Додатак А: Фенвиково стабло

|  |  |
| --- | --- |
| Аутор: Петар Величковић |  |

## Опис и анализа структуре:

**Фенвиково стабло** *(Fenwick Tree / Binary Indexed Tree / BIT)*, је елегантна структура за ефикасно чување и ажурирање кумулативних сума неког низа величине . Подржава следеће две операције:

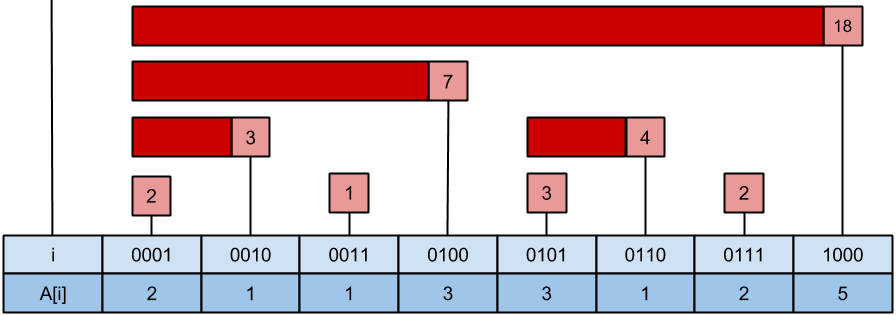
1. – повећавање -тог елемента у низу за вредност , тј. ;
2. – рачунање кумулативне суме на -тој позицији, тј. .

Ова структура користи само један низ, да би процесирала горенаведене упите. Посматраћемо индексе овог низа као бинарне бројеве (нпр, ). Сваки индекс у овом низу ће чувати неку подсуму низа да би нам олакшао да брзо реконструишемо праву кумулативну суму; конкретно, уколико је са означимо позицију последње јединице у бинарном запису индекса , онда важи следеће:

Доња граница за суму, , је еквивалентна томе да се последња јединица у бинарном запису броја премести на најмање значајну позицију. На пример:

(приметити како ).

Ради илустрације, погледати доњу слику. Најдоњи ред представља низ , док црвена поља представљају низ ; ”реп” иза сваког цревног поља представља подниз над којим се рачуна сума.



Слика . Илустрација BIT низа за А[] = {2, 1, 1, 3, 3, 1, 2, 5}

Приметимо да се сада операција може раставити на суму више елемената низа , тако да је први индекс једнак броју , следећи једнак броју који се добије када заменимо последњу јединицу у бинарном запису нулом, и тако даље. На пример (за гореприказани низ):

.

Аналогно, када извршавамо операцију , потребно је повећати само оне елементе низа који обухватају поље ; исто као што при рачунању кумулативних сума стално одузимамо последњу јединицу, ова операција је обрнута; додајемо у сваком кораку да бисмо добили следећи индекс низа који треба ажурирати (где је позиција последње јединице у тренутном индексу). Ово радимо све док је индекс унутар низа, тј. док не прекорачи . Дакле, уколико желимо за низ дат горе да извршимо , секвенца операција би била следећа:

; ;

; ;

; ;

Овиме имамо све што нам је потребно да бисмо имплементирали Фенвиково стабло; само нам треба начин да ефикасно извучемо број који има јединицу само на последњем месту као број . Ово можемо добити као вредност израза , где & представља операцију битовне конјункције (bitwise AND). Ово функционише зато што се бројеви чувају у рачунарима у форми комплемента двојке; да бисмо добили број супротан неком броју, потребно је обрнути му све битове у бинарном запису и додати му 1.

Сада можемо написати C++ код за операције и :

int bit[MAX\_N];

void update(int x, int val)

{

while (x <= n)

{

bit[x] += val;

x += (x & -x);

}

}

int read(int x)

{

int ret = 0;

while (x > 0)

{

ret += bit[x];

x -= (x & -x);

}

return ret;

}

Пошто ће обе операције највише да изврше број корака једнак броју цифара у бинарном запису броја , лако је закључити да су обе операције сложености . Фенвиково стабло, као структура која је јако једноставна за кодирање, која нуди ефикасно (и у меморијском и у временском смислу) манипулисање кумулативним сумама (које се јако често појављују као подпроблем у многим задацима), и која се лако генерализује на више димензија, је стога структура коју би сваки такмичар требао да има у свом ”арсеналу”.

**Разни задаци:**

***z-magija*** ([www.z-trening.com](http://www.z-trening.com)), ***It's a Murder!*** ([www.spoj.com](http://www.spoj.com)), ***Matrix Summation*** ([www.spoj.com](http://www.spoj.com)).