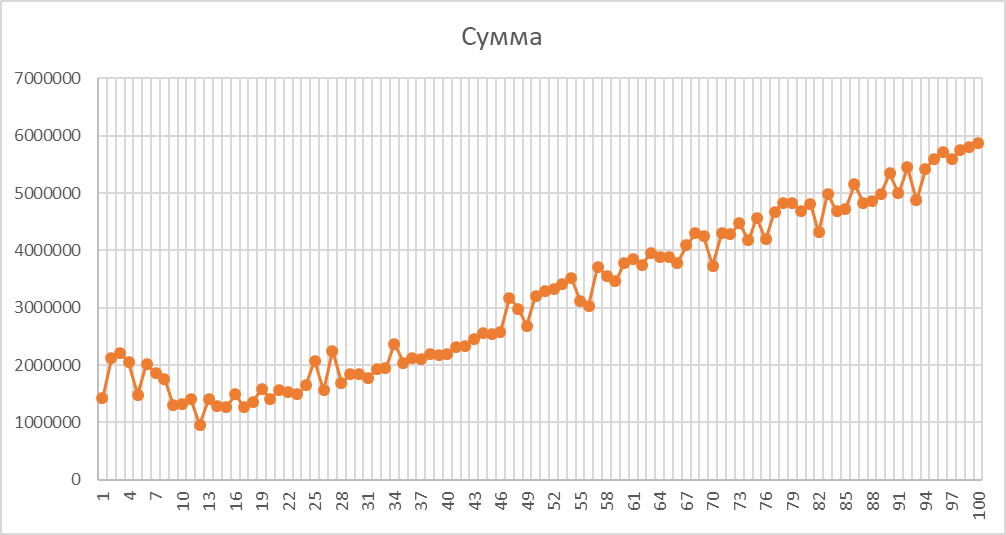
Дерево Фенвика хранит в себе только определенные суммы. Чтобы определить, где и какие суммы, переведем все числа в двоичную систему счисления. Теперь по формуле F(i) = i & ( i + 1) вычисляем, что в ячейке i хранится сумма элементов с i до F(i) (в десятичной с.с.)

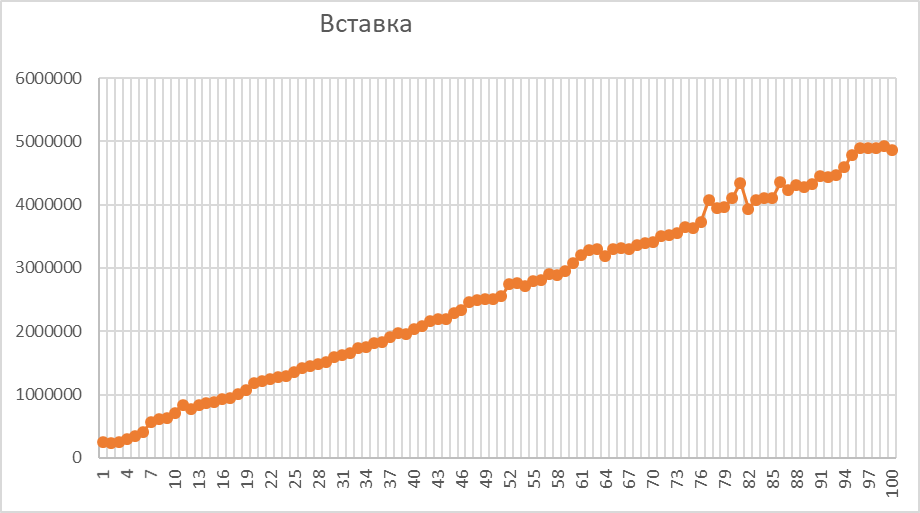
Чтобы обновлять элементы используется формула: F(i) = i | ( i + 1) (опять же i в двоичной с.с.). Теперь если захотим изменить число i на единицу, то прибавим ее к i, к F(i) и т.д, пока не выйдем за пределы массива.



Суммирование будет работать за логарифм, а точнее за количество единичных битов в записи x: на каждой итерации мы делаем x -= x & -x, то есть удаляем младший бит.

Вставка тоже будет работать за логарифм: каждую итерацию количество нулей на конце *x* увеличивается хотя бы на единицу.





