I Maratona de Programação do IFB
Número de Série Tabajara

Daniel Saad Nogueira Nunes

O problema **Número de Série Tabajara** consistia em, dado um número n, retornar o seu sucessor na ordem lexicográfica (ordem do dicionário), ou -1, caso não houvesse sucessor. Por exemplo:

Tabela 1: Entradas e saídas esperadas.

Número	Retorno
123	213
47651	51467
1135	1153
222	-1
321	-1

Como n poderia ter 100 dígitos, ele deveria ser representado via uma *string* $S[0..n-1]$. A estratégia para resolver o problema é a seguinte:

1. Varrer S da direita para a esquerda enquanto a sequência de dígitos obtida for crescente.
2. Se a varredura passou por toda a *string*, esta já é a última permutação na ordem lexicográfica. Retorno -1.
3. Caso contrário, seja i a primeira posição tal que $S[i] < S[i+1]$.
4. Ache o $j > i$ mais a direita possível tal que $S[j] > S[i]$.
5. Troque $S[i]$ com $S[j]$.
6. Reverta $S[i+1..n-1]$.

Vamos explicar o por que da estratégia adotada funcionar. Ao inspecionar S da direita para a esquerda e verificando que **todos** os dígitos estão em uma sequência **monotonicamente** crescente, temos que este é o maior número possível de ser obtido, uma vez que a troca de quaisquer dois dígitos faria com que o número de série obtido fosse menor ou igual ao número de série original, portanto o retorno -1 está justificado.

Se não é o caso que, inspecionando S da direita para a esquerda, obtemos uma sequência monotonicamente crescente então é possível obter o próximo sucessor na ordem lexicográfica. Seja i o primeiro índice da direita para esquerda tal que $S[i] < S[i+1]$. Se escolhermos um $S[j]$, $j > i$, para ser trocado com $S[i]$, está claro que um número de série maior poderá ser obtido. Por exemplo, tome o número:

47651

O primeiro i que atende a condição mencionada é $i = 0$. Trocando $S[i]$ com alguns dos dígitos a direita, podemos obter:

74651 67451 57641 17654

Qual é o dígito que devemos escolher para trocar com $S[i]$? Aquele que for o menor possível, mas que ainda seja maior do que $S[i]$. No caso do exemplo anterior, o candidato seria:

57641

Mas ainda não terminamos. Obviamente o número de série é maior que o original, mas ele não é o próximo da sequência lexicográfica. Mas note que a sequência $S[i+1..n-1]$ é monotonicamente crescente! Se invertermos a mesma, obteremos a sequência monotonicamente decrescente:

51467

Por que isto funciona? Bom, trocamos o dígito imediatamente maior da sequência crescente no lugar de $S[i]$, então é óbvio que o número gerado é maior que o original, mas como o sufixo restante continua em ordem monotonicamente crescente, obtemos o menor sufixo possível ao invertê-lo. Assim, temos que:

1. O dígito trocado com $S[i]$ é imediatamente maior que ele.
2. O restante do sufixo, contendo dígitos menos significativos, ao ser invertido, gera o menor valor possível.

Solução

De acordo com a discussão anterior, obtemos o Algoritmo 1.

Complexidade

A solução tem tempo de pior caso $O(n)$. Basta observar os laços das linhas 2 e 7.

Algorithm 1: Número de série tabajara

Input: S

Output: S' que sucede S na ordem lexicográfica, ou -1 caso seja impossível.

```
1  i ← n − 2
2  while( (i >= 0) ∧ (S[i] > S[i + 1]) )
3  |  i − −
4  if( i = −1 )
5  |  return −1
6  j ← i + 1
7  while( (S[j] > S[i]) ∧ (j < n) )
8  |  j + +;
9  SWAP(S[j − 1], S[i])
10 REVERSE(S[i + 1..n − 1])
```

Solução Alternativa

Agora que já sabemos o que está por trás da solução, podemos usar uma coisa já pronta. O `C++` possui a função `next_permutation` que nos dá a próxima *string* na ordem lexicográfica, resultando no Algoritmo 2.

Algorithm 2: numero-de-serie-tabajara.cpp

```cpp
#include <iostream>
#include <algorithm>

using namespace std;

string solve(string s){
    auto b = next_permutation(s.begin(),s.end());
    if(!b){
        return "-1";
    }
    return s;
}

int main(){
    std::ios::sync_with_stdio(false);
    string s;
    while(cin >> s && s!="-1"){
        cout << solve(s) << "\n";
    }
    return 0;
}
```