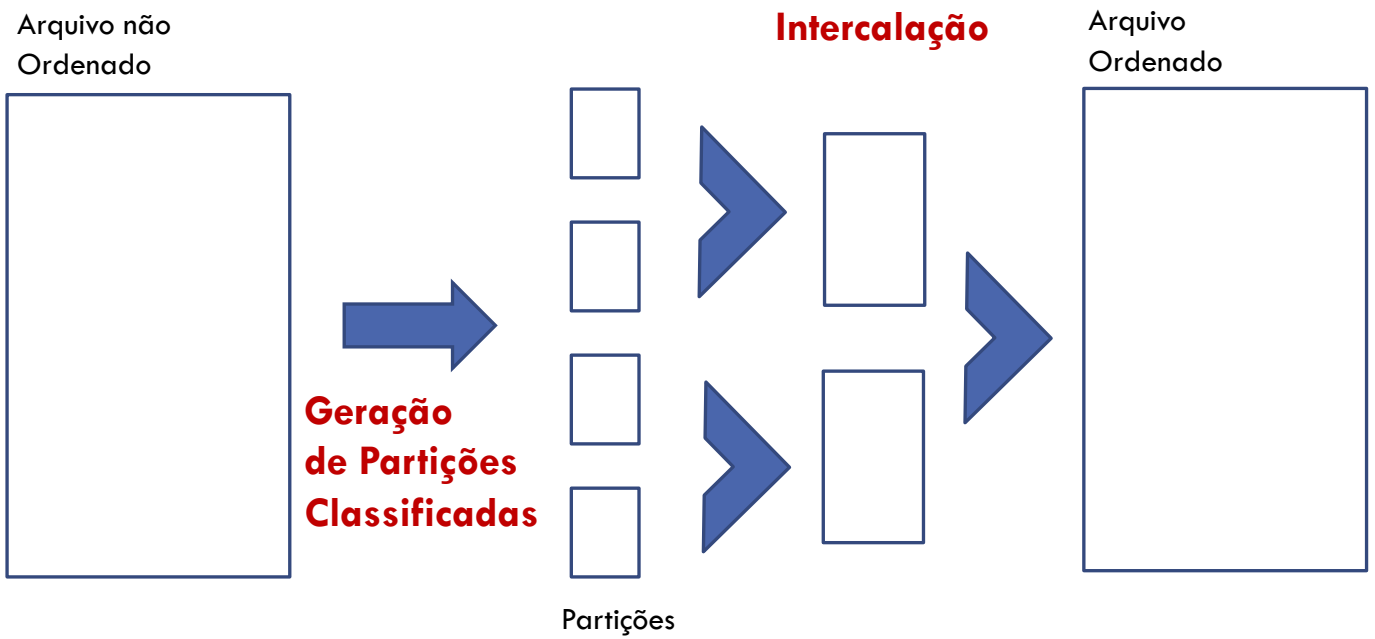
ORDENAÇÃO EXTERNA DE ARQUIVOS: INTERCALAÇÃO DE PARTIÇÕES ORDENADAS

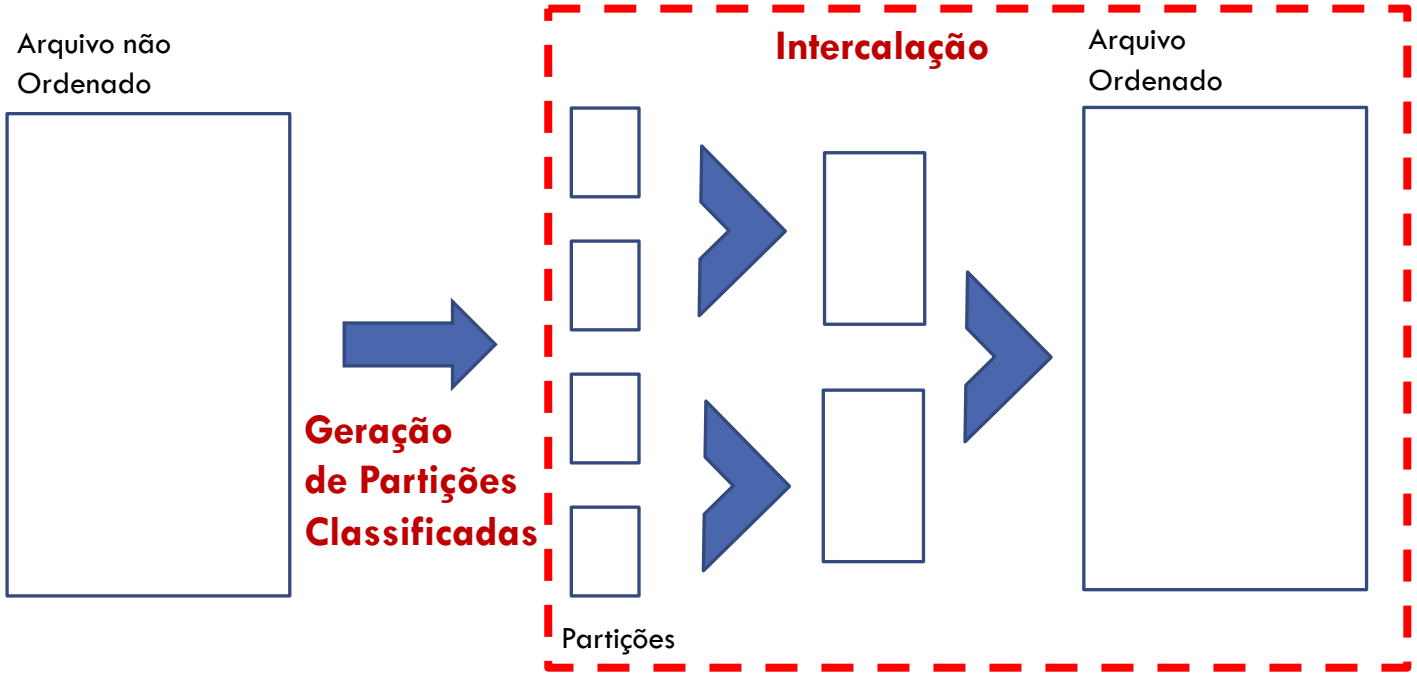
Vanessa Braganholo

Estruturas de Dados e Seus Algoritmos

RELEMBRANDO MODELO DA CLASSIFICAÇÃO EXTERNA



NESSA AULA: 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

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

Arquivos por intercalar
(cada coluna representa um arquivo)

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

Livro pág. 55

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

IMPLEMENTAÇÃO

Ver implementação do algoritmo básico de intercalação no site da disciplina

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

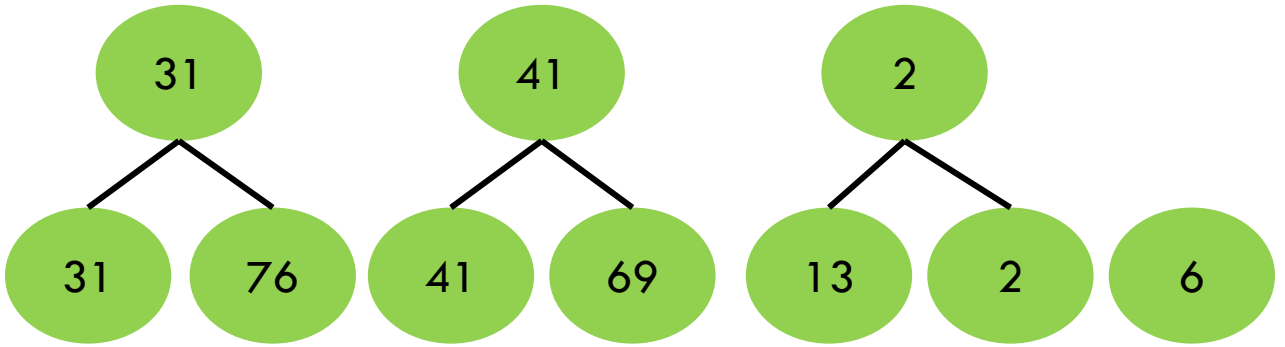
Criar os nós folha da árvore contendo o primeiro registro de cada arquivo

- Aqui usamos apenas as chaves para simplificar

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

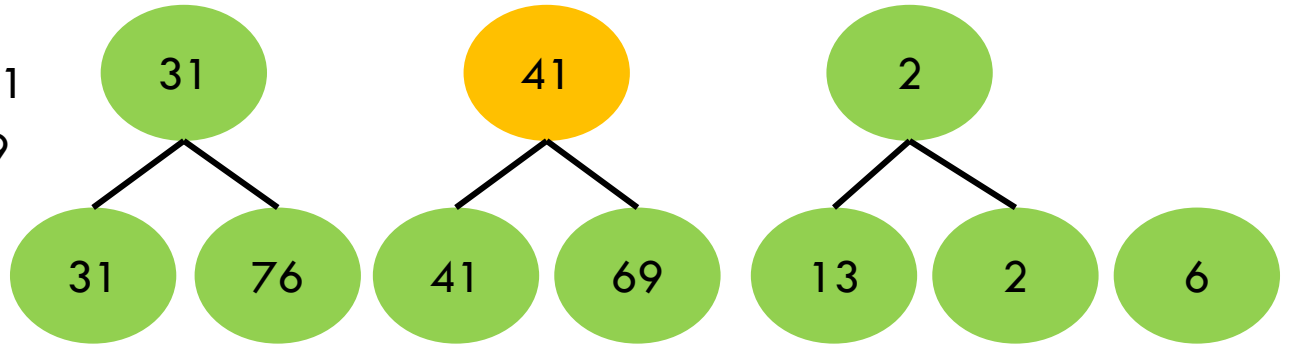


31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

ÁRVORE BINÁRIA DE VENCEDORES

Representação do nó interno 41

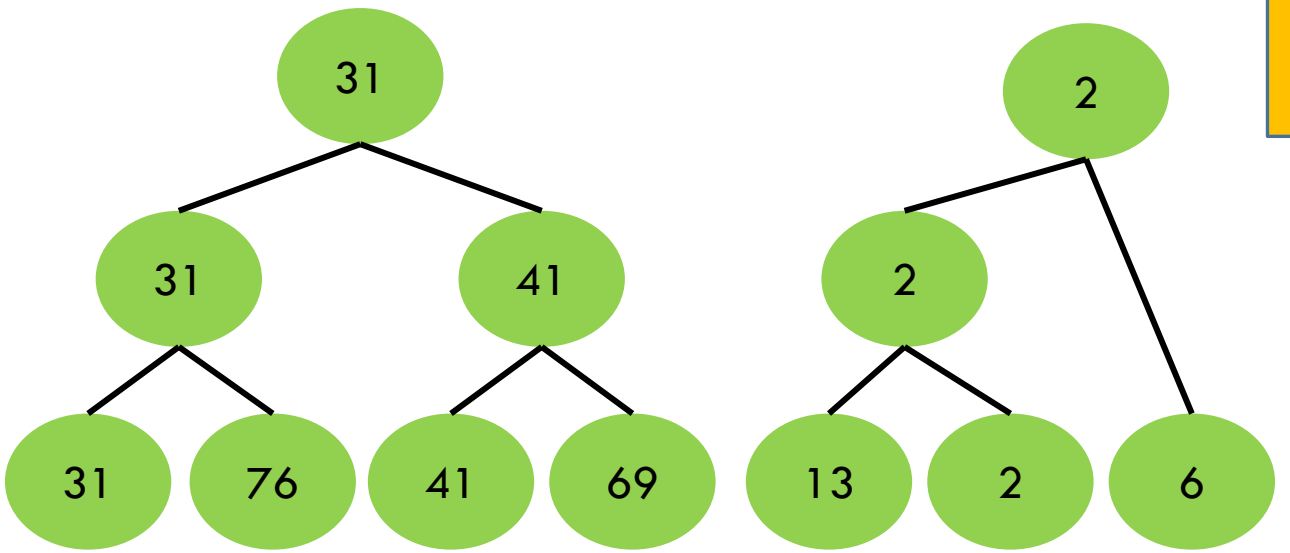
- **vencedor**: 41
- **endVencedor**: valor do ponteiro da esquerda
- **f**: f3
- **esq**: ponteiro p/ 41
- **dir**: ponteiro p/ 69



