# Fenwick Data Structure

Fenwick ဆိုတာ Array တစ်ခုရှိတယ် ဆိုပါစို့။ ထို array ရဲ့ value အားလုံးကို ပေါင်းချင်တယ်ဆိုရင် တစ်ခန်းချင်းစီထောက်ပြီး ပေါင်းရမည်။ အဲ့လိုပေါင်းနေမယ်ဆိုရင် range ကြီးလာတာနှင့်အမျှ အချိန်ကြာ လာပါမည်။ အက်တော့ prefix sum တွက်ထားပြီးသား array တစ်ခုတည်ဆောက်လိုက်ပါမယ်။

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| array | 2 | 4 | 8 | -5 | 7 | 0 | 6 | 11 | -7 | 2 |  |
| Prefix sum of array | 0 | 2 | 6 | 14 | 9 | 16 | 16 | 22 | 33 | 26 | 28 |

Table - 1

ဆက်လက်ပြီး prefix sum array ကို ဘယ်လိုတည်ဆောက်သွားလဲဆိုတာ ရှင်းပြပါမယ်။ index 0 နေရာမှာ value 0 ကိုပဲ ပြန်ထည့်ထားပါတယ်။ index 1 နေရာမှာ array ရဲ့ Index 0 ( value 2 ) ကို ထည့်ထားပါတယ်။ စတင်ပေါင်းမည့်နေရာက prefix sum ရဲ့ index 2 နေရာကနေ စတင်ပြီး တွက်ရတော့မှာပါ။ အထက်ပါ table မှာဆိုရင် prefix sum ရဲ့ index 2 သည် array ရဲ့ index 0 နှင့် 1 ရဲ့ ပေါင်းခြင်းဖြစ်ပြီး prefix sum ရဲ့ index 3 သည် arr of index (0+1+2) ပေါင်းခြင်း ဖြစ်ပါတယ်။ ထိုနည်းအတိုင်း တွက်လာပြီး နောက်ဆုံးတွင် prefix sum ရဲ့ index 10 သည် array of index 0 to 9 ကို ပေါင်းခြင်း ဖြစ်ပါသည်။ ဒီနေရာမှာ array of element တွေ အရမ်းများလာတဲ့အခါ ပေါင်းရတဲ့ကိန်းတွေများ လာတဲ့အတွက် အချိန်ပိုကြာနိုင်သလို မှားဖို့လည်း ရာခိုင်နှုန်းများလာပါတယ်။ လူမှန်ရင် အမှားနဲ့ မကင်းဘူးကိုး :3။ အဲ့ဒီတော့ ပိုလွယ်ပြီး အမှားနည်းမည့် နည်းလမ်းကို ပြောပြပါမယ်။ ဆိုကြပါစို့ prefix sum of index 4 နေရာက value ကို လိုချင်တယ်ဆိုရင် array of index 3 နှင့် prefix sum of index 3 ကို ပေါင်းလိုက်ရုံပဲ ဖြစ်ပါတယ်။ အလွယ်မှတ်နိုင်ဖို့အတွက်ဆိုရင် **index = 4; arr[index-1] +prefix[index-1] = prefix[index]** ဖြစ်ပါတယ်။ အခြားနည်းလမ်းတွေ များစွာရှိသော်လည်း ဒီနည်းလမ်းကို အလွယ်ဆုံးလို့ ယူဆတဲ့အတွက် ထည့်သွင်းဖော်ပြရခြင်း ဖြစ်ပြီး မိမိနှစ်သက်သည့် နည်းလမ်းဖြင့်လည်း တွက်ချက်နိုင်ပါ သည်။ ပိုမိုမြင်သာအောင် နောက်ထပ် array တစ်ခုတည်ဆောက်ပြပါမည်။

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| array | 5 | 8 | 10 | 6 | 7 | -8 | 7 | 12 | 2 | 4 |  |
| Prefix sum of array | 0 | 5 | 13 | 23 | 29 | 36 | 28 | 35 | 47 | 69 | 73 |

Table – 2

အထက်ပါ table-2 ကို သုံးပြီးတော့ ဆက်လက်ရှင်းပြပါမည်။ အခုဆိုရင် original array နှင့် prefix sum ကို တွက်ထားတဲ့ array နှစ်ခု ရှိနေပါပြီ။ ကျွန်တော်တို့က original array ရဲ့ index 0 to 9 ပေါင်းခြင်းကို လိုချင်တယ်ဆိုပါစို့။ prefix sum ထဲက index 9 ကို ထုတ်ပြလိုက်ရုံဖြင့် လိုချင်သည့် ရလာဒ်ကို တန်းရပါမည်။ နောက်တစ်ခုအနေနဲ့ index 2 to 7 ပေါင်းခြင်းကို လိုချင်တယ်ဆိုပါစို့။ prefix sum of index 7 ထဲကနေ prefix sum of index 2 ကို နှုတ်လိုက်ရုံပဲ ဖြစ်ပါတယ်။

