MaratonaCIn

Seletiva 2020

Problema

Problema inicial:

Existem N baldes numerados 1 a N. Deseja-se fazer as seguintes operações:

Op. 1: adicione/remova X bolas do balde i.
Op. 2: Conte quantas bolas existem entre os baldes [L,R].

Problema

MaratonaCIn

Problema inicial:

Existem N baldes numerados 1 a N. Deseja-se fazer as seguintes operações:

Op. 1: adicione/remova X bolas do balde i.
Op. 2: Conte quantas bolas existem entre os baldes [L,R].

Como resolver isso ?

Solução Trivial

MaratonaCIn

Problema inicial:

Existem N baldes numerados 1 a N. Deseja-se fazer as seguintes operações:

Op. 1: adicione/remova X bolas do balde i.
Op. 2: Conte quantas bolas existem entre os baldes [L,R].

Como resolver isso ?

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100005;
vector<int>balde(maxn);

int main(){

    int q;
    cin >> q;
    while(q--){
        int op;
        cin>>op;
        if(op==1){ // operacao 1
            int i,x;
            cin >> i >> x;
            balde[i] += x;
        }
        else{ // operacao 2
            int l,r;
            cin >> l >>r;
            int ans = 0;
            for(int i=l;i<=r;i++){
                ans += balde[i];
            }
            cout << ans << endl;
        }
    }
}
```

Solução trivial

MaratonaCIn

Problema inicial:

Existem N baldes numerados 1 a N. Deseja-se fazer as seguintes operações:

Op. 1: adicione/remova X bolas do balde i.
Op. 2: Conte quantas bolas existem entre os baldes [L,R].

Como resolver isso ?

Complexidade:

Op1 = O(1)
Op2 = O(n)

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100005;
vector<int>balde(maxn);

int main(){

    int q;
    cin >> q;
    while(q--){
        int op;
        cin>>op;
        if(op==1){ // operacao 1
            int i,x;
            cin >> i >> x;
            balde[i] += x;
        }
        else{ // operacao 2
            int l,r;
            cin >> l >>r;
            int ans = 0;
            for(int i=l;i<=r;i++){
                ans += balde[i];
            }
            cout << ans << endl;
        }
    }
}
```

Motivação

MaratonaCIn

A forma trivial de resolver este tipo de problema pode ser $O(n^2)$ no pior caso. Precisamos de um tipo de estrutura de dados que satisfaça as duas operações anteriores e reduza a complexidade.

Existem duas estruturas para isso (uhul!)

- Binary indexed Tree (BIT)
- Segment Tree (SegTree)

MaratonaCIn

Segment Tree

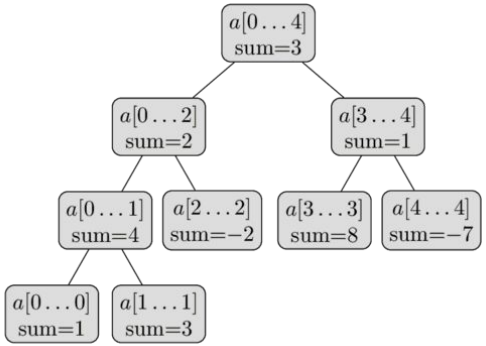
Aula 9

Segment Tree

MaratonaCIn

O que é ?

Árvore binária. Cada nó desta árvore guarda alguma informação sobre um segmento do vetor.



Se um Nó guarda informação sobre o intervalo [2,8]...

-> o filho à esquerda guardará informações sobre o intervalo [2,5]

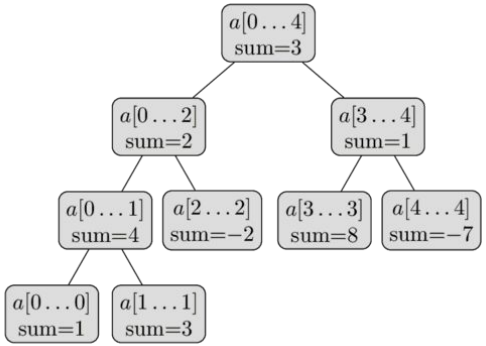
-> o filho à direita guardará informações sobre o intervalo [6,8]

Segment Tree

MaratonaCIn

O que é ?

