

Pilhas e Filas

Pilhas Monótonas

Prof. Edson Alves – UnB/FCTE

1. Definição
2. Implementação
3. Aplicações de Pilhas Monótonas

Definição

Definição de monotonicidade

Função não-decrescente e função não-crescente

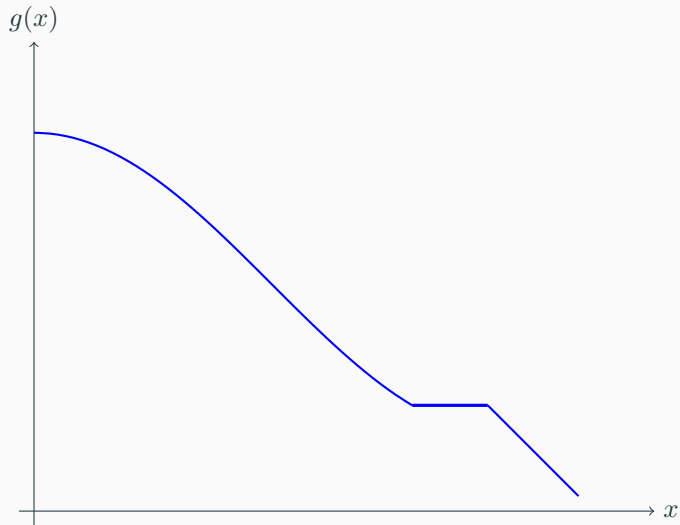
Seja $f : A \rightarrow B$ uma função. Dizemos que f é **não-decrescente** se para todo par $x, y \in A$, com $x \leq y$, temos que $f(x) \leq f(y)$.

Seja $g : A \rightarrow B$ uma função. Dizemos que g é **não-crescente** se para todo par $x, y \in A$, com $x \leq y$, temos que $g(x) \geq g(y)$.

Função monótona

Seja $h : A \rightarrow B$ uma função. Dizemos que h é **monótona** se h é não-decrescente ou não-crescente.

Exemplo de função não-decrescente



Definição de pilha monótona

Pilha monótona

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

A pilha P será **não-decrescente** se f for **não-crescente**; caso contrário, P será não-decrescente.

Inserção em pilhas monótonas

- Em pilhas monótonas é necessário manter a invariante da monotonicidade a cada inserção
- Seja P uma pilha não-decrescente e x um elemento a ser inserido em P
- Se P estiver vazia, basta inserir x em P : o invariante estará preservado
- Se P não estiver vazia, o mesmo acontece se $x \leq y$, onde y é o topo de P
- 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 topo até que x possa ser inserido em P

Exemplo de inserções em uma pilha não-decrescente

$a_n =$

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | 4 | 3 | 4 | 2 | 1 | 3 |
|---|---|---|---|---|---|---|

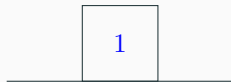
Exemplo de inserções em uma pilha não-decrescente



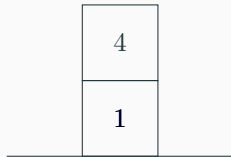
Exemplo de inserções em uma pilha não-decrescente



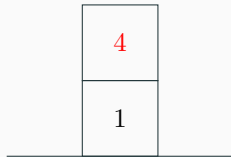
Exemplo de inserções em uma pilha não-decrescente



Exemplo de inserções em uma pilha não-decrescente



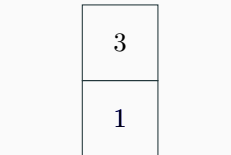
Exemplo de inserções em uma pilha não-decrescente



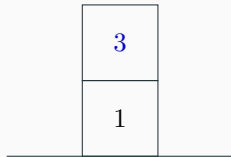
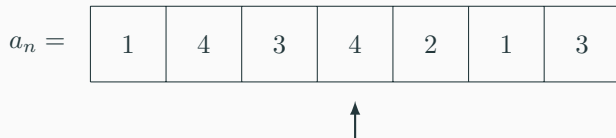
Exemplo de inserções em uma pilha não-decrescente



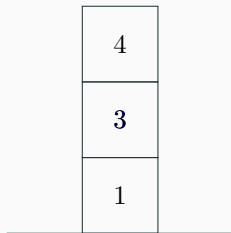
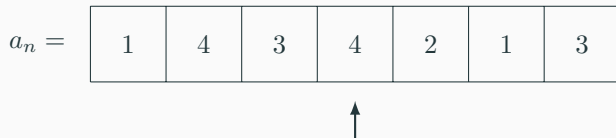
Exemplo de inserções em uma pilha não-decrescente



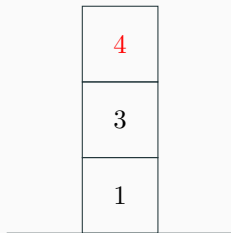
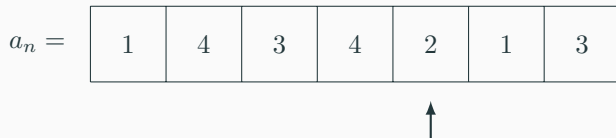
Exemplo de inserções em uma pilha não-decrescente



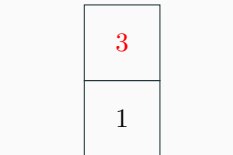
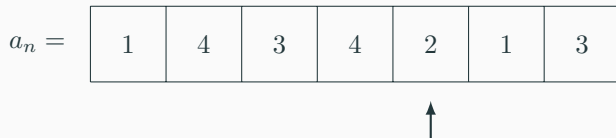
Exemplo de inserções em uma pilha não-decrescente