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

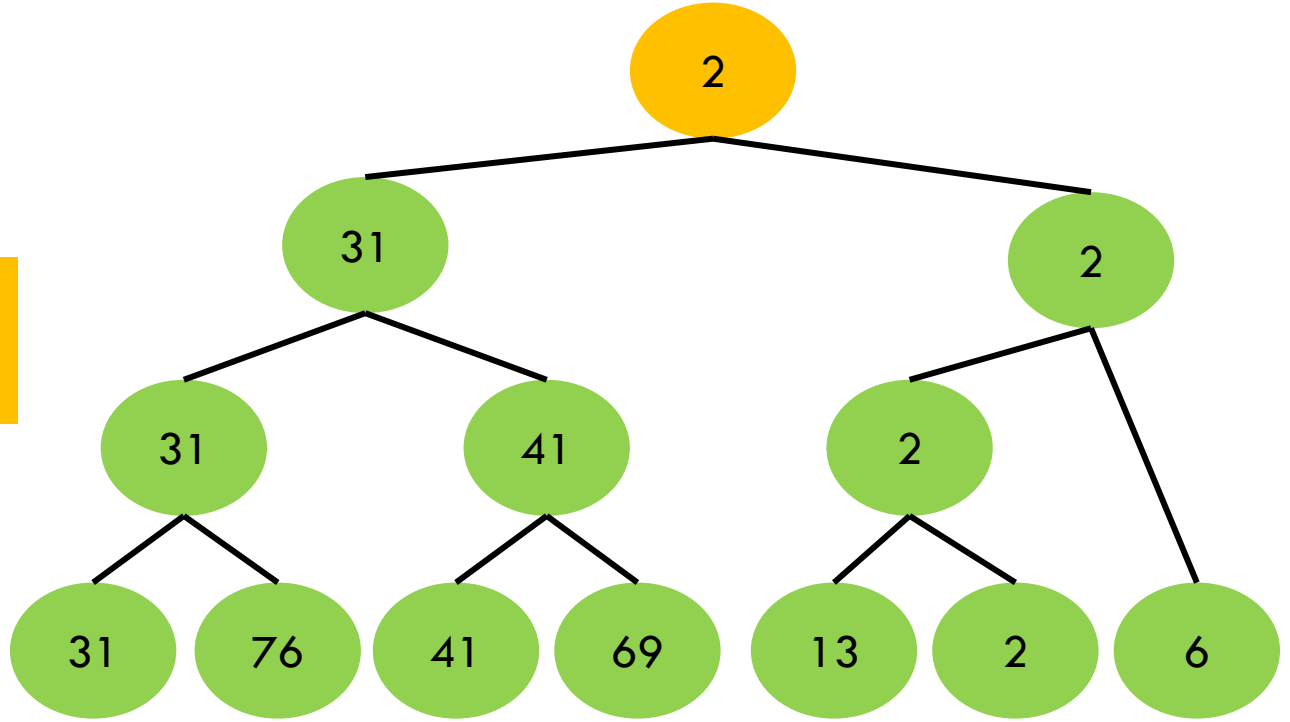
Atenção: valores de chave se repetem em vários níveis



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

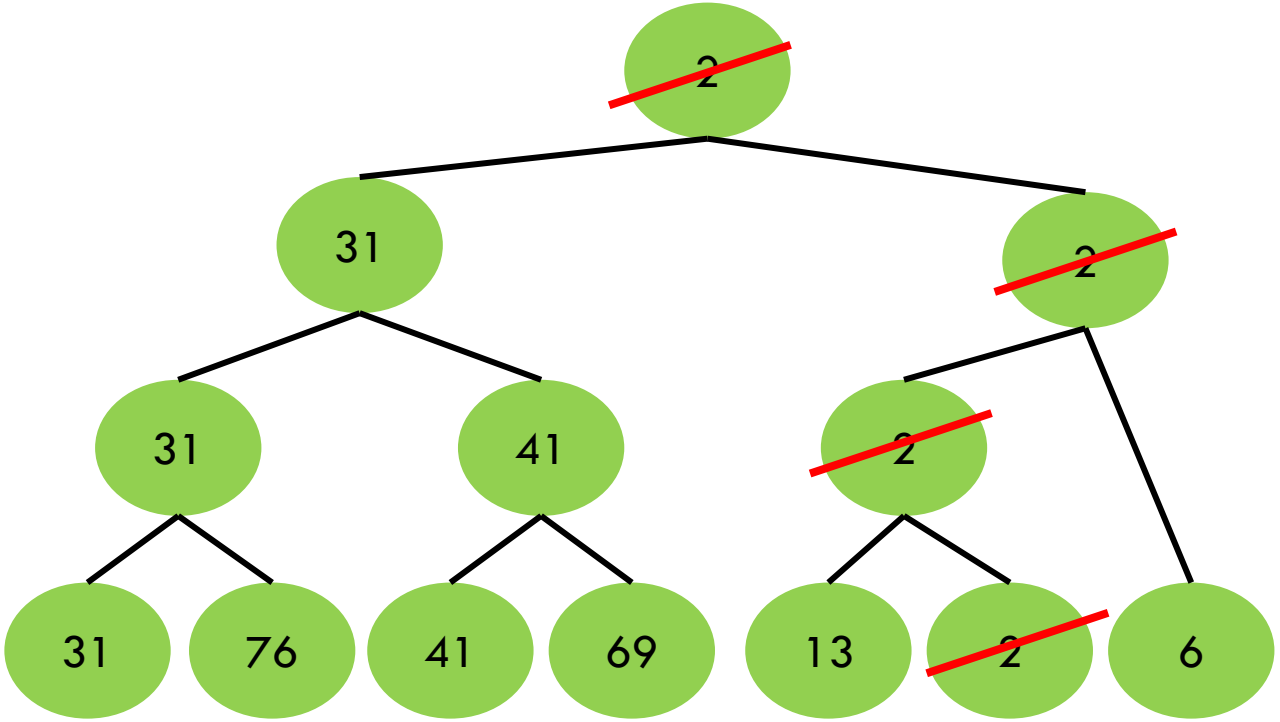


31	76	41	69	13	2	6
70				40	20	10
					51	15
						60
f1	f2	f3	f4	f5	f6	f7

USO DA ÁRVORE DE VENCEDORES NO ALGORITMO DE INTERCALAÇÃO

Chave da raíz é retirada e registro correspondente é inserido no arquivo

ÁRVORE BINÁRIA DE VENCEDORES

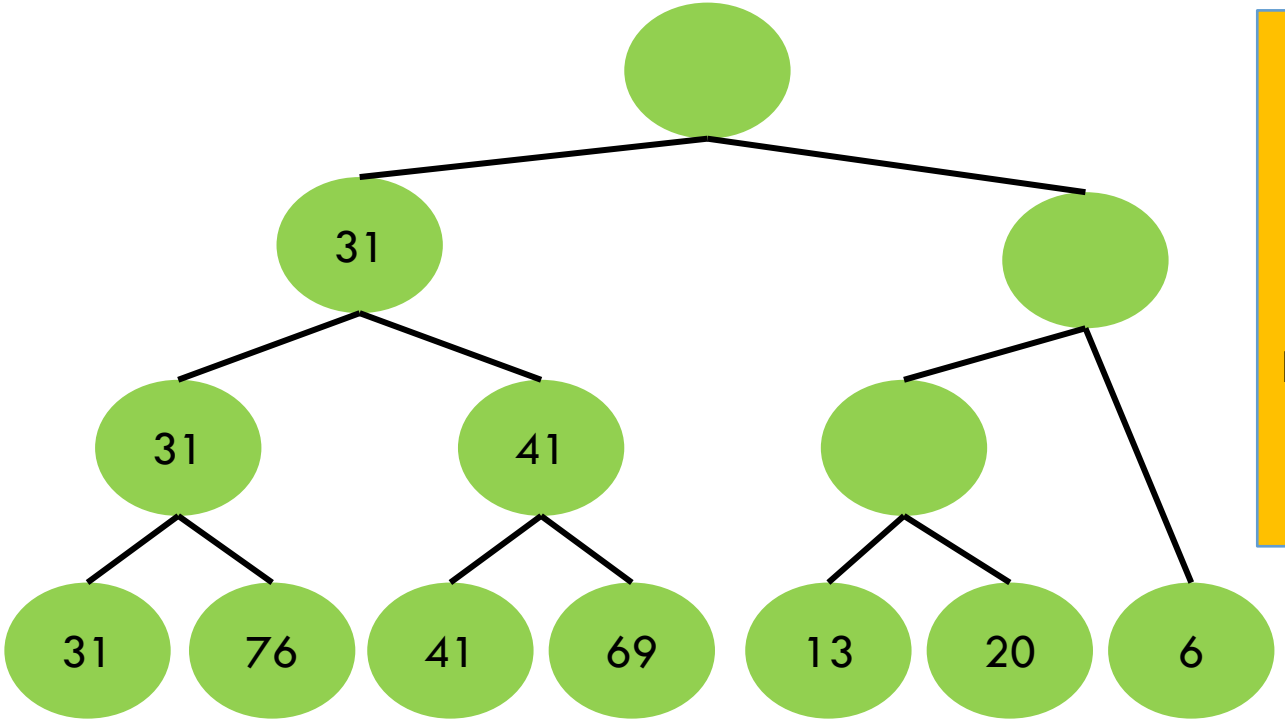


31	76	41	69	13	2	6
70				40	20	10
					51	15
						60

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

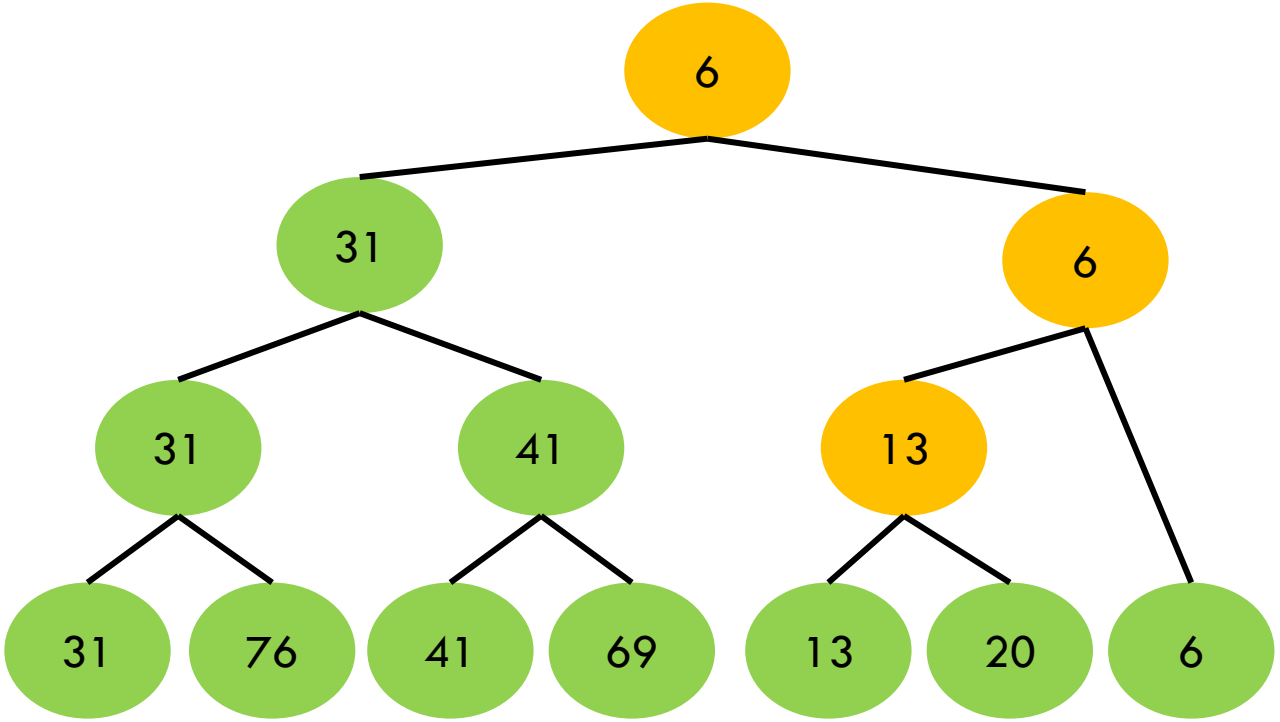
Implementação pode ter um ponteiro para o pai para facilitar