Árvore binária. Cada nó desta árvore guarda alguma informação sobre um segmento do vetor.



Se um Nó guarda informação sobre o intervalo [2,8]...

-> o filho à esquerda guardará informações sobre o intervalo [2,5]

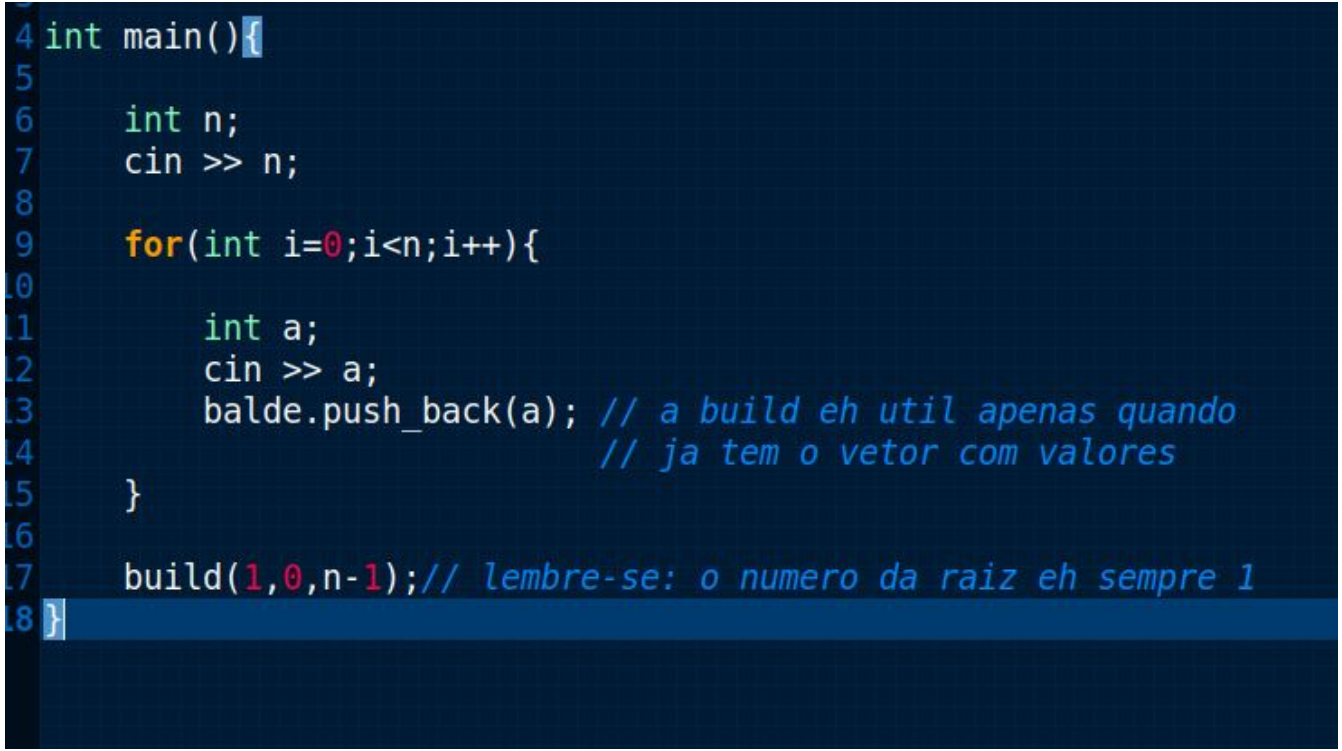
-> o filho à direita guardará informações sobre o intervalo [6,8]

Como codar isso ?

