ACH2024

Aula 26
Ordenação externa:
Seleção por substituição e intercalação polifásica

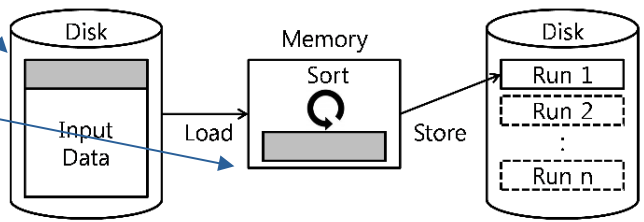
Profa. Ariane Machado Lima

Aula passada

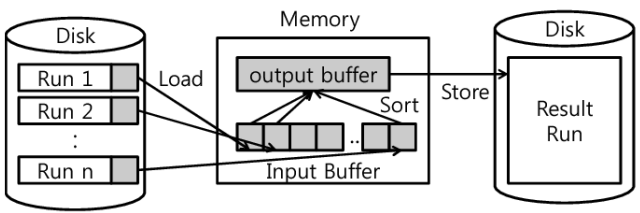
Ordenação Externa

- O método mais importante é o de ordenação por intercalação.
- Intercalar significa combinar dois ou mais blocos ordenados em um único bloco ordenado. Também conhecido como mergeSort externo
- A intercalação é utilizada como uma operação auxiliar na ordenação.
- Estratégia geral dos métodos de ordenação externa:
  1. Quebre o arquivo em blocos do tamanho da memória interna disponível.
  2. Ordene cada bloco na memória interna.
  3. Intercale os blocos ordenados, fazendo várias passadas sobre o arquivo.
  4. A cada passada são criados blocos ordenados cada vez maiores, até que todo o arquivo esteja ordenado.

Cuidado: esse bloco não é o bloco do disco – um melhor termo seria "**segmento**"



(a) Run formation phase

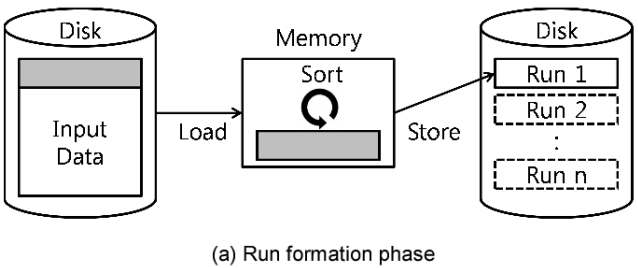


(b) Merge phase

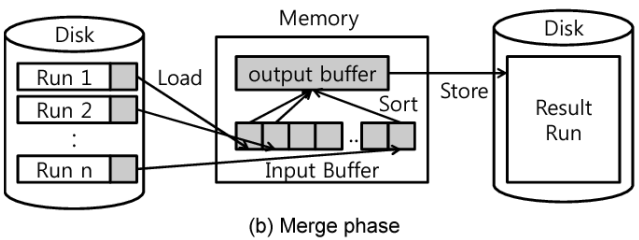
Lee, Joonhee et al. "External Mergesort for Flash-Based Solid State Drives." IEEE Transactions on Computers 65 (2016): 1518-1527.

Ordenação Externa

- Os algoritmos para ordenação externa devem reduzir o número de passadas sobre o arquivo.
- Uma boa medida de complexidade de um algoritmo de ordenação por intercalação é o número de vezes que um item é lido ou escrito na memória auxiliar.
- Os bons métodos de ordenação geralmente envolvem no total menos do que dez passadas sobre o arquivo.



