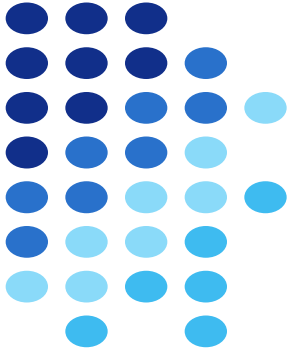
Universidade Federal de Sergipe
Departamento de Sistemas de Informação

SINF0007 – Estrutura de Dados II

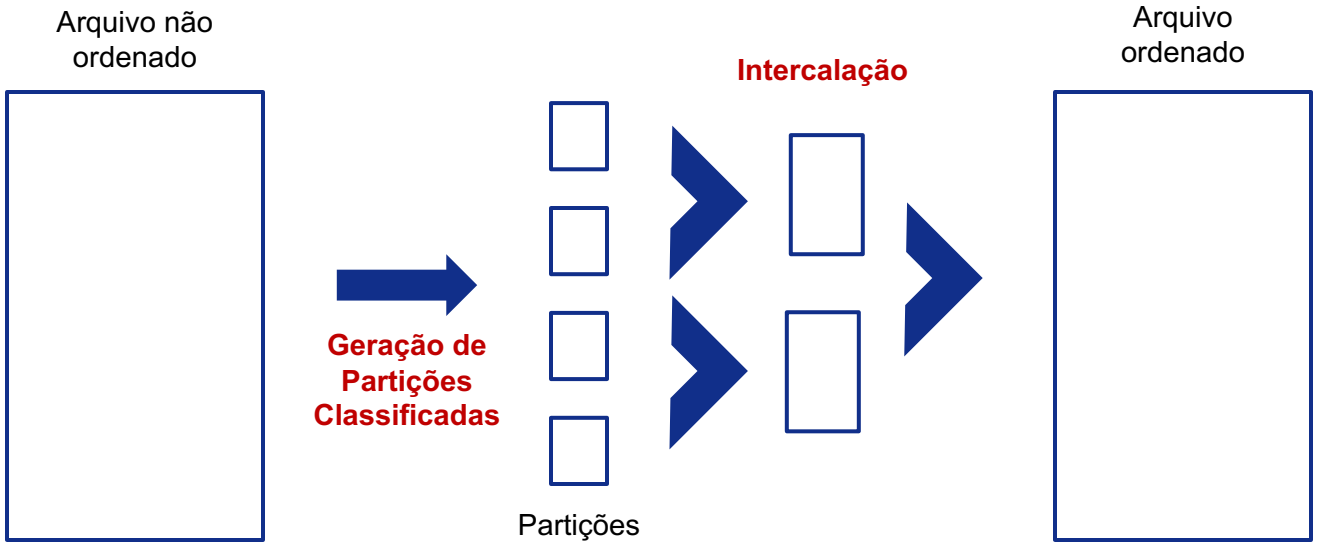
Ordenação Externa de Arquivos: Intercalação de Partições Ordenadas



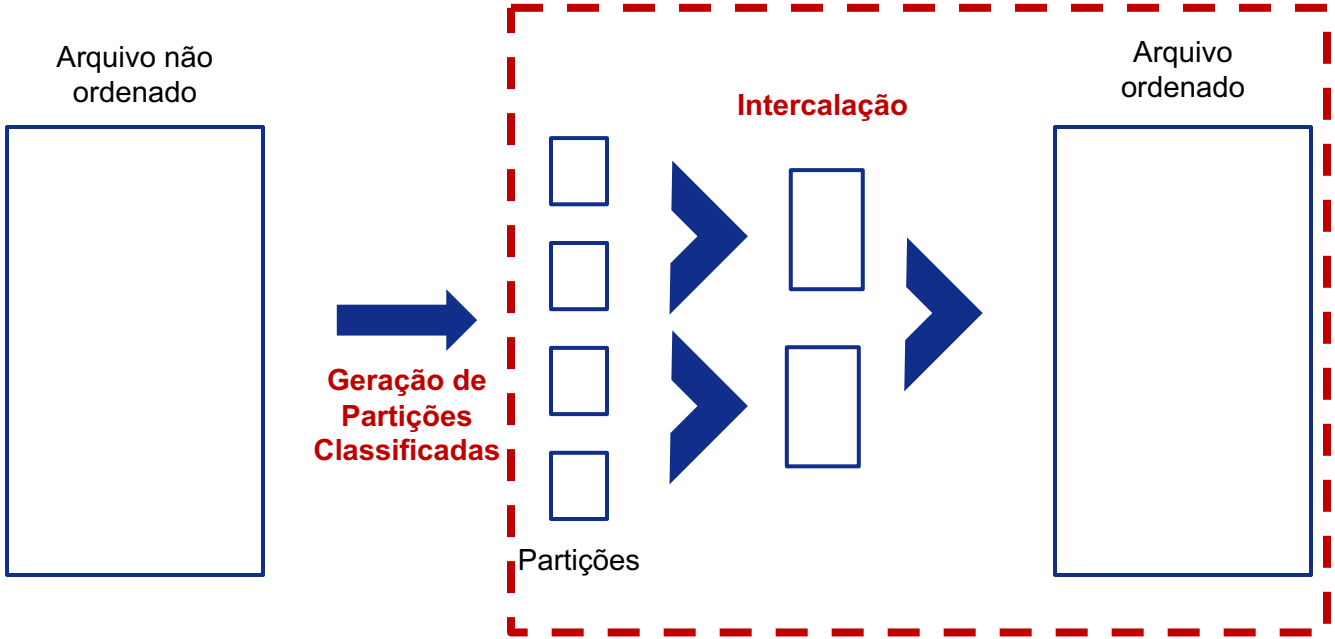
4

Prof. Dr. Raphael Pereira de Oliveira
raphael.oliveira@academico.ufs.br

Relembrando o Modelo da Classificação Externa



Nessa aula veremos: Etapa de Intercalação



Objetivo da Etapa de Intercalação

- Transformar um conjunto de partições classificadas por determinado critério, em um **único arquivo** contendo todos os registros de todas as partições originais do conjunto

- O arquivo gerado deve estar classificado pelo mesmo critério de classificação das partições iniciais

Problema

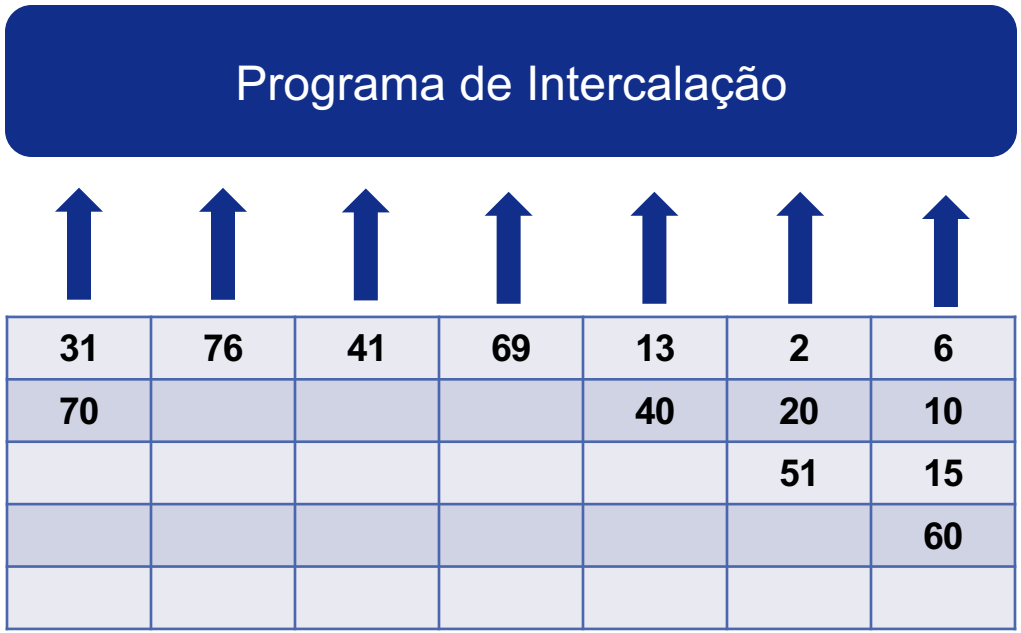
- Considere a existência de R partições geradas pelo processo de geração de partições

- Como gerar o arquivo a partir das R partições?

Algoritmo Básico

- De cada um dos arquivos a intercalar basta ter em memória um registro
- Considera-se cada arquivo como uma pilha
  - Topo da pilha: registro em memória
- Em cada iteração do algoritmo, o topo da pilha com menor chave é gravado no arquivo de saída e é substituído pelo seu sucessor
- Pilhas vazias têm topo igual a **`high value`**
- O algoritmo termina quando todos os topos da pilha tiverem **`high value`**

Esquema Básico de Intercalação



Arquivos por intercalar
(cada coluna representa um arquivo)

2
6
10
13
15
20
31
40
41
51
60
69
70
76

Número de Iterações

- A cada iteração, encontra-se a menor chave (**`O(n)`**)
  - **`n`** é o número de arquivos a intercalar
- Número de iterações = número total de registros a serem ordenados

31	76	41	69	13	2	6
70				40	20	10
					51	15
						60

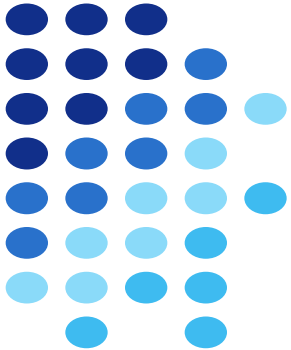
Mas...