Build


```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100005;

vector<int> baldes(maxn);
vector<int> tree(4*maxn); // sempre colocar 4*maxn. Memoria necessaria

void build(int pos,int i,int j){

    int esq = 2*pos;
    int dir = 2*pos + 1;
    int mdi = (i+j)/2;

    if(i == j){ // cheguei em uma folha
        tree[pos] = balde[i]; // i e j sao os ponteiros que identificam o intervalo do array
    }
    else{

        tree[pos] = 0; // reseta
        build(esq,i,mid);
        build(dir,mid+1,j);

        tree[pos] = tree[esq] + tree[dir];
    }
} // essa funcao eh pouco usada, mas eh bom saber...
```

Chamando a build na main



```
int main(){

    int n;
    cin >> n;

    for(int i=0;i<n;i++){

        int a;
        cin >> a;
        balde.push_back(a); // a build eh util apenas quando
                            // ja tem o vetor com valores
    }

    build(1,0,n-1);// lembre-se: o numero da raiz eh sempre 1
}
```

UPDATE


```
6     // pos = indice do node da arvore
7     // [i,j] = indices do intervalo do vetor
8     // x = posicao que quero atualizar no meu array
9     // value = valor a ser atualizado
10
11 void update(int pos,int i,int j,int x,int value){
12
13     int esq = 2*pos;
14     int dir = 2*pos + 1;
15     int mid = (i+j)/2;
16
17     if(i == j){
18
19         tree[pos] += value;
20     }
21     else{
22
23         tree[pos] += value;
24         if(x <= mid) update(esq,i,mid,x,value);
25         else update(dir,mid+1,j,x,value);
26     }
27 }
```

Chamando na main:

```
31     int x,val;
32     cin >> x >> val;
33     update(1,0,n-1,x,val);
34 }
```

Query

MaratonaCIn

```
// [l,r] =intervalo a ser consultado

int query(int pos,int i,int j,int l,int r){

    int esq = 2*pos;
    int dir = 2*pos + 1;
    int mid = (i+j)/2;
    int ret;

    if(j < l || i > r){ // o segmento atual esta fora do segmento desejado
        ret =  0;
    }
    else if(i >= l && j<=r){ // o segmento atual esta contido no segmento desejado
            ret = tree[pos];
    }
    else{ //o segmento atual possui apenas uma parte pertencente ao segmento desejado
        ret =0;

        ret += query(esq,i,mid,l,r);
        ret += query(dir,mid+1,j,l,r);
    }

    return ret;
}
```

Chamando na main

```
int main(){

    int l,r;
    cin>>l>>r;

    cout << update(1,0,n-1,l,r) << endl;

}
```

Complexidade

Como a estrutura de dados é um árvore binária, cada operação levará um tempo O(logN).

No pior caso, o algoritmo rodará em O(N*logN)

Complexidade

Como a estrutura de dados é um árvore binária, cada operação levará um tempo O(logN).

No pior caso, o algoritmo rodará em O(N*logN)

Essa estrutura de dados é bastante poderosa!!!
Será possível resolver vários novos problemas.

MaratonaCIn

Binary Indexed Tree

Aula 9

BIT

MaratonaCIn

→ Imagine que você possui um número n=10 de caixas, onde cada caixa contém uma determinada quantidade de laranjas:
{3, 1, 4, 4, 2, 7, 10, 6, 9, 5}.

→ Você agora quer carregar um caminhão com algumas dessas caixas, começando da primeira até encher a capacidade do caminhão:
Se no caminhão cabem 4, pega-se as quatro primeiras caixas{3, 1, 4, 4} somando um total de 12 laranjas.

→ Perceba que a quantidade de laranjas no caminhão será dada pela soma das caixas do intervalo (1, k), sendo k a capacidade do caminhão.

Como calcular o resultado?

É possível calcular o resultado somando todos os elementos, com uma complexidade linear.

Problema:

