Pilhas e Filas

Filas Monótonas

Prof. Edson Alves – UnB/FCTE

Sumário

Definição

Definição de fila monótona

Fila monótona

Seja F uma fila de elementos do tipo T. A fila F é dita **monótona** se, quando extraídos todos os elementos de F, eles formam uma sequência $x_1, x_2, \ldots, x_N$, onde x_i é o elemento obtido na i-ésima extração, tais que a função $F : \mathbb{N} \to T$, com $f(i) = x_i$, é monótona.

A fila F será **não-decrescente** se f for **não-decrescente**; caso contrário, F será não-decrescente.

Inserção em filas monótonas

- Em filas monótonas é necessário manter a invariante da monotonicidade a cada inserção
- Seja F uma fila não-decrescente e x um elemento a ser inserido em F
- Se F estiver vazia, basta inserir x em F: o invariante estará preservado
- Se F não estiver vazia, o mesmo acontece se $x \leq y$, onde y é o último de F
- Contudo, se $x > y$, é preciso remover y antes da inserção de x
- Após a remoção de y, é preciso confrontar x com o novo elemento que ocupará a última posição até que x possa ser inserido na última posição de F

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline\end{array}$$

$$F =$$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$\uparrow$

$F =$

1

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$\uparrow$

$F =$

1

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline \end{array}$$

$$F = \begin{array}{|c|c|} \hline 1 & 4 \\ \hline \end{array}$$

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline \end{array}$$

$$F = \begin{array}{|c|c|}\hline 1 & 4 \\ \hline \end{array}$$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \boxed{1}\boxed{4}\boxed{3}\boxed{4}\boxed{2}\boxed{1}\boxed{3}$$

$$F = \boxed{1}\boxed{3}$$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1	3

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|} \hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline \end{array}$$

$$\uparrow$$

$$F = \begin{array}{|c|c|c|} \hline 1 & 3 & 4 \\ \hline \end{array}$$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

↑

$F =$

1	3	4

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1	3

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1	2

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline \end{array}$$

$$\uparrow$$

$$F = \begin{array}{|c|c|}\hline 1 & \color{red}{2} \\ \hline \end{array}$$

Exemplo de inserções em uma pilha não-decrescente

$$a_n = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 4 & 3 & 4 & 2 & 1 & 3 \\ \hline \end{array}$$

$$F = \begin{array}{|c|}\hline 1 \\ \hline \end{array}$$

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

1	4	3	4	2	1	3

$F =$

1	1

Implementação

Implementação de uma fila não-decrescente em C++

```cpp
template<typename T>
class MonoQueue {
public:
    void push(const T& x)
    {
        while (not st.empty() and st.back() > x)
            st.pop_back();

        st.emplace_back(x);
    }

    void pop() { st.pop_front(); };
    auto back() const { return st.back(); }
    auto front() const { return st.front(); }
    bool empty() const { return st.empty(); }

private:
    deque<T> st;
};
```

Implementação de uma fila não-decrescente em C++

```cpp
template<typename T>
ostream& operator<<(ostream& os, const MonoQueue<T>& ms)
{
    auto temp(ms);

    while (not temp.empty())
    {
        cout << temp.front() << ' ';
        temp.pop();
    }

    return os;
}
```

Implementação de uma fila não-decrescente em C++

```cpp
int main()
{
    vector<int> as { 1, 4, 3, 4, 2, 1, 3 };
    MonoQueue<int> mq;

    for (auto& a : as)
    {
        mq.push(a);
        cout << mq << '\n';
    }

    return 0;
}
```

Aplicações de Filas Monótonas

Menores elementos em uma janela móvel

Definição

Seja $a_1, a_2, \ldots, a_N$ uma sequência de elementos e k um inteiro positivo. Os **menores elementos em uma janela móvel de tamanho** k correspondem à sequência $b_1, b_2, \ldots, b_M$ tal que $M = N - K + 1$ e

$$b_i = \min(a_i, a_{i+1}, \ldots, a_{i+k-1})$$

Menores elementos em uma janela móvel em $O(1)$

- Os menores elementos em uma janela móvel de tamanho k de uma sequência $a_1, a_2, \ldots, a_N$ podem ser determinados em $O(1)$ por meio de uma fila não-decrescente F
- Inicialmente, insira índices os primeiros k elementos da sequência em F (usando os valores dos elementos para manter o invariante)
- O elemento que ocupar a primeiro posição em F será o índice do elemento mínimo de dentre os k primeiros
- Inicialize dois marcadores L e R com os valores 1 e $K+1$, respectivamente
- Enquanto $R \leq N$:
    1. Remova o primeiro elemento da fila, se ele for igual a L
    2. Insira R em F
    3. O elemento da primeira posição de R será o índice do elemento mínimo de $a_{L+1}, a_{L+2}, \ldots, a_R$
    4. Incremente ambos marcadores