(a) Run formation phase



(b) Merge phase

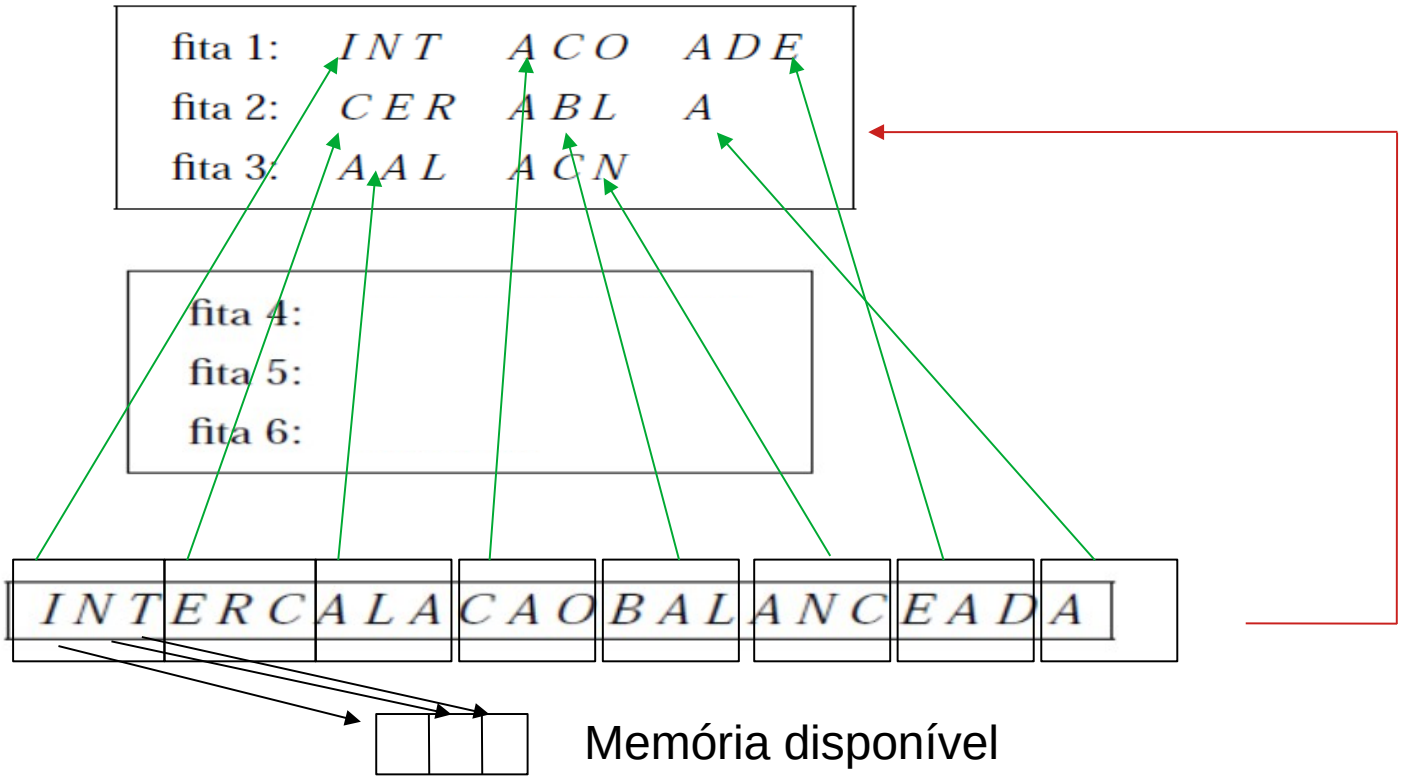
Lee, Joonhee et al. “External Mergesort for Flash-Based Solid State Drives.” IEEE Transactions on Computers 65 (2016): 1518-1527.

Principais abordagens gerais de ordenação externa

- **Intercalação balanceada**
- **Seleção por substituição**
- Intercalação **Polifásica**

Intercalação Balanceada de Vários Caminhos

- Fase de criação dos segmentos ordenados (**corridas**).



Ordena e joga de forma **balanceada** pelas f fitas (metade das fitas disponíveis inicialmente) – cada vez coloco em uma fita

Intercalação Balanceada de Vários Caminhos

- Fase de intercalação - Primeira passada:
  1. O primeiro registro de cada fita é lido.
  2. Retire o registro contendo a menor chave.
  3. Armazene-o em uma fita de saída.
  4. Leia um novo registro da fita de onde o registro retirado é proveniente.
  5. Ao ler o terceiro registro de uma corrida sua fita fica inativa.
  6. A fita é reativada quando o terceiro registro das outras fitas forem lidos.
  7. Neste instante 1 corrida de nove registros ordenados foi formado na fita de saída.
  8. Repita o processo para as corridas restantes.

fita 1:	*I N T*	*A C O*	*A D E*
fita 2:	*C E R*	*A B L*	*A*
fita 3:	*A A L*	*A C N*	

