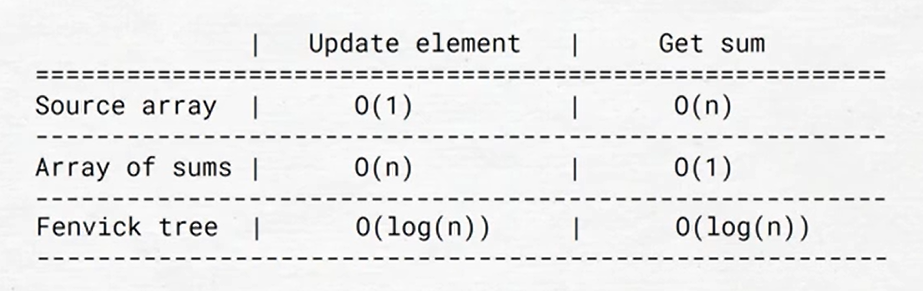
**Дерево Фенвика**

**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.

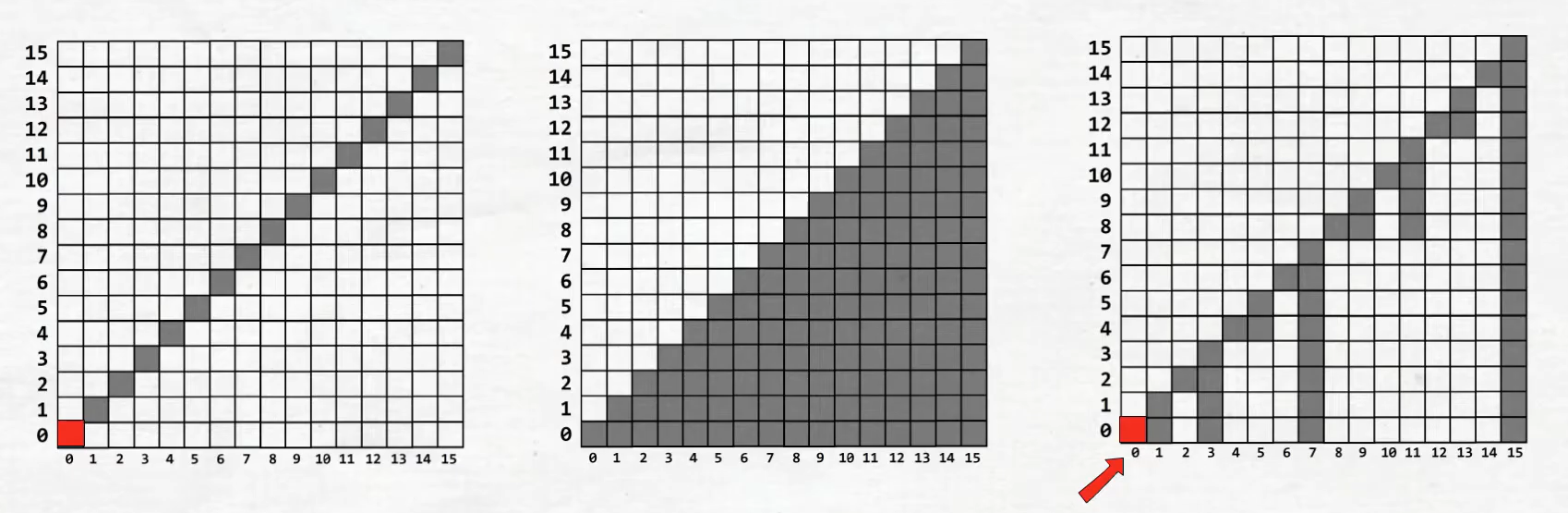
**Основной принцип.**

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 | 1664 | 320 | 100 | 51 | 288 | 88 | 5100 |
| 2 | 6688 | 336 | 200 | 52 | 492 | 152 | 5200 |
| 3 | 2064 | 112 | 300 | 53 | 220 | 132 | 5300 |
| 4 | 2067 | 100 | 400 | 54 | 331 | 96 | 5400 |
| 5 | 2131 | 108 | 500 | 55 | 196 | 72 | 5500 |
| 6 | 2244 | 100 | 600 | 56 | 388 | 108 | 5600 |
| 7 | 2915 | 104 | 700 | 57 | 440 | 88 | 5700 |
| 8 | 2820 | 124 | 800 | 58 | 364 | 88 | 5800 |
| 9 | 2708 | 124 | 900 | 59 | 199 | 96 | 5900 |
| 10 | 2407 | 188 | 1000 | 60 | 179 | 76 | 6000 |
| 11 | 2764 | 172 | 1100 | 61 | 272 | 92 | 6100 |
| 12 | 2424 | 168 | 1200 | 62 | 143 | 100 | 6200 |
| 13 | 487 | 220 | 1300 | 63 | 200 | 84 | 6300 |
| 14 | 415 | 144 | 1400 | 64 | 483 | 96 | 6400 |
| 15 | 420 | 96 | 1500 | 65 | 488 | 72 | 6500 |
| 16 | 343 | 112 | 1600 | 66 | 515 | 76 | 6600 |
| 17 | 276 | 96 | 1700 | 67 | 292 | 88 | 6700 |
| 18 | 352 | 80 | 1800 | 68 | 299 | 100 | 6800 |
| 19 | 268 | 64 | 1900 | 69 | 168 | 68 | 6900 |
| 20 | 331 | 96 | 2000 | 70 | 367 | 104 | 7000 |
| 21 | 391 | 120 | 2100 | 71 | 335 | 132 | 7100 |
| 22 | 376 | 84 | 2200 | 72 | 471 | 88 | 7200 |
| 23 | 491 | 108 | 2300 | 73 | 340 | 64 | 7300 |
