

Filas e Pilhas Encadeadas

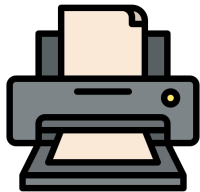
Prof. Marcelo Rosa

Algoritmos e Estrutura de Dados 2 (AE43CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

- 1 Filas
 - Fila encadeada
- 2 Pilhas
 - Pilha encadeada

- 1 Filas
 - Fila encadeada
- 2 Pilhas
 - Pilha encadeada

- Considere que uma impressora seja compartilhada em um laboratório
- Os alunos enviam documentos quase ao mesmo tempo

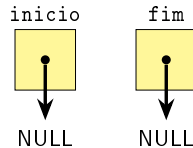


- Como gerenciar a lista de tarefas de impressão?

- Remove primeiro os objetos **inseridos há mais tempo**
- **FIFO** (First-In First-Out): primeiro a entrar é o primeiro a sair
- Operações:
 - **Enfileira** (enqueue): **adiciona** item no “**fim**”
 - **Desenfileira** (dequeue): **remove** item do “**início**”

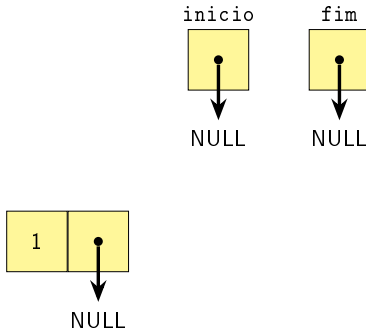
Fila: implementação com lista encadeada

Inserção de elementos



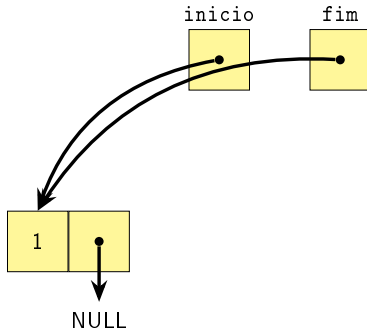
Fila: implementação com lista encadeada

Inserção de elementos



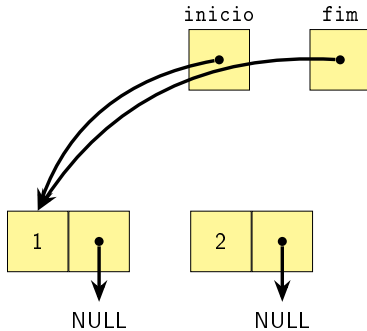
Fila: implementação com lista encadeada

Inserção de elementos



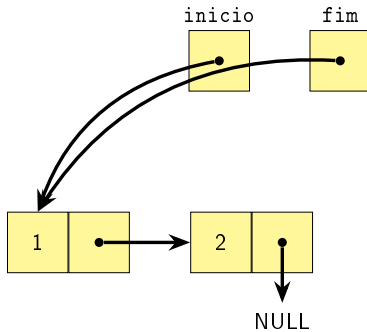
Fila: implementação com lista encadeada

Inserção de elementos



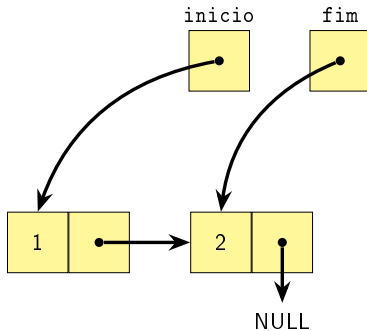
Fila: implementação com lista encadeada

Inserção de elementos



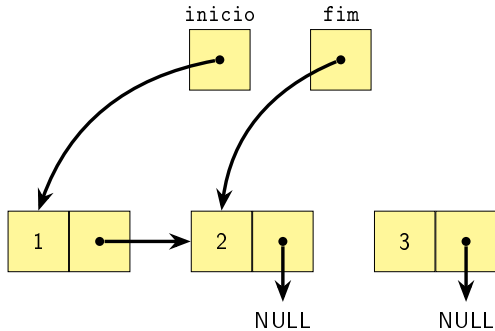
Fila: implementação com lista encadeada

Inserção de elementos



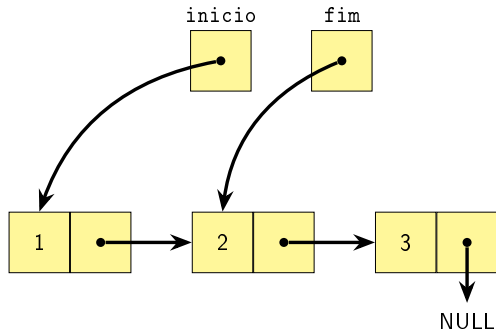
Fila: implementação com lista encadeada

Inserção de elementos



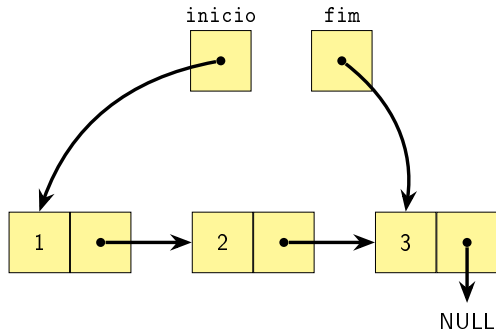
Fila: implementação com lista encadeada

Inserção de elementos



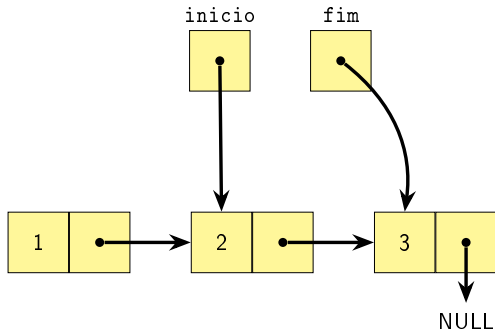
Fila: implementação com lista encadeada

Remoção de elementos



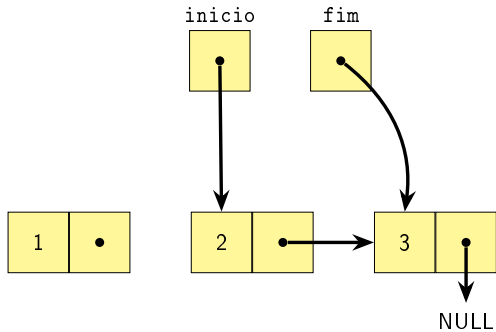
Fila: implementação com lista encadeada

Remoção de elementos



Fila: implementação com lista encadeada

Remoção de elementos



- Complexidade das operações enfileirar e desenfileirar:

| Operação | Complexidade |
