Beecrowd 1897

Jogo Esperto

Prof. Edson Alves – UnB/FGA

Problema

Enquanto Bino descansava, inventou um jogo esperto. Dado um número N e um número M, Bino quer saber qual a quantidade mínima de operações para converter N em M.

Existe seis operações permitidas.

- Operação 1: $N = N * 2$
- Operação 2: $N = N * 3$
- Operação 3: $N = N/2$
- Operação 4: $N = N/3$
- Operação 5: $N = N + 7$
- Operação 6: $N = N - 7$

Entrada e saída

Entrada

A entrada contém dois inteiros N $(0 \leq N \leq 10000)$ e M $(0 \leq M \leq 10000)$.

Saída

A saída é o número mínimo de operações para converter N em M.

Exemplo de entradas e saídas

Exemplo de Entrada	Exemplo de Saída
10 15	2
45 15	1
84 63	3

Solução com complexidade $O(|N - M|)$

- Observe que a sequência de operações 1, 2, 5, 4, 3 efetivamente acrescenta uma unidade em N:
$$N \to 2N \to 6N \to 6N + 7 \to 2N + 2 \to N + 1$$
- De forma semelhante, a sequência 1, 2, 6, 4, 3 subtrai uma unidade de N
- Assim, em no máximo $5|N - M|$ operações é possível alcançar M e o problema sempre terá solução
- Para determinar o mínimo de soluções com complexidade $O(|N - M|)$, basta armazenar N em uma fila e inserir seu valor em um conjunto, para que ele não seja processado mais do que uma vez
- A cada etapa, se o próximo elemento da fila x não for igual a M, todos os resultados das operações em x que não tiverem sido inseridos no conjunto ainda devem entrar na fila e no conjunto
- Na fila deve ser armazenado, além do valor x, quantas operações foram necessárias para encontrá-lo

Solução com complexidade $O(|N - M|)$

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

using namespace std;
using ii = pair<int, int>;

int solve(int N, int M)
{
    queue<ii> ns;
    set<int> found;

    ns.push(make_pair(N, 0));
    found.insert(N);

    while (not ns.empty())
    {
        auto [n, ops] = ns.front();
        ns.pop();

        if (n == M)
            return ops;
```

Solução com complexidade $O(|N - M|)$

```
22          vector<int> xs { n * 2, n * 3, n / 2, n / 3, n + 7, n - 7 };
23
24          for (auto x : xs)
25          {
26              if (found.count(x) == 0)
27              {
28                  ns.push(ii(x, ops + 1));
29                  found.insert(x);
30              }
31          }
32      }
33
34      return -1;
35  }
```

Solução com complexidade $O(|N - M|)$

```
37 int main()
38 {
39     int N, M;
40     cin >> N >> M;
41
42     auto ans = solve(N, M);
43
44     cout << ans << endl;
45
46     return 0;
47 }
```