EDCO4B
ESTRUTURAS DE DADOS 2

Aula 10 - Organizando Arquivos para Desempenho

Prof. Rafael G. Mantovani

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Apucarana - PR, Brasil

Universidade Tecnológica Federal do Paraná (UTFPR)
Engenharia de Computação

Licença

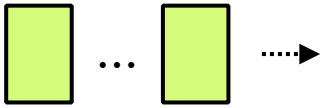
Roteiro

1. Introdução
2. Reuso em registros de tamanho fixo
3. Reuso em registros de tamanho variável
4. Revisão
5. Referências

Roteiro

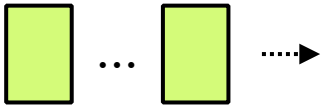
1. **Introdução**
2. **Reuso em registros de tamanho fixo**
3. **Reuso em registros de tamanho variável**
4. **Revisão**
5. **Referências**

Introdução



Objetos

Introdução

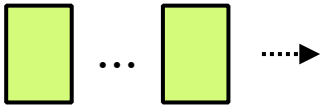


Objetos

Ames | Mary | 123 Maple | Stillwater | OK | 74075 |
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |
…
…
…
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 |

Arquivo

Introdução



Objetos

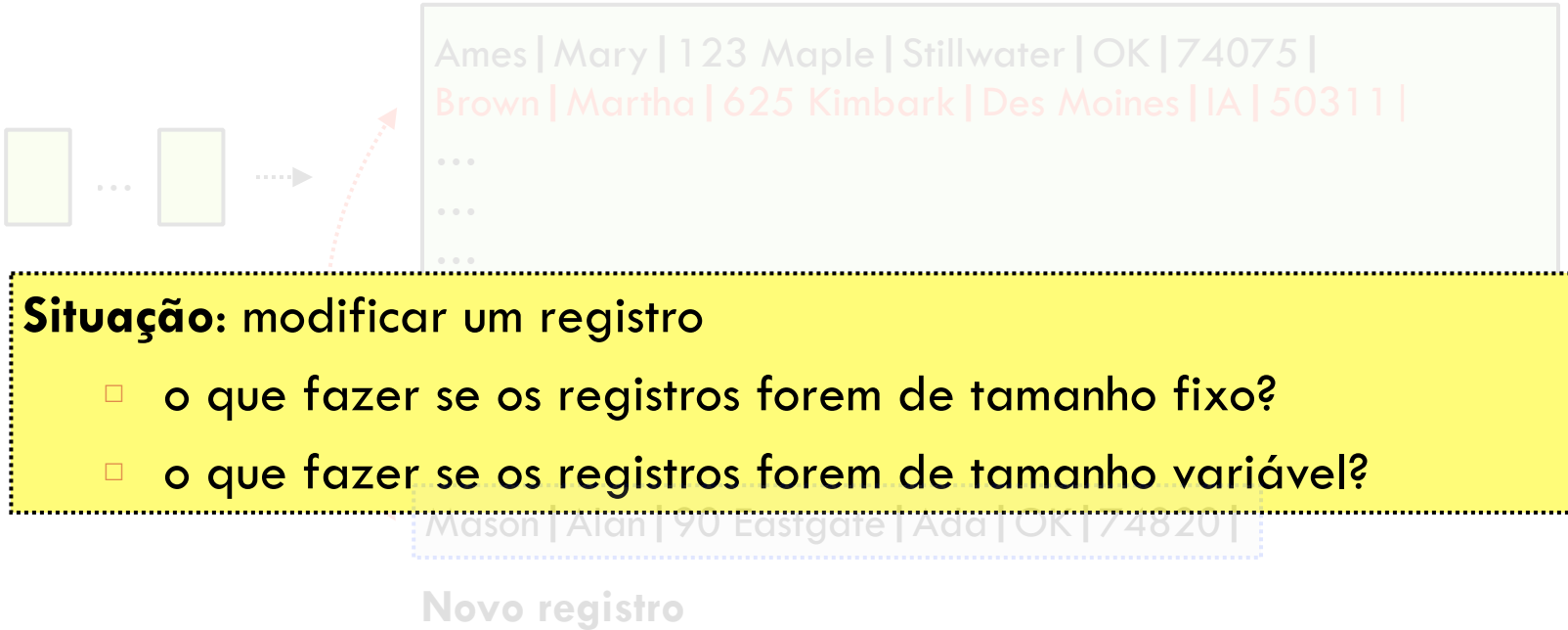
Ames | Mary | 123 Maple | Stillwater | OK | 74075 |
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |
…
…
…
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 |

Arquivo

Mason | Alan | 90 Eastgate | Ada | OK | 74820 |

Novo registro

Introdução



Situação: modificar um registro

- o que fazer se os registros forem de tamanho fixo?
- o que fazer se os registros forem de tamanho variável?

Introdução

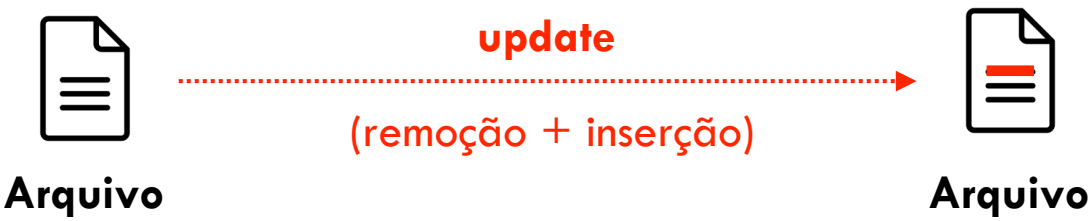
- Organização do arquivo se deteriora conforme o arquivo é modificado
  - adição de registros
  - remoção de registros
  - update de registros

Introdução

- Organização do arquivo se deteriora conforme o arquivo é modificado
  - adição de registros
  - remoção de registros
  - update de registros

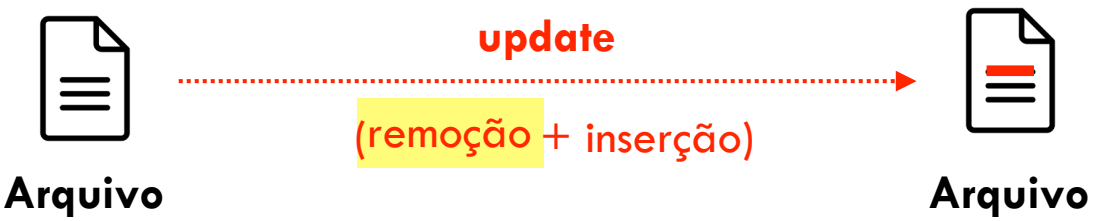
Introdução

- Organização do arquivo se deteriora conforme o arquivo é modificado
  - adição de registros
  - remoção de registros
  - update de registros



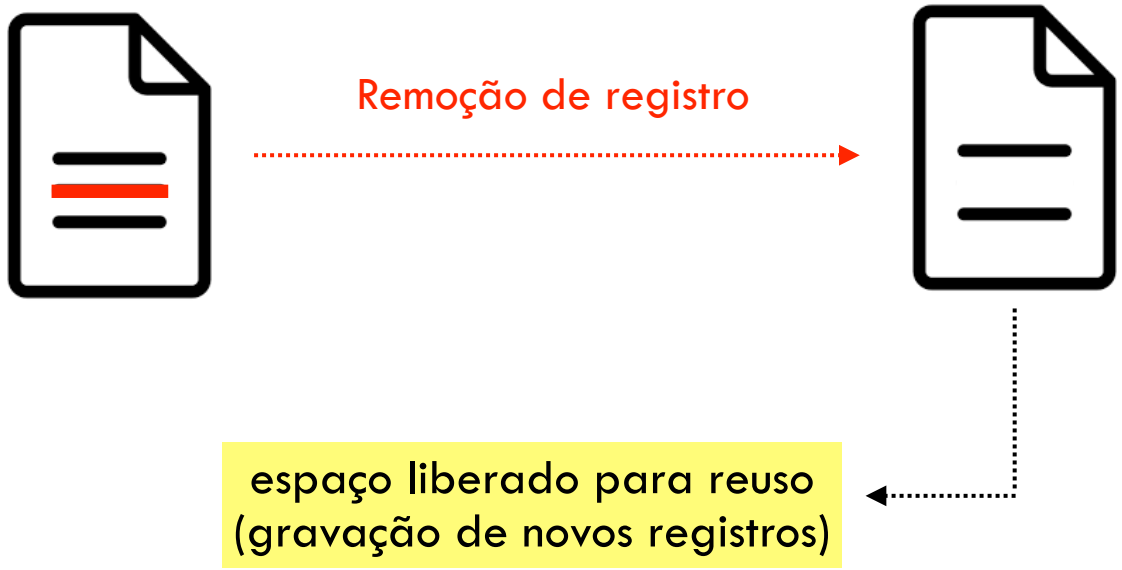
Introdução

- Organização do arquivo se deteriora conforme o arquivo é modificado
  - adição de registros
  - remoção de registros
  - update de registros



