Team Notebook

September 23, 2025

Contents

1 01-utilidades

1.1 00-base

```
// g++ file.cpp -o prog -std=c++17 -Wfatal-errors
// ./prog < in

#include <bits/stdc++.h>
using ll = long long; //opcionais, so alias para long long e
      unsigned long long
using ull = unsigned long long; //opcionais, so alias para
     long long e unsigned long long

using namespace std;
int main() {
    //para otimizar a entrada e saida de dados, podemos usar
         o seguinte:
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout.tie(0);

    return 0;
}
```

1.2 03-comandos-uteis

```
Comandos Uteis (Linux)

< arquivo - usa o conteudo como entrada
> arquivo - grava a saida no arquivo
time - mede quanto tempo o codigo levou
diff A B - compara o conteudo dos arquivos
chmod +x - torna executavel (ex.: chmod +x script.sh)
cat - imprime um arquivo
grep - busca uma palavra/padrao
head / tail - mostra inicio/fim de arquivos
mkdir - cria diretorio
rm / cp / mv - remove / copia / move arquivos
```

2 02-manipulacao-binaria

2.1 00-Operações-Bitwise

```
/* Operaoes  bit a bit:
       ~x op  binria  complementar
       x & y op E bit a bit entre x e y
       x | y op OU bit a bit entre x e y
       x ^ y op XOR bit a bit entre x e y
       x << int op SHIFT LEFT de i posies
       x >> int op SHIFT RIGHT de i posies
*/
```

2.2 01-isSet-and-setBit

```
bool isSet(int bitPosition, int number)
{
    bool ret = ((number & (1 << bitPosition)) != 0);
    return ret;
} //essa  funo
//serve para verificar se o bit na posio `bitPosition` est "
     ligado" (=1)

bool setBit(int bitPosition, int number)
{
    return (number | (1 << bitPosition));
}
// essa  funo  serve para ativar o bit na `bitPosition`
```

2.3 02-builtin

```
__builtin_ctz(x)      // Numero de zeros a direita
__builtin_clz(x)      // Numero de zeros a esquerda
__builtin_popcount(x) // Numero de bits 1
     __builtin_popcountll(x) para long long x
__builtin_ffs(x)      // Posicao do primeiro bit 1 (1-indexed
     )

// Adicionar ll ao final para long long [__builtin_clzll(x)]
```

2.4 AND

```
/*
     Nesse caso, a  comparao  de bits resulta 1 apenas
          quando os dois bits comparados forem 1.

     0b1100'1100 (204)
     0b1010'1010 (170)
     -----------
     0b1000'1000 (136)
   */
    uint8_t a = 204;
    uint8_t b = 170;
    uint8_t c = a & b; // vai retornar 136
```

2.5 NOT

```
uint8_t x = 170; // 0b1010'1010
uint8_t y = ~x; // 0b0101'0101 (85)

/* ! a inverso  no    troca de lugar. o que  zero vira um
    e vice e versa
    */
    /* O operador ~ inverte os bits:
       1010 1010 (170)

       0101 0101 (85)

       Como uint8_t  um  nmero  SEM SINAL, o valor final  85.
    */
```

2.6 OR

```
// OPERAO  OR (|) - retorna 1 apenas quando pelo menos 1
     dos bits for = a 1:
   /*
      0b1100'1100
      0b1010'1010
      -----------
      0b1110'1110
   */

int a = 10;
int b = 5;
int c = a | b; // resulta 15
```

2.7 SHIFTS

```
 //OPERADOR SHIFT LEFT (<<) - desloca bits para a esquerda e
      multiplica por 2
   uint8_t x = 5;    // 0000 0101
   uint8_t y = x << 1; // 0000 1010 = 10
   //em termos de  multiplicao , seria 2
```

```
   //outro exemplo:
   int x = 3;
   int y = x << 4; // 3 * 2 = 3 * 16 = 48


   //OPERADOR SHIFT RIGHT (<<) - desloca bits para a direita
        e divide por 2
   uint8_t x = 20;    // 0001 0100
   uint8_t y = x >> 2; // 0000 0101 = 5

}
```

2.8 XOR

```
 //OPERADOR XOR (^) - retorna 1 apenas quando os bits forem
      diferentes:
   /*
      0b1100'1100
      0b1010'1010
      -----------
      0b0110'0110
   */


int a = 10;  // 1010 em binrio
int b = 5;   // 0101 em binrio
int c = a ^ b; //resulta em 15 (1111)
```