**prefix[7]-prefix[2] = sum of arr [2,7]**

prefix[7] = 35

prefix[2] = 13

sum of arr [2,7] = prefix[7] – prefix[2]

= 35 – 13

= 22

အထက်ပါ နည်းလမ်းမှန်မမှန်ကို စစ်ဆေးနိုင်ရန် original array ရဲ့ index 2 to 6 ကို ပေါင်းပေးရပါမယ်။ range တွက်တဲ့အခါ နောက်ဆုံး range ကို တစ်လျှော့ပြီး တွက်ပေးရတာမို့ပါ။ index 2 to 7 ဆိုရင် index 2+3+4+5+6 ကို ပေါင်းပေးရမှာပါ။

Sum of arr [2,7] = 10+6+7-8+7

= 22

Prefix sum of array ဖြင့် နောက်ထပ် array အသစ်တစ်ခု တည်ဆောက်ခြင်း နည်းလမ်းကို အသုံးပြုပြီး မိမိတို့လိုချင်သည့် index range ကို တွက်ယူနိုင်ပါတယ်။ သို့သော် original array ထဲက value တစ်ခုခုကို ပြောင်းလဲလိုက်မည်ဆိုလျှင် အစကနေ အကုန်လုံးကို ပြန်တွက်ရပြန်ပါတယ်။ အဲ့ဒီလို မတွက်ရအောင် ဘယ်လိုဖြေရှင်းမလဲပေါ့။ ဒီနေရာမှာ ပိုမိုမြန်ဆန်အောင် fenwick tree ကို စမတွက်ခင် ကတည်းက ဆောက်ထားပေးလိုက်တာပါ။ ဆက်လက်ပြီး fenwick tree ဆောက်နည်း အကြောင်းကို ရှင်းပြသွားပါမည်။ binary to decimal သို့ ပြောင်းလဲခြင်း ကဲ့သို့သော numbering system ဖြင့် ဆက်နွယ်သော အပိုင်းများကို သိရှိပြီးသားဟု ယူဆထားသည့်အတွက် ထည့်သွင်း မဖော်ပြတော့ပါ။

Fenwick Tree အကြောင်းကို မရှင်းပြခင် (LSB) least significant bit ကို အရင်ဆုံးသိဖို့ လိုအပ်ပါတယ်။ LSB ကို တွက်ချက်ရာတွင် နည်းလမ်းနှစ်မျိုးဖြင့် တွက်ချက်နိုင်ပါတယ်။ တစ်မျိုးသည် သင်္ချာတွက်ရာတွင် အဆင်ပြေစေပြီး နောက်တစ်မျိုးသည် program များရေးသားရာတွင် အဆင်ပြေစေ ပါသည်။ ဥပမာ - index 10 ဆိုပါစို့။ binary အနေဖြင့်ဆိုလျှင် 1010 ရပါတယ်။ LSB ကို ရှာမည်ဆိုလျှင် binary value ကို ညာဘက်ကနေ စတင်ရေတွက်ရပြီး နောက်ဆုံး 1 ဂဏန်းသည် LSB ဖြစ်ပါသည်။ 1010 တွင် LSB သည် 2 ဖြစ်ပါသည်။ index 9 ဆိုလျှင် binary အရ 1001 ဖြစ်ပြီး LSB သည် 1 ဖြစ်ပါသည်။ ဆိုတော့ index 8 အရ binary 1000 ဆိုလျှင် LSB သည် 8 ဖြစ်မှာပါ။

ဆက်လက်ပြီး ရလာသော LSB ကို 2**LSB**-1 ဖြင့် ဆက်လက်တွက်ပေးရပါမယ်။ ထိုအခါမှသာ index တစ်ခုချင်းစီရဲ့ range value အသီးသီးကို ရရှိမှာပါ။

Index 10 (decimal) = 1010 (binary)

LSB (least significant bit) = 2 (decimal)

