AtCoder Beginner Contest 098

Problema D: *Xor Sum 2*

Prof. Edson Alves – UnB/FGA

Problema

There is an integer sequence A of length N.

Find the number of the pairs of integers l and r $(1 \leq l \leq r \leq N)$ that satisfy the following condition:

- $A_l \ xor \ A_{l+1} \ xor \ \ldots \ xor \ A_r = A_l + A_{l+1} + \ldots + A_r$

Here, xor denotes the bitwise exclusive OR.

Entrada e saída

Constraints

- $1 \leq N \leq 2 \times 10^5$
- $0 \leq A_i < 2^{20}$
- All values in input are integers.

Input

Input is given from Standard Input in the following format:

$$N$$
$$A_1 \ A_2 \ \ldots \ A_N$$

Output

Print the number of the pairs of integers l and r $(1 \leq l \leq r \leq N)$ that satisfy the condition.

Exemplo de entradas e saídas

Sample Input

```
4
2 5 4 6

9
0 0 0 0 0 0 0 0 0

19
885 8 1 128 83 32 256 206 639 16 4
128 689 32 8 64 885 969 1
```

Sample Output

```
5

45

37
```

Solução com complexidade $O(N)$

- Dados dois inteiros x e y, a igualdade $x\ xor\ y = x + y$ só é verdadeira se x e y não tiverem nenhum *bit* em comum
- Assim, um intervalo de índices $[l, r]$ atenderá o critério do problema somente se, para cada par de elementos A_i, A_j, com $i, j \in [l, r]$ e $i \neq j$, vale que $A_i\ and\ A_j = 0$ (isto é, A_i e A_j não tem *bits* em comum)
- Como existem $O(N^2)$ pares de índices (l, r), com $1 \leq l \leq r \leq N$, uma solução que verifique cada um destes pares teria um veredito TLE
- Porém, é possível resolver o problema em $O(N)$, por meio da técnica dos dois ponteiros
- Para isso, inicie o ponteiro L no primeiro elemento do vetor

Solução com complexidade $O(N)$

- Inicie com zero uma variável x, que conterá a disjunção (ou) dos elementos do intervalo $[L, R)$
- Para cada valor de L, inclua A_L em x
- O ponteiro R deve ser o maior dentre L e o próprio R
- Enquanto R apontar para um elemento do vetor e A_R não tiver *bits* em comum com x, inclua A_R em x e incremente R
- Ao final do processo, qualquer intervalo $[L, r]$, com $r \in [L, R)$, será um intervalo válido do problema
- Assim, a resposta deve ser acrescida em $R - L$
- Após a atualização da resposta, remova o elemento A_L de x e incremente L (observe que $R - L > 0$ em qualquer iteração)

Solução com complexidade $O(N)$

```cpp
#include <bits/stdc++.h>

using namespace std;
using ll = long long;

ll solve(int N, const vector<int>& xs)
{
    ll ans = 0;
    auto L = 0, R = 0, x = 0;

    while (L < N)
    {
        R = max(L, R);

        while (R < N and (x & xs[R]) == 0) {
            x |= xs[R];
            ++R;
        }

        ans += (R - L);
```

Solução com complexidade $O(N)$

```
22          x &= ~xs[L];
23          ++L;
24      }
25
26      return ans;
27  }
28
29  int main()
30  {
31      ios::sync_with_stdio(false);
32
33      int N;
34      cin >> N;
35
36      vector<int> xs(N);
37
38      for (int i = 0; i < N; ++i)
39          cin >> xs[i];
40
41      auto ans = solve(N, xs);
```

Solução com complexidade $O(N)$

```
43      cout << ans << endl;
44
45      return 0;
46  }
```