Introdução

- Quando se deleta um registro, desejamos **reutilizar** esse espaço



Roteiro

1. Introdução
2. **Reuso em registros de tamanho fixo**
3. Reuso em registros de tamanho variável
4. Revisão
5. Referências

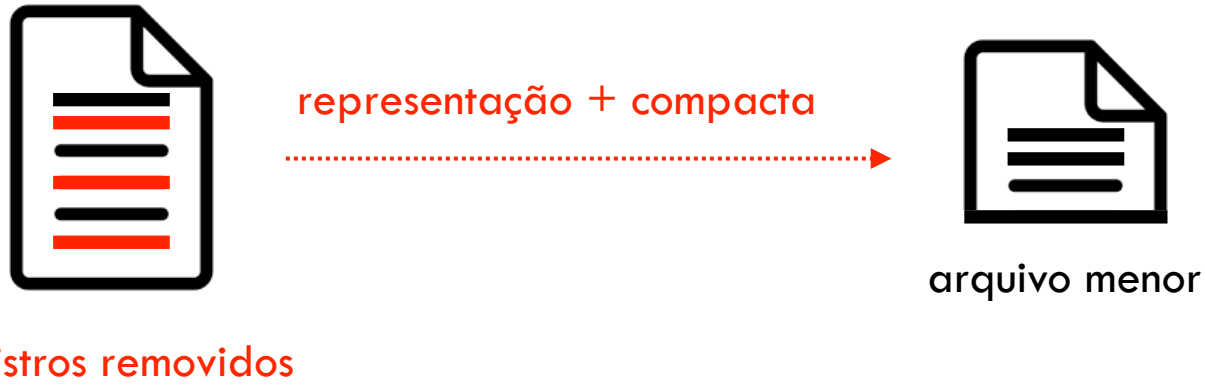
Introdução

- ***Storage Compaction*** → tornar o arquivo menor

registros removidos

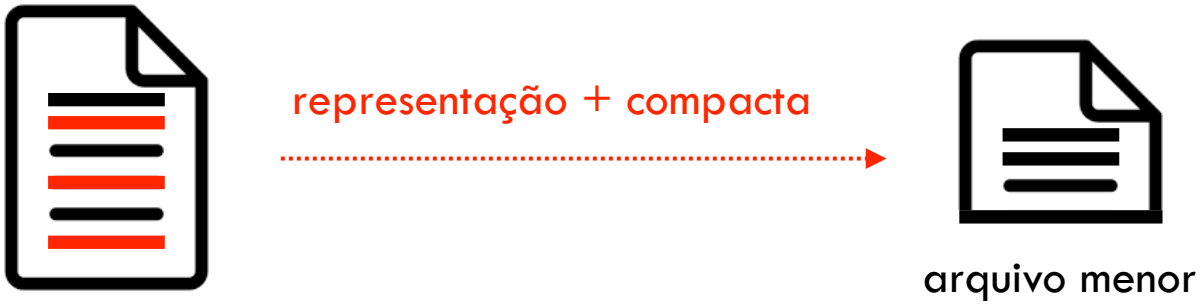
Introdução

- ***Storage Compaction*** → tornar o arquivo menor



Introdução

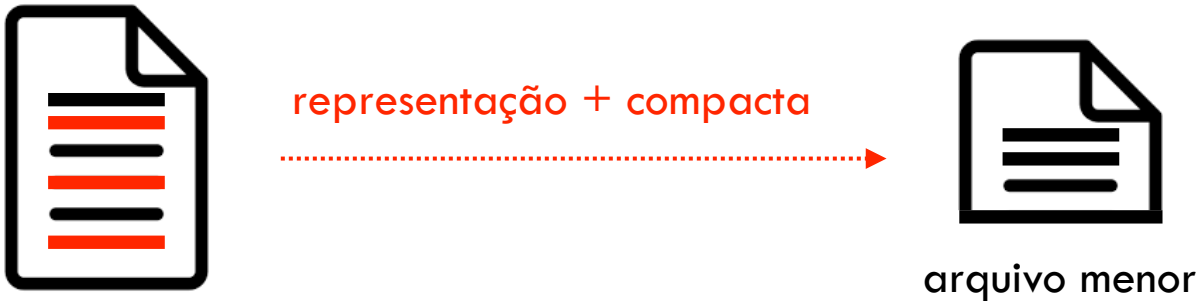
- ***Storage Compaction*** → tornar o arquivo menor



Precisamos de uma forma de identificar os registros removidos (sem realmente remove-los)

Introdução

- ***Storage Compaction*** → tornar o arquivo menor



Precisamos de uma forma de identificar os registros removidos (sem realmente remove-los)
Exemplo: marcar registros com *

Storage Compaction

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 | ……
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 | ……..

Storage Compaction

Remover: Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 | ……
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 | ……..

Storage Compaction

Remover: Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |

* Encontrar o registro (via chave)

Storage Compaction

Remover: Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
* | *rrison* | *Sebatian* | *9035 South Hill* | *Forest Village* | OK | *74820* |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 | ……
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 | ……..

* Marcá-lo como removido (*) sem remove-lo

Storage Compaction

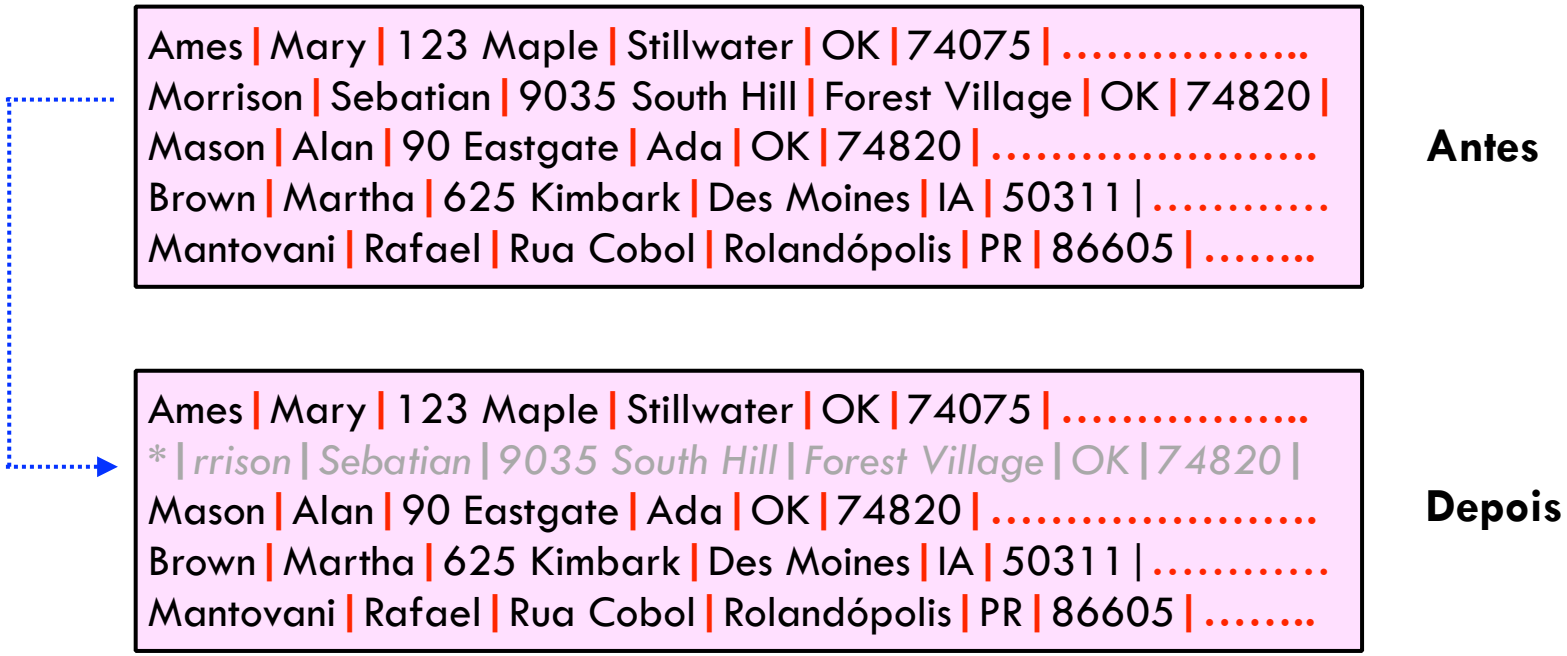
Remover: Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
*| *rrison* | *Sebatian* | *9035 South Hill* | *Forest Village* | *OK* | *74820* |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 | ……
Mantovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 | ……..