Exemplo de inserções em uma pilha não-decrescente



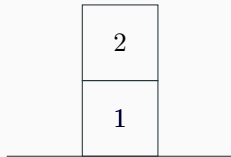
Exemplo de inserções em uma pilha não-decrescente



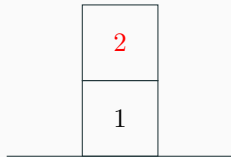
Exemplo de inserções em uma pilha não-decrescente



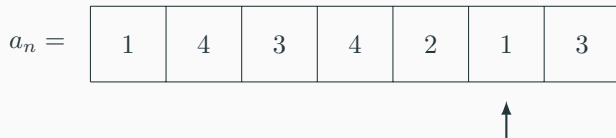
Exemplo de inserções em uma pilha não-decrescente



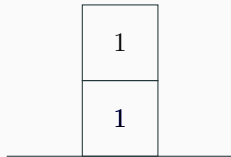
Exemplo de inserções em uma pilha não-decrescente



Exemplo de inserções em uma pilha não-decrescente



Exemplo de inserções em uma pilha não-decrescente



Implementação

Implementação de uma pilha não-decrescente em C++

```
5 template<typename T>
6 class MonoStack {
7 public:
8     void push(const T& x)
9     {
10         while (not st.empty() and st.top() > x)
11             st.pop();
12
13         st.emplace(x);
14     }
15
16     void pop() { st.pop(); };
17     auto top() const { return st.top(); }
18     bool empty() const { return st.empty(); }
19
20 private:
21     stack<T> st;
22 };
```

Implementação de uma pilha não-decrescente em C++

```
24 template<typename T>
25 ostream& operator<<(ostream& os, const MonoStack<T>& ms)
26 {
27     auto temp(ms);
28
29     while (not temp.empty())
30     {
31         cout << temp.top() << ' ';
32         temp.pop();
33     }
34
35     return os;
36 }
```

Implementação de uma pilha não-decrescente em C++

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

Aplicações de Pilhas Monótonas

Maior elemento à esquerda (ou a direita)

Definição

Seja a_1, a_2, \dots, a_N uma sequência de elementos. O **maior elemento à esquerda (à direita)** de a_i , se existir, é o elemento a_j tal que j é o menor (maior) índice tal que $j > i$ ($j < i$) e $a_j > a_i$.

Maior elemento à esquerda em $O(N)$

- É possível determinar o maior elemento à esquerda para todos os elementos de uma sequência a_1, a_2, \dots, a_N em $O(N^2)$ por meio de uma busca completa
- Para cada índice i , é preciso avaliar todos os elementos a_j , com $j = 1, 2, \dots, i - 1$
- Contudo, é possível determinar estes valores em $O(N)$ com uma modificação no método de inserção de uma pilha não-crescente
- A inserção em uma pilha não-crescente ocorre em duas etapas: manutenção do invariante e inserção do novo elemento
- Finalizada a manutenção do invariante, os elementos que restam na pilha são todos maiores do que a_i e o elemento do topo será o maior elemento à esquerda de a_i
- Em algumas implementações são mantidos os índices e não os valores da sequência propriamente ditos (ou pares com ambas informações)

Implementação do maior elemento à esquerda em C++

```
5 template<typename T>
6 class MonoStack {
7 public:
8     MonoStack(int start_idx = 0, inc = 1) : pos(start_idx), invalid(start_idx - 1) {}
9
10    int push(const T& x) {
11        while (not st.empty() and st.top().second <= x)
12            st.pop();
13
14        auto i = st.empty() ? invalid : st.top().first;
15        st.emplace(pos, x);
16        pos += inc;
17
18        return i;
19    }
20
21    void pop() { st.top(); };
22    auto top() const { return st.top(); }
23    bool empty() const { return st.empty(); }
```


Implementação do maior elemento à esquerda em C++

```
25 private:
26     stack<pair<int, T>> st;
27     int pos, invalid;
28 };
29
30 auto pge(const vector<int>& xs)
31 {
32     MonoStack<int> ms;
33     vector<int> ans;
34
35     for (auto x : xs)
36         ans.emplace_back(ms.push(x));
37
38     return ans;
39 }
```

Implementação do maior elemento à esquerda em C++

```
41 int main()
42 {
43     vector<int> as { 1, 4, 3, 4, 2, 1, 3 };
44     auto is = pge(as);
45
46     cout << "i:  ";
47     for (size_t i = 0; i < as.size(); ++i)
48         cout << setw(2) << setfill(' ') << i << (i + 1 == as.size() ? '\n' : ' ');
49
50     cout << "as:  ";
51     for (size_t i = 0; i < as.size(); ++i)
52         cout << setw(2) << setfill(' ') << as[i] << (i + 1 == as.size() ? '\n' : ' ');
53 }
```

Maior elemento à direita em $O(N)$

- É possível determinar o maior elemento à direita usando estratégia semelhante
- Basta inserir os elementos da direita para a esquerda em uma pilha não-crescente
- Caso sejam usados os índices, é preciso iniciar o marcador de posição no índice $N - 1$ e decrementá-lo a cada inserção
- Também é possível computar ambos elementos de uma só vez: o elemento mais à direita de a_i será o elemento a_j que remove a_i da pilha na manutenção do invariante
- Em uma implementação direta, sem uso de classes, é possível obter ambos vetores (*pge* – *previous greater element* – e *nge* – *next greater element*) com uma única chamada de função

1. **GeeksForGeeks.** [Introduction to Monotonic Stack - Data Structure and Algorithm Tutorials](#), acesso em 14/08/2025.