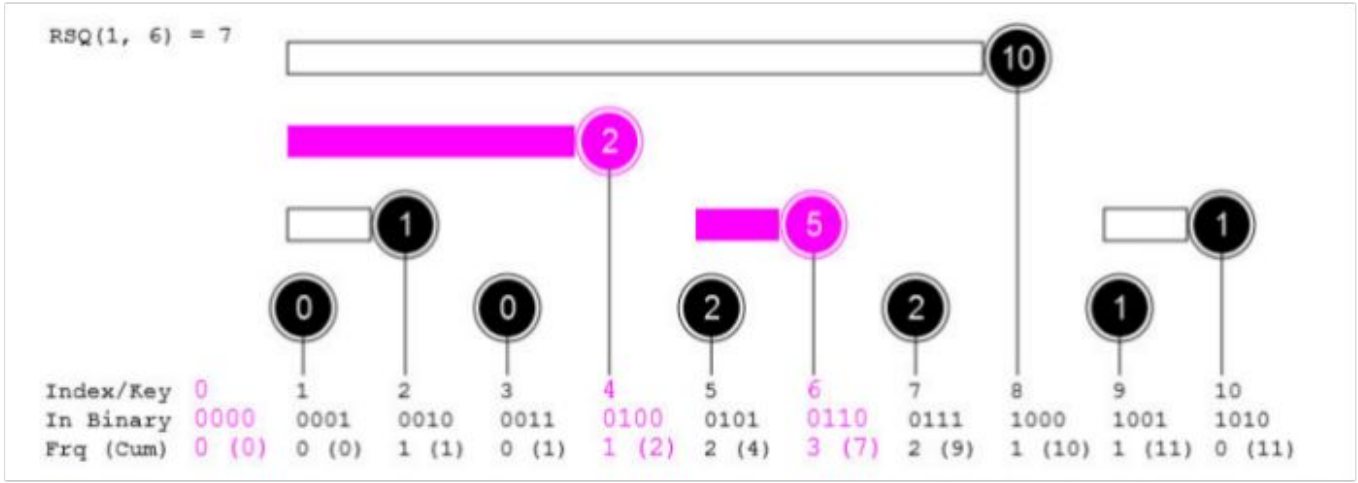
Imagine agora que o número de laranjas nessas caixas podem mudar, e existem vários caminhões diferentes que podem transportar as laranjas. Você quer saber quanto um determinado caminhão disponível carregaria se fosse usado naquel momento.

BIT

Nesse caso, somar todos os elementos do intervalo várias vezes pode ser custoso, então para resolver o problema utilizaremos uma BIT.
A ideia da BIT é um array, onde cada índice é responsável por um determinado intervalo de valores de acordo com a cadeia de bits do número do indice.
Cada índice é responsável pelo intervalo de [i-LSO(i)+1, i],sendo LSO(i) o número que representa o menos significativo bit 1 setado de i.
EX: 7 = 0111, LSO(7) = 1 = 0001.
12 = 1100, LSO(12) = 4 = 0100.

```
int lso(int i){
    int least_significant_one = i & (-i);
    return least_significant_one;
    /*retorna o numero que contem
    apenas o bit menos significante de
    i como 1;*/
}
```

BIT



→ O a BIT acima representa o vetor v {0, 1, 0, 1, 2, 3, 2, 1, 1, 0}

→ Perceba que o 0 não é usado, o 1 guarda v[1], o 2 guarda a soma [v[1]+v[2]], o 3 guarda v[3], o 4 guarda [v[1]+v[2]+v[3]+v[4]].

BIT

MaratonaCIn

Função para alterar um valor na BIT

```
void ajuste(int pos, int val){
    if(pos == 0)return;//não insira no 0;
    ft[pos]+=val;
    for(pos = pos+lso(pos); pos <= n; pos += lso(pos)){
        ft[pos]+=val;
    }
}
```

Função para calcular a resposta de um intervalo

```
ll intervalo(int pos){
    ll ans = ft[pos];
    for(pos = pos-lso(pos); pos > 0; pos -= lso(pos)){
        ans += ft[pos];
    }
    return ans;
}
```

BIT

E se eu quiser calcular um intervalo[a, b] que a não é 1?
Calcule intervalo(b) – intervalo(a-1)

```
ll intervalo(int ini, int fim){
    ll res = acumulado(fim) - acumulado(ini-1);
    return res;
}
```

```
#define maxn 212345
typedef long long ll;

ll ft[maxn+1]; //Minha BIT
int n;
```

Lembre-se de colocar um tamanho Máximo+1,
pois o 0 não é utilizado!

Complexidade

A estrutura consegue resolver uma query ou ajustar um valor numa complexidade O(logN), pois utiliza operaçõe com os bits dos índices.

Caso seja realizada N querys, a complexidade é O(N*logN)