* registro fica “invalidado" e seu espaço pode ser reutilizado

Storage Compaction

Remover: Morrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |



Storage Compaction

- Como **reutilizar** o espaço?

Storage Compaction

- Como **reutilizar** o espaço?

- estratégias baseadas no ***storage compatcion*** não reusam os espaços por um determinado tempo
- os registros são marcados como removidos (*) e deixados no arquivo por um tempo
- programas devem incluir uma lógica para ignorar registros removidos
- **vantagem:** remoção pode ser desfeita sem muito esforço

Storage Compaction

- Recuperação do espaço de todos registros deletados acontece uma única vez (dentro de um intervalo de tempo)
  - Depois de acumular registros, o programa reconstrói o arquivo sem os espaços em branco

Storage Compaction

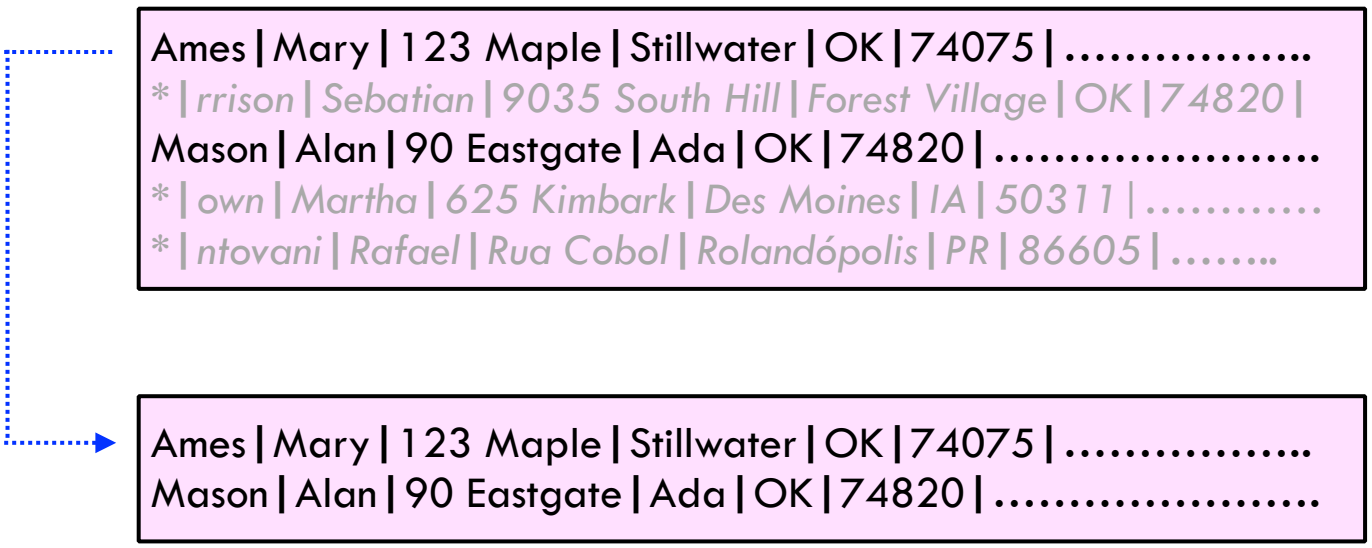
Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
*| *rrison* | *Sebatian* | *9035 South Hill* | *Forest Village* | *OK* | *74820* |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
*| *own* | *Martha* | *625 Kimbark* | *Des Moines* | *IA* | *50311* | ……
*| *ntovani* | *Rafael* | *Rua Cobol* | *Rolandópolis* | *PR* | *86605* | ……..

Storage Compaction

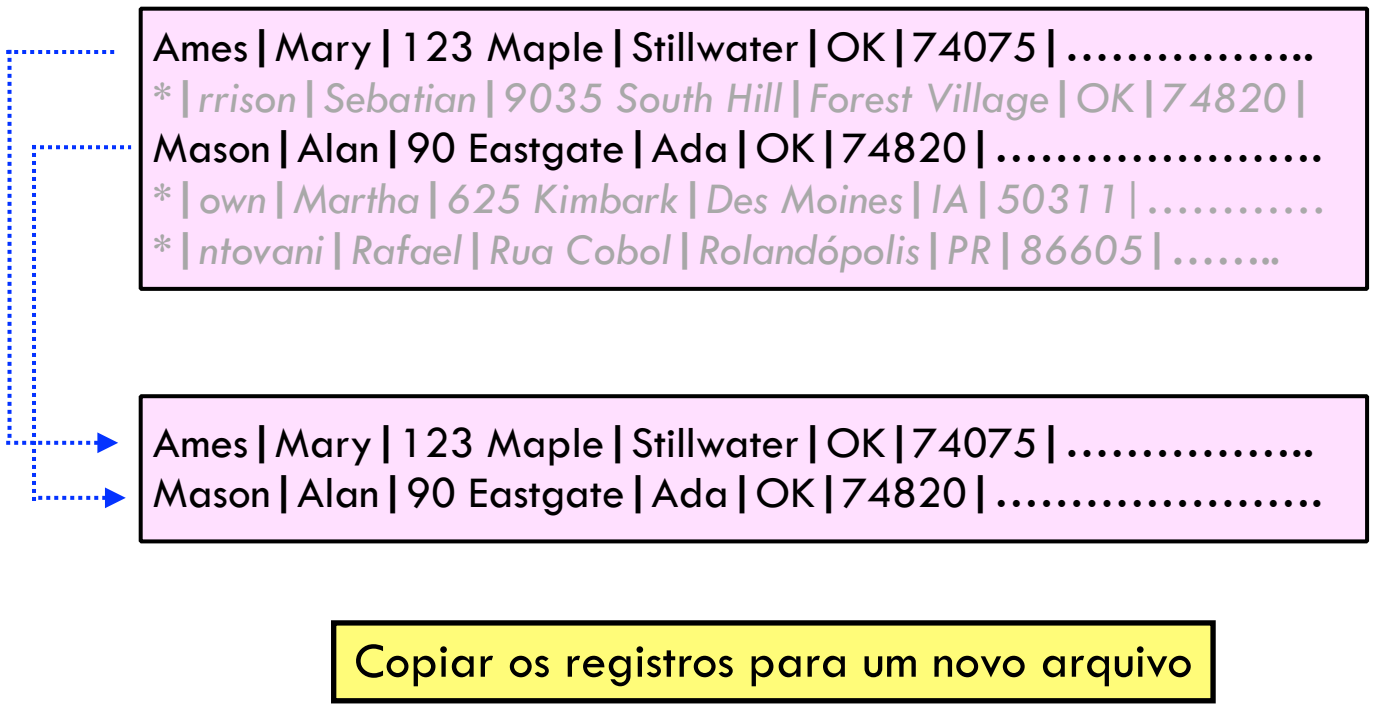
Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
*| *rrison* | *Sebatian* | *9035 South Hill* | *Forest Village* | *OK* | *74820* |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..
*| *own* | *Martha* | *625 Kimbark* | *Des Moines* | *IA* | *50311* | ……
*| *ntovani* | *Rafael* | *Rua Cobol* | *Rolandópolis* | *PR* | *86605* | ……..