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

Arquivo de saída

2

ÁRVORE BINÁRIA DE VENCEDORES

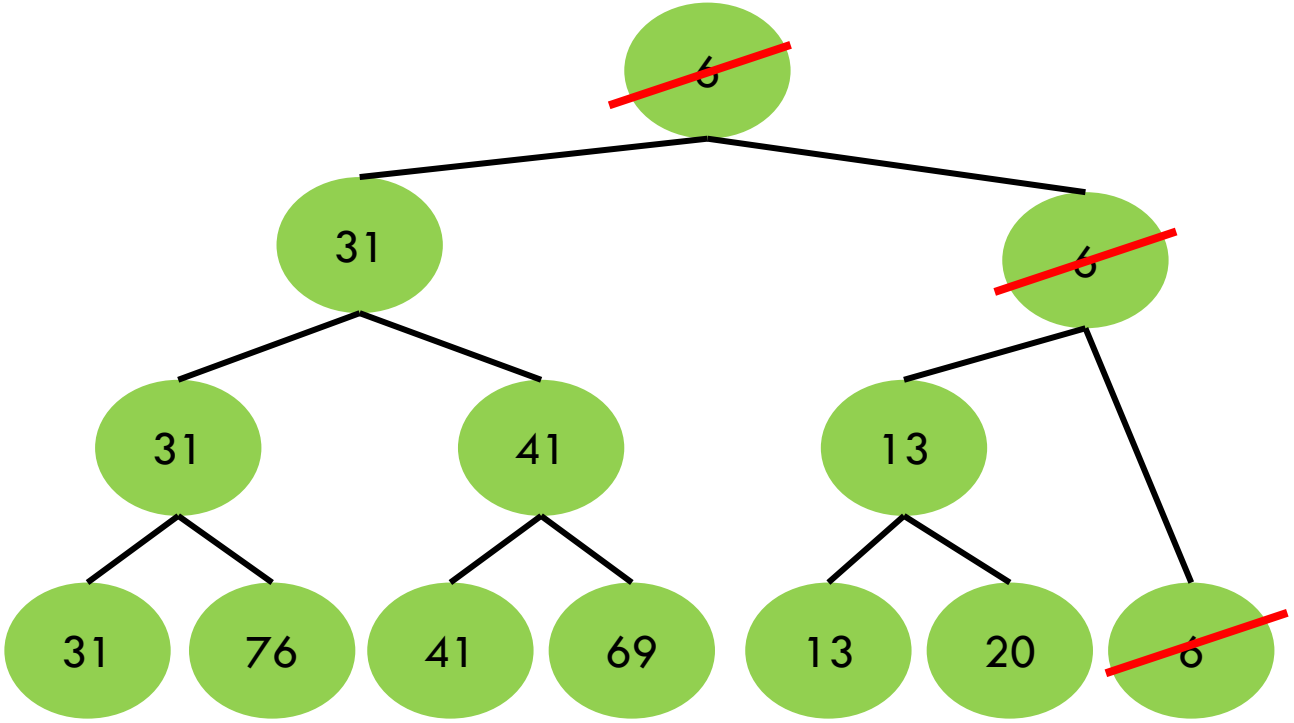


31	76	41	69	13	20	6
70				40	51	10
						15
						60

Arquivo de saída

2

ÁRVORE BINÁRIA DE VENCEDORES

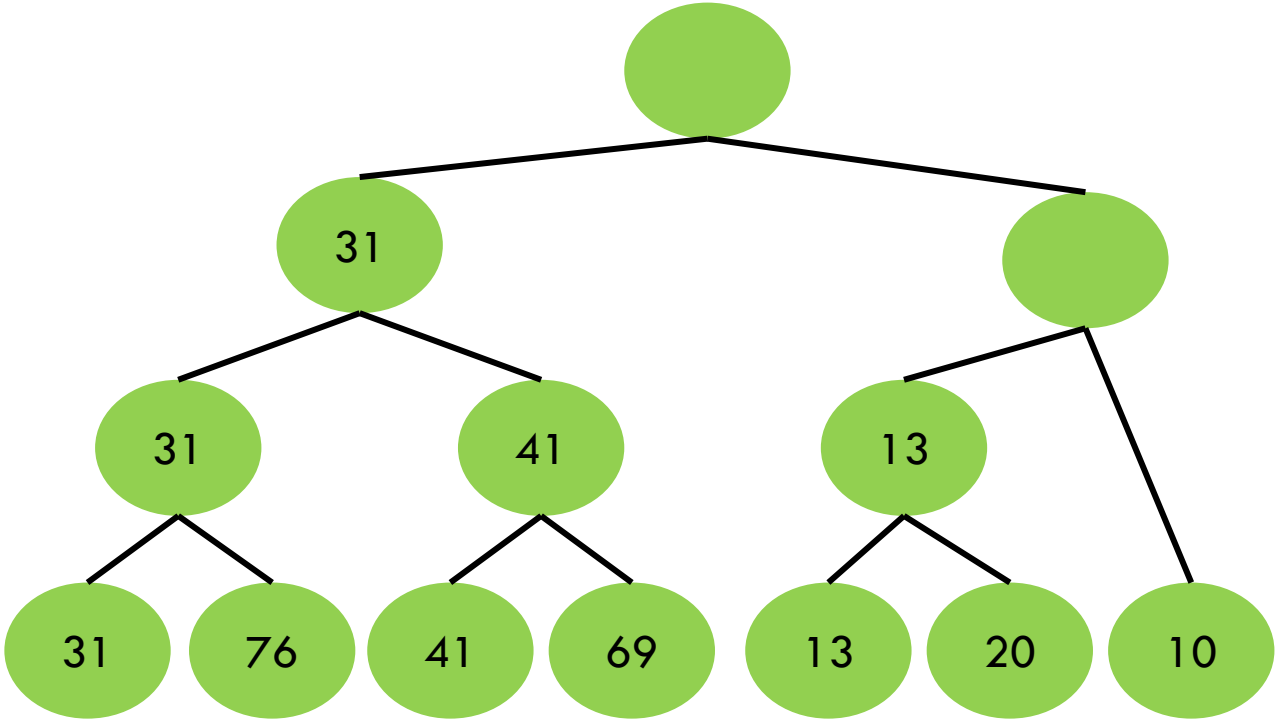


31	76	41	69	13	20	6
70				40	51	10
						15
						60

Arquivo de saída

2
6

ÁRVORE BINÁRIA DE VENCEDORES

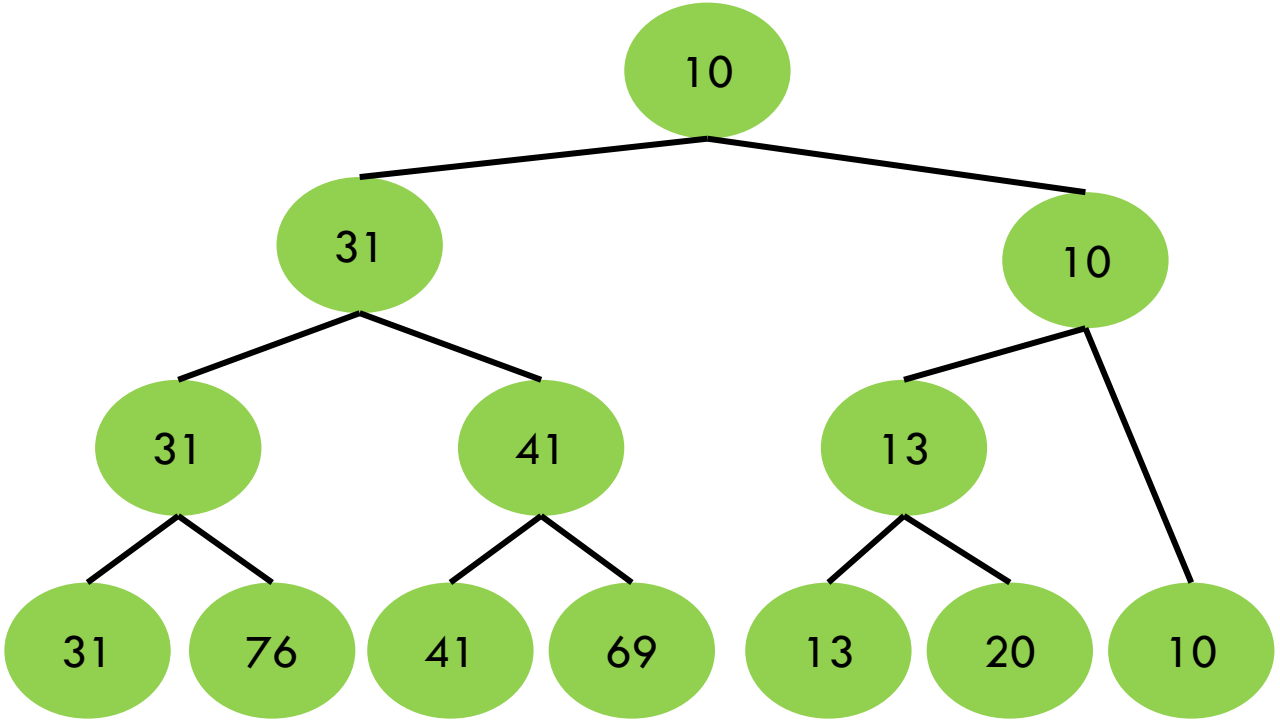


31	76	41	69	13	20	10
70				40	51	15
						60

Arquivo de saída

2
6

ÁRVORE BINÁRIA DE VENCEDORES

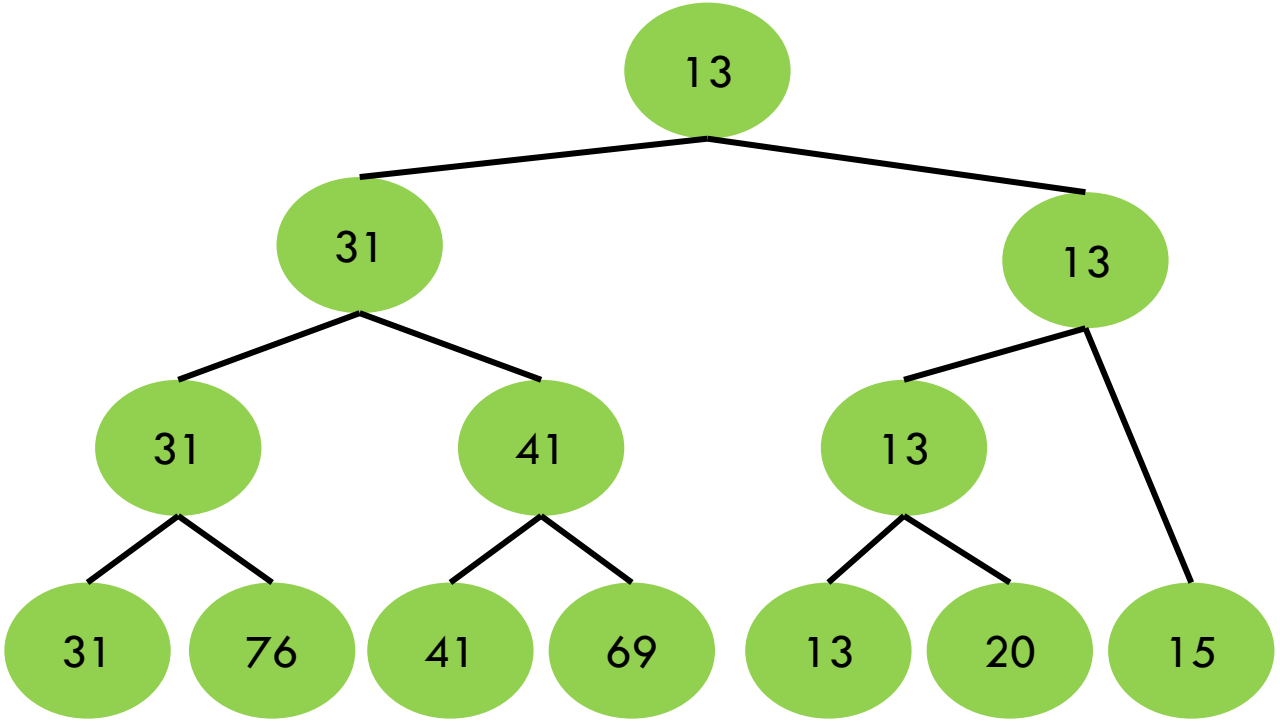


31	76	41	69	13	20	10
70				40	51	15
						60

Arquivo de saída

2
6
10

ÁRVORE BINÁRIA DE VENCEDORES

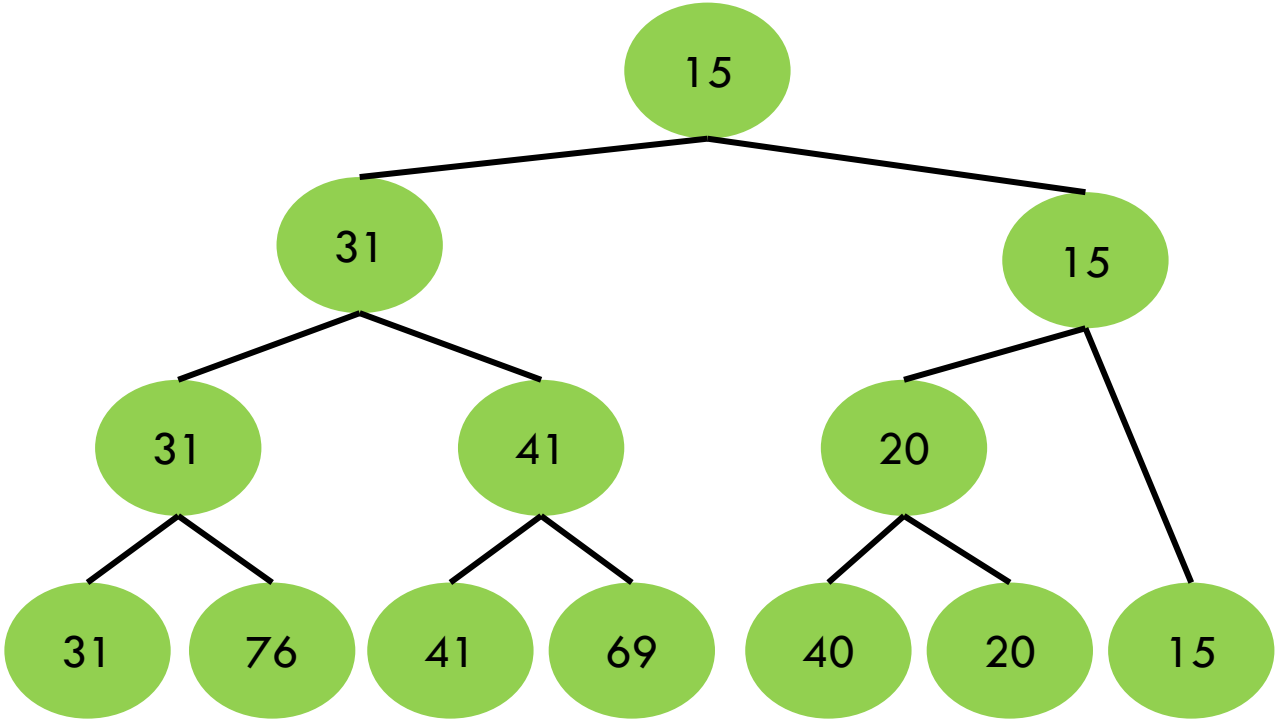


31	76	41	69	13	20	15
70				40	51	60

Arquivo de saída

2
6
10
13

ÁRVORE BINÁRIA DE VENCEDORES

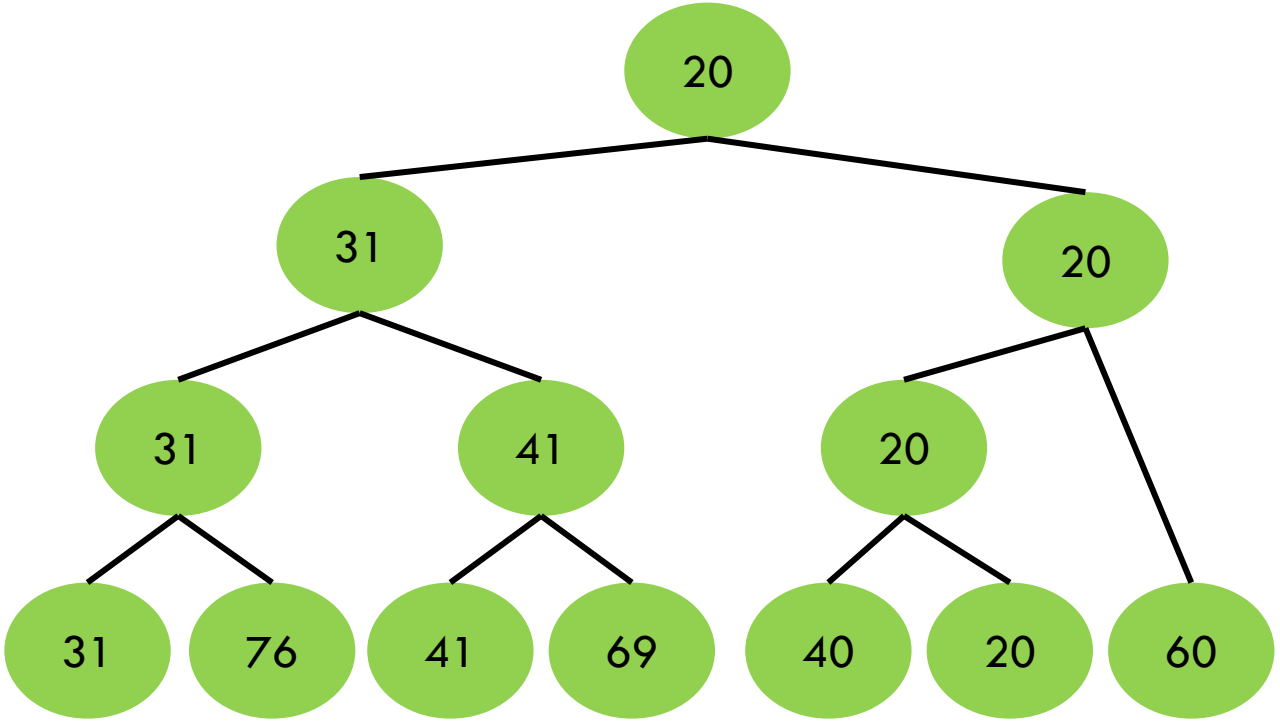


31	76	41	69	40	20	15
70					51	60

Arquivo de saída

2
6
10
13
15