|---------------|--------------|
| Enfileirar | $\Theta(1)$ |
| Desenfileirar | $\Theta(1)$ |

Fila encadeada

Implementando em C

```
1 typedef struct No No
2
3 struct No{
4     int item;
5     No *next;
6 };
```

Fila encadeada

Implementando em C

```
1 typedef struct No No
2
3 struct No{
4     int item;
5     No *next;
6 };
```

```
7 typedef struct FilaE FilaE
8
9 struct FilaE{
10     No *inicio;
11     No *fim;
12 };
```

Fila encadeada

Implementando em C

```
1 FilaE* criar_filaE(){
2     FilaE *f = (FilaE*) malloc(sizeof(FilaE));
3
4     f->inicio = NULL;
5     f->fim = NULL;
6
7     return f;
8 }
```

Fila encadeada

Implementando em C

```
1 void enfileirar(FilaE* f, int key){
2     No *novo; // Novo nó
3
4     // Caso a fila encadeada seja nula, alocar um espaço para essa estrutura
5     if (f == NULL)
6         f = criar_filaE();
7
8     // Criar novo nó
9     novo = criar_no(key);
10
11    // Caso a fila ainda esteja vazia, o primeiro e o último elemento são iguais
12    if (f->inicio == NULL)
13        f->inicio = f->fim = novo;
14    else{
15        // Caso a fila não esteja vazia, basta atualizar o final da estrutura
16        f->fim->next = novo;
17        f->fim = novo;
18    }
```

Fila encadeada

Implementando em C

```
1  int desenfileirar(FilaE* f){
2      No *primeiro; // Primeiro elemento da fila
3      int item = INT_MIN; // Valor que está no início da fila
4
5      if (!filaE_vazia(f)){
6          // obter o primeiro elemento da fila
7          primeiro = f->inicio;
8
9          f->inicio = primeiro->next;
10
11         if (f->inicio == NULL)
12             f->fim = NULL;
13
14         // Acessar o conteúdo do nó
15         item = primeiro->item;
16
17         // Liberar o nó desenfileirado
18         free(primeiro);
19     }
20
21     return item;
22 }
```

- 1 Filas
 - Fila encadeada
- 2 Pilhas
 - Pilha encadeada

- **Remove** primeiro o **objeto inserido a menos tempo**.
- **LIFO** (Last-IN First-Out): último a entrar é o primeiro a sair.