- E se a quantidade de arquivos a intercalar for muito grande?
  - Encontrar o menor valor de chave pode ser uma tarefa custosa
  - Operação de busca da menor chave tem que ser repetida várias e várias vezes, até os arquivos terminarem

Otimização do Algoritmo

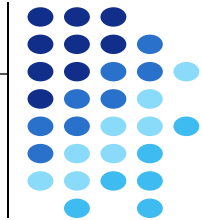
- **Árvore Binária de Vencedores**

Árvore Binária de Vencedores



Árvore Binária de Vencedores

- Nós folha representam as chaves que estão nos topos das pilhas dos arquivos a intercalar
- Cada nó interno representa o menor de seus dois filhos
- A raiz representa o menor nó da árvore



Árvore Binária de Vencedores

- Cada nó da árvore tem os seguintes componentes
  - **`vencedor`**: valor da menor chave daquela sub-árvore
  - **`endVencedor`**: ponteiro para o nó folha que tem aquela chave
  - **`f`**: variável FILE atrelada ao arquivo do vencedor
  - **`esq`**: ponteiro para o filho da esquerda
  - **`dir`**: ponteiro para o filho da direita

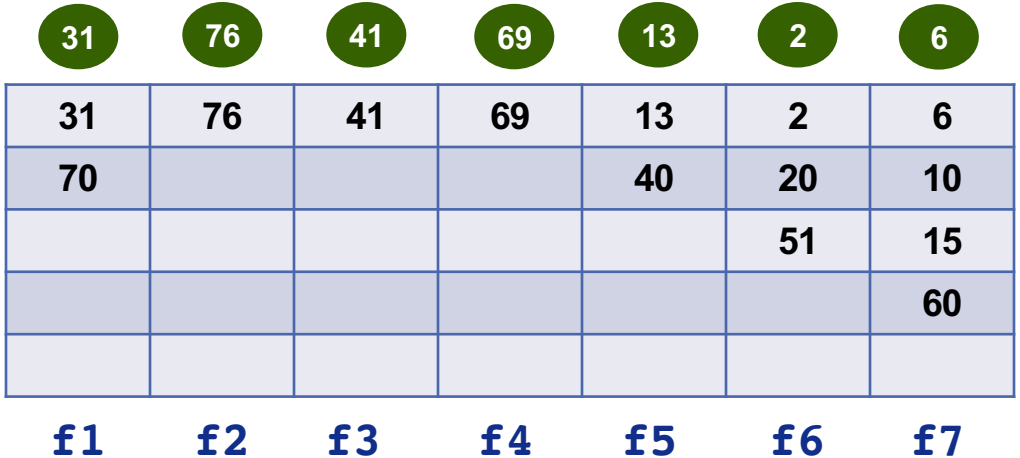
Exemplo

- Arquivos a serem ordenados
  - Cada coluna abaixo representa um arquivo com suas respectivas chaves
  - As variáveis **`FILE`** associadas a cada arquivo nesse exemplo são **`f1`**, **`f2`**, **`...`**, **`f7`**

31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
`f1`	**`f2`**	**`f3`**	**`f4`**	**`f5`**	**`f6`**	**`f7`**

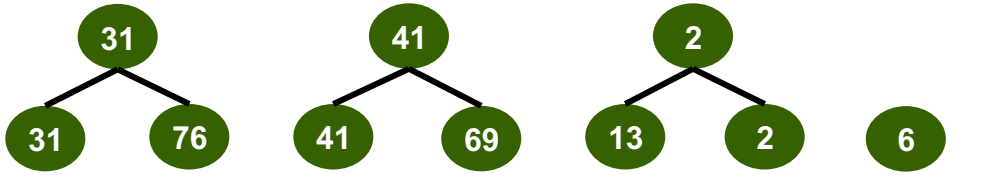
Árvore Binária de Vencedores

- Criar os nós folha da árvore contendo o primeiro registro de cada arquivo
  - Aqui usamos apenas as chaves para simplificar



Árvore Binária de Vencedores

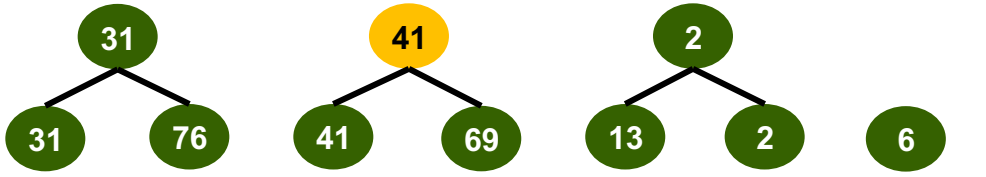
- Criar um nó raiz para cada 2 nós folha, com o menor dos dois valores



31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

Árvore Binária de Vencedores

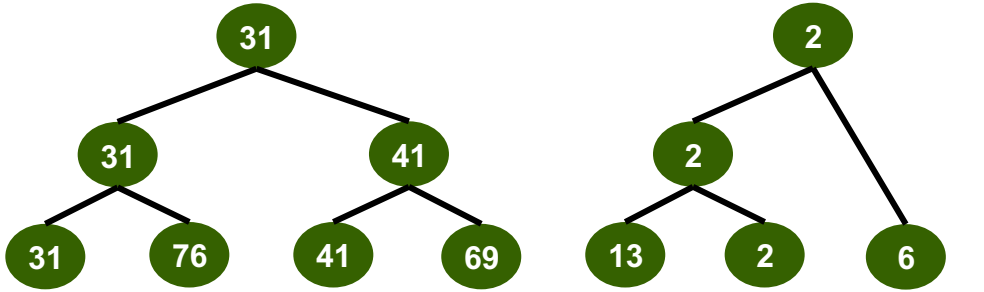
- Criar um nó raiz para cada 2 nós folha, com o menor dos dois valores
  - Representação do nó interno 41
    - **`vencedor`**: **41** (valor da menor chave daquela sub-árvore)
    - **`endVencedor`**: **valor do ponteiro da esquerda** (ponteiro para o nó folha que tem aquela chave)
    - **`f`**: **f3** (variável FILE atrelada ao arquivo do vencedor)
    - **`esq`**: **ponteiro para 41** (ponteiro para o filho da esquerda)
    - **`dir`**: **ponteiro para 69** (ponteiro para o filho da direita)



31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

Árvore Binária de Vencedores

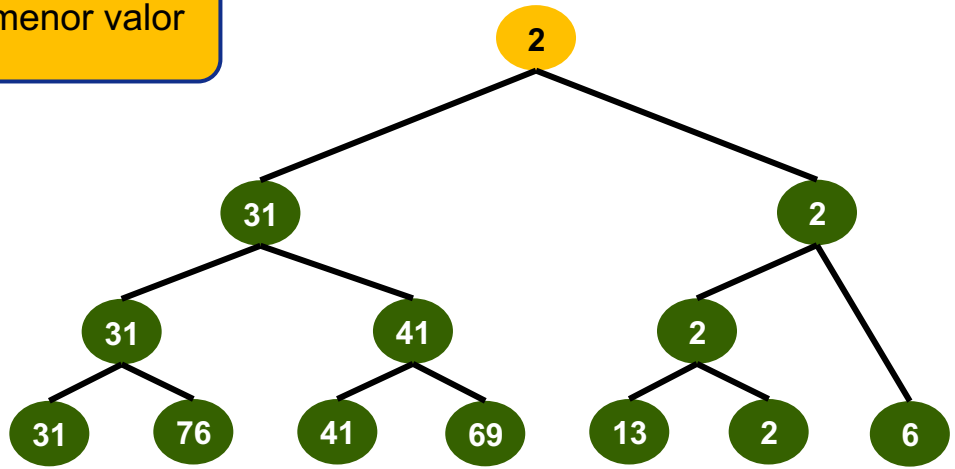
Atenção: Valores de chave se repetem em vários níveis



31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

Árvore Binária de Vencedores

Raiz tem chave de menor valor

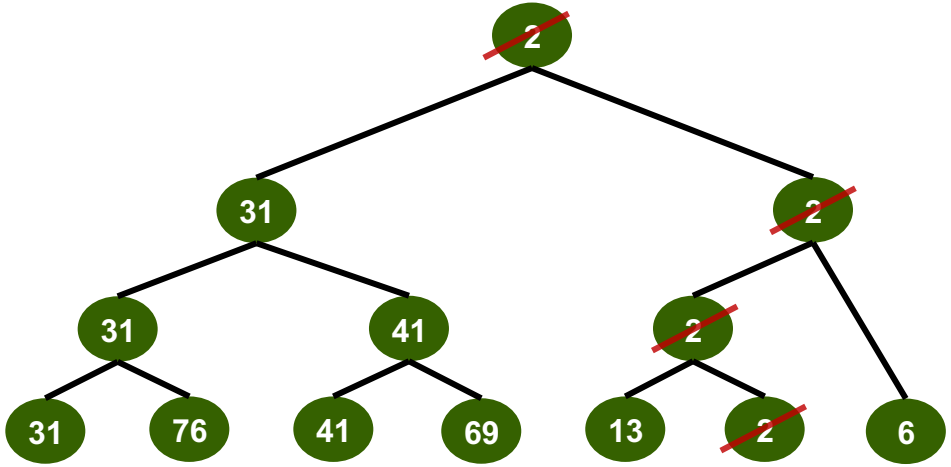


31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

Árvore Binária de Vencedores

- Como usar a árvore de vencedores no algoritmo de Intercalação?
  - A chave da raiz é retirada e o registro correspondente é inserido no arquivo de saída

Árvore Binária de Vencedores



f1	f2	f3	f4	f5	f6	f7
31	76	41	69	13	~~2~~	6
70				40	20	10
					51	15
						60

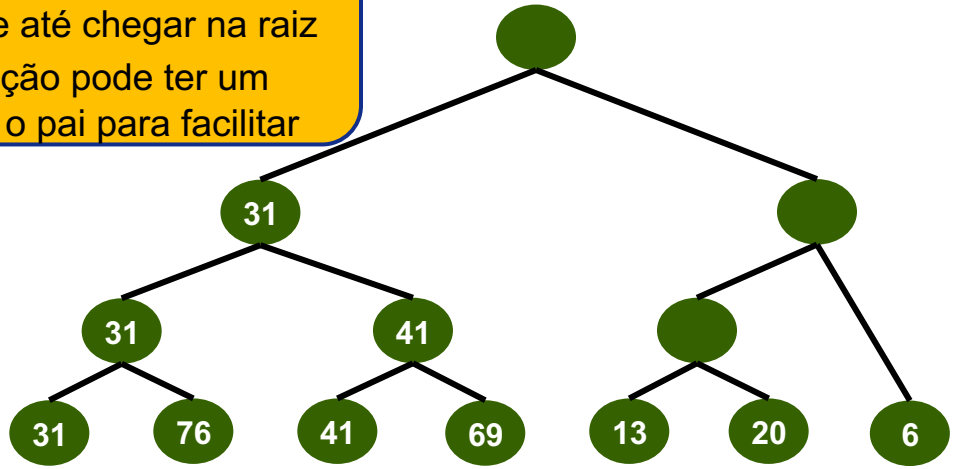
Arquivo de saída

2

Árvore Binária de Vencedores

Só é necessário comparar **20** com os nós que fazem parte daquele ramo da árvore até chegar na raiz