| 24 | 236 | 92 | 2400 | 74 | 316 | 96 | 7400 |
| 25 | 120 | 72 | 2500 | 75 | 556 | 72 | 7500 |
| 26 | 148 | 64 | 2600 | 76 | 523 | 88 | 7600 |
| 27 | 159 | 84 | 2700 | 77 | 183 | 92 | 7700 |
| 28 | 284 | 104 | 2800 | 78 | 403 | 84 | 7800 |
| 29 | 164 | 104 | 2900 | 79 | 383 | 68 | 7900 |
| 30 | 220 | 76 | 3000 | 80 | 300 | 108 | 8000 |
| 31 | 99 | 96 | 3100 | 81 | 204 | 104 | 8100 |
| 32 | 155 | 144 | 3200 | 82 | 460 | 92 | 8200 |
| 33 | 167 | 156 | 3300 | 83 | 227 | 104 | 8300 |
| 34 | 167 | 148 | 3400 | 84 | 243 | 96 | 8400 |
| 35 | 212 | 84 | 3500 | 85 | 267 | 148 | 8500 |
| 36 | 176 | 120 | 3600 | 86 | 315 | 80 | 8600 |
| 37 | 151 | 64 | 3700 | 87 | 423 | 96 | 8700 |
| 38 | 312 | 84 | 3800 | 88 | 368 | 96 | 8800 |
| 39 | 463 | 76 | 3900 | 89 | 475 | 136 | 8900 |
| 40 | 244 | 68 | 4000 | 90 | 624 | 88 | 9000 |
| 41 | 256 | 96 | 4100 | 91 | 396 | 112 | 9100 |
| 42 | 256 | 108 | 4200 | 92 | 283 | 96 | 9200 |
| 43 | 268 | 96 | 4300 | 93 | 215 | 120 | 9300 |
| 44 | 168 | 112 | 4400 | 94 | 396 | 88 | 9400 |
| 45 | 148 | 68 | 4500 | 95 | 419 | 80 | 9500 |
| 46 | 204 | 64 | 4600 | 96 | 536 | 116 | 9600 |
| 47 | 152 | 112 | 4700 | 97 | 283 | 156 | 9700 |
| 48 | 255 | 96 | 4800 | 98 | 399 | 84 | 9800 |
| 49 | 112 | 104 | 4900 | 99 | 456 | 132 | 9900 |
| 50 | 383 | 96 | 5000 | 100 | 299 | 136 | 10000 |

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