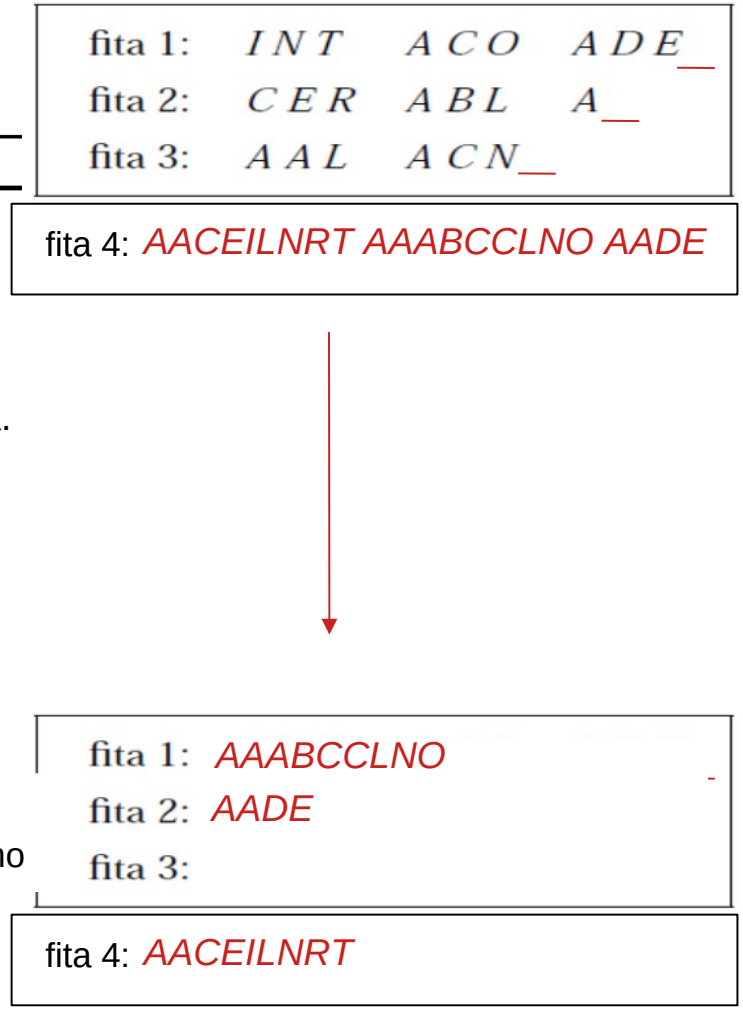
fita 4:	*AACEILNRT*
fita 5:	*AAABCCLNO*
fita 6:	*AADE*

Próxima passada faz o mesmo, mas agora usando fitas 4 a 6 como entrada e as fitas 1 a 3 como saída, e assim sucessivamente até gerar uma única corrida

$$P(n) = \lceil \log_f \lceil \frac{n}{m} \rceil \rceil$$

Intercalação Balanceada de Vários Caminhos

- No exemplo foram utilizadas $2f$ fitas para uma intercalação-de-f-caminhos.
- É possível usar apenas $f + 1$ fitas:
  - Encaminhe todas as corridas intercaladas para uma única fita de saída.
  - Redistribuia as corridas entre as fitas de onde elas foram lidas.
  - O custo envolvido é uma passada a mais em cada intercalação.
- No caso do exemplo de 22 registros, apenas quatro fitas seriam suficientes:
  - A intercalação das corridas a partir das fitas 1, 2 e 3 seria toda dirigida para a fita 4.
  - Ao final, a segunda e a terceira corridas ordenadas de nove registros seriam transferidas de volta para as fitas 1 e 2 , e a fita 3 usada como fita de saída



Aula da hoje

- **Intercalação balanceada**
- **Seleção por substituição**
- Intercalação **Polifásica**

Dá para diminuir o número de passadas?

Lembrando que há duas fases na ordenação externa: geração das corridas iniciais (ordenadas) e intercalação

Número de passos de intercalação P(N) = teto($\log_f$ teto(N/M)), sendo teto(N/M) o número de corridas iniciais

Se diminuirmos o número de corridas (gerando corridas mais longas), podemos diminuir o número de passos...

Implementação por meio de Seleção por Substituição

- A implementação do método de intercalação balanceada pode ser realizada utilizando filas de prioridades.
- As duas fases do método podem ser implementadas de forma eficiente e elegante.
- Operações básicas para formar corridas ordenadas:
  - Obter o menor dentre os registros presentes na memória interna.
  - Substituí-lo pelo próximo registro da fita de entrada.
- Estrutura ideal para implementar as operações: *heap*.
- Operação de substituição:
  - Retirar o menor item da fila de prioridades.
  - Colocar um novo item no seu lugar.
  - Reconstituir a propriedade do *heap*.