Implementação pode ter um ponteiro para o pai para facilitar

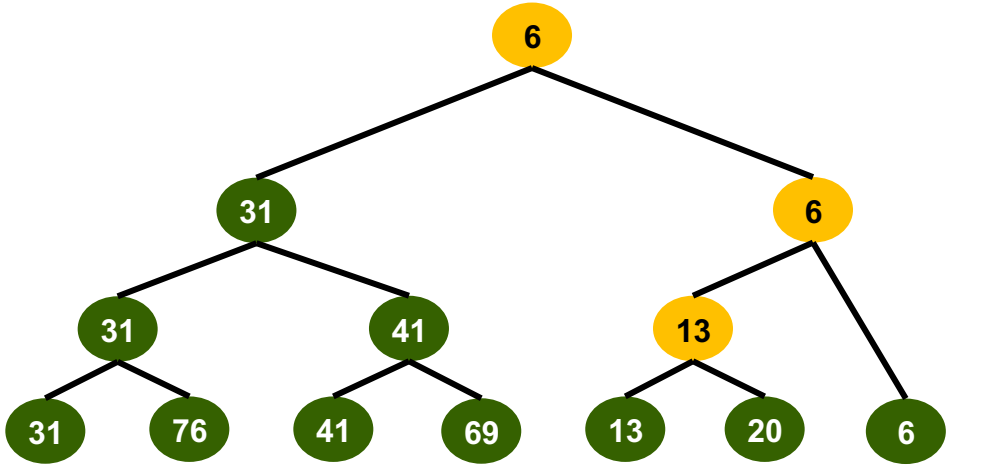


31	76	41	69	13	20	6
70				40	51	10
						15
						60
f1	f2	f3	f4	f5	f6	f7

Arquivo de saída

2

Árvore Binária de Vencedores

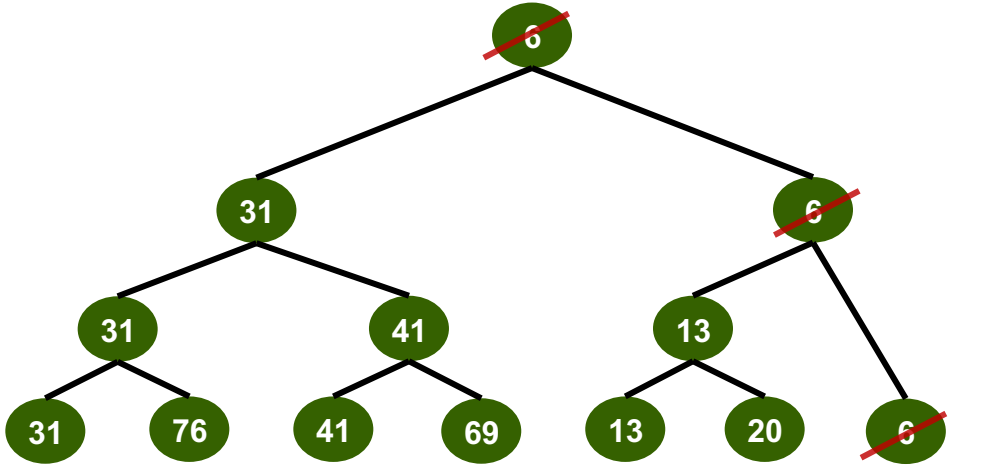


f1	f2	f3	f4	f5	f6	f7
31	76	41	69	13	20	6
70				40	51	10
						15
						60

Arquivo de saída

2

Árvore Binária de Vencedores

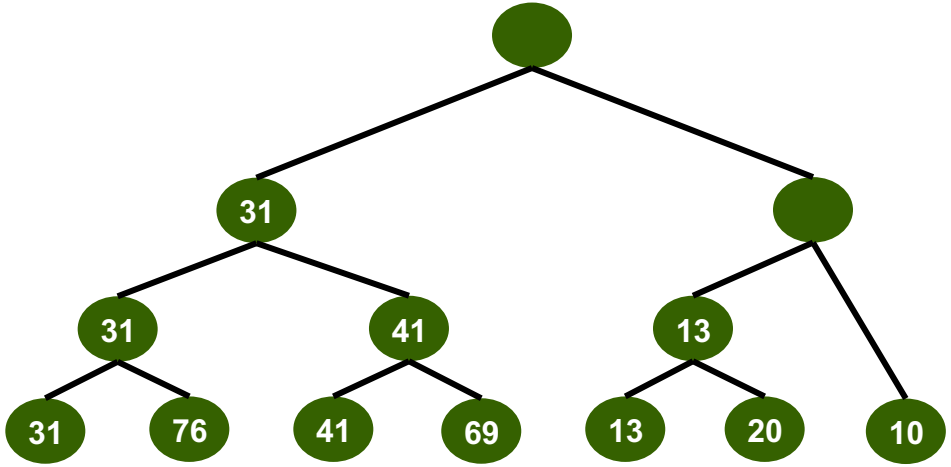


f1	f2	f3	f4	f5	f6	f7
31	76	41	69	13	20	6
70				40	51	10
						15
						60

Arquivo de saída

2
6

Árvore Binária de Vencedores

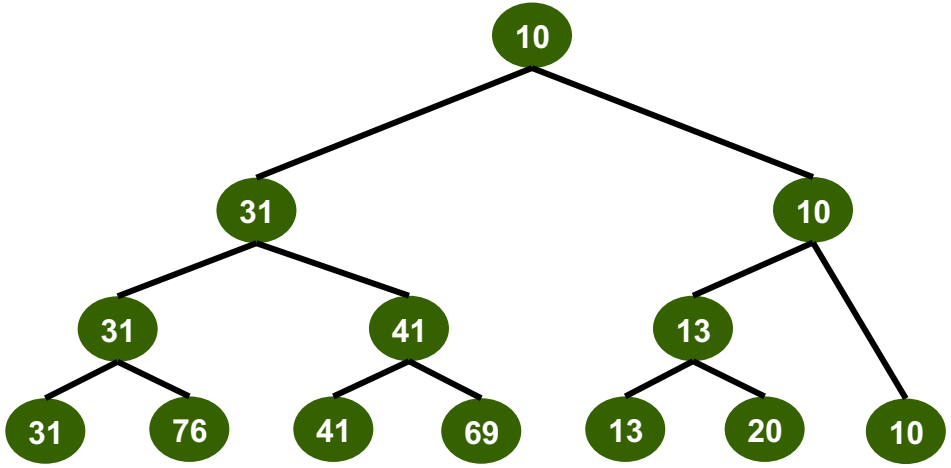


f1	f2	f3	f4	f5	f6	f7
31	76	41	69	13	20	10
70				40	51	15
						60

Arquivo de saída

2
6

Árvore Binária de Vencedores

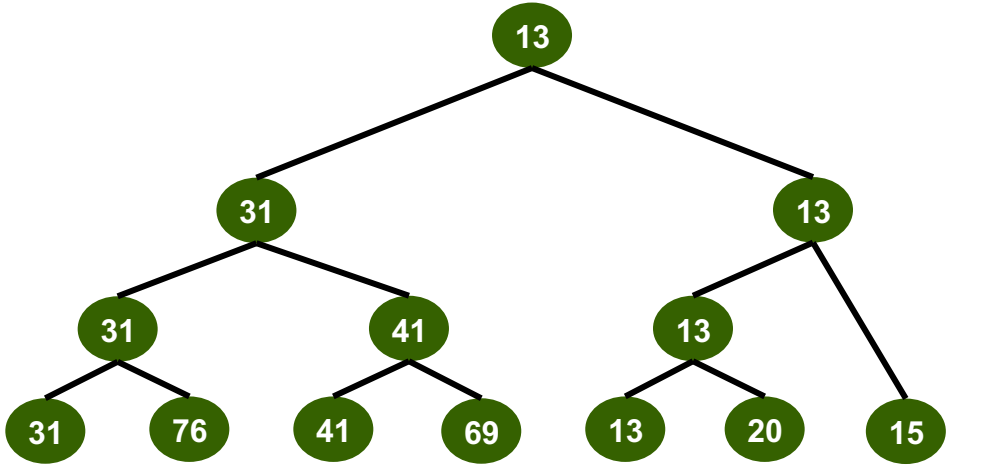


f1	f2	f3	f4	f5	f6	f7
31	76	41	69	13	20	10
70				40	51	15
						60

Arquivo de saída

2
6
10

Árvore Binária de Vencedores



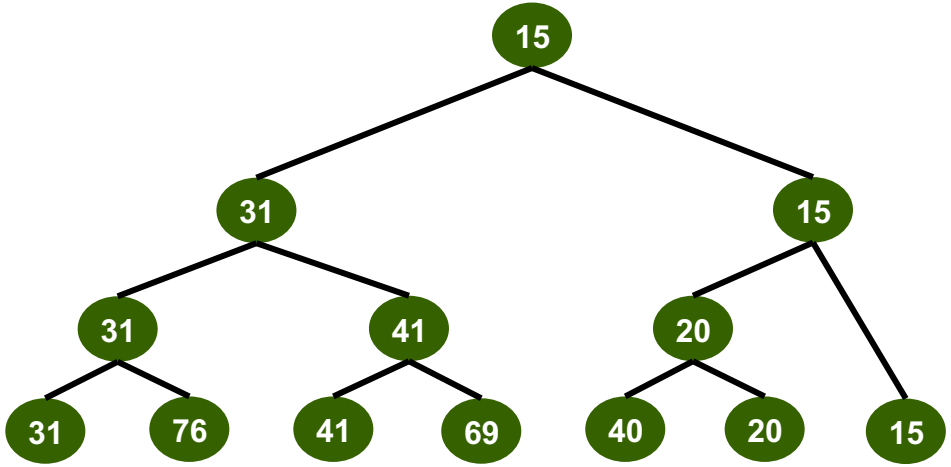
31	76	41	69	13	20	15
70				40	51	60

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13

Árvore Binária de Vencedores



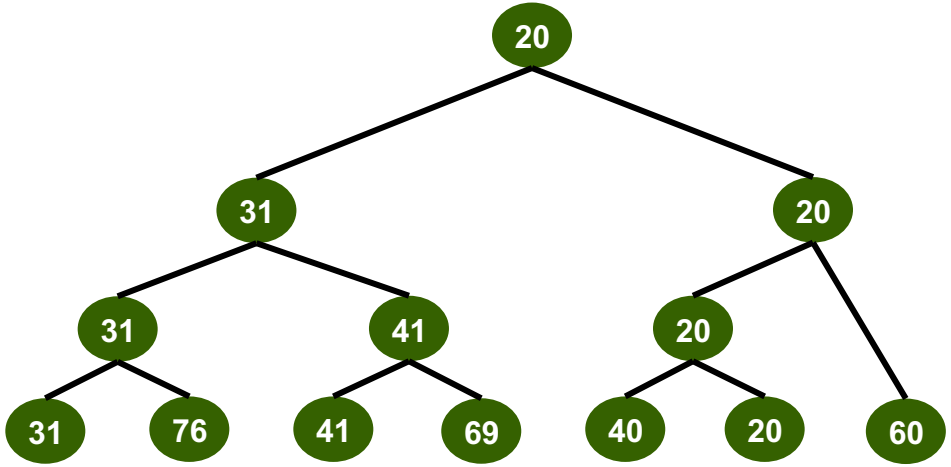
31	76	41	69	40	20	15
70					51	60

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13
15

Árvore Binária de Vencedores

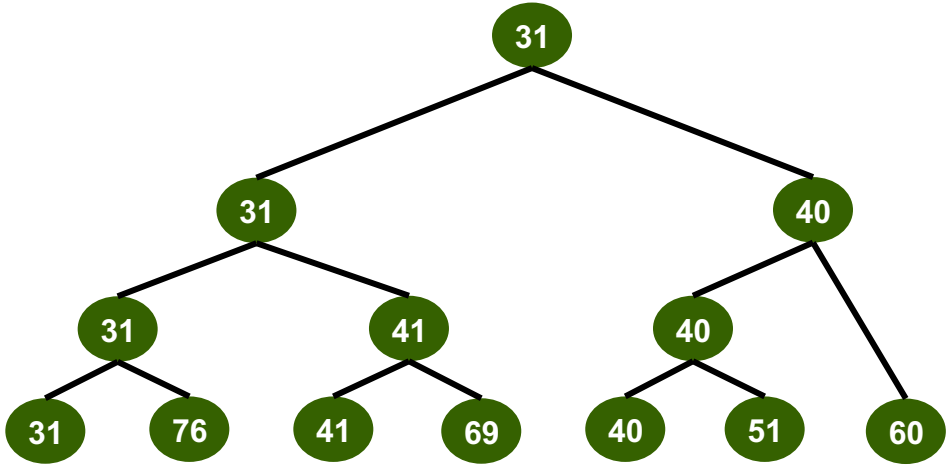


f1	f2	f3	f4	f5	f6	f7
31	76	41	69	40	20	60
70					51	

Arquivo de saída

2
6
10
13
15
20

Árvore Binária de Vencedores



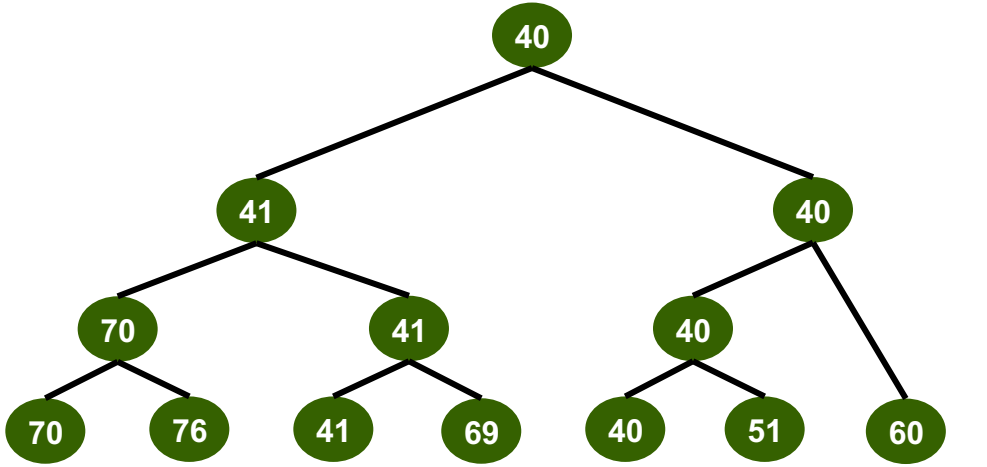
31	76	41	69	40	51	60
70						

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13
15
20
31