ÁRVORE BINÁRIA DE VENCEDORES

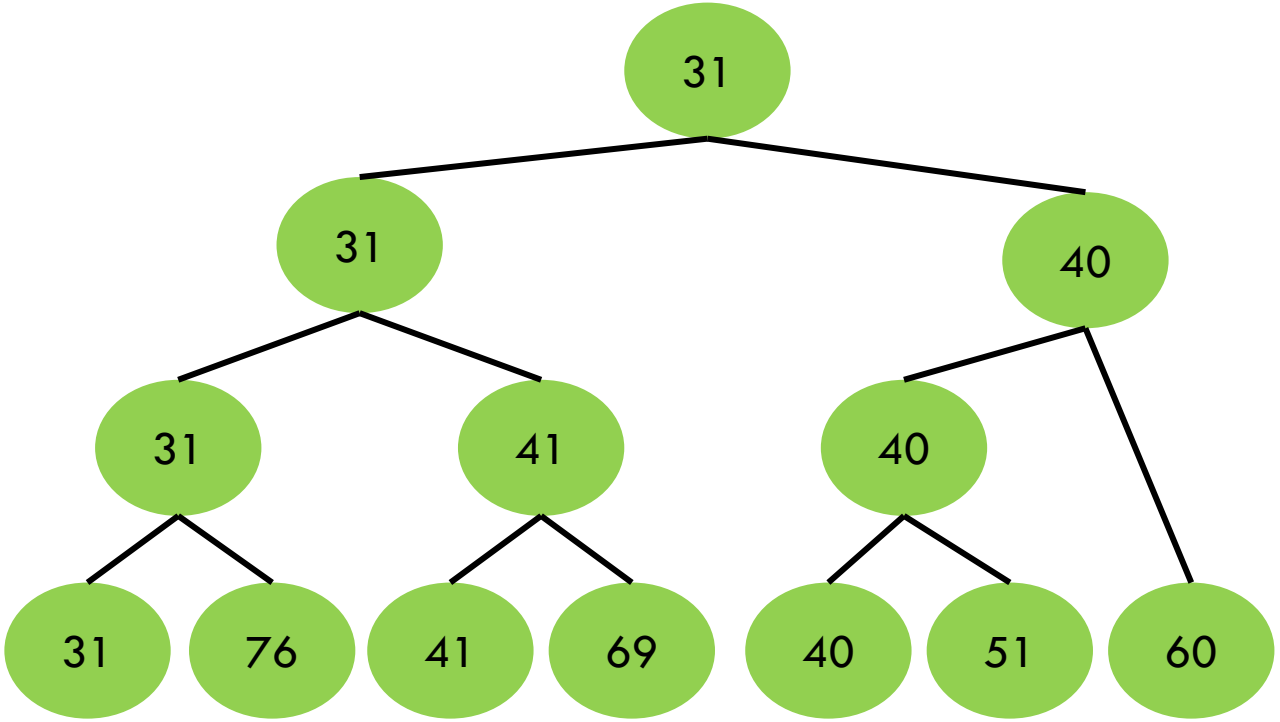


31	76	41	69	40	20	60
70					51	

Arquivo de saída

2
6
10
13
15
20

ÁRVORE BINÁRIA DE VENCEDORES

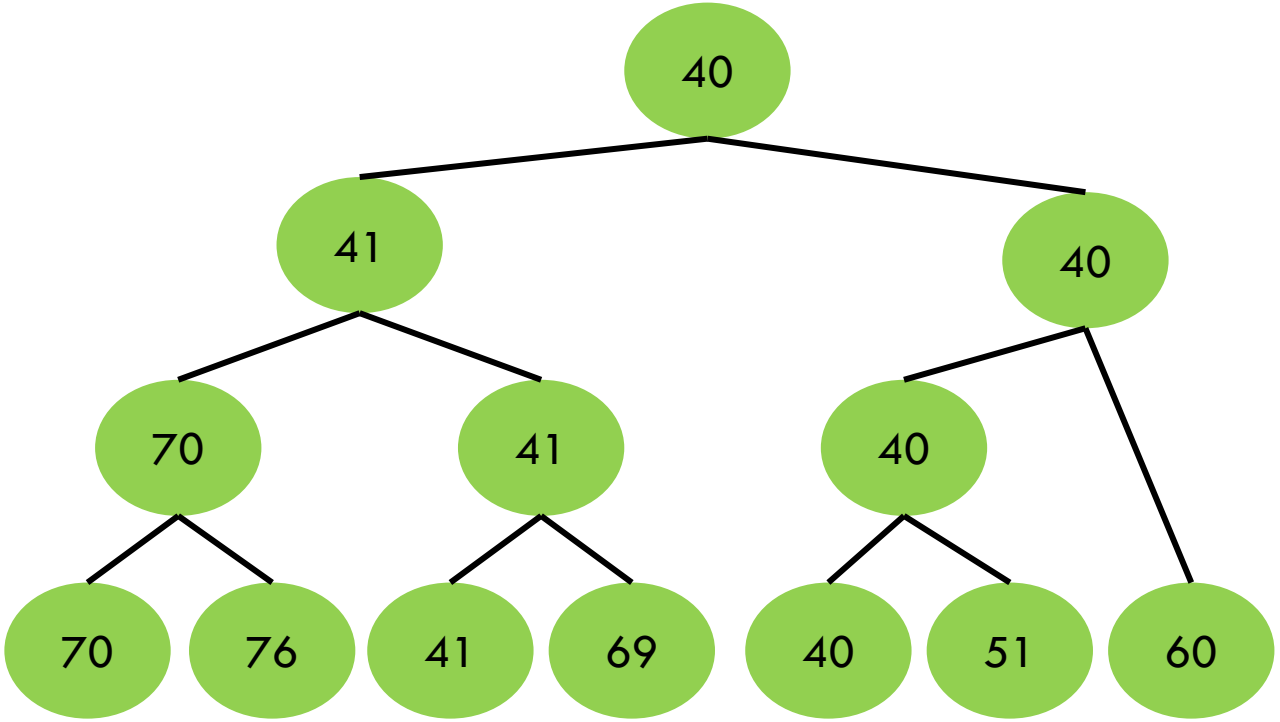


31	76	41	69	40	51	60
70						

Arquivo de saída

2
6
10
13
15
20
31

ÁRVORE BINÁRIA DE VENCEDORES

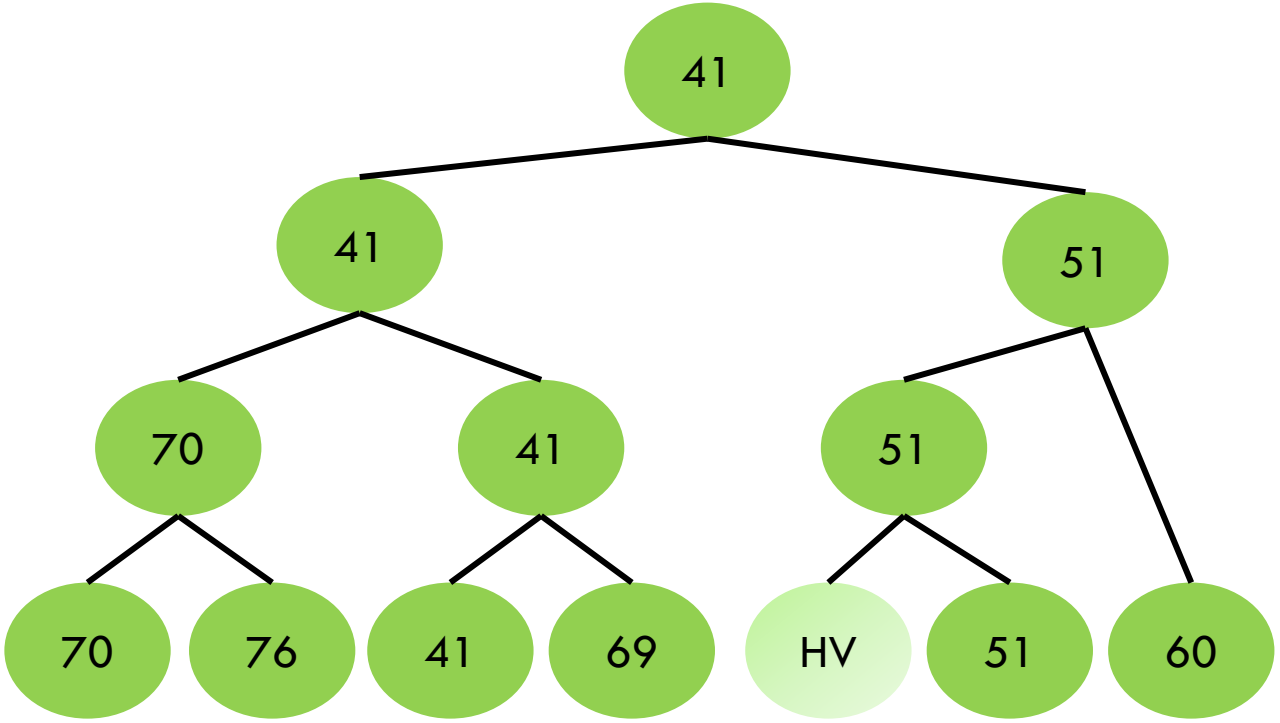


70	76	41	69	40	51	60

Arquivo de saída

2
6
10
13
15
20
31
40

ÁRVORE BINÁRIA DE VENCEDORES

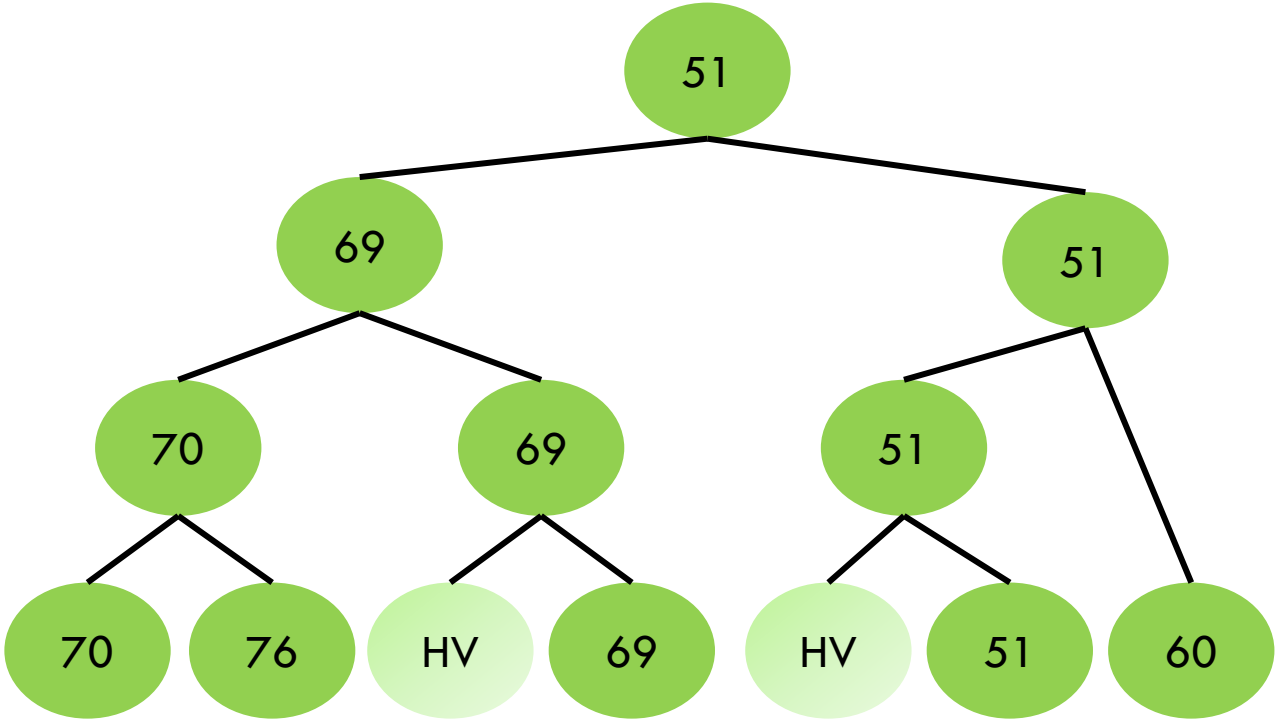


70	76	41	69		51	60

Arquivo de saída

2
6
10
13
15
20
31
40
41

ÁRVORE BINÁRIA DE VENCEDORES

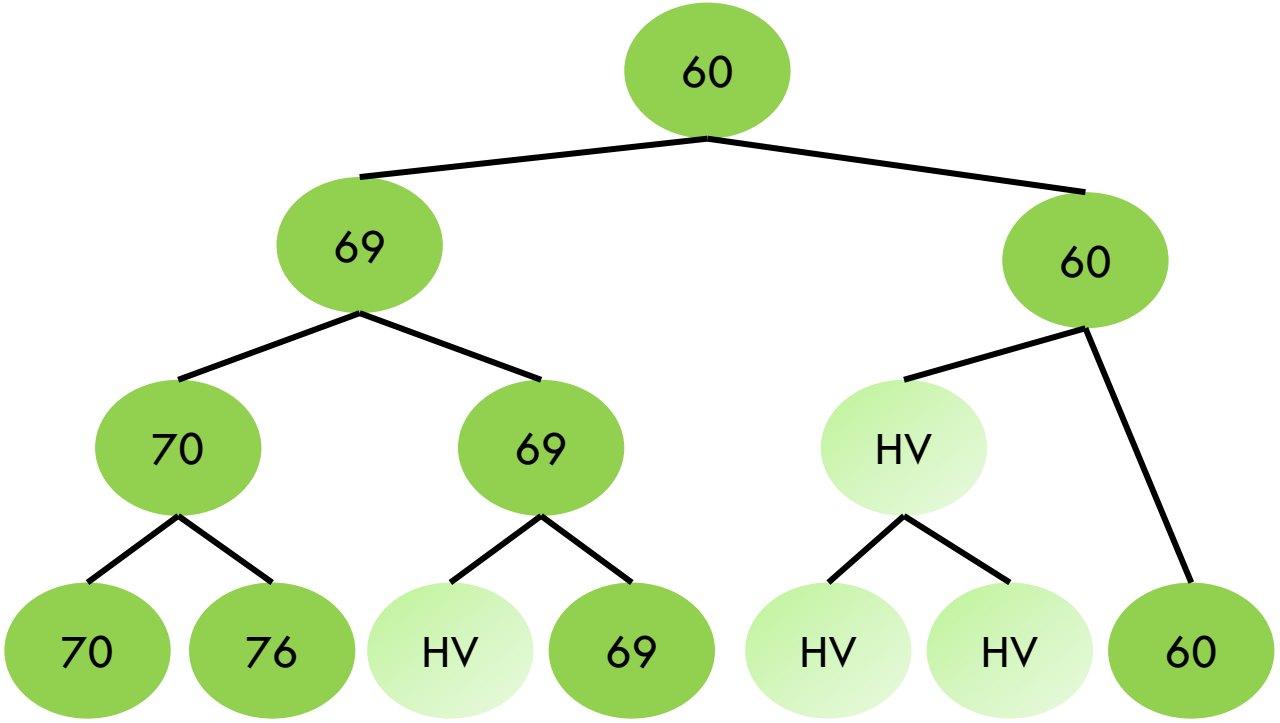


70	76		69		51	60

Arquivo de saída

2
6
10
13
15
20
31
40
41
51

ÁRVORE BINÁRIA DE VENCEDORES

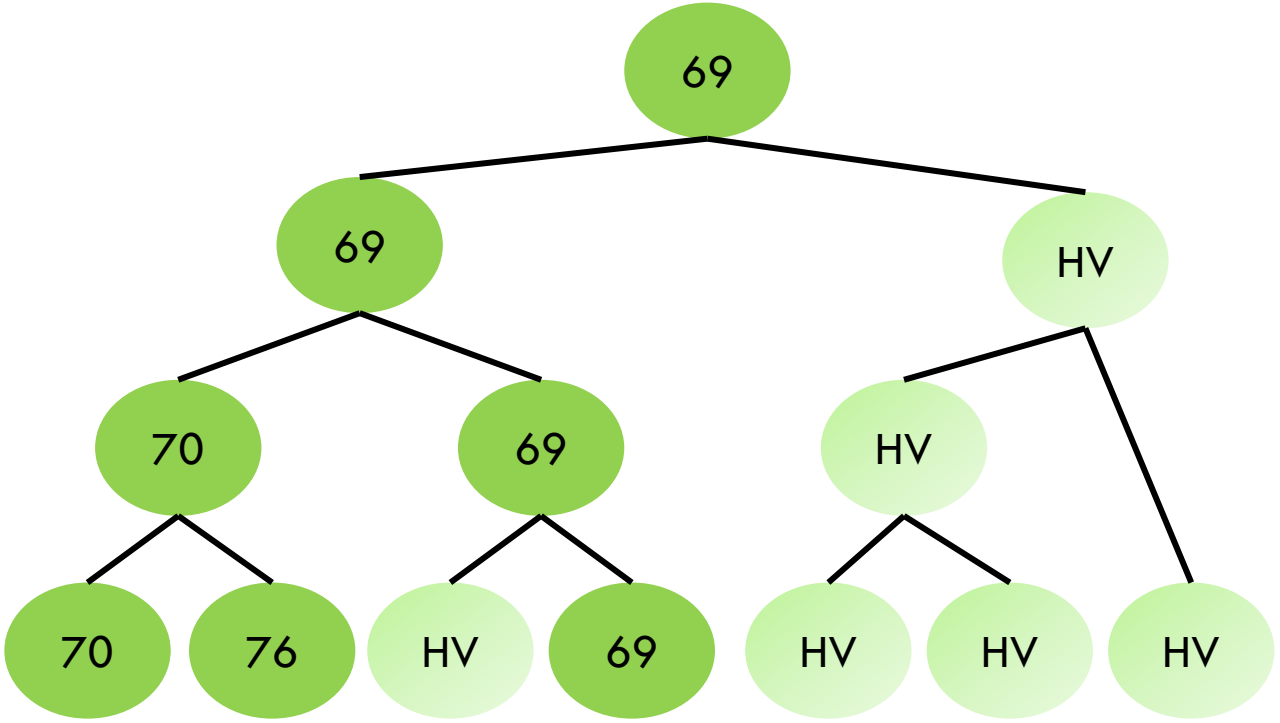


70	76		69			60

Arquivo de saída

2
6
10
13
15
20
31
40
41
51
60

ÁRVORE BINÁRIA DE VENCEDORES

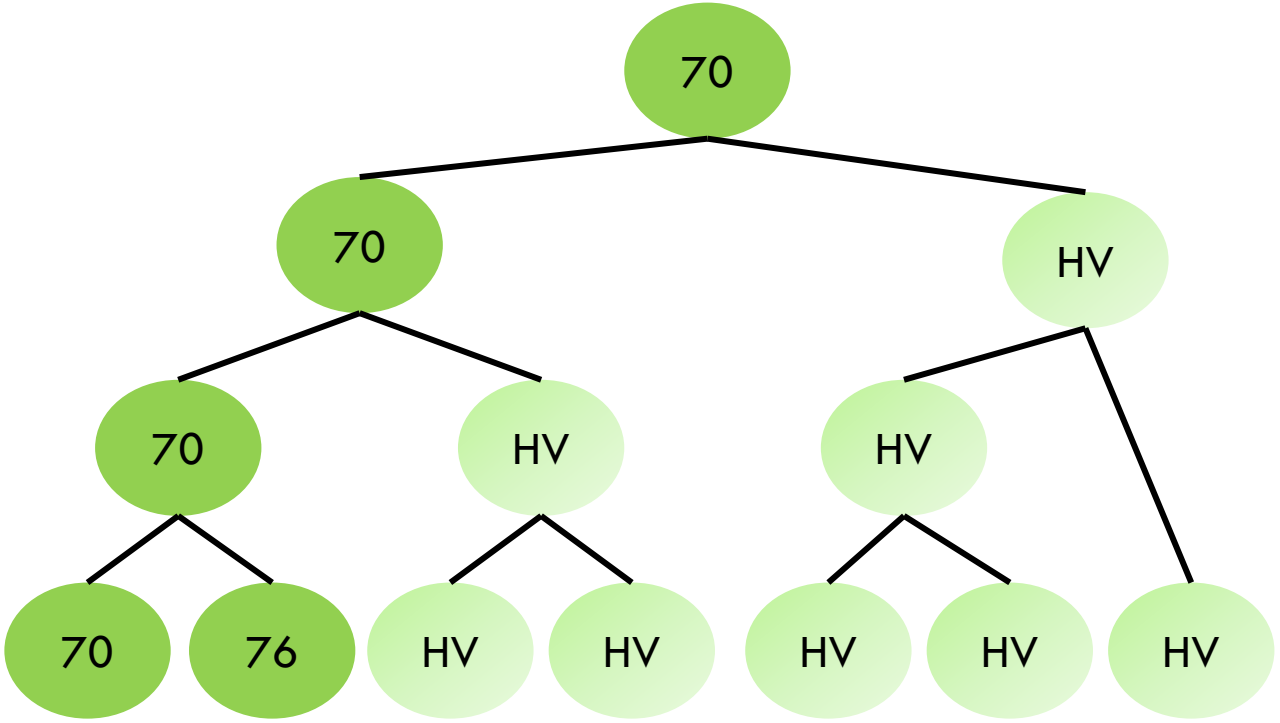


70	76		69			

Arquivo de saída