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | ……..
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | ……..

Storage Compaction



Storage Compaction



Copiar os registros para um novo arquivo

Storage Compaction

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | .....
* | rrison | Sebatian | 9035 South Hill | Forest Village | OK | 74820 |
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | .....
* | own | Martha | 625 Kimbark | Des Moines | IA | 50311 | .....
* | ntovani | Rafael | Rua Cobol | Rolandópolis | PR | 86605 | .....

Ames | Mary | 123 Maple | Stillwater | OK | 74075 | .....
Mason | Alan | 90 Eastgate | Ada | OK | 74820 | .....

Deletar o arquivo antigo e manter apenas o novo

Exercícios

1) Implemente uma função que realiza a **remoção** (**delete**) de registros em um arquivo de tamanho fixo. Use os registros de heróis da aula passada, levemente alterado, como formato de registro (gravação/ record).

```
// arq: arquivo de entrada com os registros gravados
// key: chave no formato canônico que será manipulada e removida
// Returns: retorna 1 se deletou com sucesso
//          retorna 0 se não deletou, pq o registro não foi encontrado
int deleteRecord (FILE *arq, char* key);
```

Exercícios

```
// struct para representar um Heroi
 typedef struct {
      char primeiroNome[16];        // primeiro nome
      char sobrenome[16];           //  último nome do heroi
      char nomeHeroi[16];           // nome de heroi, alias
      char poder[16];               // poder
      char fraqueza[21];            // fraqueza
      char cidade[21];              //  cidade onde vive
      char profissao[21];           //  profissão
 } Heroi;
```

Exercício

2) Implemente uma função que realize o processo de storage compaction de um arquivo. A função recebe um arquivo que contém uma coleção de registros, e remove um determinado registro que possua a chave canônica solicitada. Além disso, elabore uma função principal para testar as suas funções de remoção de registros e compactação de dados.

```
// arq: arquivo de entrada com os registros gravados
// key: chave no formato canônico que será manipulada e removida
// recria o arquivo arq sem os registros deletados
void storageCompaction (FILE *arq, char* key);
```

Removendo: registros de tamanho-fixo

- Existem aplicações que são muito voláteis e interativas para se usar o ***storage compatction***
  - recuperar o espaço o quanto antes
  - recuperação dinâmica de armazenamento

- recuperação dinâmica demanda:
  - marcar de alguma forma os registros removidos (*)
  - meio de encontrar os registros removidos e reusar o espaço para adicionar novos registros

Removendo: registros de tamanho-fixo

- Existem aplicações que são muito voláteis e interativas para se usar o ***storage compatction***
  - recuperar o espaço o quanto antes
  - recuperação dinâmica de armazenamento

- recuperação dinâmica demanda:
  - marcar de alguma forma os registros removidos (*)
  - **meio de encontrar os registros removidos e reusar o espaço para adicionar novos registros**

Recuperação Dinâmica

1. percorrer o arquivo, registro por registro, até que um registro removido seja encontrado
2. se chegou ao fim do arquivo sem encontrar algum registro marcado (*), adiciona o novo registro ao fim do arquivo (append)
3. senão substitui o registro deletado pelo novo registro

Recuperação Dinâmica

1. percorrer o arquivo, registro por registro, até que um registro removido seja encontrado
2. se chegou ao fim do arquivo sem encontrar algum registro marcado (*), adiciona o novo registro ao fim do arquivo (append)
3. senão substitui o registro deletado pelo novo registro

Problemas ? Pode ser muito lenta essa varredura (linear)

Removendo: registros de tamanho-fixo

- Melhoramos a solução:
  - adicionando um meio de saber imediatamente se existem espaços para reuso
  - e ter um meio de acessar diretamente um destes espaços (se existirem)
  - O que fazer/explorar para melhorar mais ainda?

Removendo: registros de tamanho-fixo

- Melhoramos a solução:
  - adicionando um meio de saber imediatamente se existem espaços para reuso
  - e ter um meio de acessar diretamente um destes espaços (se existirem)
  - O que fazer/explorar para melhorar mais ainda?

Usar Listas Lineares (Pilhas) !!!

Removendo: registros de tamanho-fixo

- Listas Lineares
  - podemos mover pela lista procurando novas posições disponíveis para inserção de registros
  - Lista de disponibilidade (avail list): quando a lista contém espaços disponíveis para reuso

- Com registros de tamanho-fixo, todas as posições são “iguais" então o novo registro pode ser inserido em qualquer um dos espaços disponíveis
  - Listas Lineares —> Pilhas!!!

Removendo: registros de tamanho-fixo

- Pilhas
  - contém os RRNs dos registros removidos
  - o espaço mais recente é reutilizado

Algoritmo (Update)

Se (Pilha estiver vazia) :
adiciona novo registro no fim do arquivo
Senão:
Reusa o espaço disponível mais recente,
usando RRN para acessar a posição

Removendo: registros de tamanho-fixo

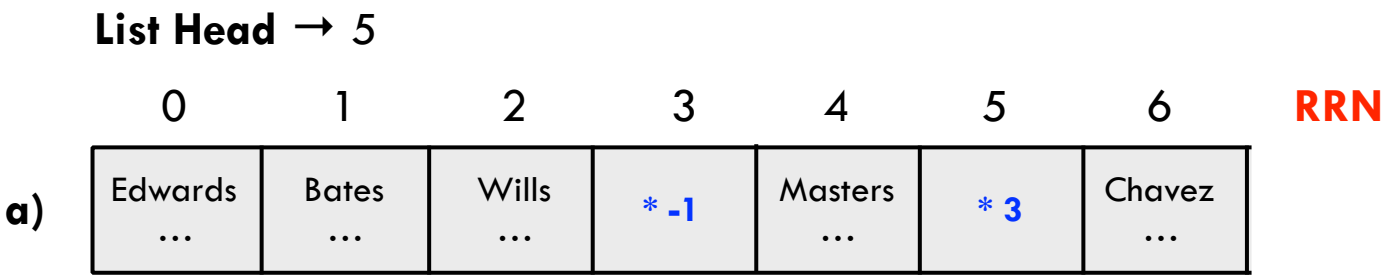
- Pilhas
  - contém os RRNs dos registros removidos
  - o espaço mais recente é reutilizado

Algoritmo (Update)

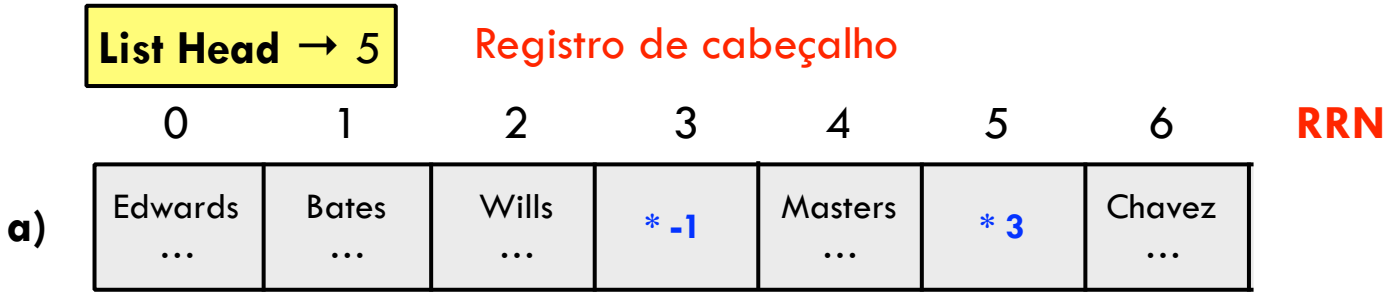
Se (Pilha estiver vazia) :
adiciona novo registro no fim do arquivo
Senão:
Reusa o espaço disponível mais recente,
usando RRN para acessar a posição