Ordenação do tipo
Seleção por Substituição

Geração das corridas iniciais: (conseguiremos gerar corridas iniciais maiores que m !!! usando um heap de tamanho m…) - m sendo o tamanho da memória disponível

Algoritmo:

Exemplo: m = 3

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

heap

Entra	1	2	3
E	I	N	T
R			
C			
A			
L			
A			
C			
A			
O			
B			
A			
L			
A			
N			
C			
E			
A			
D			
A			

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T

R
C
A
L
A
C
A
O
B
A
L
A
N
C
E
A
D
A

Sai I, entra E

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C			
A			
L			
A			
C			
A			
O			
B			
A			
L			
A			
N			
C			
E			
A			
D			
A			

Sai I, entra E

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A			
L			
A			
C			
A			
O			
B			
A			
L			
A			
N			
C			
E			
A			
D			
A			

Sai I, entra E
Sai N, entra R

Algoritmo:

Exemplo: m = 3

heap

Entra	1	2	3
E	*I*	*N*	*T*
R	*N*	*E**	*T*
C	*R*	*E**	*T*
A	*T*	*E**	*C**
L			
A			
C			
A			
O			
B			
A			

Sai I, entra E
Sai N, entra R
Sai R, entra C

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:

    - Marque-o como membro da próxima corrida
    - Trate-o como maior que todos os itens da corrida atual

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*

A C A O B A L A N C E A D A

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

4. Se um item marcado for para o topo da fila de prioridades então:
   - A corrida atual é encerrada
   - Uma nova corrida ordenada é iniciada

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*
A			
C			
A			
O			
B			
A			
L			
A			
N			
C			
E			
A			
D			
A			

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A

Corrida atual: INRT

Algoritmo:

Exemplo: m = 3

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

4. Se um item marcado for para o topo da fila de prioridades então:
   - A corrida atual é encerrada
   - Uma nova corrida ordenada é iniciada

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*
A	C*	E*	L*
C			
A			
O			
B			
A			
L			
A			
N			
C			
E			
A			
D			
A			

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A
Nova corrida:
sai A, entra L

Algoritmo:

Exemplo: m = 3

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

4. Se um item marcado for para o topo da fila de prioridades então:
   - A corrida atual é encerrada
   - Uma nova corrida ordenada é iniciada

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*
A	C*	E*	L*
C	E*	A	L*
A	L*	A	C
O	A	A	C
B	A	O	C
A	B	O	C
L	C	O	A*
A	L	O	A*
N	O	A*	A*
C	A*	N*	A*
E	A*	N*	C*
A	C*	N*	E*
D	E*	N*	A
A	N*	D	A
	A	D	A
	A	D	
	D		

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A
Nova corrida:
sai A, entra L
...

- Primeira passada sobre o arquivo exemplo.
- Os asteriscos indicam quais chaves pertencem a blocos diferentes.

Algoritmo:

Exemplo: m = 3

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:
   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

4. Se um item marcado for para o topo da fila de prioridades então:
   - A corrida atual é encerrada
   - Uma nova corrida ordenada é iniciada

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*
A	C*	E*	L*
C	E*	A	L*
A	L*	A	C
O	A	A	C
B	A	O	C
A	B	O	C
L	C	O	A*
A	L	O	A*
N	O	A*	A*
C	A*	N*	A*
E	A*	N*	C*
A	C*	N*	E*
D	E*	N*	A
A	N*	D	A
	A	D	A
	A	D	
	D		

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A
Nova corrida:
sai A, entra L
...

5 corridas ao invés de 8

- Primeira passada sobre o arquivo exemplo.
- Os asteriscos indicam quais chaves pertencem a blocos diferentes.

Algoritmo:

1. Inserir m elementos do arquivo original de entrada na fila de prioridades

2. Substituir o menor item da fila de prioridades pelo próximo item do arquivo

3. Se esse próximo item é menor do que o que saiu:

   - Marque-o como membro da próxima corrida
   - Trate-o como maior que todos os itens da corrida atual

Lembrando de refazer o heap!

4. Se um item marcado for para o topo da fila de prioridades então:

   - A corrida atual é encerrada
   - Uma nova corrida ordenada é iniciada

Exemplo: m = 3

heap

Entra	1	2	3
E	I	N	T
R	N	E*	T
C	R	E*	T
A	T	E*	C*
L	A*	E*	C*
A	C*	E*	L*
C	E*	A	L*
A	L*	A	C
O	A	A	C
B	A	O	C
A	B	O	C
L	C	O	A*
A	L	O	A*
N	O	A*	A*
C	A*	N*	A*
E	A*	N*	C*
A	C*	N*	E*
D	E*	N*	A
A	N*	D	A
	A	D	A
	A	D	
	D		

Sai I, entra E
Sai N, entra R
Sai R, entra C
Sai T, entra A
Nova corrida:
sai A, entra L
...