Árvore Binária de Vencedores



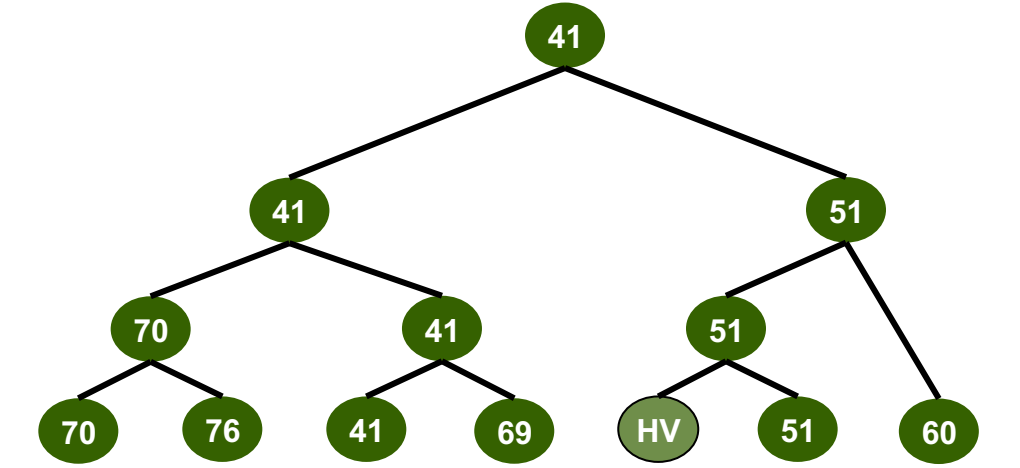
70	76	41	69	40	51	60

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13
15
20
31
40

Árvore Binária de Vencedores



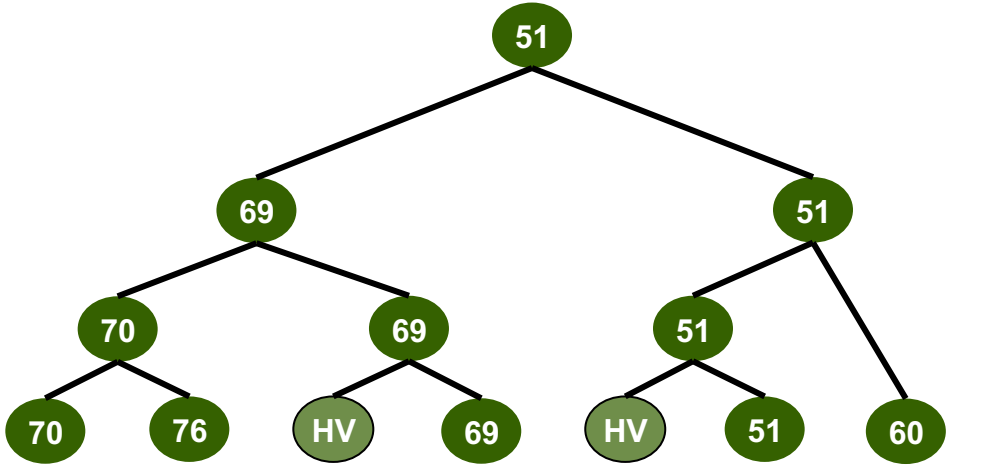
70	76	41	69		51	60

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13
15
20
31
40
41

Árvore Binária de Vencedores

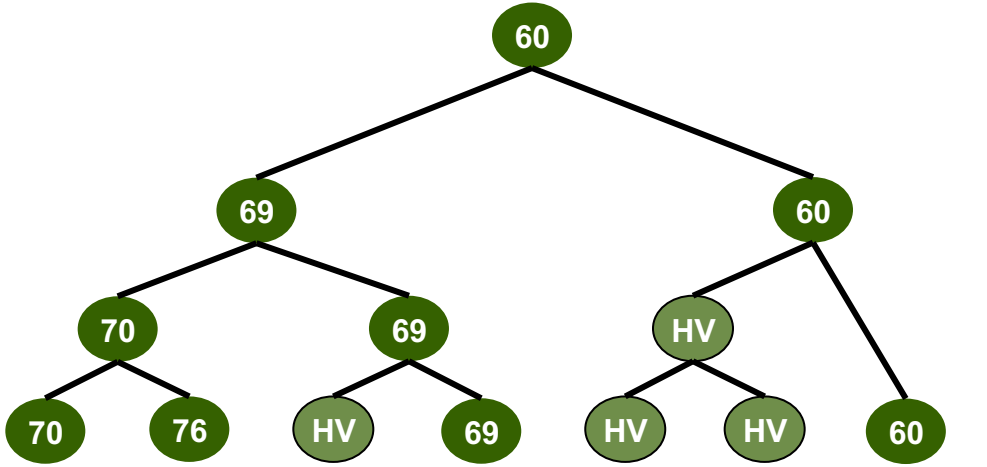


f1	f2	f3	f4	f5	f6	f7
70	76		69		51	60

Arquivo de saída

2
6
10
13
15
20
31
40
41
51

Árvore Binária de Vencedores

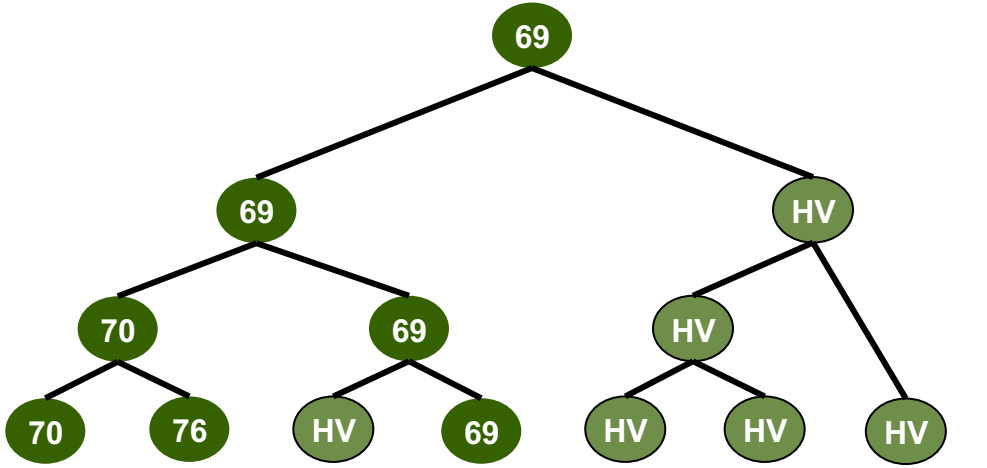


f1	f2	f3	f4	f5	f6	f7
70	76		69			60

Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60

Árvore Binária de Vencedores

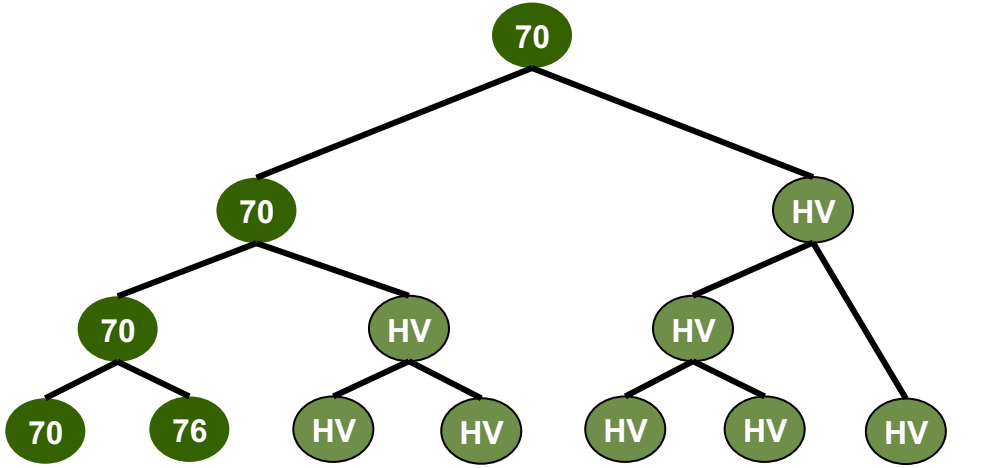


f1	f2	f3	f4	f5	f6	f7
70	76		69			

Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60
69

Árvore Binária de Vencedores

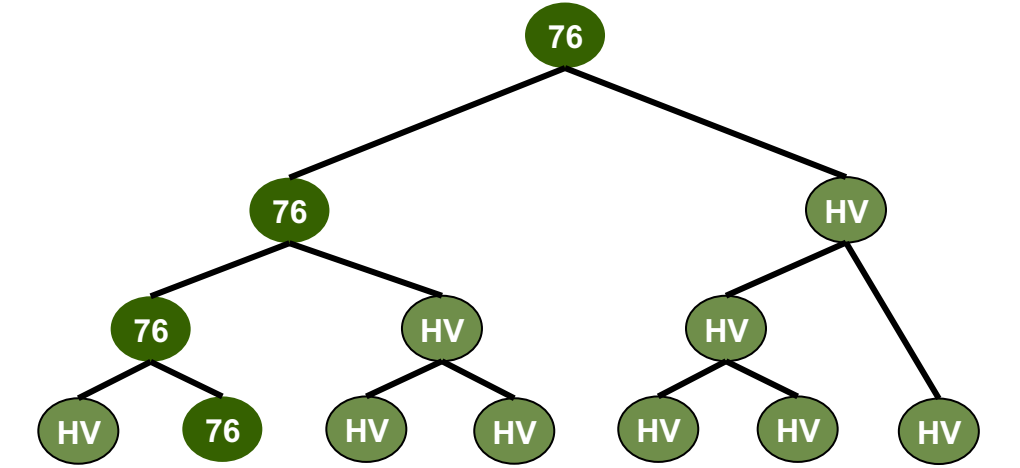


f1	f2	f3	f4	f5	f6	f7
70	76					

Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60
69
70

Árvore Binária de Vencedores



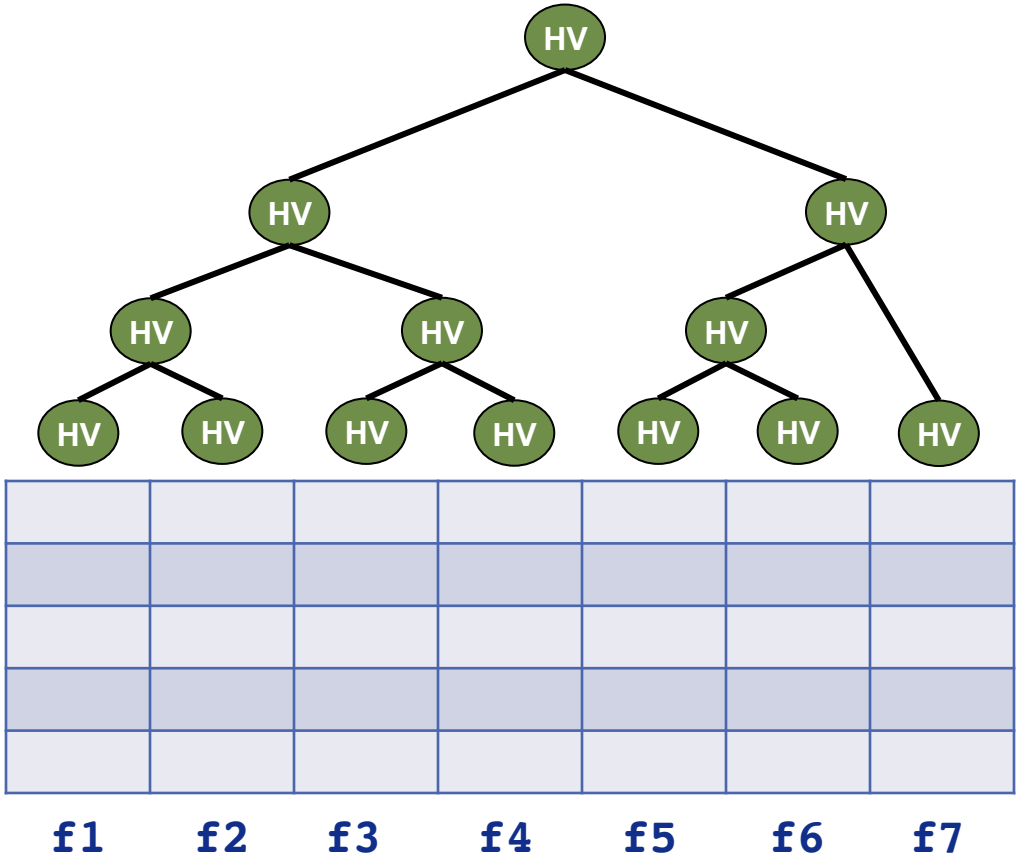
	76					

f1 f2 f3 f4 f5 f6 f7

Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60
69
70
76

Árvore Binária de Vencedores



Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60
69
70
76

Discussão

- Montagem da árvore: **`O(n)`**
- A cada iteração, faz-se **`log n`** comparações (**`n`** é o número de arquivos a comparar)
- Número de iterações: número total de registros a serem ordenados

Exercício

- Montar a árvore de vencedores para a seguinte situação

2	55	40	3	13	7	12	6	45	43	15