Obs: Não implementamos uma estrutura Pilha separada, mas usamos a estrutura do registro para organizar uma Pilha

Removendo: registros de tamanho-fixo

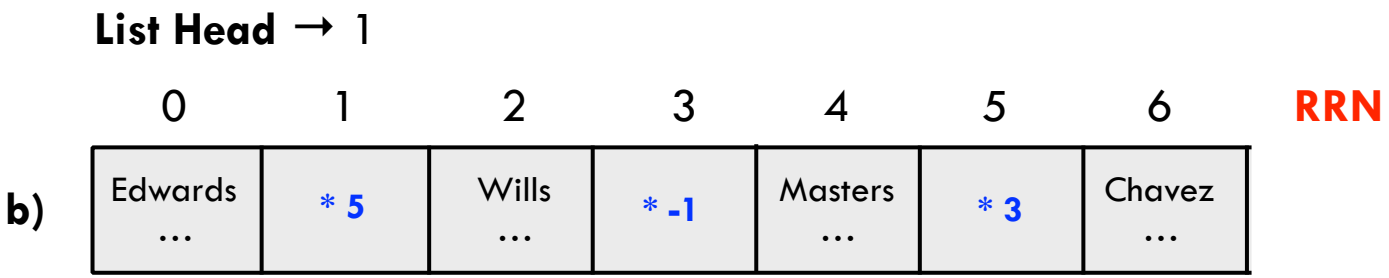


Removendo: registros de tamanho-fixo

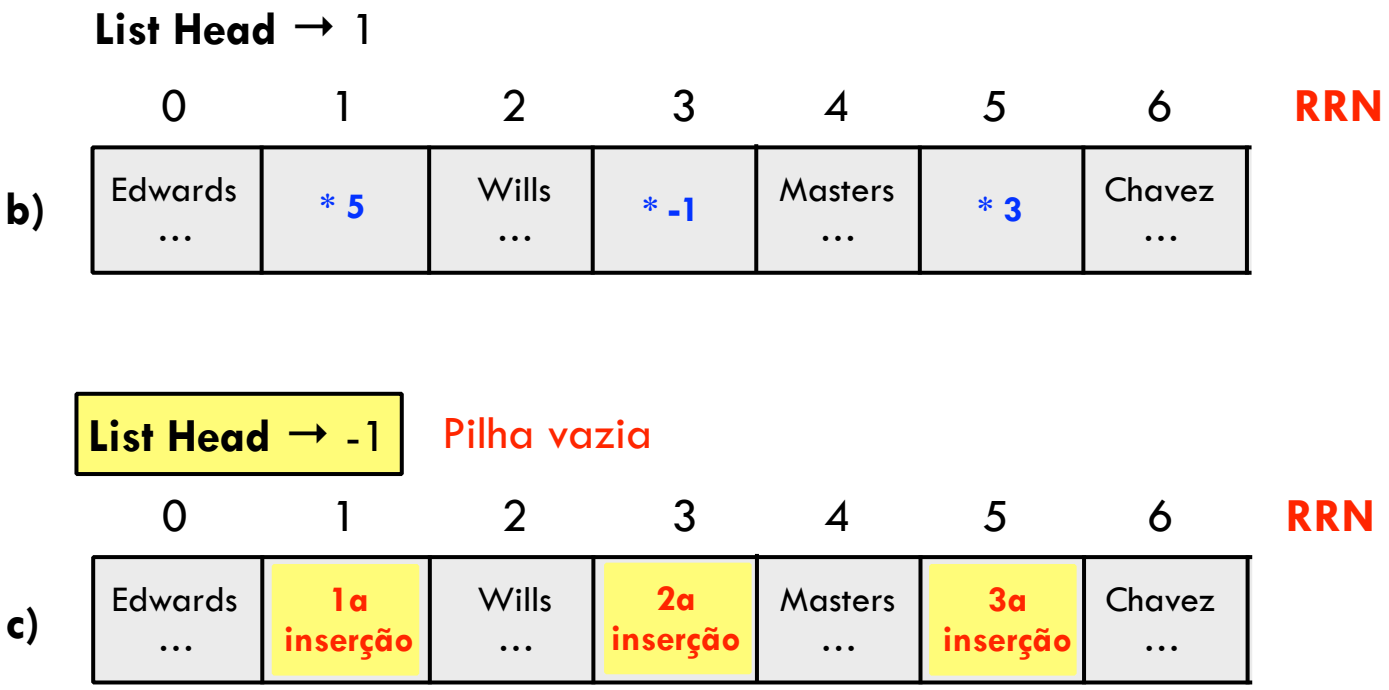


Removendo: registros de tamanho-fixo

Removendo: registros de tamanho-fixo



Removendo: registros de tamanho-fixo



Removendo: registros de tamanho-fixo

- Na prática demanda a necessidade da implementação de uma função que:
  - retorna o RRN de um espaço para ser reutilizado, ou
  - retorna o RRN do novo registro que será adicionado ao fim do arquivo, pois não há espaços para reuso

Roteiro

1. Introdução
2. Reuso em registros de tamanho fixo
3. **Reuso em registros de tamanho variável**
4. Revisão
5. Referências

Removendo: registros de tamanho-variável

- Quando temos registros de tamanho-variável, necessitamos:

Removendo: registros de tamanho-variável

- Quando temos registros de tamanho-variável, necessitamos:
  - uma forma de ligar os registros removidos em uma lista
  - um algoritmo para adicionar novos registros removidos na lista de disponibilidade
  - um algoritmo para encontrar e remover registros da lista de disponibilidade quando eles forem reutilizados pelo programa/aplicação

Removendo: registros de tamanho-variável

- Listas Lineares !!!
  - controlar os registros removidos com uma marca (*)
  - não podemos usar os RRNs, mas sim os ***byte offsets***
    - registros tem tamanhos variáveis

Removendo: registros de tamanho-variável

Head.First_Avail: -1
40 Ames | Mary | 123 Maple | Stillwater | OK | 74075 | **64** Sebastian | 9035 South Hillcrest | Forest Village | OK | 74820 | **45** Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |

Antes

Removendo: registros de tamanho-variável

Head.First_Avail: -1
40 Ames | Mary | 123 Maple | Stillwater | OK | 74075 | **64** Sebastian | 9035 South Hillcrest | Forest Village | OK | 74820 | **45** Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |

Antes

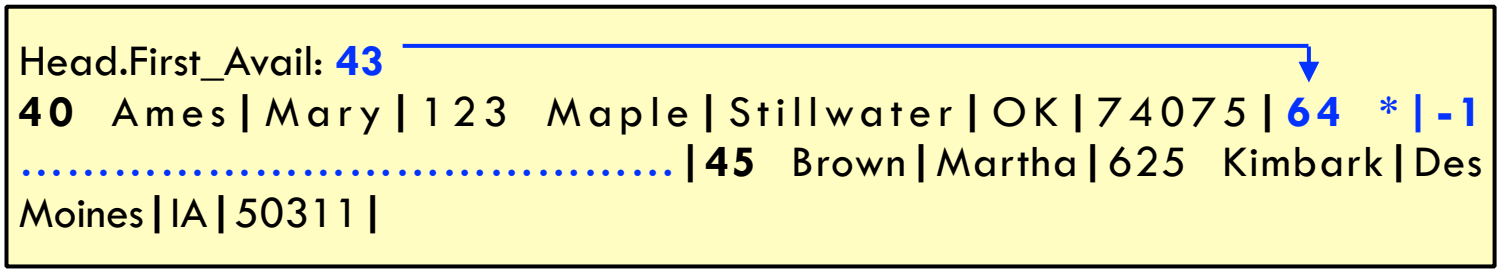
Head.First_Avail: **43**
40 Ames | Mary | 123 Maple | Stillwater | OK | 74075 | **64** * | **-1**
..... | **45** Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |

Depois

Removendo: registros de tamanho-variável

Head.First_Avail: -1
40 Ames | Mary | 123 Maple | Stillwater | OK | 74075 | **64** Sebastian | 9035 South Hillcrest | Forest Village | OK | 74820 | **45** Brown | Martha | 625 Kimbark | Des Moines | IA | 50311 |

Antes



Depois

Removendo: registros de tamanho-variável

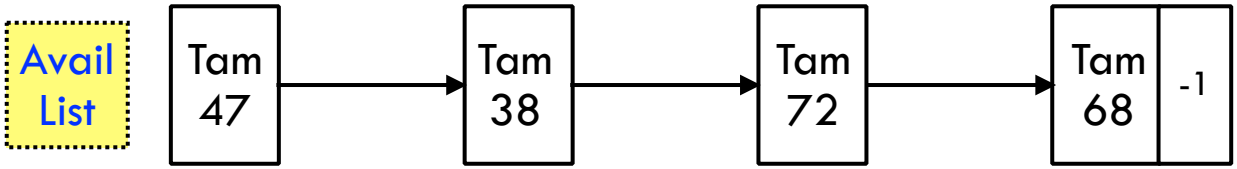
- Como adicionar e remover os registros?