5 corridas ao invés de 8

Voltando a pensar em fitas: cada corrida gerada vai para uma das f fitas de intercalação (uma mesma fita pode conter várias corridas)

Ordenação do tipo Seleção por Substituição

fita 1:	*INRT*	*AACEN*
fita 2:	*ACEL*	*AAD*
fita 3:	*AABCLO*	
fita 4:		

heap

1	2	3
A	A	I

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da atual corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f (<= m)
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1:	*INRT*	*AACEN*
fita 2:	*ACEL*	*AAD*
fita 3:	*AABCLO*	
fita 4:	*A*	

heap

1	2	3
A	A	I
A	C	I

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1:	*INRT*	*AACEN*
fita 2:	*ACEL*	*AAD*
fita 3:	*AABCLO*	
fita 4:	*AA*	

heap

1	2	3
A	A	I
A	C	I
A	C	I

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1:	*INRT*	*AACEN*
fita 2:	*ACEL*	*AAD*
fita 3:	*AABCLO*	
fita 4:	*AAA*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAAB*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT*	*AACEN*
fita 2: *ACEL*	*AAD*
fita 3: *AABCLO*	
fita 4: *AAABC*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT*	*AACEN*
fita 2: *ACEL*	*AAD*
fita 3: *AABCLO*	
fita 4: *AAABCC*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAABCCE*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT*	*AACEN*
fita 2: *ACEL*	*AAD*
fita 3: *AABCLO*	
fita 4: *AAABCCEI*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAABCCEIL*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAABCCEILL*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou um heap de tamanho máximo f, contendo o menor elemento de cada fita precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT*	*AACEN*
fita 2: *ACEL*	*AAD*
fita 3: *AABCLO*	
fita 4: *AAABCCEILLN*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	
O	R	

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou um heap de tamanho máximo f, contendo o menor elemento de cada fita precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT*	*AACEN*
fita 2: *ACEL*	*AAD*
fita 3: *AABCLO*	
fita 4: *AAABCCEILLNO*	

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	
O	R	
R		

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou
um heap de tamanho máximo f, contendo o menor elemento de cada fita
precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAABCCEILLNOR*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	
O	R	
R		
T		

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou
um heap de tamanho máximo f, contendo o menor elemento de cada fita
precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *INRT* *AACEN*
fita 2: *ACEL* *AAD*
fita 3: *AABCLO*
fita 4: *AAABCCEILLNORT*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	
O	R	
R		
T		
-		

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou
um heap de tamanho máximo f, contendo o menor elemento de cada fita
precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *AACEN*
fita 2: *AAD*
fita 3:
fita 4: *AAABCCEILLNORT*

heap

1	2	3
A	A	I
A	C	I
A	C	I
B	C	I
C	C	I
C	I	L
E	I	L
I	L	L
L	L	N
L	N	O
N	O	
O	R	
R		
T		
-		

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou
um heap de tamanho máximo f, contendo o menor elemento de cada fita
precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *AACEN*
fita 2: *AAD*
fita 3:
fita 4: *AAABCCEILLNORT AAAACDEN*

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *AAAACDEN*
fita 2:
fita 3:
fita 4: *AAABCCEILLNORT*

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1: *AAAACDEN*
fita 2: *AAAAAAABCCCDEEILLNNORT*
fita 3:
fita 4: *AAABCCEILLNORT*

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

fita 1:
fita 2: *AAAAAAABCCCDEEILLNNORT*
fita 3:
fita 4:

Fases de Intercalação:

Algoritmo:

Ideia: Há f fitas para intercalar, cada uma com no máximo C corridas. Vou manter um heap de tamanho máximo f, contendo o menor elemento de cada fita (mais precisamente, da primeira corrida de cada fita)

Para cada conjunto de i-ésimas corridas das fitas (i = 1, …, C)

1. Monte uma fila de prioridades de tamanho f
2. Enquanto não estiver vazia:
   - Substitua o menor item k (do topo da fila de prioridades) pelo próximo item da mesma corrida da mesma fita do item k (se houver) Lembrando de refazer o heap!
   - Imprima na fita de saída o item k substituído