Menores elementos em uma janela móvel em C++

```cpp
auto mesw(int n, int k, const vector<int>& xs)
{
    deque<int> q;

    for (int i = 0; i < k; ++i)
    {
        while (not q.empty() and xs[q.back()] >= xs[i])
            q.pop_back();

        q.emplace_back(i);
    }

    vector<int> ms { xs[q.front()] };

    for (int L = 0, R = k; R < n; ++L, ++R)
    {
        if (q.front() == L)
            q.pop_front();
```

Menores elementos em uma janela móvel em C++

```cpp
        while (not q.empty() and xs[q.back()] >= xs[R])
            q.pop_back();

        q.emplace_back(R);
        ms.emplace_back(xs[q.front()]);
    }

    return ms;
}
```

Menores elementos em uma janela móvel bidimensional

Definição

Seja $A_{N\times M}$ uma matriz. A matriz B contém todos os menores elementos de A para todas as janelas de tamanho $H \times W$ se B tem dimensões $(N-H+1)\times(M-W+1)$ e

$$b_{ij} = \min \begin{bmatrix} a_{ij} & a_{i(j+1)} & \cdots & a_{i(j+W-1)} \\ a_{(i+1)j} & a_{(i+1)(j+1)} & \cdots & a_{(i+1)(j+W-1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{(i+H-1)j} & a_{(i+H-1)(j+1)} & \cdots & a_{(i+H-1)(j+W-1)} \end{bmatrix}$$

Menores elementos em uma janela móvel bidimensional em $O(NM)$

- Para computar a matriz B dos menores elementos de A em uma janela móvel bidimensional de tamanho $H \times B$, inicialmente aplique o algoritmo unidimensional em cada linha de A, produzindo uma matriz intermediária C com N linhas e $M - W + 1$ colunas
- Em seguida, aplique o algoritmo unidimensional nas colunas de C para obter a matriz B

$$\begin{bmatrix} a_{11} & \dots & a_{1M} \\ a_{21} & \dots & a_{2M} \\ \vdots & \ddots & \vdots \\ a_{N1} & \dots & a_{NM} \end{bmatrix} \Rightarrow \begin{bmatrix} c_{11} & \dots & c_{1(M-W+1)} \\ c_{21} & \dots & c_{2(M-W+1)} \\ \vdots & \ddots & \vdots \\ c_{N1} & \dots & c_{N(M-W+1)} \end{bmatrix} \Rightarrow \begin{bmatrix} b_{11} & \dots & b_{1(M-W+1)} \\ b_{21} & \dots & b_{2(M-W+1)} \\ \vdots & \ddots & \vdots \\ b_{(N-H+1)1} & \dots & b_{(N-H+1)(M-W+1)} \end{bmatrix}$$

Menores elementos em uma janela móvel bidimensional em C++

```cpp
auto mesw_2D(int NA, int MA, int H, int W, const vector<vector<int>>& A)
{
    // Aplica o mesw 1D em todas as linhas de A
    auto NC = NA, MC = MA - W + 1;
    vector<vector<int>> C(NC, vector<int>(MC));

    for (int row = 0; row < NA; ++row)
    {
        deque<int> q;

        for (int col = 0; col < W; ++col)
        {
            while (not q.empty() and A[row][q.back()] >= A[row][col])
                q.pop_back();

            q.emplace_back(col);
        }

        C[row][0] = A[row][q.front()];
```

Menores elementos em uma janela móvel bidimensional em C++

```cpp
        for (int L = 0, R = W; R < MA; ++L, ++R)
        {
            if (q.front() == L)
                q.pop_front();

            while (not q.empty() and A[row][q.back()] >= A[row][R])
                q.pop_back();

            q.emplace_back(R);
            C[row][L + 1] = A[row][q.front()];
        }
    }

    // Aplica o mesw 1D em todas as colunas de C
    auto NB = (NC - H + 1), MB = MC;
    vector<vector<int>> B(NB, vector<int>(MB));
```

Menores elementos em uma janela móvel bidimensional em C++

```
    for (int col = 0; col < MC; ++col)
    {
        deque<int> q;

        for (int row = 0; row < H; ++row)
        {
            while (not q.empty() and C[q.back()][col] >= C[row][col])
                q.pop_back();

            q.emplace_back(row);
        }

        B[0][col] = C[q.front()][col];

        for (int L = 0, R = H; R < NC; ++L, ++R)
        {
            if (q.front() == L)
                q.pop_front();
```

Menores elementos em uma janela móvel bidimensional em C++

```
                q.emplace_back(R);
                B[L + 1][col] = C[q.front()][col];
            }
        }

    return B;
}
```

Referências

1. **Ali Ibrahim**. Monotonic Queue, acesso em 15/08/2025.
2. **bicsi**. Minima/maxima over all fixed-size arrays (multi-dimensional), acesso em 18/08/2025.
3. **Li Yin**. Monotonic Queue Explained with LeetCode Problems, acesso em 15/08/2025.
4. **YouKn0wwho Academy**. 125. Monotonic Queue, aceso em 15/08/2025.
5. **YouKn0wwho Academy**. 126. Monotonic Queue 2D, aceso em 15/08/2025.