70			67	41	21	17		49	57	16
79			80			82				23
98										25
f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11

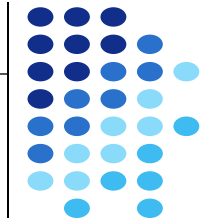
Problema na Intercalação

Seria ideal poder intercalar todas as partições de uma só vez e obter o arquivo classificado, utilizando, por exemplo, a árvore de vencedores, mas:

(i) O número de arquivos a intercalar pode gerar uma árvore de vencedores **maior do que a capacidade da memória**

(ii) Sistemas Operacionais estabelecem número máximo de arquivos abertos simultaneamente

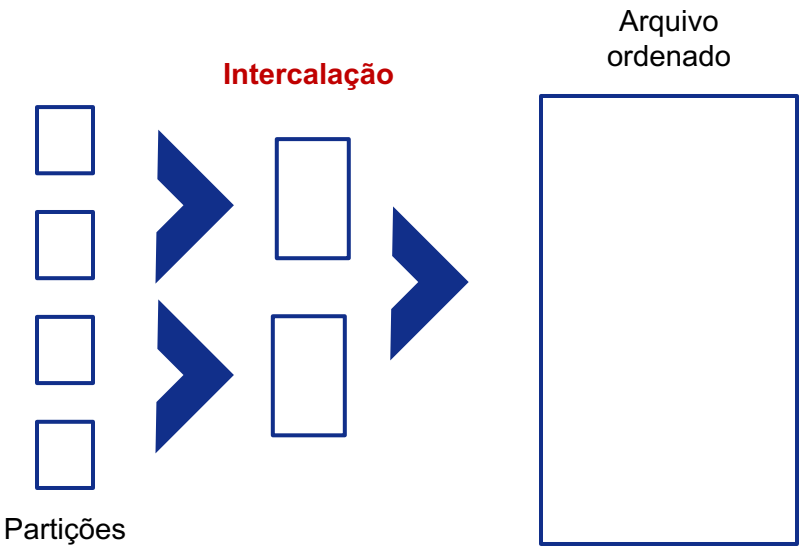
- Esse número pode ser bem menor do que o número de partições existentes
- Curiosidade: ver o número máximo de arquivos que podem ser abertos no `linux`:
  - **`ulimit –Hn`**
- Mesmo em sistemas onde não há esse limite (`MacOS`, por exemplo), a barreira é a memória disponível para armazenar os descritores dos arquivos (variáveis **`FILE`**)

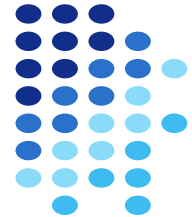


Solução de Contorno

A intercalação vai exigir uma série de fases durante as quais registros são lidos de um conjunto de arquivos e gravados em outro (partições)

Exemplo

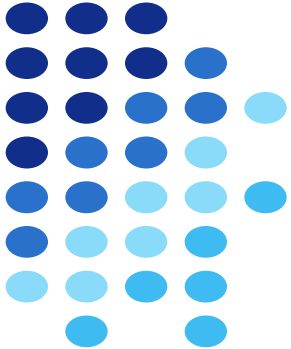




Algoritmo que Resolve o Problema

- **Intercalação Ótima**

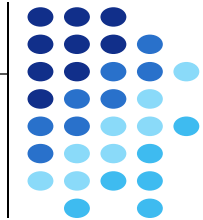
Intercalação Ótima



Intercalação Balanceada de N Caminhos

- **Primeiro passo**: determinar o número de arquivos **F** que o algoritmo irá manipular
  - **F – 1** arquivos serão usados para leitura (entrada)
  - **1** arquivo será usado para escrita (saída)