- Remove primeiro o **objeto inserido a menos tempo**.
- **LIFO** (Last-IN First-Out): último a entrar é o primeiro a sair.

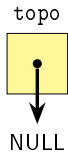
É como uma pilha de pratos:

- **Empilha** os pratos sobre os quais já estão na pilha.
- **Desempilha** o prato de cima (em geral).



Pilha: implementação com lista encadeada

Inserção de elementos



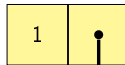
Pilha: implementação com lista encadeada

Inserção de elementos

topo



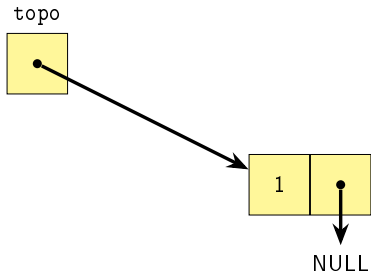
NULL



NULL

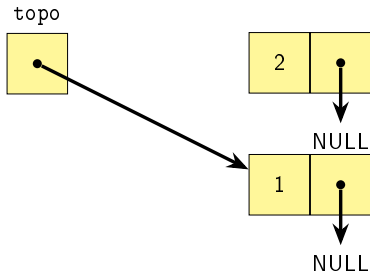
Pilha: implementação com lista encadeada

Inserção de elementos



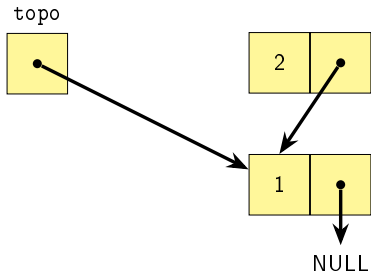
Pilha: implementação com lista encadeada

Inserção de elementos



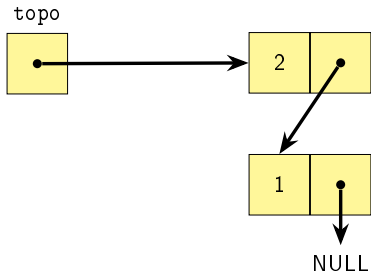
Pilha: implementação com lista encadeada

Inserção de elementos



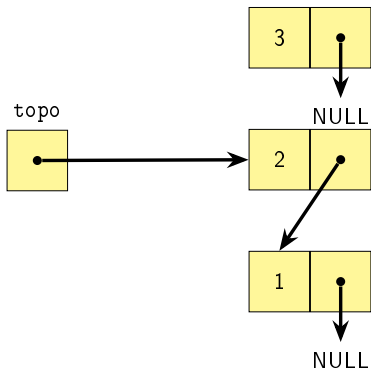
Pilha: implementação com lista encadeada

Inserção de elementos



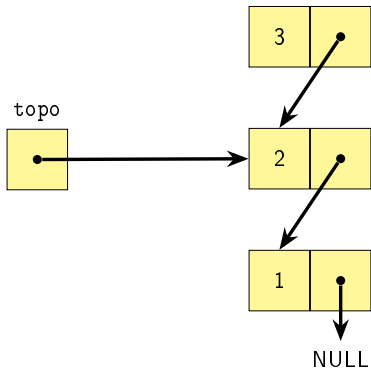
Pilha: implementação com lista encadeada

Inserção de elementos



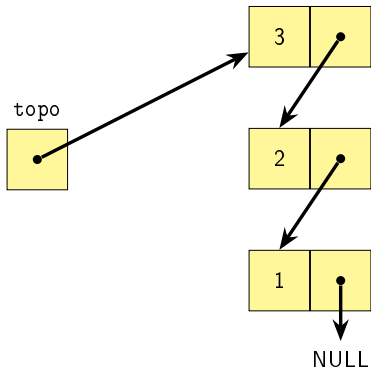
Pilha: implementação com lista encadeada