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

ÁRVORE BINÁRIA DE VENCEDORES

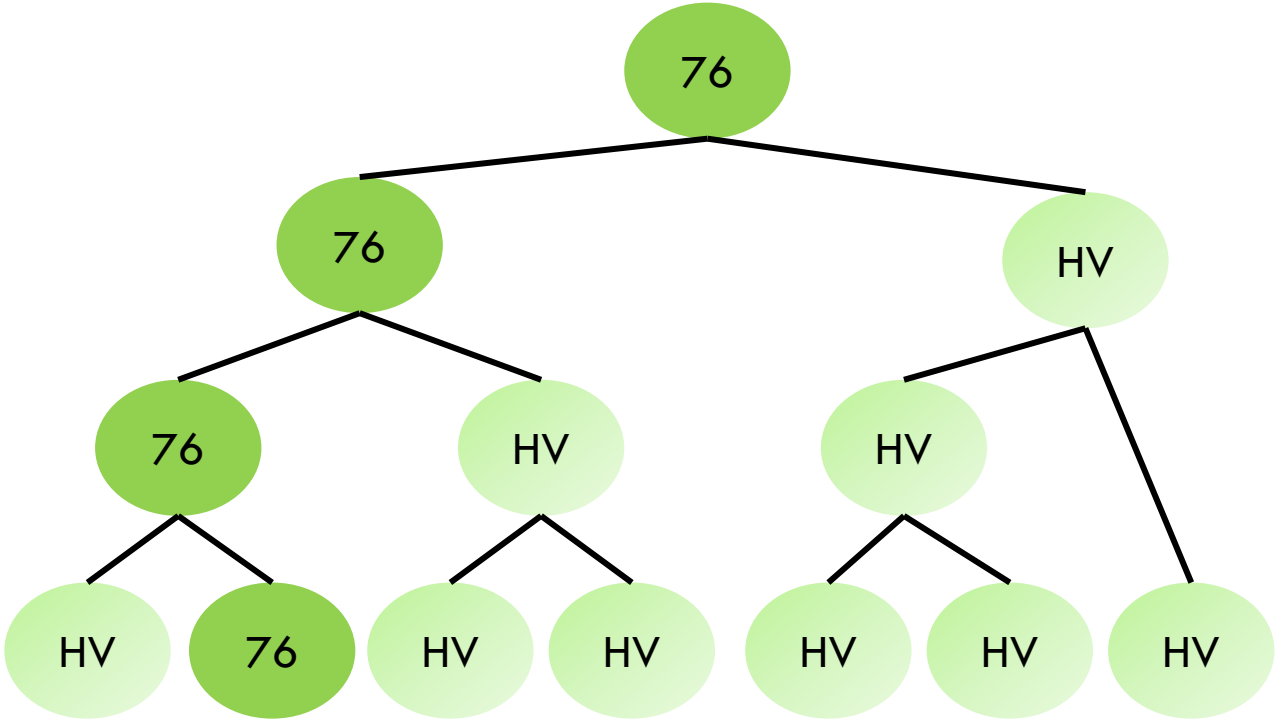


70	76					

Arquivo de saída

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

ÁRVORE BINÁRIA DE VENCEDORES

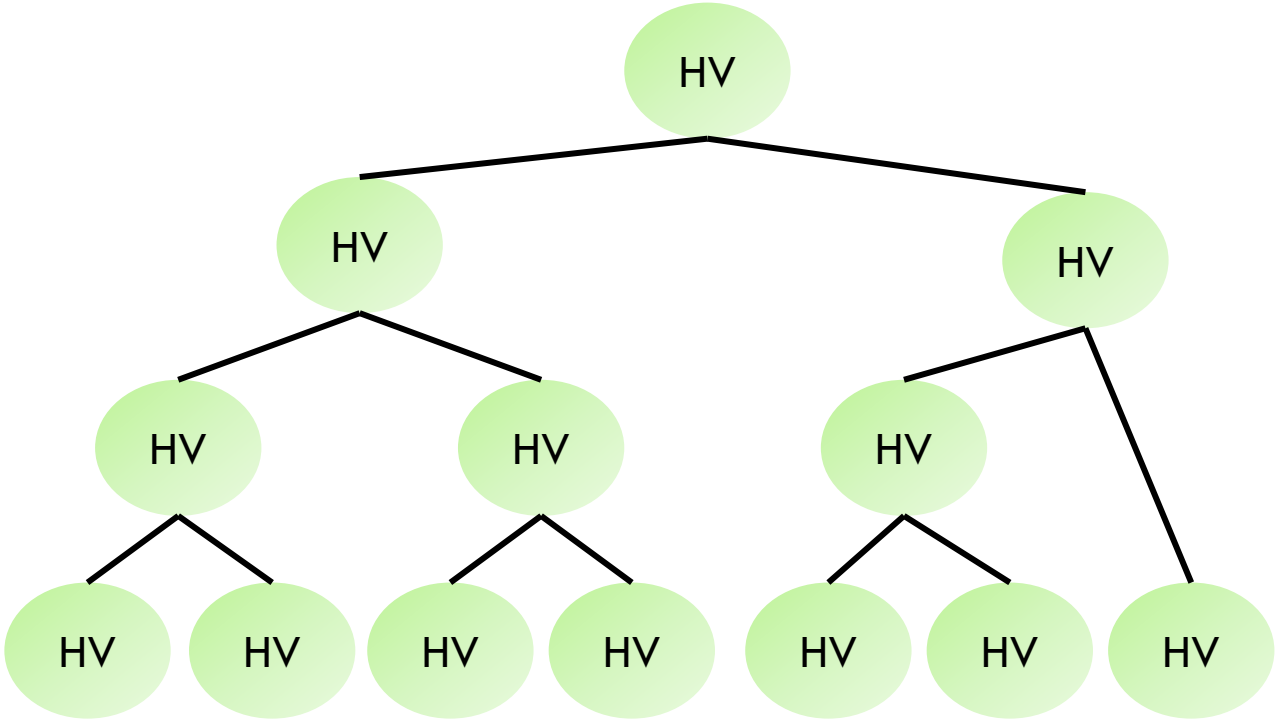


	76					

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

Pensar no **algoritmo** (em alto nível) de montagem da árvore

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

EXERCÍCIO

Implementar o algoritmo de intercalação **utilizando árvore binária de vencedores**

- Entrada:
  - Lista com os nomes dos arquivos de partições (entrada)
  - Quantidade de arquivos de entrada
  - Nome do arquivo de saída
- Estrutura dos arquivos: Clientes (cod_cliente, nome)

MONTAGEM DA ÁRVORE DE VENCEDORES

Entrada:

- Lista com os nomes dos arquivos de partições (entrada)
- Quantidade de arquivos de entrada
- Nome do arquivo de saída

Algoritmo:

1. Criar uma lista de nós vazia
2. Criar nós folha da árvore, e adicioná-los na lista
3. Enquanto lista tiver mais de 1 elemento:
    1. Retirar os 2 primeiros nós da lista
    2. Criar um nó **p** para ser o pai desses dois, escolhendo o vencedor e ajustando os campos do nó criado de acordo
    3. Adicionar **p** no final da lista
4. O elemento que sobrou na lista é a raiz da árvore de vencedores

ESTRUTURA DOS NÓS

```
typedef struct No {
    TCliente *vencedor;
    struct No *endVencedor;
    FILE *f;
    struct No *pai;
    struct No *dir;
    struct No *esq;
} TNo;
```

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**

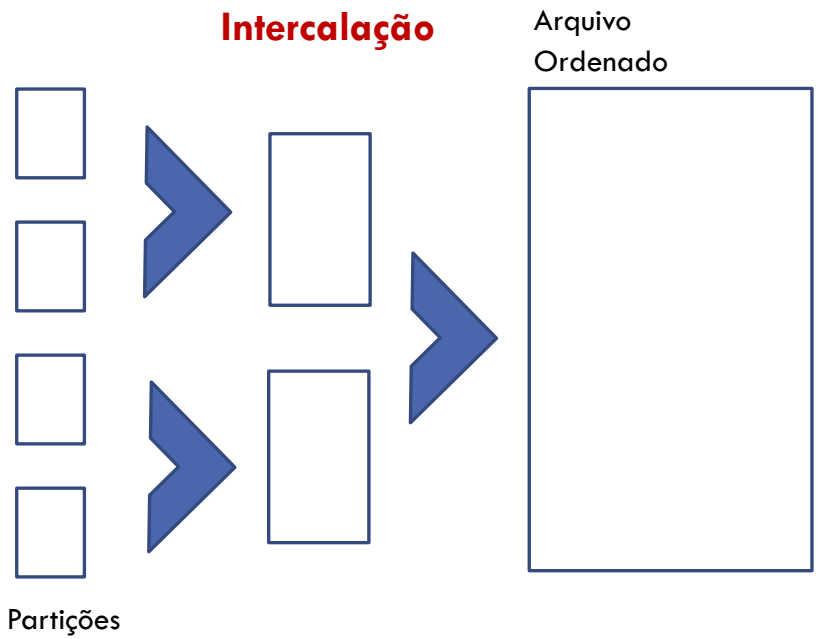
(ii) Sistemas Operacionais estabelecem **número máximo de arquivos abertos simultaneamente**

- 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)

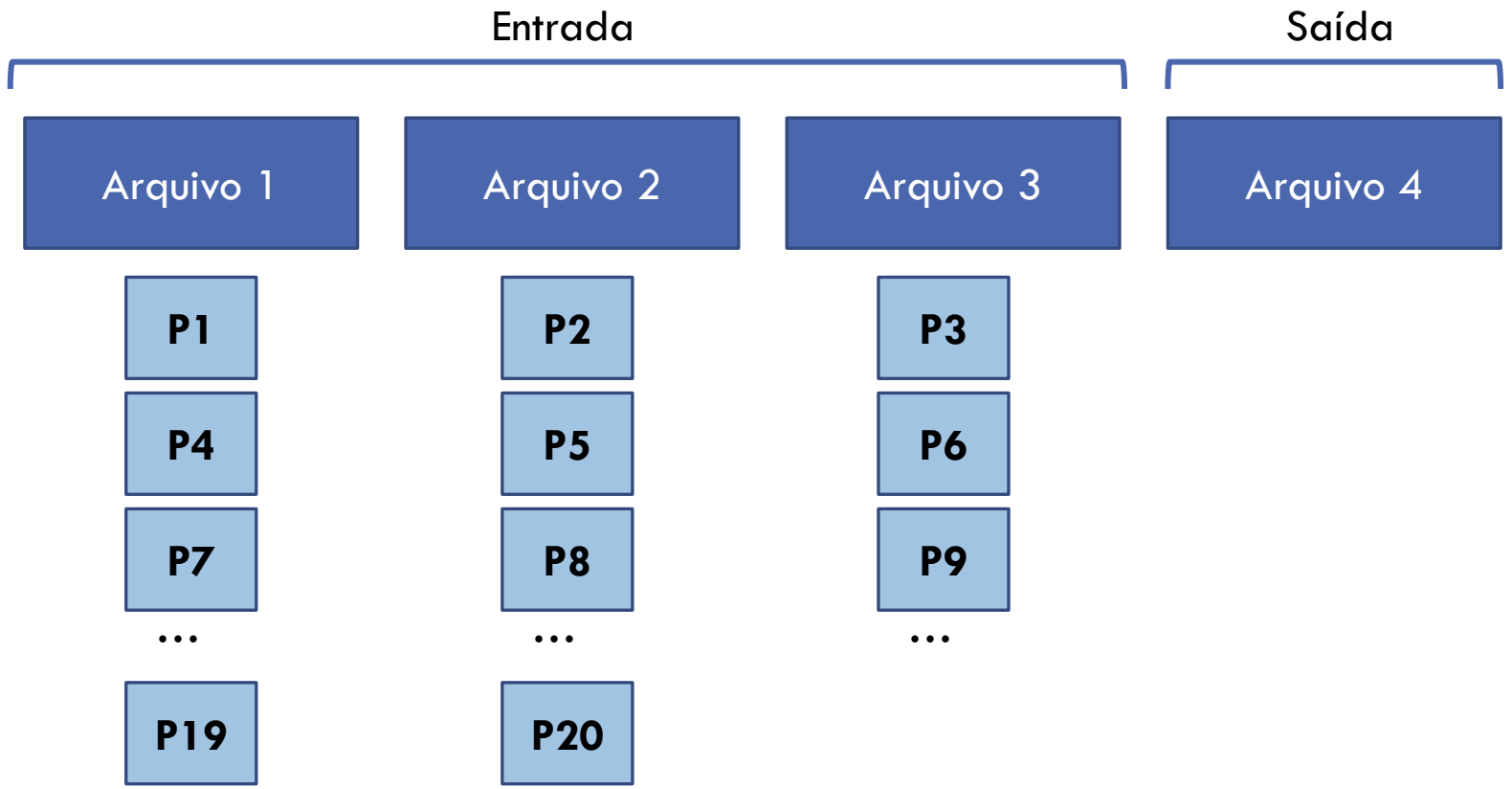
EXEMPLO COM F = 4

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)