Range value = 2**2-1** =2**1**= 2

အထက်ပါ example တွင် index 10 ၏ range value သည် 2 ဖြစ်ပါသည်။ အခြား index များကို လည်း ထိုနည်းအတိုင်း ဆက်လက်တွက်ချက်နိုင်ပါသည်။ ဒုတိယနည်းလမ်းမှာ x & (-x) formula ကို အသုံးပြုထားပြီး တွက်ချက်ခြင်း ဖြစ်ပါသည်။ အက်တာကို မပြောခင် အရင်ဆုံး value တွေကို (-) အနှုတ်ကိန်း ပြောင်းလိုက်သည့်အခါ binary အဖြစ် ဘယ်လိုပြောင်းရလဲဆိုတာ နားလည်ဖို့ လိုပါတယ်။ ဥပမာ index 10 ဆိုလျှင် binary အရ 1010 ဖြစ်ပါတယ်။ range တွက်တဲ့ formula x & (-x) အရ x တော့ ရပါပြီ။ (-x) အတွက် ကျန်နေပါသေးတယ်။ အဲ့ဒီတော့ binary ကို အနှုတ်ကိန်းပြောင်းလိုလျှင် ပထမဆုံး 1’s complement လုပ်ပေးရပါတယ်။ 1’s complement ဆိုတာ ပြောင်းပြန် (invert) ပြုလုပ်ပေးခြင်း ဖြစ်ပါတယ်။ 1010 ရဲ့ invert ဆိုတော့ 0101 ရပါမယ်။ ထိုကိန်းကို 2’s complement ပြုလုပ်ပေးလိုက်လျှင် ကျွန်တော်တို့ အလိုရှိသော အနှုတ်ကိန်းကို ရပါပြီ။ ဆိုတော့ 1’s complement ကို 1 ပေါင်းပေးလိုက်လျှင် 2’s complement ရပါပြီ။ စာနဲ့မရှင်းလျှင် math အနေဖြင့် ပိုမိုနားလည်စေရန် အောက်တွင် ဖော်ပြထားပါသည်။

Index 10 = 1010

1’s complement = 0101

+ 1

2’s complement = 0110

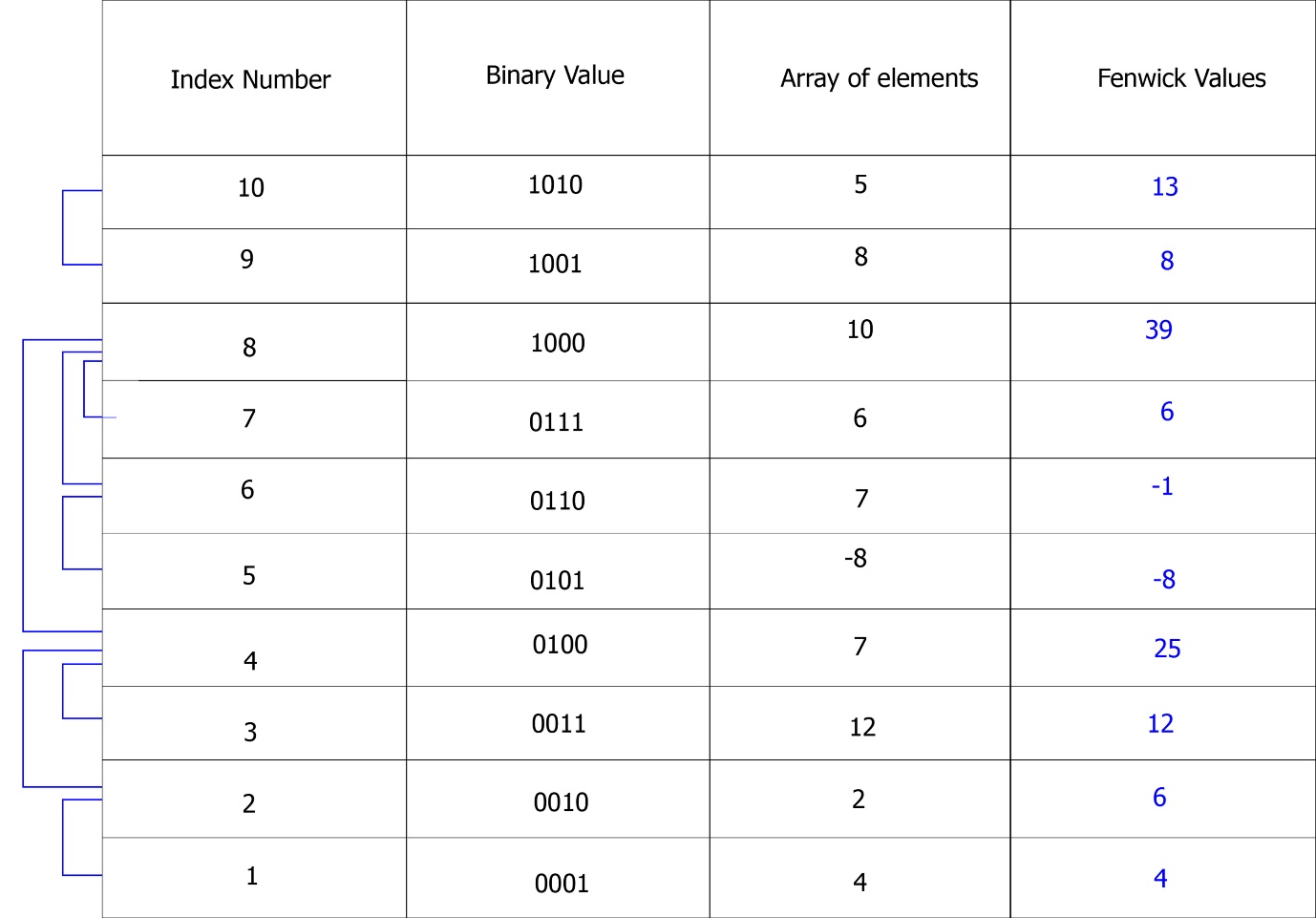
ရလာတဲ့ 2’s complement ကို decimal ပြောင်းကြည့်လျှင် အနှုတ်ကိန်းကို အဖြစ်ပြန်ရပါမည်။ ဆိုတော့ index 10 ရဲ့ range ကို ဆက်လက်တွက်ချက်ကြည့်ရအောင်။