Removendo: registros de tamanho-variável

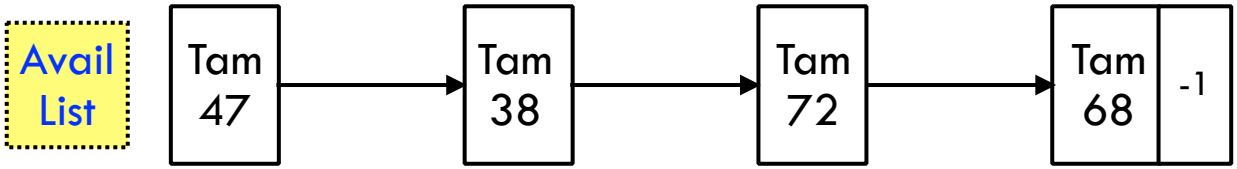
- Como adicionar e remover os registros?
  - não podemos usar Pilhas porque os registros tem tamanhos diferentes
  - mais uma condição: encontrar um espaço que seja **grande o suficiente** para receber o novo registro
  - necessitamos: percorrer a lista e encontrar um espaço que satisfaça essa condição

Removendo: registros de tamanho-variável

Lista de Disponibilidade

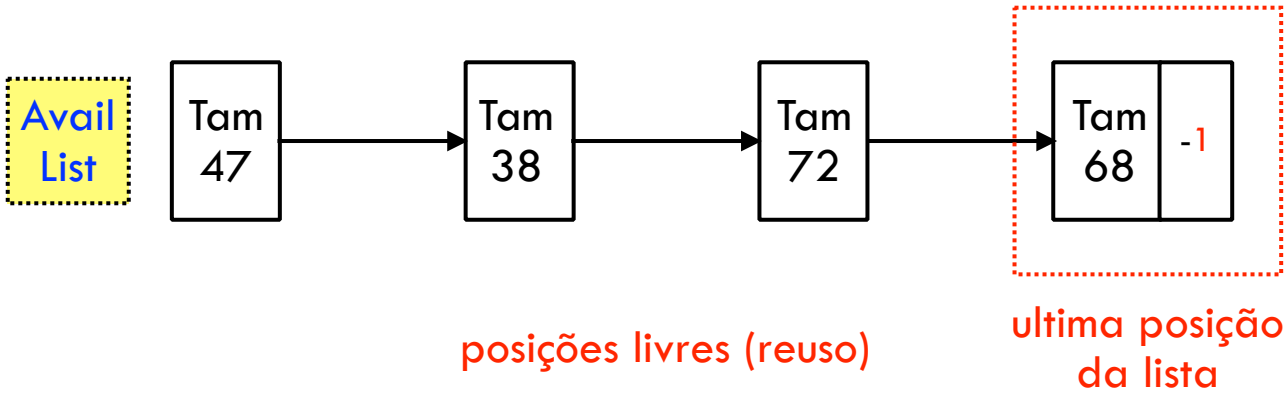


Removendo: registros de tamanho-variável

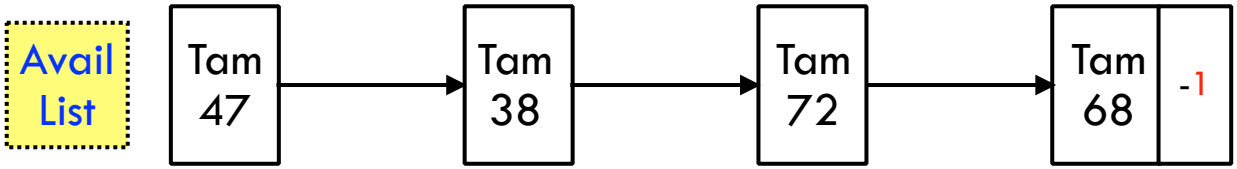


posições livres (reuso)