Exemplo com F = 4

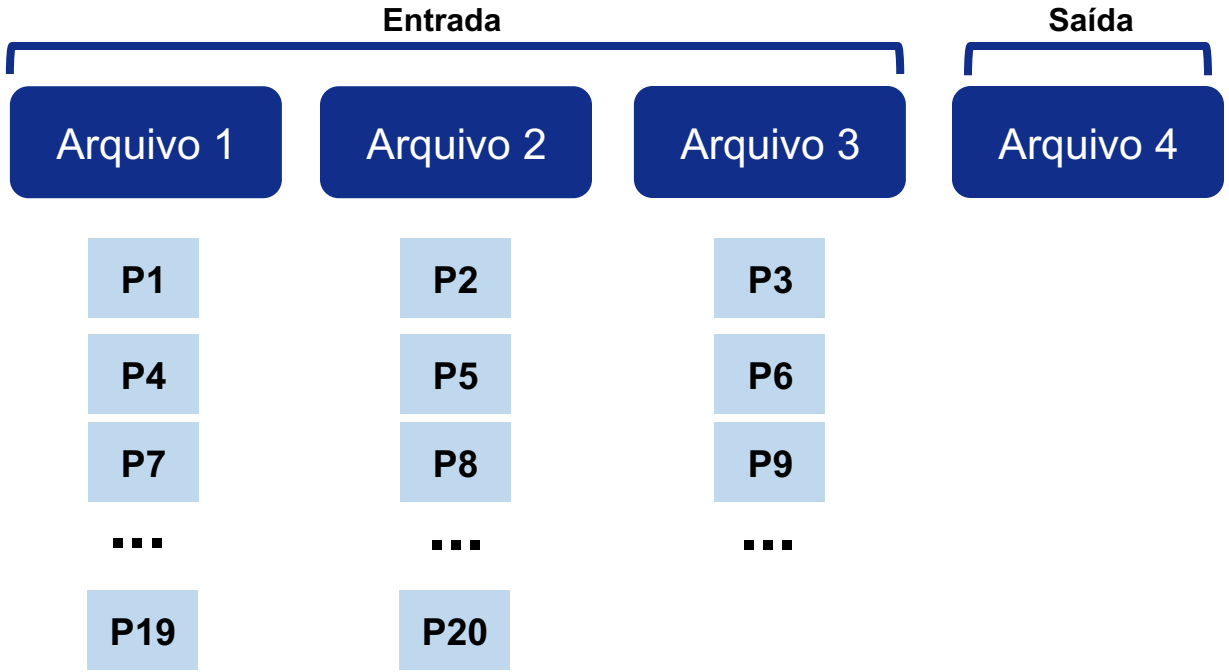


Intercalação Ótima

- Durante cada fase do algoritmo, **F – 1** partições são intercaladas e gravadas no arquivo de saída

Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



Intercalação Ótima

- Do conjunto inicial de partições removem-se as partições intercaladas e a ele agrega-se a partição gerada na intercalação

- Algoritmo termina quando este conjunto tiver apenas uma partição

Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)

Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)

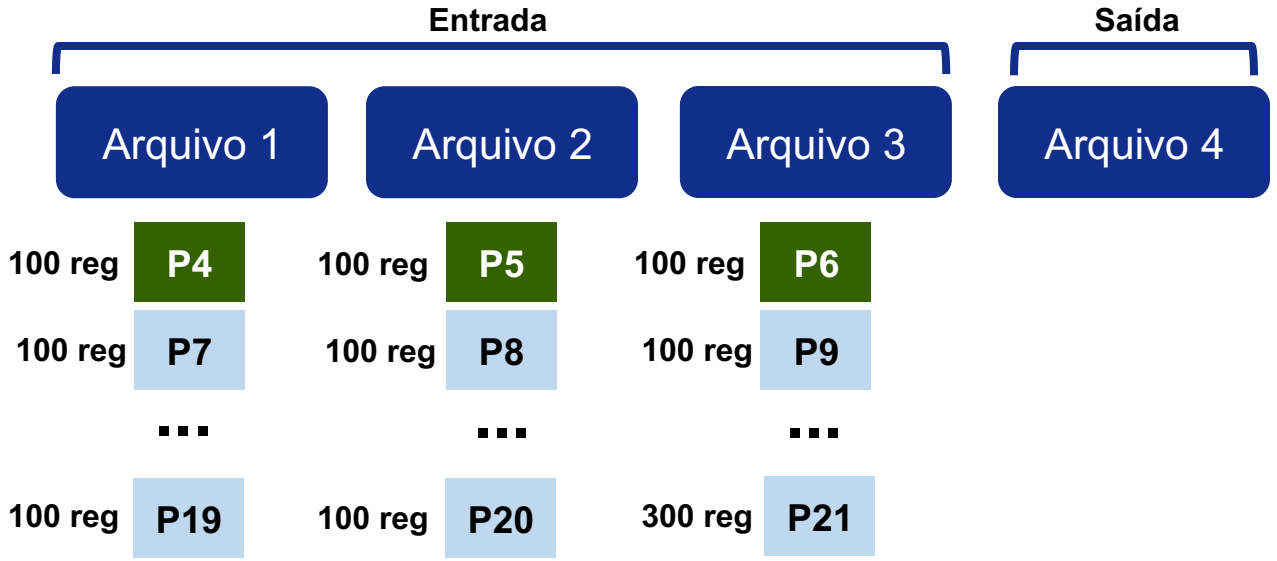
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)

100 reg P4	100 reg P5	100 reg P6
100 reg P7	100 reg P8	100 reg P9
...	...	...
100 reg P19	100 reg P20	300 reg P21

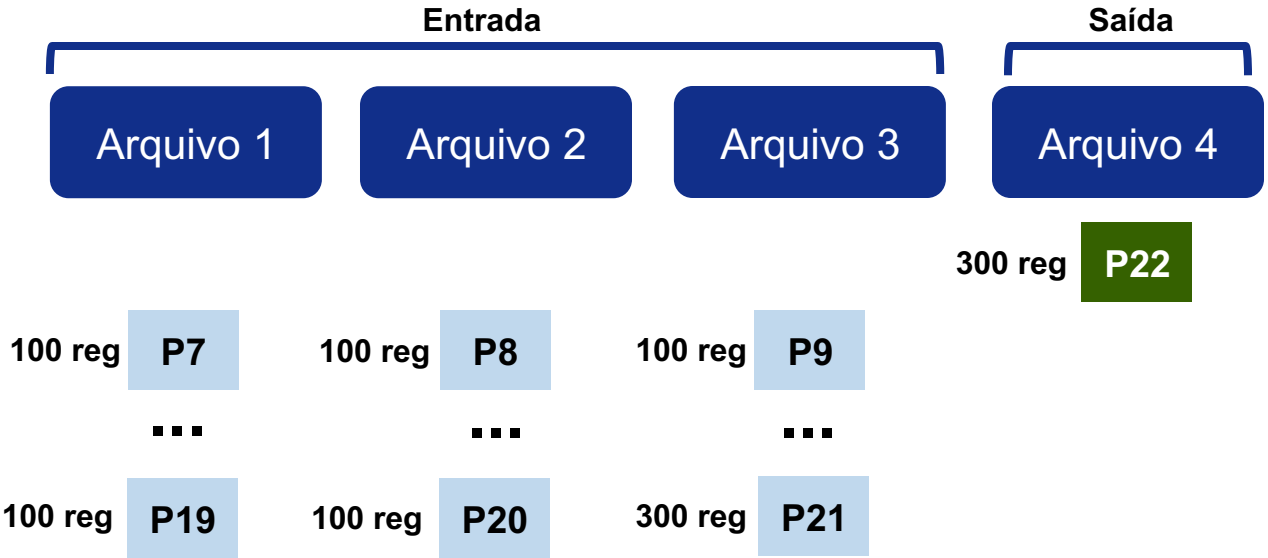
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



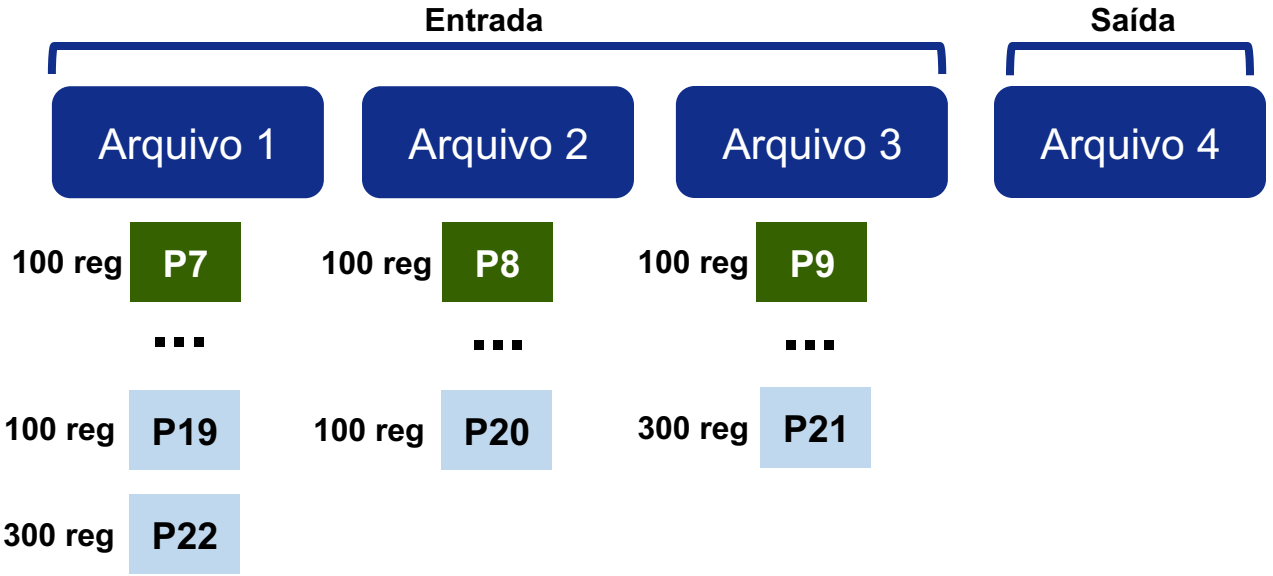
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)

100 reg P7	100 reg P8	100 reg P9
...	...	...
100 reg P19	100 reg P20	300 reg P21
300 reg P22		

