Pilhas e Filas

Filas: problemas resolvidos

Prof. Edson Alves - UnB/FGA

2020

Sumário

OJ 10935 – Throwing Cards Away I

Problema

Given is an ordered deck of n cards numbered 1 to n with card 1 at the top and card n at the bottom. The following operation is performed as long as there are at least two cards in the deck:

Throw away the top card and move the card that is now on the top of the deck to the bottom of the deck.

Your task is to find the sequence of discarded cards and the last, remaining card.

Entrada e saída

Input

Each line of input (except the last) contains a number $n \leq 50$. The last line contains '`0`' and this line should not be processed.

Output

For each number from the input produce two lines of output. The first line presents the sequence of discarded cards, the second line reports the last remaining card. No line will have leading or trailing spaces. See the sample for the expected format.

Exemplo de entradas e saídas

Sample Input

```
7
19
10
6
0
```

Sample Output

```
Discarded cards: 1, 3, 5, 7, 4, 2
Remaining card: 6
Discarded cards: 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 4, 8, 12, 16, 2, 10, 18, 14
Remaining card: 6
Discarded cards: 1, 3, 5, 7, 9, 2, 6, 10, 8
Remaining card: 4
Discarded cards: 1, 3, 5, 2, 6
Remaining card: 4
```

Solução com complexidade $O(N)$

- O processo descrito no problema pode ser simulado através do uso de uma fila
- Os elementos removidos podem ser armazenados ou em uma outra fila ou em um vetor
- Como, a cada ciclo, o tamanho da fila diminui em uma unidade, o algoritmo tem complexidade $O(N)$
- A saída deve ser formatada com cuidado: no caso $n = 1$ não há elementos a serem removidos

Solução com complexidade $O(N)$

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

using namespace std;

vector<int> solve(int n)
{
    queue<int> q;

    for (int i = 1; i <= n; ++i)
        q.push(i);

    vector<int> ans;

    while (q.size() > 1)
    {
        auto x = q.front();
        q.pop();

        ans.push_back(x);

```

Solução com complexidade $O(N)$

```
        auto y = q.front();
        q.pop();

        q.push(y);
    }

    ans.push_back(q.front());

    return ans;
}

int main()
{
    int n;

    while (cin >> n, n)
    {
        auto order = solve(n);

        cout << "Discarded cards:";

```

Solução com complexidade $O(N)$

```
        for (int i = 0; i < n - 1; ++i)
            cout << (i ? ", " : " ") << order[i];

        cout << "\nRemaining card: " << order.back() << '\n';
    }

    return 0;
}
```

LeetCode 933 - Number of Recent Calls

Problema

Write a class RecentCounter to count recent requests.

It has only one method: `ping(int t)`, where t represents some time in milliseconds.

Return the number of pings that have been made from 3000 milliseconds ago until now.

Any ping with time in $[t - 3000, t]$ will count, including the current ping.

It is guaranteed that every call to ping uses a strictly larger value of t than before.

Entrada e saída

Example 1:

Input: inputs = ["RecentCounter","ping","ping","ping","ping"],
inputs = [[],[1],[100],[3001],[3002]]
Output: [null,1,2,3,3]

Note:

1. Each test case will have at most 10000 calls to ping.
2. Each test case will call ping with strictly increasing values of t.
3. Each call to ping will have $1 \leq t \leq 10^9$.

Solução com complexidade $O(N)$

- Seja N o número de chamadas do método `ping()`
- Para determinar quantas chamadas se encontram no intervalo $[t, t + 3000]$, é preciso manter uma fila com os valores de t já invocados
- O primeiro elemento da fila conterá o valor t_a da chamada mais antiga
- O último elemento da fila será o valor t_b da chamada atual
- Enquanto $t_b - t_a > 3000$, o primeiro elemento da fila deve ser descartado
- O retorno da chamada deve ser o tamanho da fila
- Como cada elemento será inserido ou removido uma única vez, a complexidade desta solução é $O(N)$

Solução AC com complexidade $O(N)$

```cpp
class RecentCounter {
public:
    queue<int> q;

    RecentCounter() {

    }

    int ping(int t) {

        q.push(t);

        while (not q.empty() and t - q.front() > 3000)
            q.pop();

        return q.size();
    }
};
```

Referências

1. OJ 10935 – Throwing cards away I
2. LeetCode 933 - Number of Recent Calls