Ordenação do tipo Seleção por Substituição

- Vantagens:
  - número de comparações para achar o menor item (fase de intercalacão)
    - Intercalação balanceada de f caminhos: f-1
    - Seleção por substituição:
      - $\log_2 f$ (usando heap binário)
  - Corridas iniciais de tamanho médio 2*m => número menor de passadas sobre o arquivo para ordená-lo (intercalacões)

- Vantajoso quando f é grande (maior ou igual a 8)

Considerações Práticas

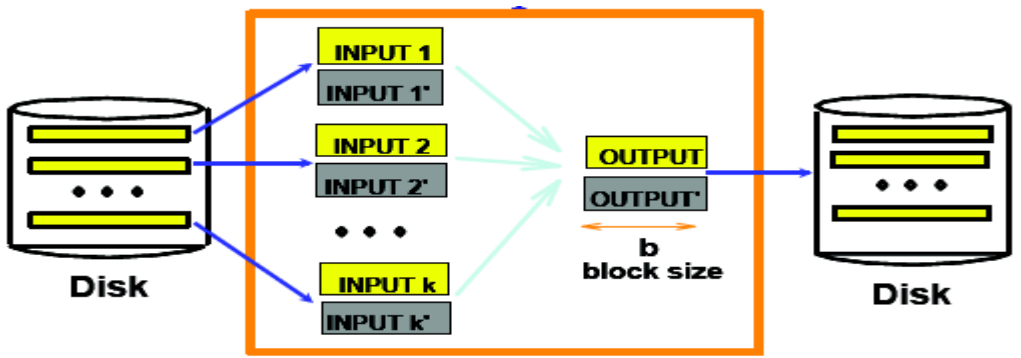
- As operações de entrada e saída de dados devem ser implementadas eficientemente.
- Deve-se procurar realizar a leitura, a escrita e o processamento interno dos dados de forma simultânea.
- Os computadores de maior porte possuem uma ou mais unidades independentes para processamento de entrada e saída.
- Assim, pode-se realizar processamento e operações de E/S simultaneamente.

Considerações Práticas

- Técnica para obter superposição de E/S e processamento interno:
  - Utilize $2f$ áreas de entrada e $2f$ de saída. (ou 2f + 2, como o desenho abaixo)
  - Para cada unidade de entrada ou saída, utiliza-se duas áreas de armazenamento:
    1. Uma para uso do processador central
    2. Outra para uso do processador de entrada ou saída.
  - Para entrada, o processador central usa uma das duas áreas enquanto a unidade de entrada está preenchendo a outra área.
  - Depois a utilização das áreas é invertida entre o processador de entrada e o processador central.
  - Para saída, a mesma técnica é utilizada.

Vantagem: ?

Desvantagem: ?



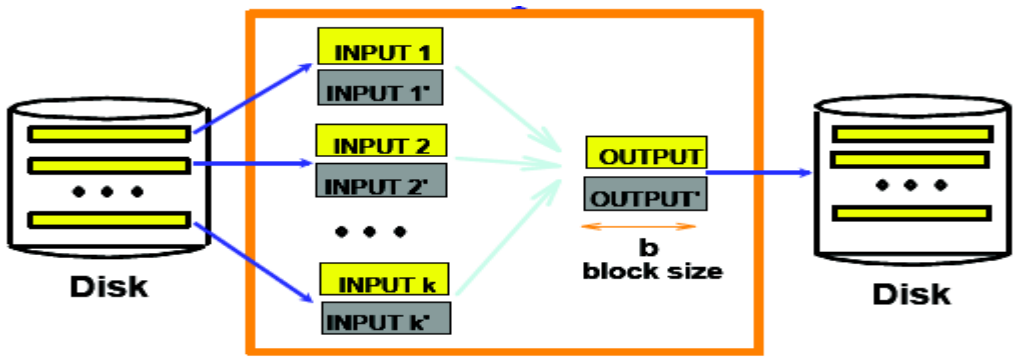
B main memory buffers, k-way merge

Considerações Práticas

- Técnica para obter superposição de E/S e processamento interno:
  - Utilize $2f$ áreas de entrada e $2f$ de saída. (ou 2f + 2, como o desenho abaixo)
  - Para cada unidade de entrada ou saída, utiliza-se duas áreas de armazenamento:
    1. Uma para uso do processador central
    2. Outra para uso do processador de entrada ou saída.
  - Para entrada, o processador central usa uma das duas áreas enquanto a unidade de entrada está preenchendo a outra área.
  - Depois a utilização das áreas é invertida entre o processador de entrada e o processador central.
  - Para saída, a mesma técnica é utilizada.