Данный алгоритм может помочь во всех задачах, где надо быстро изменять и определять результат операции. Потенциальное применение бинарного индексированного дерева — обработка очень больших алфавитов. Например, при арифметическом сжатии на основе слов. В этом случае мы воспользуемся преимуществом компактной структуры данных и логарифмической стоимости доступа.

Плюсы структуры данных:

* Скорость работы. (Дерево Фенвика занимает в константное значение раз меньше памяти, чем дерево отрезков. Это следует из того, что дерево Фенвика хранит только значение операции для каких-то элементов, а дерево отрезков хранит сами элементы и частичные результаты операции на подотрезках, поэтому оно занимает как минимум в два раза больше памяти).
* Простота реализации
* Двоичное индексированное дерево занимает линейную память

Минусы:

* Функция для дерева Фенвика должна быть обратимой, а это значит, что минимум и максимум это дерево считать не может.

Литература:

* Википедия - <https://en.wikipedia.org/wiki/Fenwick_tree>
* Кормен. Алгоритмы - построение и анализ.

Код программы:

import java.util.\*;  
public class FenwickTree {  
  
 private final int[] btree;  
  
 public FenwickTree(int size) {  
 this.btree = new int[size];  
 }  
  
 public FenwickTree(ArrayList<Integer> initValues) {  
 this(initValues.size());  
 for (int i = 0; i < initValues.size(); i++) {  
 setValue(i, initValues.get(i));  
  
 }  
 }  
  
 public long get(int index) {  
 long result = getSum(index);  
  
 return index > 0 ? result - getSum(index - 1) : result;  
 }  
  
 public void setValue(int index, int value) {  
 long difference = value - get(index);  
 for (; index < btree.length; index = (index | (index+1)))  
 btree[index] += difference;  
 }  
  
 public long getSum(int from, int to) {  
 return getSum(to) - getSum(from - 1);  
 }  
  
 public long getSum(int to) {  
 long result = 0;  
 while (to >= 0) {  
 result += btree[to];  
 if (to == 0) {  
 break;  
 }  
 to &= to + 1;  
 to--;  
 }  
 return result;  
 }  
 public String toString(){  
 return Arrays.*toString*(Arrays.*copyOf*(btree, btree.length));  
 }