2.9 zPrint$_{b}$*inario*

```
//as  funes  a seguir printam o valor em binrio de um nmero
void printBinary(uint32_t value) {
   for (int i = 31; i >= 0; --i) {
       cout << ((value >> i) & 1u);
       if (i % 8 == 0 && i != 0) cout << ' '; // deixa um
            espao a cada byte
   }
}

void printBinary8(uint8_t value) {
   for (int i = 7; i >= 0; --i) {
       cout << ((value >> i) & 1u);
       if (i == 4) cout << '\''; // separa os dois nibbles (4
            bits)
   }
}
```

3 03-matematica

3.1 crivo-eratostenes

```
// Tabua para realizar consultas se um numero e primo ou nao
vector<int> primos_ate_n(int N)
{
    vector<int> marcacao(N, 1); // 1 = possivel primo, 0 =
         com certeza nao primo
    vector<int> primos;
    for (int x = 2; x < N; x++)
        if (marcacao[x] == 1)
        {
            primos.push_back(x);
            for (int m = x + x; m < N; m += x)
            {
                marcacao[m] = 0; // nao e primo
            }
        }
    return primos;
}
```

4 04-busca-binaria

4.1 busca-binaria-em-vectors

```
vector<int> v;
v.push_back(1);
v.push_back(3);
v.push_back(5);
v.push_back(7);
v.push_back(9);
// procure pelo primeiro elemento maior que 3 nesse vetor
auto u = upper_bound(v.begin(),v.end(),3);
// para nao termos que escrever
// vector<int>::iterator, vamos usar auto

//*u = 5(o primeiro elemento maior que 3 e 5)

u = lower_bound(v.begin(),v.end(),3);

//*u = 3(o primeiro elemento maior ou igual a 3 e 3).
```

4.2 busca-binaria-maximo

```
// Valor maximo (Maior valor que torna check verdadeiro)

int l = a;
int r = b;
while (r > l + 1)
{
    int mid = (l + r) / 2;
    if (check(mid))
    {           // mid e valido
        l = mid; // como queremos maximizar a resposta, e mid
             e uma resposta valida
                // descartamos tudo a esquerda de mid (mas
                     nao mid)
    }
    else
    {
        r = mid - 1; // Se mid nao e valido, descartamos ele
             e tudo acima.
    }
}
int ans = r;
if (check(l))
{
    ans = l;
}
```

4.3 busca-binaria-minimo

```
// Valor minimo (Menor valor que torna check verdadeiro)

int l = a; // sei que a resposta nao e menos que a
int r = b; // sei que a resposta nao e mais que b (as vezes
     esse chute tem que ser bom, para evitar overflow)

while (r > l + 1)
{ // repita enquanto o intervalo tiver tamanho > 2
    int mid = (l + r) / 2;
    if (check(mid))
    {           // mid e valido
        r = mid; // como queremos minimizar a resposta, e mid
             e uma resposta valida
                // descartamos tudo a direita de mid (mas nao
                     mid)
    }
    else
    {
        l = mid + 1; // Se mid nao e valido, descartamos ele
             e tudo abaixo.
```

```
    }
}
// Ao final desse laco, a resposta pode estar em l ou r.
// Queremos minimizar a resposta, entao se l for valido,
// ficaremos com l, e caso contrario, com r
int ans = r;
if (check(l))
{
    ans = l;
}
```

5 05-Estruturas-de-Dados

5.1 fila

```
queue<int> fila;

fila.push(1);
fila.push(2);

int t = fila.size();
int proximo = fila.front();
fila.pop();
while (!fila.empty())
{
    int proximo = fila.front();
    fila.pop();
}
```

5.2 min-max

```
int menor = min(v[5], v[3]); // menor = 1
int maior = max(v[5], v[3]); // maior = 412
```

5.3 pilha

```
stack<int> pilha;

pilha.push(1);
pilha.push(2);
pilha.push(3);

int tamanho = pilha.size();
int topo = pilha.top();
pilha.pop();

while (!pilha.empty())
{
    int elemento = pilha.top();
    pilha.pop();
}
```

5.4 reverse

```
reverse(v.begin(), v.end()); // Inverte a ordem dos
    elementos de v
```

5.5 swap

```
swap(v[5], v[3]); // troca os valores de v[5] e v[3]
```

5.6 to-string

```
int x = 1320;
string a = to_string(x); // a = "1320"
```

5.7 vector

```
vector<int> v; // declaracao de um vector;
v.push_back(9);
v.push_back(2);
int n = v.size();
vector<int>::iterator i = v.begin(); //i aponta para v[0];
i++; // i aponta para v[1];
//
// adicionar cin em vetores:
for(int i = 0; i < v.size(); i++){
 cin >> v[i]; //onde V  o seu vetor e i  a  posio
}

sort(v.begin(),v.end()); // Ordenacao de v
// se quiser ordena-lo de tras para frente:
sort(v.end(), v.begin());
```