## Binary Indexed Trees (Fenwick Trees)

Ulises Tirado Zatarain <sup>1</sup> (ulises.tirado@cimat.mx)

<sup>1</sup> Algorists Group

August, 2015



### Outline

- Introduction
  - Basic Problem
  - Basic Idea
- Operations
  - The last bit
  - Reading
  - Writing
  - Get single
- Problems
- Multidimensional BITs





### Introduction - Basic problem

- ullet We have  $\mathfrak n$  boxes and we need to perform two operations:
  - **1** Add marbles to box k:  $\mathcal{O}(1)$
  - ② Sum marbles from box i to j:  $\mathcal{O}(n)$
- Suppose we make q queries. The worst case has time complexity  $\mathcal{O}(qn)$ .
- Using Segment Trees/RMQ: O(q|gn).





#### Introduction - Basic idea

- Taking the binary representation of an integer k, we can compute the sum of elements from 1 to k as sum of set of sums.
- Each set contains son successive number of non-overlapping sums. Then, we can build a data structure indexed by the binary representation:

#### Binary Indexed Tree

Let k is an index of our BIT and r is the position in k of the last bit 1, then  $t_k$  is sum of boxes from index  $k-2^r+1$  to index k. In other words, we mean that k is responsible for indexes in  $[k-2^r+1,k]\cap \mathbb{Z}$ .





#### Introduction - Basic idea

Responsibility table																
k	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
R	1	12	3	14	5	56	7	18	9	910	11	912	13	131	415	116
$\nu_k$	1	0	2	1	1	3	0	4	2	5	2	2	3	1	0	2
$t_k$	1	1	2	4	1	4	0	12	2	7	2	11	3	4	0	29
$\Sigma_{\rm k}$	1	1	3	4	5	8	8	12	14	19	21	23	26	27	27	29

• For example, if we are looking the sum of the first 13 elements, 13 is equal to 1101<sub>2</sub>. Then, we will compute:

$$\Sigma_{1101_2} = t_{1101_2} + t_{1100_2} + t_{1000_2}$$





### Isolating the last bit 1

• Let  $x \in \mathbb{Z}$ , then we can write x in binary representation as  $[x]_2 = y1z$ , where z consists of all zeros, so  $\tilde{z}$  consists of all ones. Then, we have:

$$[-x]_{2} = \widetilde{y1z} + 1$$

$$= \widetilde{y}0\widetilde{z} + 1$$

$$= \widetilde{y}01...1 + 1$$

$$= \widetilde{y}10...0$$

$$= \widetilde{y}1z$$

• Using bitwise operator AND between x and -x, we can get:





## Reading $\Sigma_k \mathcal{O}(\lg n)$

• If we need the sum for some integer k, we add to sum  $t_k$ , substractnlas bit k from itself and repeat this until  $k \le 0$ .

```
\begin{array}{c|c} \text{function read}(k \in \mathbb{Z}^+) \text{begin} \\ & sum \longleftarrow 0; \\ & \text{while } k > 0 \text{ do} \\ & | sum \longleftarrow sum + t_k; \\ & | k \longleftarrow k - (k \ \& \ -k); \\ & \text{end} \\ & \text{return sum}; \\ & \text{end} \end{array}
```





# Writing $t_k : \mathcal{O}(|gn|)$

• When we want to update the k-th element we need to update all indexes whose responsible for the element k.

```
\label{eq:function} \begin{array}{l} \text{function add}(k \in \mathbb{Z}^+, value \in \mathbb{Z}) \, \text{begin} \\ & | \quad \text{while } k \leqslant n \, \, \text{do} \\ & | \quad t_k \longleftarrow t_k + value \, ; \\ & | \quad k \longleftarrow k + (k \, \& \, -k); \\ & \text{end} \end{array}
```





# Get single $v_k$ : $O(\lg n)$

```
function get (k \in \mathbb{Z}^+) begin
     sum \leftarrow t_k;
     if k > 0 then:
         z \leftarrow k - (k \& -k);
         while k \neq z do
           sum \leftarrow sum - t_k; 
k \leftarrow k - (k \& -k);
          end
     end
     return sum;
end
```



#### **Problems**

- There is an array of n cards. Each card is putted face down on table. You have two queries:
  - T(i,j) turn cards from index i to index j, include i-th and j-th card card which was face down will be face up; card which was face up will be face down
  - 2 Q(i) answer 0 if i-th card is face down else answer 1
- Floating Median

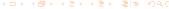




# 2D BITs $\mathcal{O}(q | g m | g n)$

```
function add(x,y \in \mathbb{Z}^+,value \in \mathbb{Z}) begin
       sum \leftarrow 0:
       while x \le n do
               \hat{\mathbf{y}} \longleftarrow \mathbf{y};
               while \hat{y} \leqslant m do
                \begin{array}{c} t_{x,\hat{y}} \longleftarrow t_{x,\hat{y}} + value; \\ \hat{y} \longleftarrow \hat{y} + (\hat{y} \& -\hat{y}); \end{array}
                end
               x \leftarrow x + (x \& -x);
       end
end
```





### References |

- TopCoder
- Math Porn @ tumblr
- Multidimensional BITs