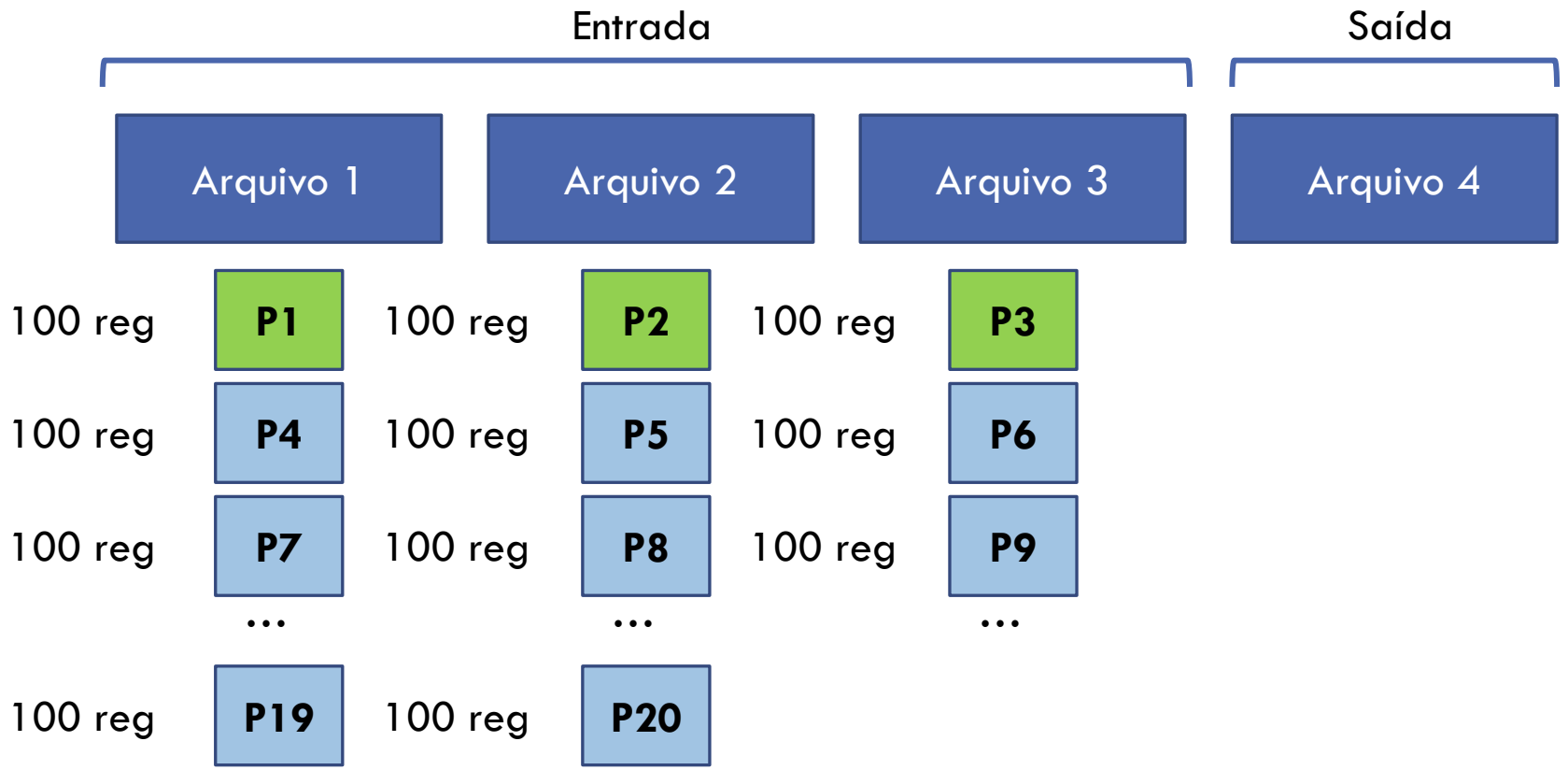
INTERCALAÇÃO ÓTIMA

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

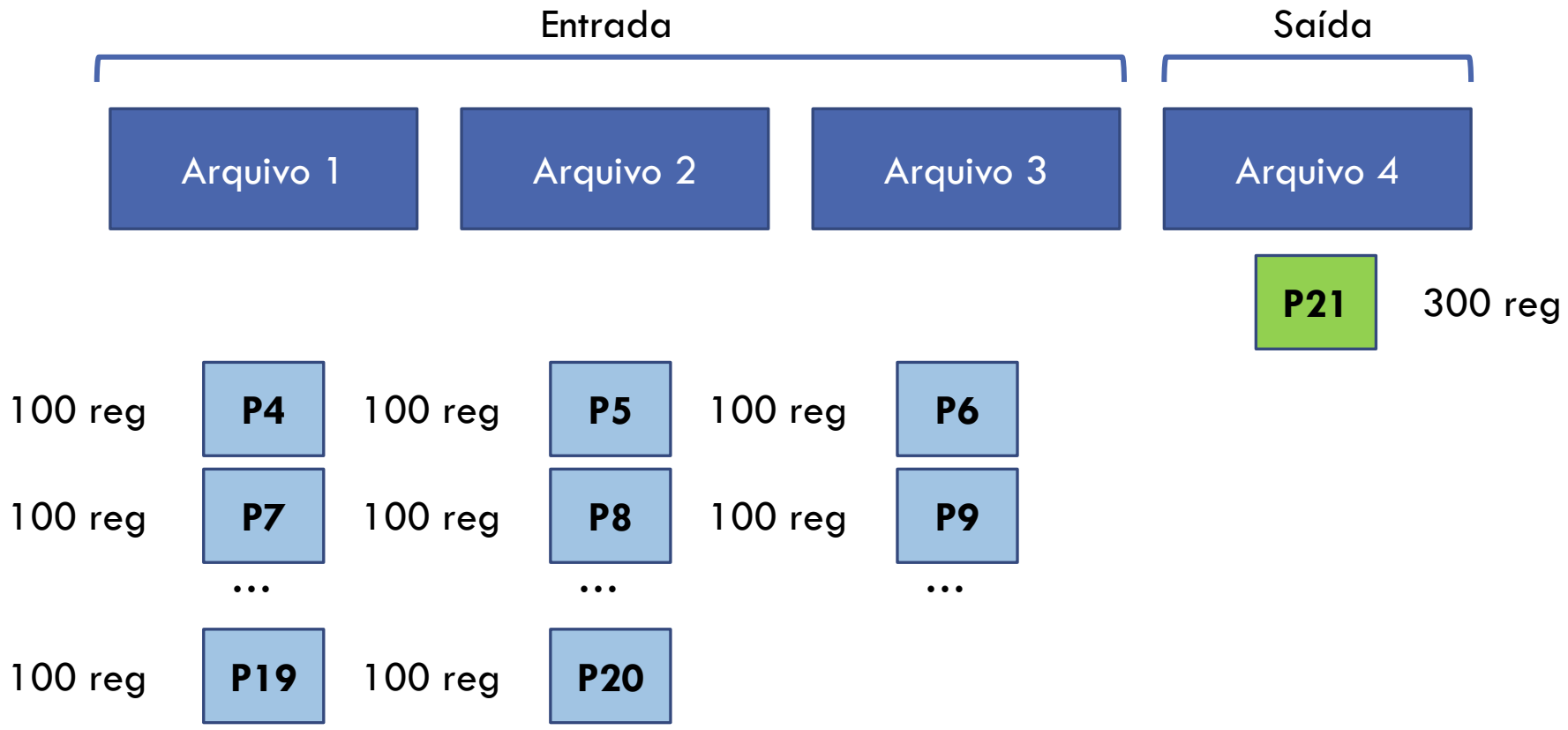
EXEMPLO COM F = 4

Intercalar 20 partições com 100 registros cada



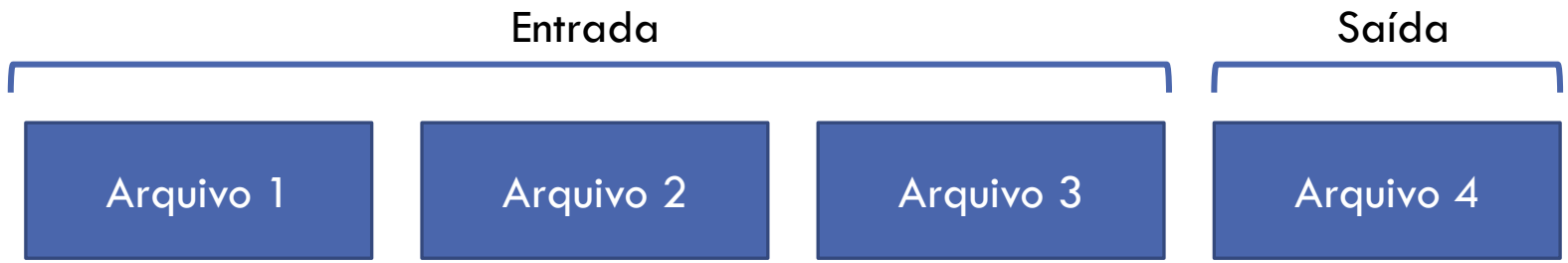
EXEMPLO COM F = 4

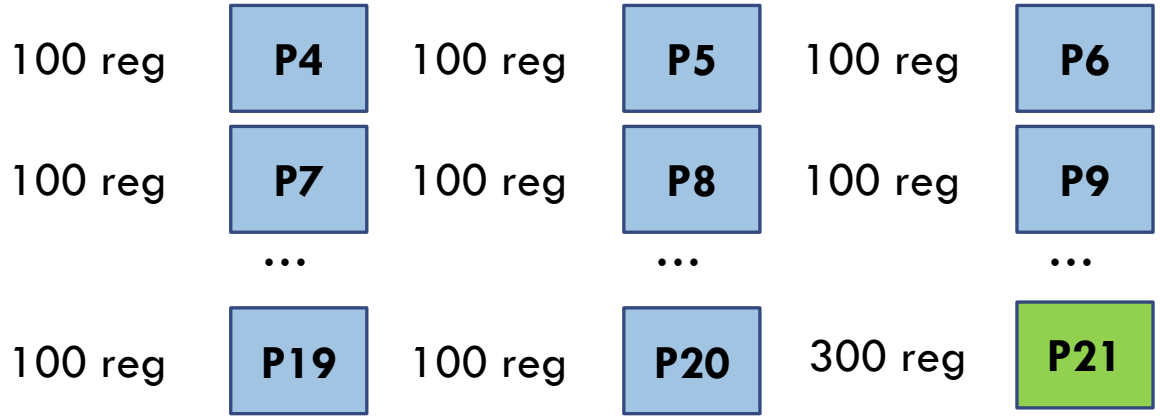
Intercalar 20 partições com 100 registros cada