Приложение

Таблица полученных значений времени работы (в наносекундах), в зависимости от размера данных.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Номер файла** | **Сумма** | **Вставка** | **Элементов в файле** |  | **Номер файла2** | **Сумма2** | **Вставка2** | **Элементов в файле2** |
| 1 | 1424350 | 244212 | 100 |  | 51 | 3290816 | 2553308 | 5100 |
| 2 | 2115812 | 239872 | 200 |  | 52 | 3332880 | 2751400 | 5200 |
| 3 | 2220208 | 247616 | 300 |  | 53 | 3406556 | 2769136 | 5300 |
| 4 | 2051472 | 299392 | 400 |  | 54 | 3514284 | 2714336 | 5400 |
| 5 | 1484780 | 342932 | 500 |  | 55 | 3110948 | 2793824 | 5500 |
| 6 | 2025376 | 406548 | 600 |  | 56 | 3034616 | 2816472 | 5600 |
| 7 | 1861156 | 559972 | 700 |  | 57 | 3712896 | 2906472 | 5700 |
| 8 | 1762308 | 614384 | 800 |  | 58 | 3559380 | 2882348 | 5800 |
| 9 | 1300864 | 629144 | 900 |  | 59 | 3461904 | 2947264 | 5900 |
| 10 | 1317212 | 714200 | 1000 |  | 60 | 3781700 | 3079264 | 6000 |
| 11 | 1409504 | 837980 | 1100 |  | 61 | 3847692 | 3203268 | 6100 |
| 12 | 962748 | 774648 | 1200 |  | 62 | 3752488 | 3276756 | 6200 |
| 13 | 1402332 | 836188 | 1300 |  | 63 | 3953020 | 3291408 | 6300 |
| 14 | 1282044 | 858076 | 1400 |  | 64 | 3881716 | 3193816 | 6400 |
| 15 | 1269804 | 883944 | 1500 |  | 65 | 3879916 | 3296728 | 6500 |
| 16 | 1492028 | 927044 | 1600 |  | 66 | 3789248 | 3311792 | 6600 |
| 17 | 1273860 | 940416 | 1700 |  | 67 | 4094508 | 3303284 | 6700 |
| 18 | 1362792 | 1008108 | 1800 |  | 68 | 4300516 | 3358016 | 6800 |
| 19 | 1579012 | 1072400 | 1900 |  | 69 | 4248776 | 3400048 | 6900 |
| 20 | 1412108 | 1182176 | 2000 |  | 70 | 3731800 | 3404384 | 7000 |
| 21 | 1568216 | 1216632 | 2100 |  | 71 | 4300004 | 3499084 | 7100 |
| 22 | 1536888 | 1244476 | 2200 |  | 72 | 4293456 | 3518048 | 7200 |
| 23 | 1495832 | 1284468 | 2300 |  | 73 | 4470832 | 3544772 | 7300 |
| 24 | 1647988 | 1292868 | 2400 |  | 74 | 4181356 | 3646732 | 7400 |
| 25 | 2072632 | 1361220 | 2500 |  | 75 | 4570532 | 3636324 | 7500 |
| 26 | 1560520 | 1412132 | 2600 |  | 76 | 4193736 | 3724896 | 7600 |
| 27 | 2253904 | 1454404 | 2700 |  | 77 | 4679452 | 4069532 | 7700 |
| 28 | 1693676 | 1485812 | 2800 |  | 78 | 4835400 | 3946468 | 7800 |
| 29 | 1838256 | 1515960 | 2900 |  | 79 | 4826856 | 3958744 | 7900 |
| 30 | 1841512 | 1587368 | 3000 |  | 80 | 4685004 | 4109748 | 8000 |
| 31 | 1766928 | 1620772 | 3100 |  | 81 | 4818780 | 4343936 | 8100 |
| 32 | 1930556 | 1652580 | 3200 |  | 82 | 4317912 | 3931852 | 8200 |
| 33 | 1947024 | 1738028 | 3300 |  | 83 | 4988956 | 4078172 | 8300 |
| 34 | 2376808 | 1751856 | 3400 |  | 84 | 4686528 | 4096708 | 8400 |
| 35 | 2044228 | 1817228 | 3500 |  | 85 | 4726580 | 4108008 | 8500 |
| 36 | 2126160 | 1832716 | 3600 |  | 86 | 5153808 | 4349812 | 8600 |
| 37 | 2100788 | 1910924 | 3700 |  | 87 | 4835840 | 4234452 | 8700 |
| 38 | 2190368 | 1966532 | 3800 |  | 88 | 4861960 | 4313588 | 8800 |
| 39 | 2181328 | 1958216 | 3900 |  | 89 | 4977948 | 4274144 | 8900 |
| 40 | 2195280 | 2033588 | 4000 |  | 90 | 5344760 | 4332340 | 9000 |
| 41 | 2318224 | 2083664 | 4100 |  | 91 | 4995228 | 4447376 | 9100 |
| 42 | 2329944 | 2168832 | 4200 |  | 92 | 5447576 | 4440676 | 9200 |
| 43 | 2451708 | 2195068 | 4300 |  | 93 | 4878156 | 4467692 | 9300 |
| 44 | 2561436 | 2186372 | 4400 |  | 94 | 5418828 | 4593728 | 9400 |
| 45 | 2551100 | 2286116 | 4500 |  | 95 | 5599660 | 4786480 | 9500 |
| 46 | 2586524 | 2340392 | 4600 |  | 96 | 5719412 | 4890020 | 9600 |
| 47 | 3173240 | 2462668 | 4700 |  | 97 | 5590004 | 4897028 | 9700 |
| 48 | 2986208 | 2500424 | 4800 |  | 98 | 5753204 | 4893612 | 9800 |
| 49 | 2687368 | 2510800 | 4900 |  | 99 | 5803628 | 4919924 | 9900 |
| 50 | 3211784 | 2508764 | 5000 |  | 100 | 5873200 | 4865076 | 10000 |

Входные данные: <https://github.com/kk0kc/lisa_asd/tree/master/FenwickTree/FilesForTest>