Inserção de elementos



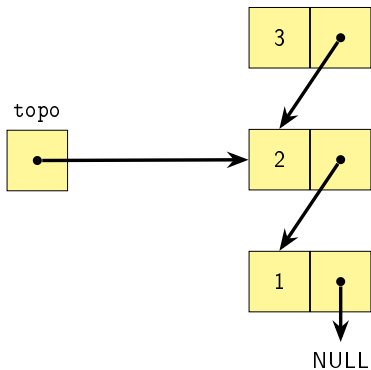
Pilha: implementação com lista encadeada

Remoção de elementos



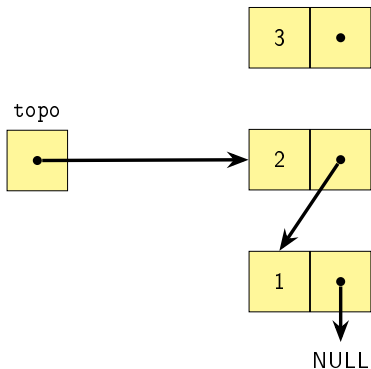
Pilha: implementação com lista encadeada

Remoção de elementos



Pilha: implementação com lista encadeada

Remoção de elementos



Pilha encadeada

Complexidade

- Complexidade das operações empilhar e desempilhar:

| Operação | Complexidade |
|-------------|--------------|
| Empilhar | $\Theta(1)$ |
| Desempilhar | $\Theta(1)$ |

Pilha encadeada

Implementação em C

```
1 typedef struct PilhaE PilhaE
2
3 struct PilhaE{
4     No *topo;
5 };
```

Pilha encadeada

Implementação em C

```
1 PilhaE* criar_pilhaE(){  
2     PilhaE* p = (PilhaE*) malloc(sizeof(PilhaE));  
3  
4     p->topo = NULL;  
5  
6     return p;  
7 }
```

Pilha encadeada

Implementação em C





```
1 void empilhar(PilhaE *p, int key){
2     No *novo; // Novo nó
3
4     // Caso a pilha encadeada seja nula,
5     // alocar um espaço para essa estrutura
6     if (p == NULL)
7         p = criar_pilhaE();
8
9     // Criar novo nó
10    novo = criar_no(key);
11
12    // Apontar o próximo do novo nó para o topo da pilha
13    novo->next = p->topo;
14
15    // Atualizar topo da pilha para novo nó
16    p->topo = novo;
17 }
```


Pilha encadeada


Implementação em C


```
1  int desempilhar(PilhaE *p){
2      No *aux; // Topo da pilha a ser removido
3      int item = INT_MIN; // Valor que está no topo
4
5      if (!pilhaE_vazia(p)){
6          // obter o nó do topo da pilha
7          aux = p->topo;
8
9          // Acessar o conteúdo do nó
10         item = aux->item;
11
12         // Atualizar o topo da pilha
13         p->topo = aux->next;
14
15         // Liberar o nó desempilhado
16         free(aux);
17     }
18
19     return item;
20 }
```

- 1 Implemente uma pilha utilizando somente as operações de lista encadeada.
- 2 Implemente uma fila utilizando somente as operações de lista encadeada.
- 3 Simule uma fila utilizando duas pilhas.
- 4 Simule uma pilha utilizando duas filas.

-  Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.
-  Schouery, R. C. S.
Pilhas, Filas, Listas. Estrutura de Dados.
Slides. Engenharia de Computação. Unicamp, 2019.
-  Oliva, J. T.
Pilhas Encadeadas. AE22CP – Algoritmos e Estrutura de Dados II.
Notas de Aula. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2024.
-  Oliva, J. T.
Filas Encadeadas. AE22CP – Algoritmos e Estrutura de Dados II.
Notas de Aula. Engenharia de Computação. Dainf/UTFPR/Pato Branco, 2024.

 Szwarcfiter, J.; Markenzon, L.
Estruturas de Dados e Seus Algoritmos.
LTC, 2010.

 Tenenbaum, A.; Langsam, Y.
Estruturas de Dados usando C.
Pearson, 1995.

 Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.