Index 10 = (x) & (-x)

= 1010 & 0110

= 0010 (binary) = 2 (decimal)

အထက်ပါ နမူနာတွင် (x) & (-x) formula ကို အသုံးပြုထားပြီး bitwise operator မှ & ကို အသုံးပြုထားပါသည်။ နှစ်ခုလုံးတူမှ 1 ရမှာပါ။ သို့မဟုတ်လျှင် 0 သာ ရမှာ ဖြစ်ပါသည်။ range တွက်နည်း ရပြီဆိုတော့ tree ဆောက်နည်းကို ဆက်လက်ရှင်းပြသွားပါမည်။



ပထမဆုံး index 1 ရဲ့ range value ကိုတွက်ရပါမည်။ ပြီးလျှင် ထို range value အတိုင်း ဆက်လက်တွက်ယူရမှာ ဖြစ်ပါသည်။ range value တွေရပြီဆိုရင် range query ကို စဉ်းစားတတ်ဖို့ လိုအပ်ပါသေးတယ်။

ဥပမာ - index 7 ဆိုပါစို့။ သူ့ရဲ့ range value က 1 ဖြစ်တဲ့အတွက် tree တည်ဆောက်ရာတွင် index 8 ကို ဆက်ကြည့်ပေးရပါမည်။ Index 8 ရဲ့ range value က 8 ဖြစ်တဲ့အတွက် index 8 နေ နောက်ထက် 8 လုံးရွေ့စရာမရှိတဲ့အတွက် ignore လုပ်လိုက်ပါတယ်။ ဆက်လက်ပြီးတော့ အထက်ပါ table ကို ရှင်းပြပါမည်။

Range value of index 1 = 1

Range 1 ဆိုတော့ index 2 ကို index 1 ရဲ့ value ဖြင့် ပေါင်းပေးရပါမယ်။ အဲ့ဒီတော့

index[1] + index[2] = fenwick[2]

4 + 2 = 6

ဆက်လက်ပြီး index 2 ရဲ့ range value က 2 ဖြစ်ပြီး နှစ်နေရာတိုးရမှာ ဆိုတော့ index 4 ဖြစ်သွားပါတယ်။ ခုန ရထားတဲ့ fenwick[2] ကို index[4] ဖြင့် ပေါင်းလိုက်လျှင် fenwick[4] ကို ရရှိမှာ ဖြစ်ပါတယ်။ ထိုနည်းအတိုင်း ဆက်လက်တွက်ချက်လျှင် binary index tree or fenwick tree ကို ရရှိပါပြီ။ ထို value ထဲမှာ point တစ်ခုခုကို Update ပြုလုပ်လျှင် fenwick tree တစ်ခုလုံးကို ပြောင်းစရာမလိုပါဘူး။ ဥပမာ - index 7 ရဲ့ value ကို ပြောင်းလဲလိုလျှင် fenwick[7],fenwick[8] နှစ်မျိုးကိုသာ ပြောင်းပေးရမှာပါ။ ဘာကြောင့်လဲဆိုတော့ index 7 ရဲ့ range value က 1 ပါ။ အဲ့ဒီတော့ index 8 ဖြစ်သွားပါတယ်။ index 8 ရဲ့ range value က 8 ဆိုတော့ ignore ဖြစ်သွားပါတယ်။

Fenwick tree တည်ဆောက်လိုက်တာကြောင့် index 7ကို ချိန်းမယ်ဆိုလျှင် အားလုံးကို ချိန်းစရာမလိုတော့ပဲ index 8 နှင့် index 7 နှစ်မျိုးကို ချိန်းလိုက်လျှင် ဆက်လက်အသုံးပြုရန် မှန်ကန်ပြီ ဖြစ်ပါတယ်။

စာမူကြမ်းပြီးချိန်::: 7:47 AM 12/27/2021 MON