Train IV

Oct 3. Ye Cao.

Close up Range (Sum) Query

0	1	2	3	4	5	6	7	8	9
2	1	7	3	1	8	9	19	21	31

Cumulative array $O(N)$ Update
$O(1)$ Sum

Square root decomposition $O(\sqrt{n})$ Sum
$O(1)$ Update

Fenwick Tree $O(Log(N))$ Sum
$O(Log(N))$ Update

How to accomplish a tree with merely an array?

Recall Heap array representation:

- Child 1 : $i * 2 + 1$
- Child 2 : $i * 2 + 2$

Lesson: Beautiful Number theory property.

Question to consider:

"Is there any number theory property that allows us to construct a tree on array that's efficient on sum query and point update"?

First we need to understand what BIT Manipulation is :

Bit here refers to binary representation of a number

$2_{bit} = 10$
$3_{bit} = 11$
$4bit = 100$

Specifically, two types of manipulation method 'll be used

- Inverse

$$(a1b)^{-} = a^{-1}1b$$

- And

$1 \& 1 = 1$
$1 \& 0 = 0$

$$\begin{aligned}(a1b)^{-} &= a^{-}0b^{-}+1 \\ &= a^{-1}0(11\cdots1)+1 \\ &= a^{-1}1b\end{aligned}$$

Now consider

$$\begin{aligned}(a1b) \& (a1b)^{-} &= a1b \& a^{-1}1b \\ &= 1b\end{aligned}$$

Last bit extraction

+= Last bit Go to parent (Update)

-= Last bit Go to Child (Sum Query)

Example:

$P(1) = 10$

$(10) = 100$

$(11) = 100$

$(100) = 1000$

$(1010) = 1100$

$(1001) = 1010$

It's actually not very appropriate to say children and parent because if you inspect closely Par(4) = 8, P(5) = 6, yet Child(5) = 4. Thus here parent and child are not inverse of each other i.e. Child(a) = b Not imply Parent(b) = a.

It's more important to understand what's happening under the hood:

Say Query(5):

1. rest = 0
2. rest += BIT[5];
3. rest += BIT[4];