Vantagem: processa mais rápido

Desvantagem: diminui f => aumenta o nr de passos de intercalacão



B main memory buffers, k-way merge

Considerações Práticas

- Escolha da ordem de intercalação f:
  - Para fitas magnéticas:
    - f deve ser igual ao número de unidades de fita disponíveis menos um.
    - A fase de intercalação usa f fitas de entrada e uma fita de saída.
    - O número de fitas de entrada deve ser no mínimo dois.
  - Para discos magnéticos:
    - O mesmo raciocínio acima é válido.
    - O acesso seqüencial é mais eficiente.
  - Sedegwick (1988) sugere considerar f grande o suficiente para completar a ordenação em poucos passos.
  - Porém, a melhor escolha para f depende de vários parâmetros relacionados com o sistema de computação disponível.

Principais abordagens gerais de ordenação externa

- **Intercalação balanceada**
- Intercalação usando **Seleção por substituição**
- Intercalação **Polifásica**

Intercalação Polifásica

- Problema com a intercalação balanceada de vários caminhos:
  - Necessita de um grande número de fitas.
  - Faz várias leituras e escritas entre as fitas envolvidas.
  - Para uma intercalação balanceada de f caminhos são necessárias $2f$ fitas.
  - Alternativamente, pode-se copiar o arquivo quase todo de uma única fita de saída para f fitas de entrada.
  - Isso reduz o número de fitas para $f + 1$.
  - Porém, há um custo de uma cópia adicional do arquivo.
- Solução:
  - **Intercalação polifásica**. (para otimizar o uso das fitas)

não balanceada

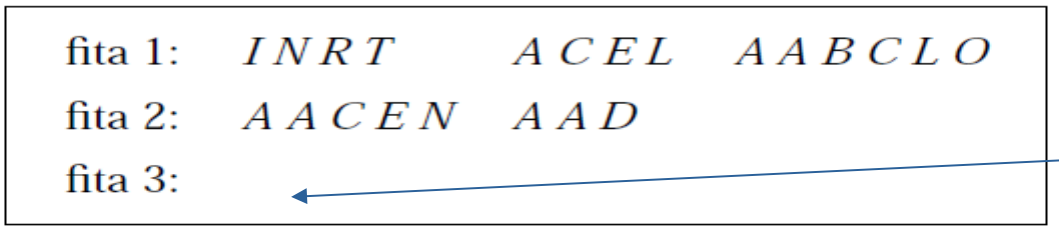
Intercalação Polifásica

- As corridas ordenadas são distribuídas de forma desigual entre as fitas disponíveis.
- Uma fita é deixada livre. (fará o papel de fita de saída)
- Em seguida, a intercalação de corridas ordenadas é executada até que uma das fitas esvazie. (que se tornará a nova fita de saída)
- Neste ponto, a antiga fita de saída troca de papel com a fita de entrada.

Isso elimina a necessidade de redistribuicão do conteúdo da fita de saída

Intercalação Polifásica

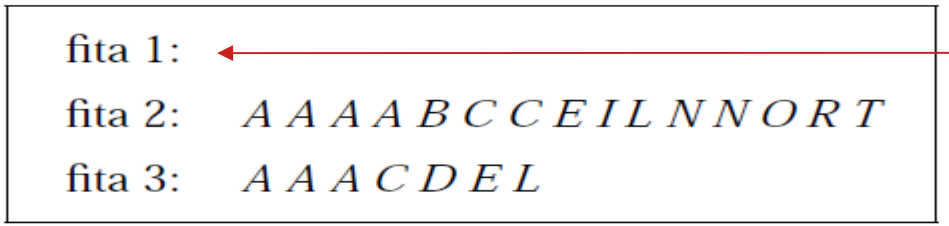
- Exemplo: (posso combinar seleção por substituicão com intercalacão polifásica)
  - Corridas ordenadas obtidas por meio de seleção por substituição:



  - Configuração após uma intercalação-de-2-caminhos das fitas 1 e 2 para a fita 3:

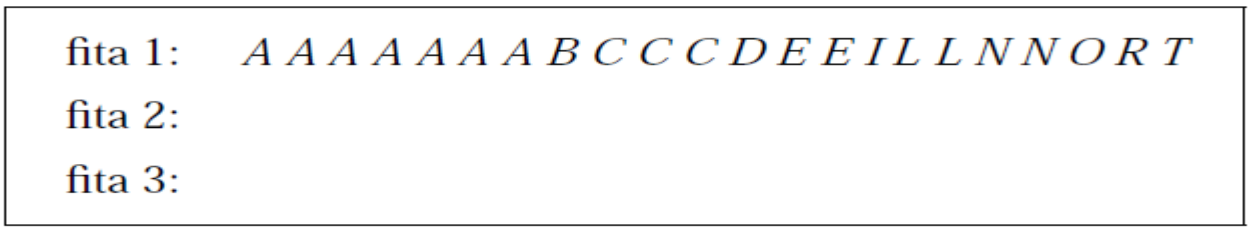
Intercalação Polifásica

- Exemplo:
  - Depois da intercalação-de-2-caminhos das fitas 1 e 3 para a fita 2:



Fita 1 ficou vazia (então se torna a nova fita de saída)

  - Finalmente:



  - A intercalação é realizada em muitas fases.
  - As fases não envolvem todas as corridas.
  - Nenhuma cópia direta entre fitas é realizada.

Intercalação Polifásica

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial das corridas ordenadas entre as fitas.
- Distribuição das corridas nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5

(fita de saída sublinhada de vermelho)

Intercalação Polifásica

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial das corridas ordenadas entre as fitas.
- Distribuição das corridas nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3

(fita de saída sublinhada de vermelho)

Intercalação Polifásica

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial das corridas ordenadas entre as fitas.
- Distribuição das corridas nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3
0	1	1	2

(fita de saída sublinhada de vermelho)

Intercalação Polifásica

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial das corridas ordenadas entre as fitas.
- Distribuição das corridas nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3
0	1	1	2
1	0	0	1

(fita de saída sublinhada de vermelho)

Intercalação Polifásica

- A implementação da intercalação polifásica é simples.
- A parte mais delicada está na distribuição inicial das corridas ordenadas entre as fitas.
- Distribuição das corridas nas diversas etapas do exemplo:

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3
0	1	1	2
1	0	0	1

(fita de saída sublinhada de vermelho)

Estratégia: planejar de baixo para cima:

- Última linha: escolha uma fita para ter a corrida final – ela recebe 1 e o restante 0
- A cada linha l, de baixo para cima:
  - Considere o maior valor v da linha l;
  - Na linha de cima (l-1), zere a fita que contém o valor v e adicione v ao valores das demais fitas
  - Contabilize o total de corridas da linha
- Se no final o número total de corridas calculado for maior que o real, considere a diferença como corridas “dummy” (que não interferem na intercalação)

fita 1	fita 2	fita 3	Total
3	2	0	5
1	0	2	3
0	1	1	2
1	0	0	1

(fita de saída sublinhada de vermelho)

Estratégia: planejar de baixo para cima:

- l ← última linha: escolha uma fita para ter a corrida final – ela recebe 1 e o restante 0; total de corridas ← 1
- Enquanto o total da linha l <= número de corridas iniciais
  - Considere o maior valor v da linha l;
  - Na linha de cima (l-1), zere a fita que contém o valor v e adicione v ao valores das demais fitas
  - Contabilize o total de corridas da linha
  - l ← l ← 1
- Se no final o número total de corridas calculado for maior que o real, considere a diferença como corridas “dummy” (que não interferem na intercalação)

Exercício

Escreva em C a função que faz essa distribuição

Parâmetros:

- N: número de corridas iniciais
- F: número de dispositivos disponíveis (como as fitas)
- D: número do dispositivo que deve conter a corrida final

Moral da História...

- Fatores importantes para uma rápida ordenação externa:
  - Crie as corridas iniciais o mais longas que for possível
  - Sobreponha processamento do algoritmo, leitura e escrita o quanto puder
  - Use o máximo de memória possível
  - Se possível, use o maior número de discos que puder para ter acesso sequencial

Referências

- Cap 4 do livro do Ziviani (parte final)
- ELMASRI, R.; NAVATHE, S. B. **Fundamentals of Database Systems**. 4 ed. Ed Pearson/Addisonn-Wesley. Seção 15.2
- RAMAKRISHNAN, R.; GEHRKE, J. **Database Management Systems**. 3 ed. Ed. McGraw-Hill. Cap 13