Removendo: registros de tamanho-variável

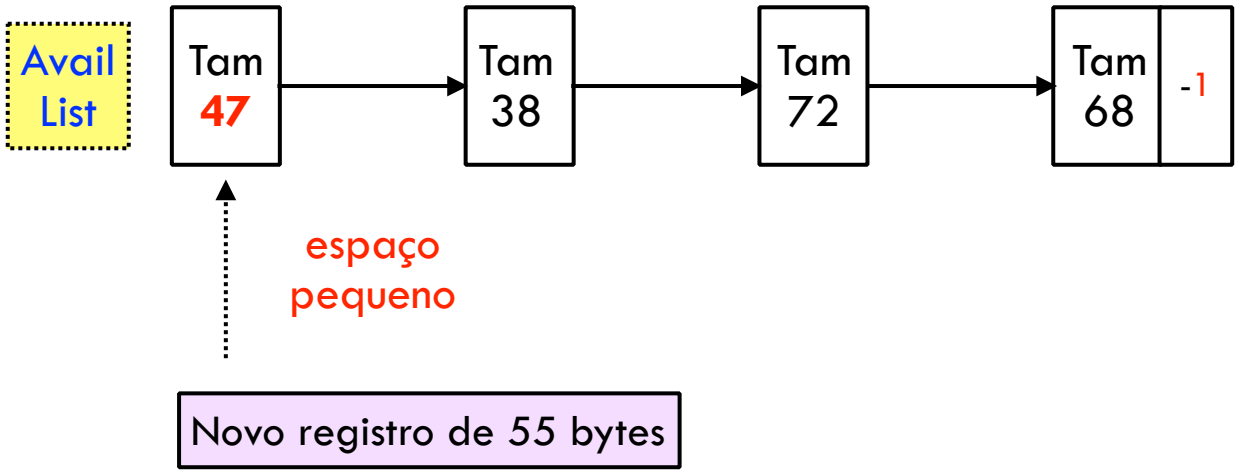


Removendo: registros de tamanho-variável

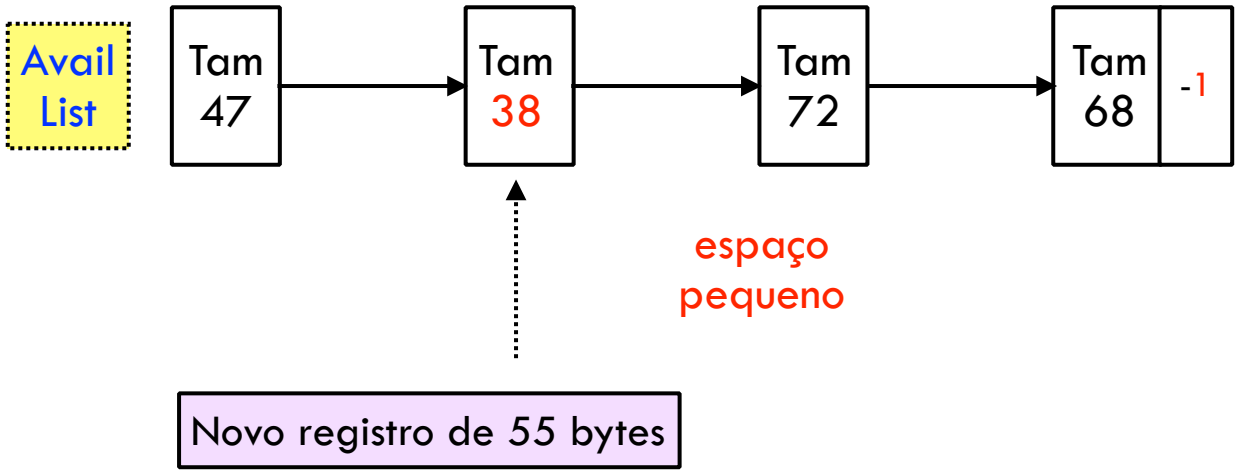


Novo registro de 55 bytes

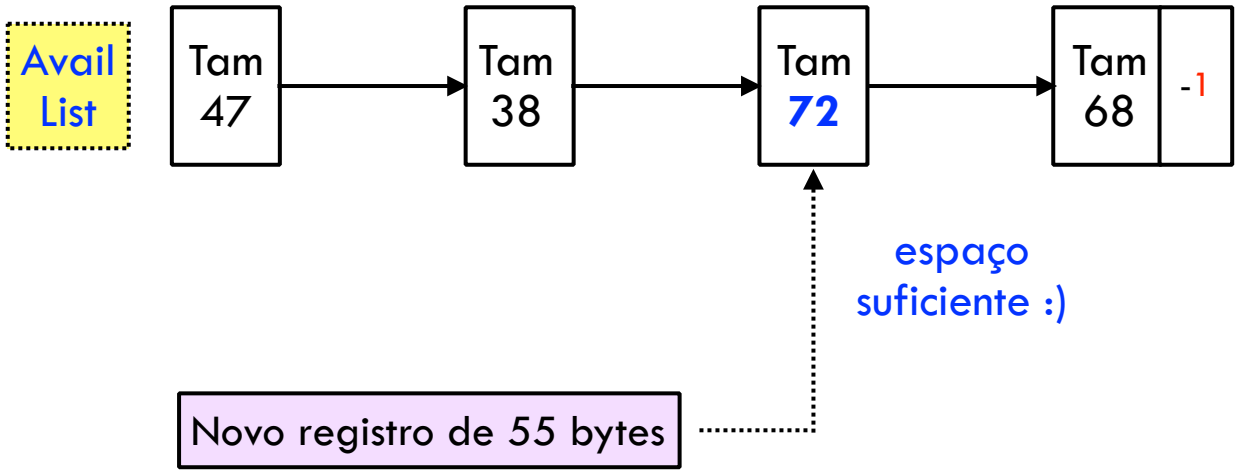
Removendo: registros de tamanho-variável



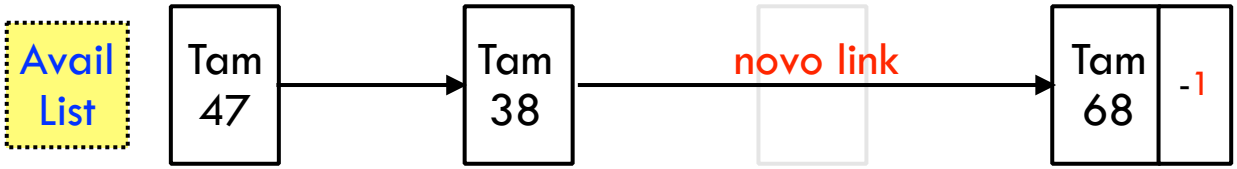
Removendo: registros de tamanho-variável



Removendo: registros de tamanho-variável



Removendo: registros de tamanho-variável



espaço reutilizado

Novo registro de 55 bytes

Removendo: registros de tamanho-variável

- O que fazer quando um registro de tamanho variável é removido e substituído por um registro menor?

Removendo: registros de tamanho-variável

Head.First_Avail: 43
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 *|-1……..
……|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

Removendo: registros de tamanho-variável

Head.First_Avail: 43
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 *|-1……..
……|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

Novo registro: Ham|Al|28 Elm|Ada|OK|70332| 26 bytes

Removendo: registros de tamanho-variável

Head.First_Avail: 43
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 *|-1……..
……|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

Novo registro: Ham|Al|28 Elm|Ada|OK|70332| 26 bytes

Head.First_Avail: **-1**
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 Ham|Al|28 Elm|Ada|
OK|70332|……|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

Removendo: registros de tamanho-variável

Head.First_Avail: 43
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 *|-1.....
.....|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

Novo registro: Ham|Al|28 Elm|Ada|OK|70332| 26 bytes

Head.First_Avail: **-1**
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 Ham|Al|28 Elm|Ada|
OK|70332|.....|45 Brown|Martha|625 Kimbark|Des Moines|
IA|50311|

espaço perdido: 37 bytes
(fragmentação interna)

Removendo: registros de tamanho-variável

- Como resolver Fragmentação Interna ?

Head.First_Avail: **-1**
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 Ham|Al|28 Elm|Ada|OK|70332|.....|45 Brown|Martha|625 Kimbark|Des Moines|IA|50311|

Removendo: registros de tamanho-variável

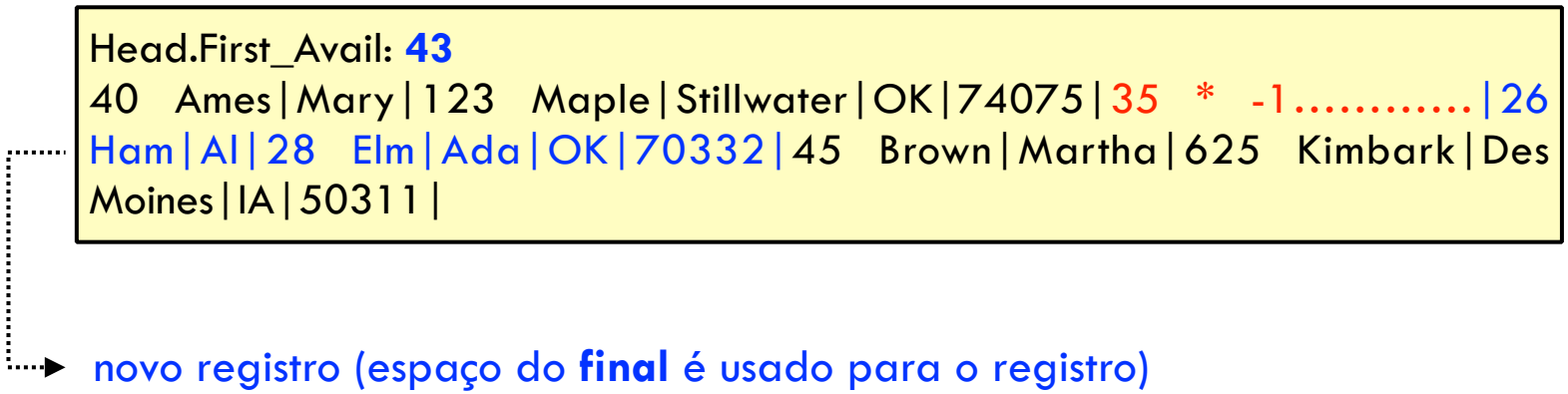
- Como resolver Fragmentação Interna ?

