#### Pilhas e Filas

Filas Monótonas

Prof. Edson Alves – UnB/FCTE

#### Sumário

- 1. Definição
- 2. Implementação
- 3. Aplicações de Filas Monótonas

# 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.

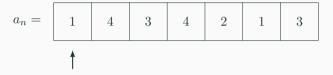
A fila F será **não-decrescente** se f for **não-decrescente**; caso contrário, F será não-decrescente.

#### Inserção em filas monótonas

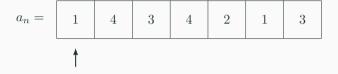
- Em filas monótonas é necessário manter a invariante da monotonicidade a cada inserção
- ullet Seja F uma fila não-decrescente e x um elemento a ser inserido em F
- ullet Se F estiver vazia, basta inserir x em F: o invariante estará preservado
- Se F não estiver vazia, o mesmo acontece se  $x \leq y$ , onde y é o último de F
- ullet Contudo, se x>y, é preciso remover y antes da inserção de x
- ullet 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

$a_n =$	1	4	3	4	2	1	3
---------	---	---	---	---	---	---	---

F =

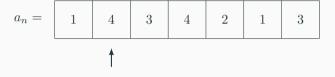


F =

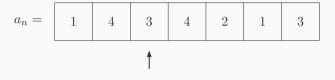


$$F = 1$$

$$F = \begin{bmatrix} 1 \end{bmatrix}$$



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



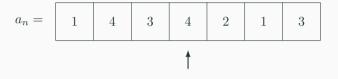
$$F = \begin{bmatrix} 1 & 4 \end{bmatrix}$$

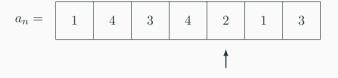
$$F = \begin{bmatrix} 1 \end{bmatrix}$$



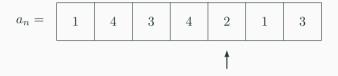
$$F = \begin{bmatrix} 1 & 3 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 & 3 \end{bmatrix}$$

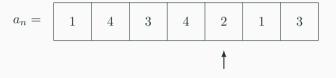




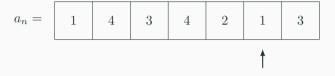
$$F = \left| \begin{array}{c|c} 1 & 3 & 4 \end{array} \right|$$



$$F = \begin{bmatrix} 1 \end{bmatrix}$$

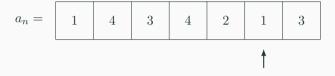


$$F = \begin{bmatrix} 1 & 2 \end{bmatrix}$$



$$F = \begin{bmatrix} 1 & 2 \end{bmatrix}$$

$$F = \begin{bmatrix} 1 \end{bmatrix}$$



$$F = \begin{bmatrix} 1 & 1 \end{bmatrix}$$

Implementação

#### Implementação de uma fila não-decrescente em C++

```
5 template<typename T>
6 class MonoQueue {
public:
      void push(const T& x)
9
          while (not st.empty() and st.back() > x)
10
              st.pop_back();
          st.emplace_back(x);
14
      void pop() { st.pop_front(); };
16
      auto back() const { return st.back(); }
      auto front() const { return st.front(): }
1.8
      bool empty() const { return st.empty(); }
19
20
21 private:
     deque<T> st;
23 };
```

#### Implementação de uma fila não-decrescente em C++

```
25 template<typename T>
26 ostream& operator<<(ostream& os, const MonoOueue<T>& ms)
27 {
      auto temp(ms);
28
29
      while (not temp.empty())
30
31
          cout << temp.front() << ' ';</pre>
32
          temp.pop();
34
35
      return os;
36
37 }
```

#### Implementação de uma fila não-decrescente em C++

```
39 int main()
40 {
      vector<int> as { 1, 4, 3, 4, 2, 1, 3 };
41
      MonoQueue<int> mq;
42
43
      for (auto& a : as)
44
45
          mq.push(a);
46
          cout << mg << '\n':
47
48
49
      return 0;
50
51 }
```

# \_\_\_\_

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

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

### 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)
- ullet 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  $R \ {\sf em} \ F$
  - 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++

```
5 auto mesw(int n, int k, const vector<int>& xs)
6 {
      deque<int> q:
      for (int i = 0: i < k: ++i)
9
10
          while (not q.empty() and xs[q.back()] >= xs[i])
              g.pop_back();
12
          g.emplace_back(i);
14
15
16
      vector<int> ms { xs[q.front()] };
18
      for (int L = 0, R = k; R < n; ++L, ++R)
19
20
          if (q.front() == L)
21
              q.pop_front();
```

#### Menores elementos em uma janela móvel em C++

```
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;
```

#### 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

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

- ullet 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
- ullet Em seguida, aplique o algoritmo unidimensional nas colunas de C para obter a matriz B

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

```
16 auto mesw_2D(int NA, int MA, int H, int W, const vector<vector<int>>>& A)
17 {
      // Aplica o mesw 1D em todas as linhas de A
18
      auto NC = NA. MC = MA - W + 1:
19
      vector<vector<int>>> C(NC, vector<int>(MC));
20
      for (int row = \emptyset: row < NA: ++row)
          deque<int> q;
24
          for (int col = 0; col < W; ++col)</pre>
26
              while (not q.empty() and A[row][q.back()] >= A[row][col])
28
                   g.pop_back():
29
30
              q.emplace_back(col):
31
          C[row][0] = A[row][q.front()];
```

```
for (int L = 0, R = W; R < MA; ++L, ++R)
36
              if (q.front() == L)
38
                  q.pop_front();
39
40
              while (not q.empty() and A[row][q.back()] >= A[row][R])
41
                  q.pop_back();
42
43
              g.emplace_back(R);
44
              C[row][L + 1] = A[row][q.front()];
45
46
47
48
      // Aplica o mesw 1D em todas as colunas de C
49
      auto NB = (NC - H + 1), MB = MC:
      vector<vector<int>>> B(NB, vector<int>(MB));
```

```
for (int col = 0; col < MC; ++col)</pre>
53
54
          deque<int> q;
55
56
           for (int row = \emptyset: row < H: ++row)
57
58
               while (not q.empty() and C[q.back()][col] >= C[row][col])
59
                   q.pop_back();
60
61
               q.emplace_back(row);
62
63
64
          B[0][col] = C[q.front()][col];
65
66
           for (int L = 0, R = H; R < NC; ++L, ++R)
67
68
               if (q.front() == L)
69
                   q.pop_front();
70
```

#### 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.