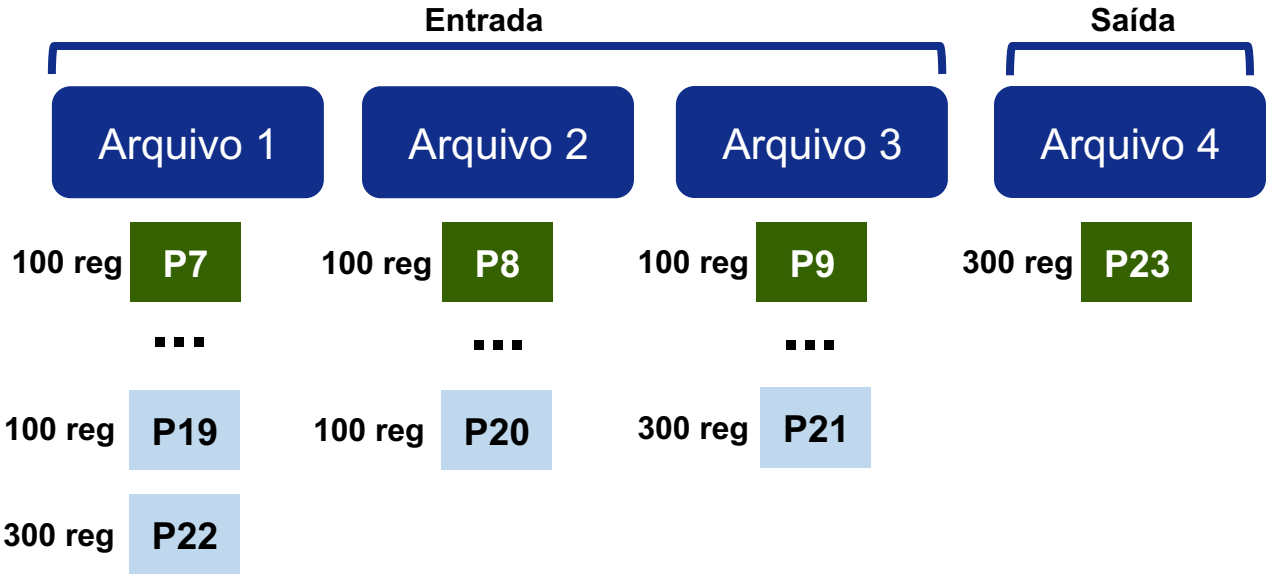
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



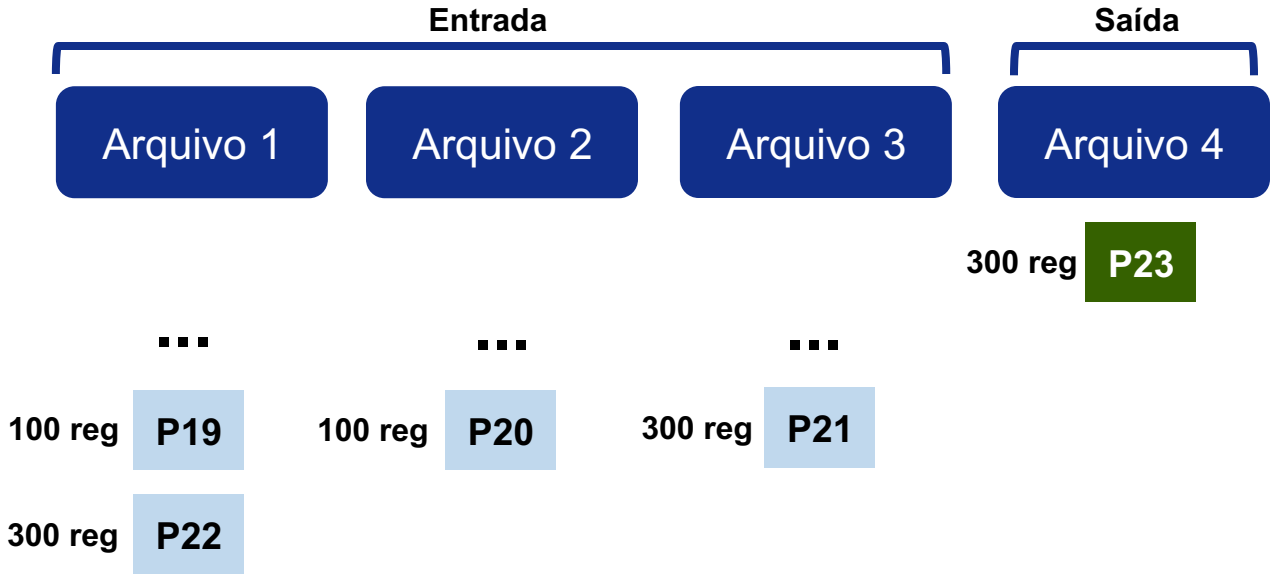
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



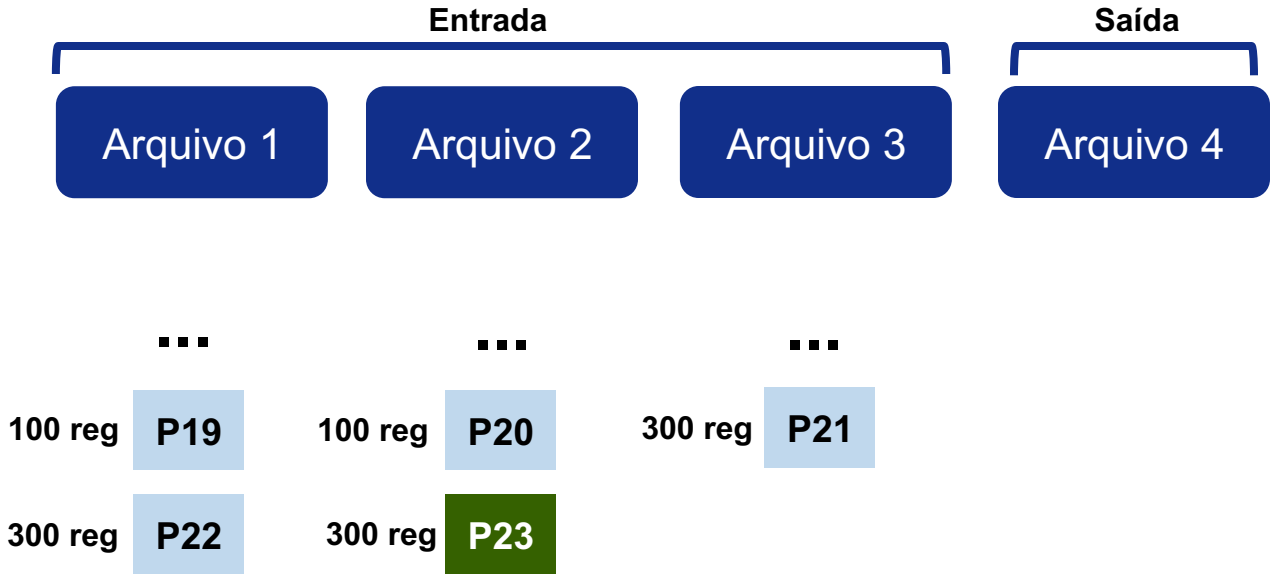
Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



Exemplo com F = 4

- Intercalar 20 partições com 100 registros cada (2000 no total)



Resumindo...

Fase	Arquivo1	Arquivo2	Arquivo3	Arquivo4	N°. de leituras
1	1:100	2:100	3:100	21:300	300
2	4:100	5:100	6:100	22:300	300
3	7:100	8:100	9:100	23:300	300
4	10:100	11:100	12:100	24:300	300
5	13:100	14:100	15:100	25:300	300
6	16:100	17:100	18:100	26:300	300
7	19:100	20:100	21:300	27:500	500
8	22:300	23:300	24:300	28:900	900
9	25:300	26:300	27:500	29:1100	1100
10	28:900	29:1100	----------	30:2000	2000
				TOTAL	6300

Medida de Eficiência

- Uma medida de eficiência do estágio de intercalação é dada pelo **número de passos** sobre os dados:

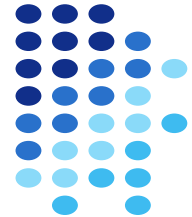
$$\textit{Número de Passos} = \frac{\textit{Número total de registros lidos}}{\textit{Número total de registros no arquivo classificado}}$$

- Número de passos representa o **número médio de vezes que um registro é lido** (ou gravado) durante o estágio de intercalação

Resumindo...

Fase	Arquivo1	Arquivo2	Arquivo3	Arquivo4	N°. de leituras
1	1:100	2:100	3:100	21:300	300
2	4:100	5:100	6:100	22:300	300
3	7:100	8:100	9:100	23:300	300
4	10:100	11:100	12:100	24:300	300
5	13:100	14:100	15:100	25:300	300
6	16:100	17:100	18:100	26:300	300
7	19:100	20:100	21:300	27:500	500
8	22:300	23:300	24:300	28:900	900
9	25:300	26:300	27:500	29:1100	1100
10	28:900	29:1100	----------	30:2000	2000
				TOTAL	6300

$$\textit{Número de Passos} = \frac{6300}{2000} = 3,15$$



Implementação

1. Criar uma lista com os nomes dos arquivos a intercalar

2. Enquanto houver mais de 1 arquivo na lista

   i. Retirar os **F-1** primeiros itens da lista e intercalá-los

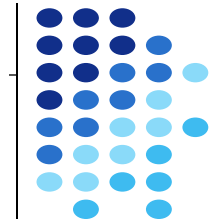
   ii. Colocar o arquivo resultante no final da lista

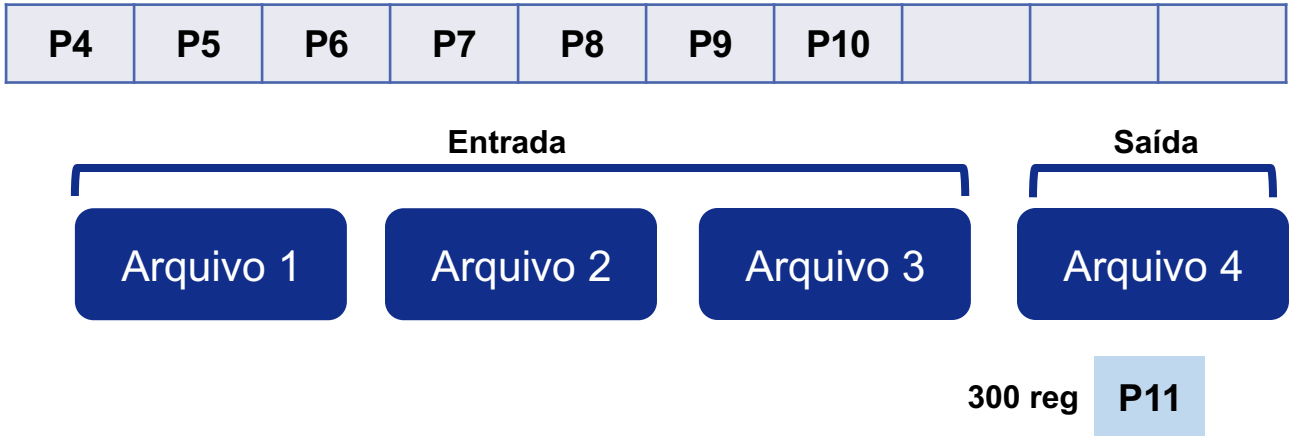
3. O arquivo que sobrar na lista será o arquivo resultante (arquivo completo que contém todos os registros)

Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições

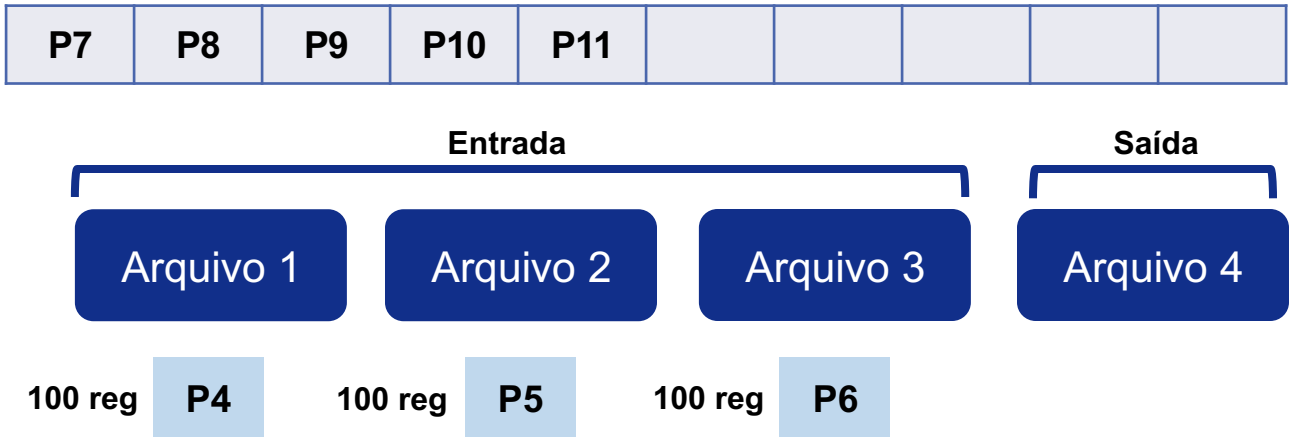
Exemplo com F = 4 e 10 Partições



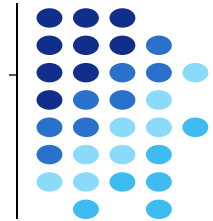


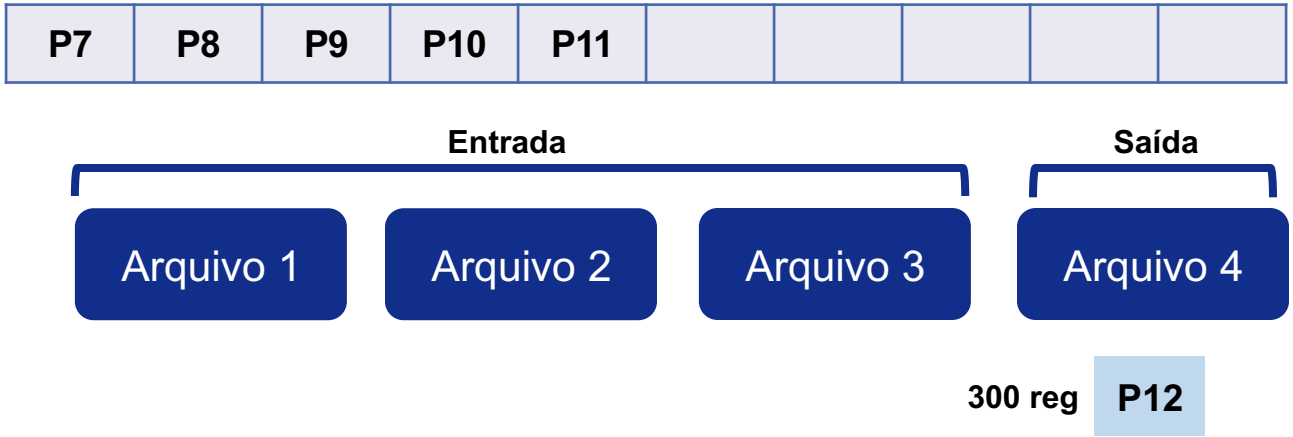
Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições



Exemplo com F = 4 e 10 Partições





Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições

Entrada

Saída

Arquivo 1 | Arquivo 2 | Arquivo 3 | Arquivo 4

100 reg P7 | 100 reg P8 | 100 reg P9 | 300 reg P13

Exemplo com F = 4 e 10 Partições

Entrada

Saída

Arquivo 1

Arquivo 2

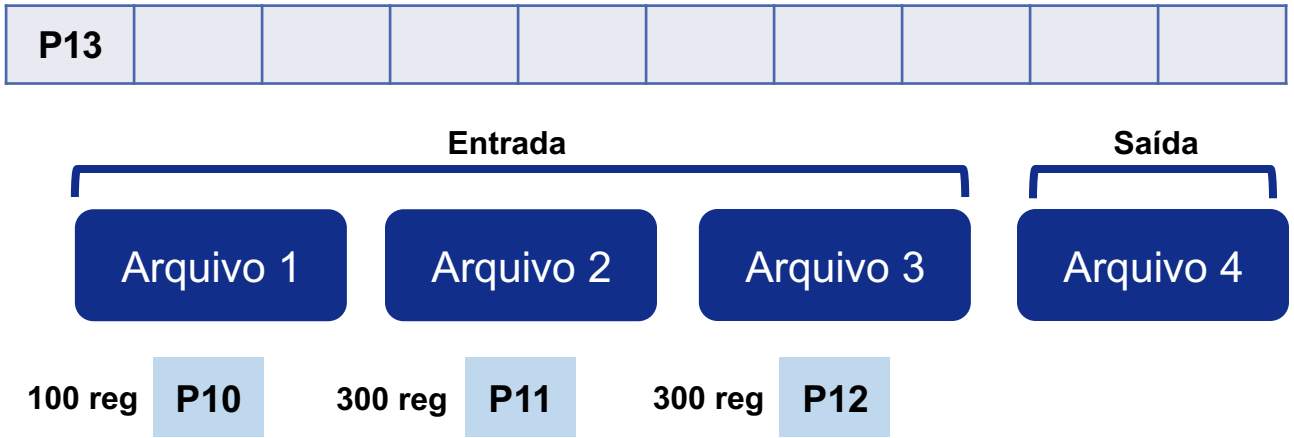
Arquivo 3

Arquivo 4

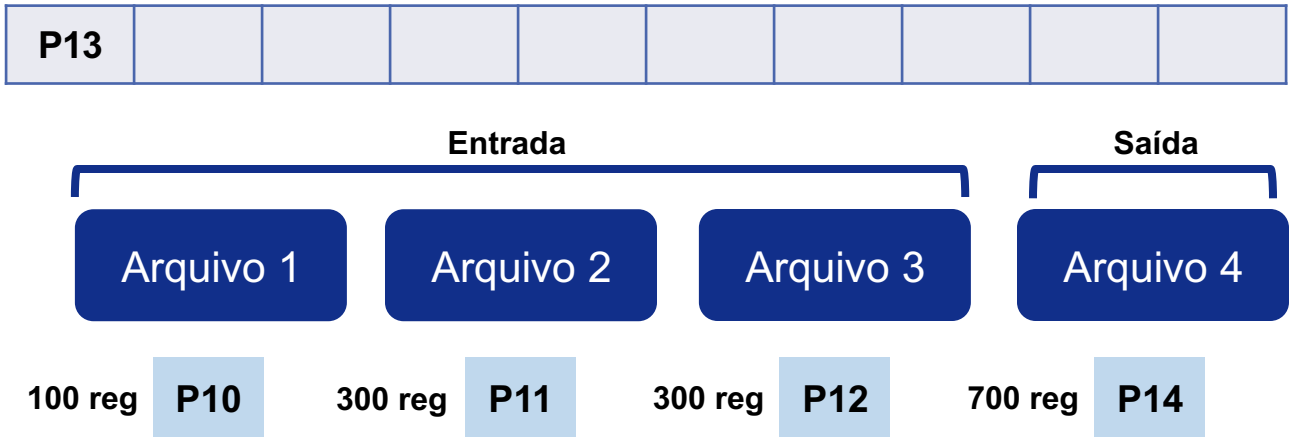
300 reg **P13**

Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições



Exemplo com F = 4 e 10 Partições



Exemplo com F = 4 e 10 Partições

700 reg **P14**

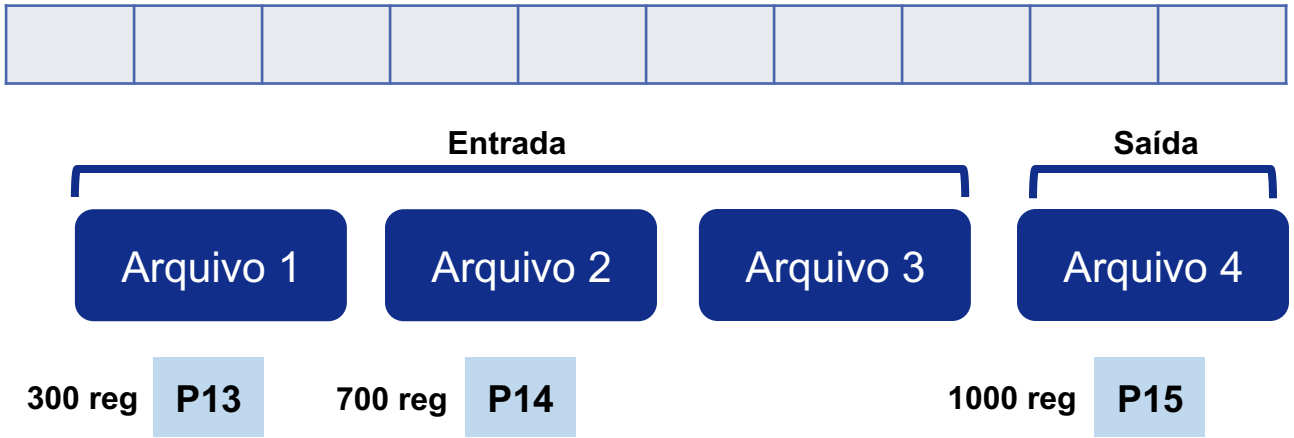
Exemplo com F = 4 e 10 Partições

Exemplo com F = 4 e 10 Partições

300 reg **P13**

700 reg **P14**

Exemplo com F = 4 e 10 Partições



Exemplo com F = 4 e 10 Partições

1000 reg **P15**

Exemplo com F = 4 e 10 Partições

DICA: Agora basta renomear a partição P15 para o nome do arquivo de saída desejado

Referências

- Material baseado nos slides de **Vanessa Braganholo**, Disciplina de Estruturas de Dados e Seus Algoritmos. Instituto de Computação. Universidade Federal Fluminense (UFF), Niterói, Brasil.
- Inhaúma Neves Ferraz. Programação Com Arquivos. 2003. Editora: manole.
- Schildt, H. C Completo e Total. Ed. McGraw-Hill.