Head.First_Avail: **-1**
40 Ames|Mary|123 Maple|Stillwater|OK|74075|64 Ham|Al|28 Elm|Ada|OK|70332|.....|45 Brown|Martha|625 Kimbark|Des Moines|IA|50311|

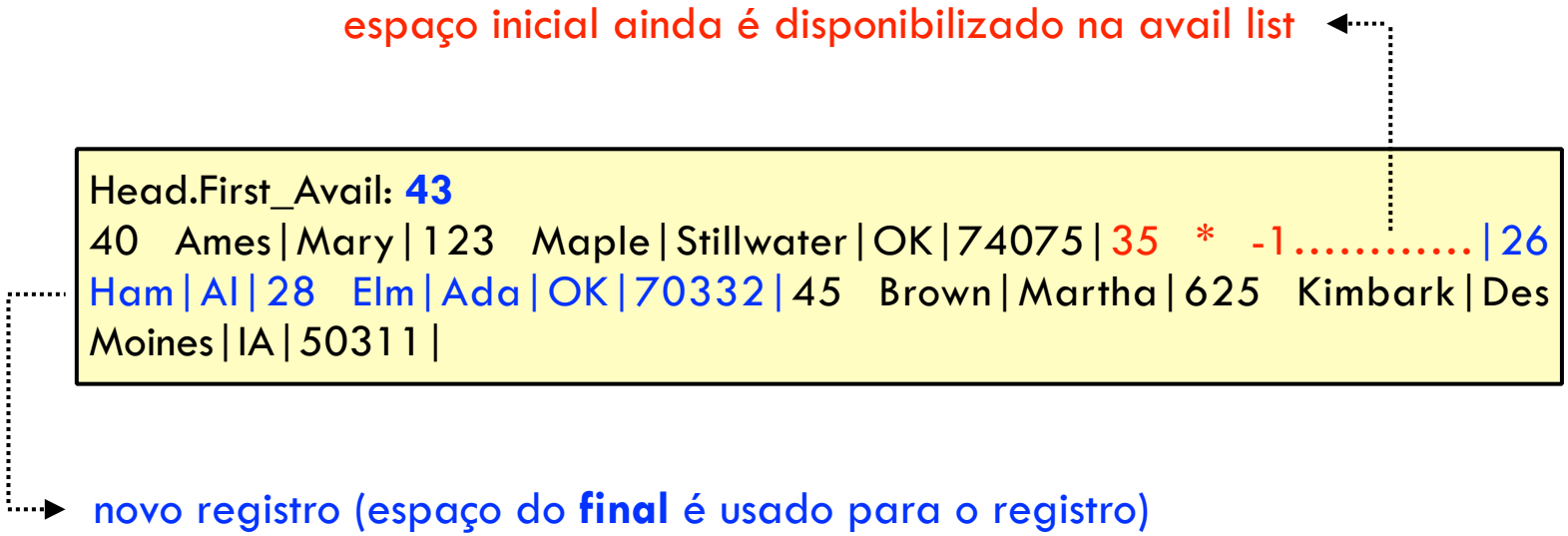
Dividimos o espaço em 2: parte ainda disponível, e parte alocada para o novo registro

Head.First_Avail: **43**
40 Ames|Mary|123 Maple|Stillwater|OK|74075|35 * -1.....|26 Ham|Al|28 Elm|Ada|OK|70332|45 Brown|Martha|625 Kimbark|Des Moines|IA|50311|

Removendo: registros de tamanho-variável



Removendo: registros de tamanho-variável



Removendo: registros de tamanho-variável

- Estratégias de gerenciamento da lista:
  - first-fit - lista não ordenada
  - best-fit - lista ordenada crescentemente
  - worst-fit: lista ordenada decrescentemente

Removendo: registros de tamanho-variável

- Estratégias de gerenciamento da lista:
  - first-fit - lista não ordenada
  - best-fit - lista ordenada crescentemente
  - worst-fit: lista ordenada decrescentemente

Tarefa: pensar nas implicâncias de cada uma destas alternativas e como implementá-las.

Roteiro

1. Introdução
2. Reuso em registros de tamanho fixo
3. Reuso em registros de tamanho variável
4. Revisão
5. Referências

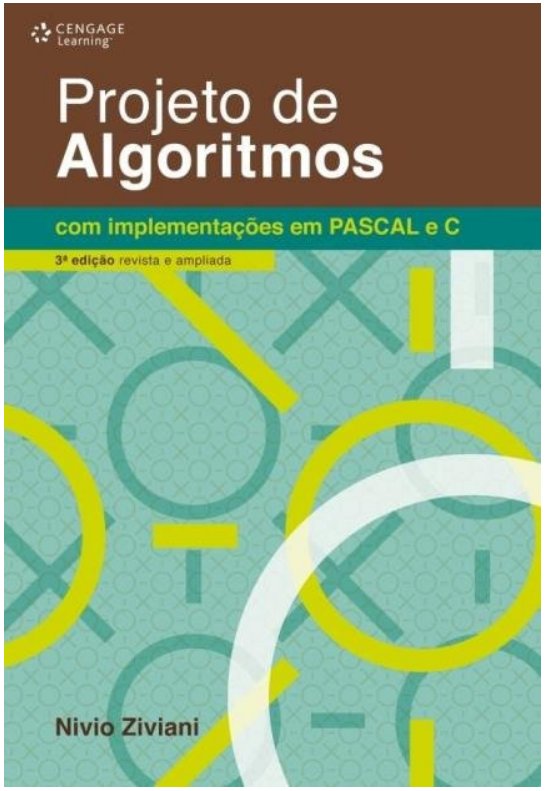
Revisão

- ***Storage Compaction***: envolve criar um novo arquivo (de tempos em tempos)
- Recuperação Dinâmica
  - registros de tamanho fixo
    - Pilhas + RRN
  - registros de tamanho variado
    - Listas + byte offset
- Tudo implementado direto no arquivo
  - Registro de cabeçalho para ajudar na manipulação

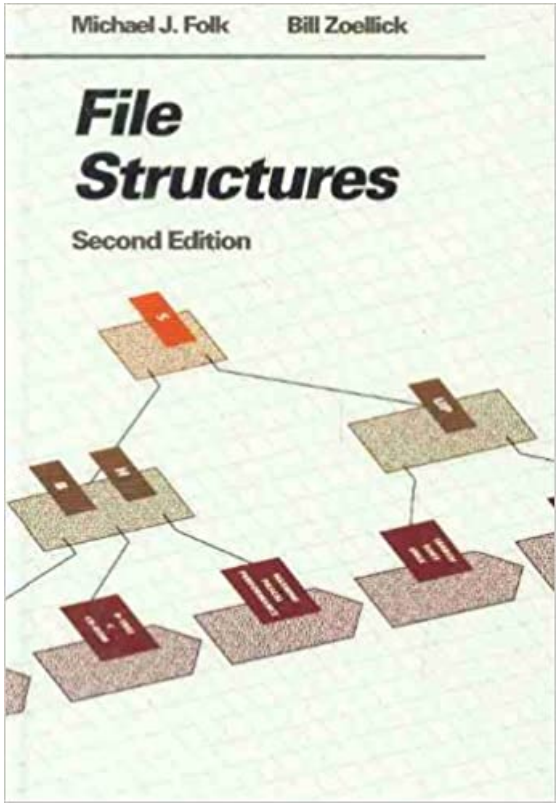
Roteiro

1. Introdução
2. Reuso em registros de tamanho fixo
3. Reuso em registros de tamanho variável
4. Revisão
5. Referências

Referências sugeridas

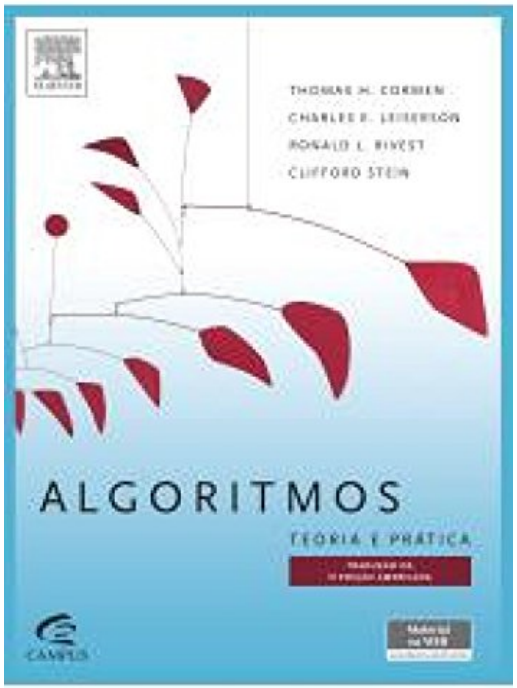


[Ziviani, 2010]



[Folk & Zoellick, 1992]

Referências sugeridas



[Cormen et al, 2018]

[Drozdek, 2017]

Perguntas?

Prof. Rafael G. **Mantovani**

rafaelmantovani@utfpr.edu.br