EXEMPLO COM F = 4

Intercalar 20 partições com 100 registros cada



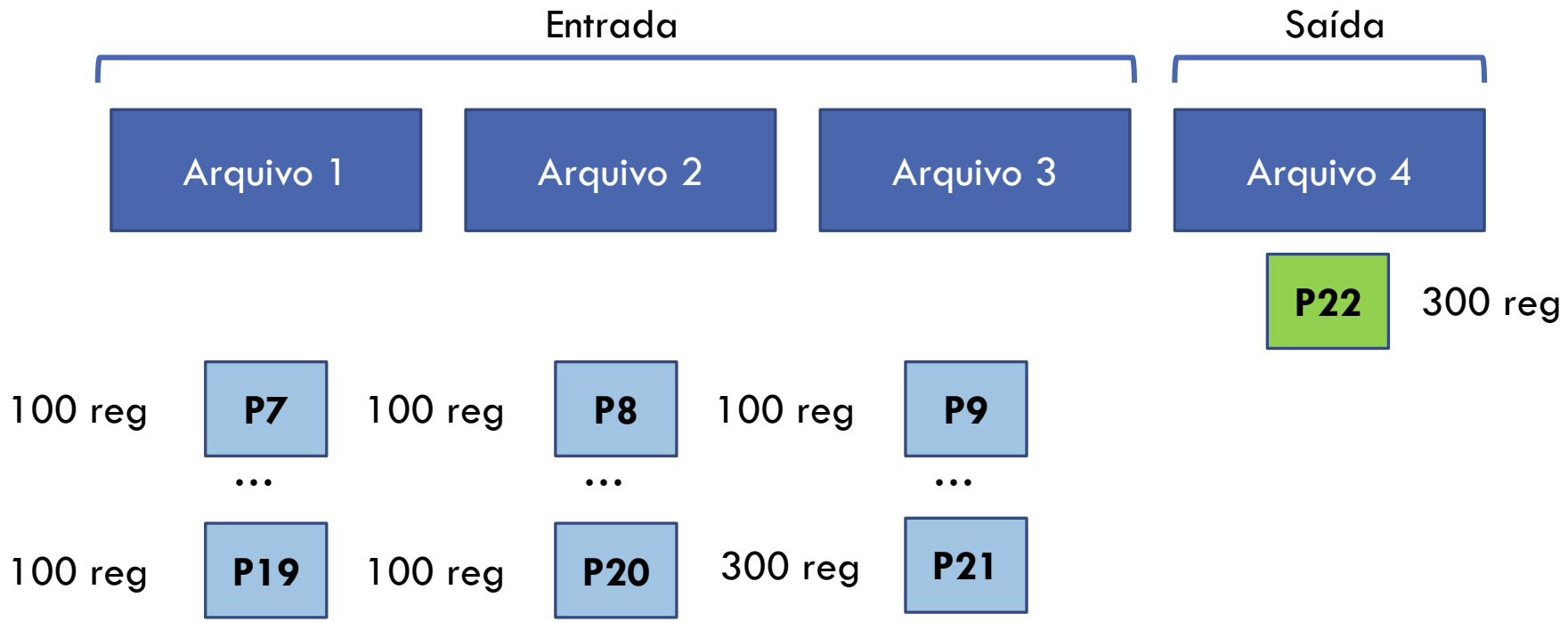


EXEMPLO COM F = 4

Intercalar 20 partições com 100 registros cada

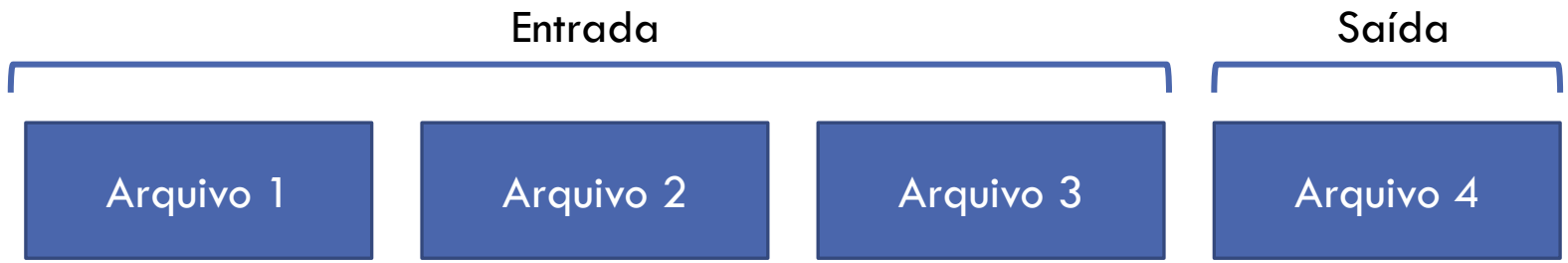
EXEMPLO COM F = 4

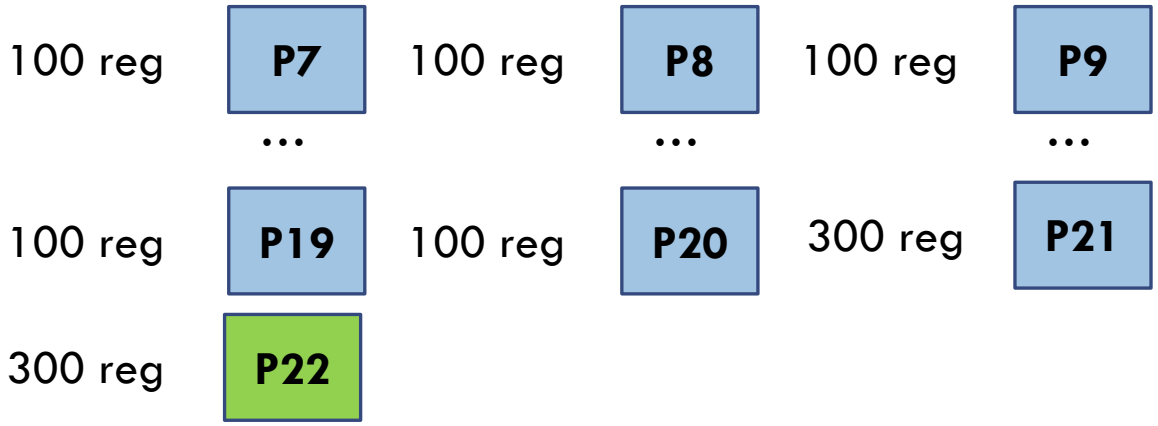
Intercalar 20 partições com 100 registros cada



EXEMPLO COM F = 4

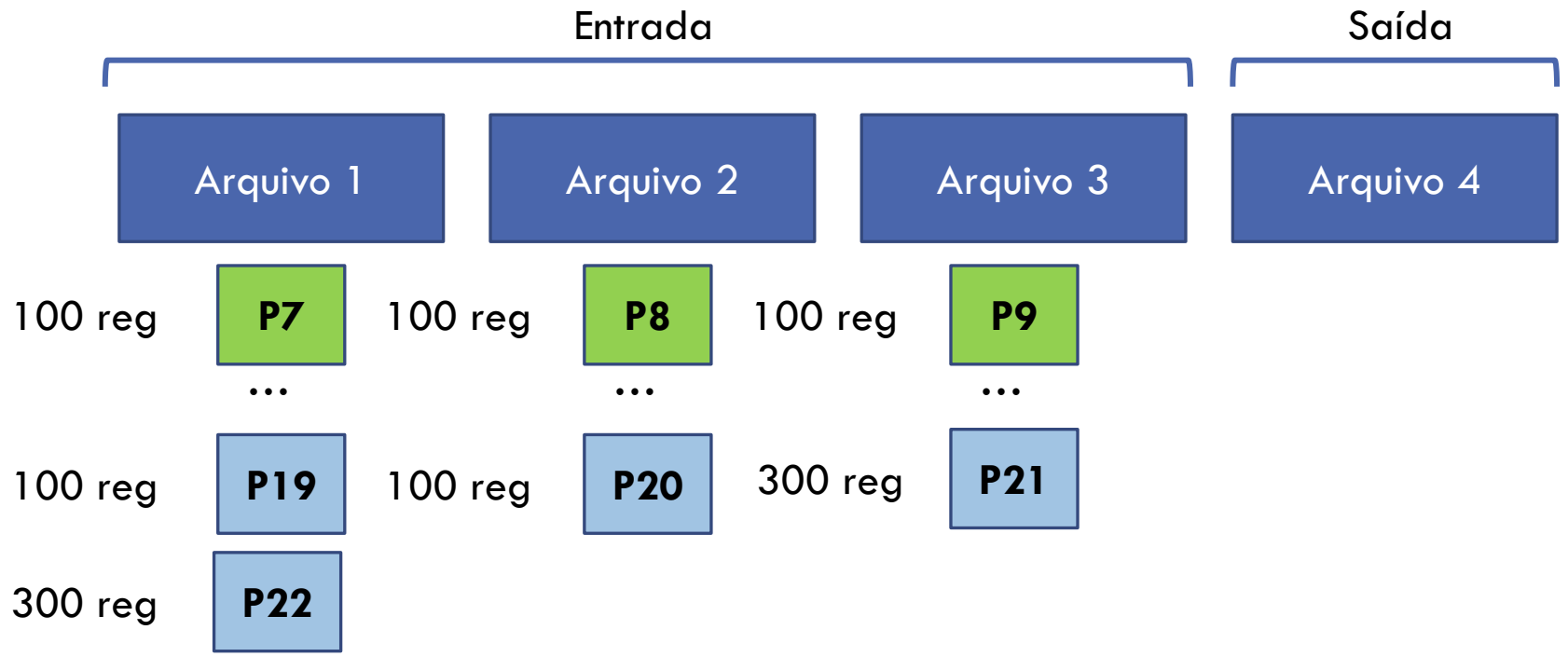
Intercalar 20 partições com 100 registros cada





EXEMPLO COM F = 4

Intercalar 20 partições com 100 registros cada



RESUMO DO EXEMPLO

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:

$$\text{Número de passos} = \frac{\text{No. total de registros lidos}}{\text{No. total de registros no arquivo classificado}}$$

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

RESUMO DO EXEMPLO

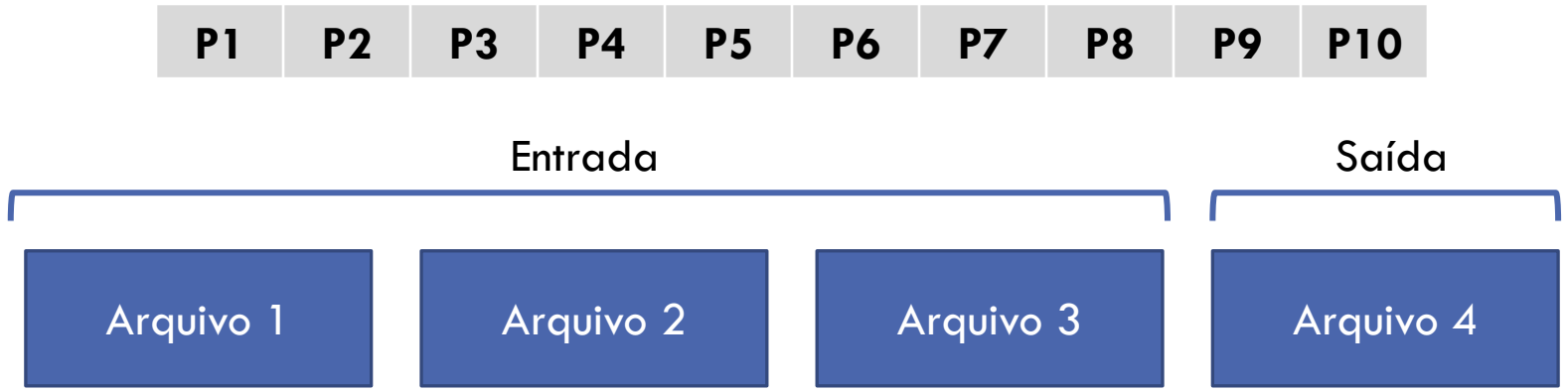
Fase	Arquivo1	Arquivo2	Arquivo3	Arquivo4	N°. de leituras
1	1:100	2:100	3:100	21:300	300
2	4:100	5:100	6:100	22:300	300
3	7:100	8:100	9:100	23:300	300
4	10:100	11:100	12:100	24:300	300
5	13:100	14:100	15:100	25:300	300
6	16:100	17:100	18:100	26:300	300
7	19:100	20:100	21:300	27:500	500
8	22:300	23:300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

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

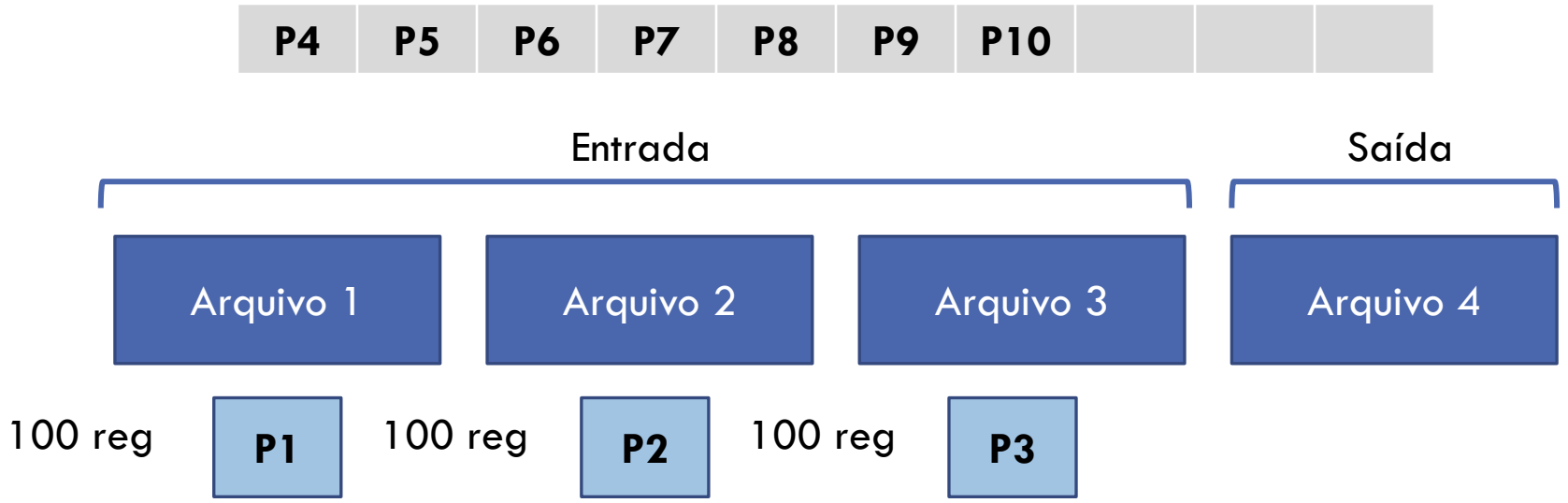
IMPLEMENTAÇÃO

1. Criar uma lista com os nomes dos arquivos a intercalar
2. Enquanto houver mais de 1 arquivo na lista
    1. Retirar os F-1 primeiros itens da lista e intercalá-los
    2. 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)

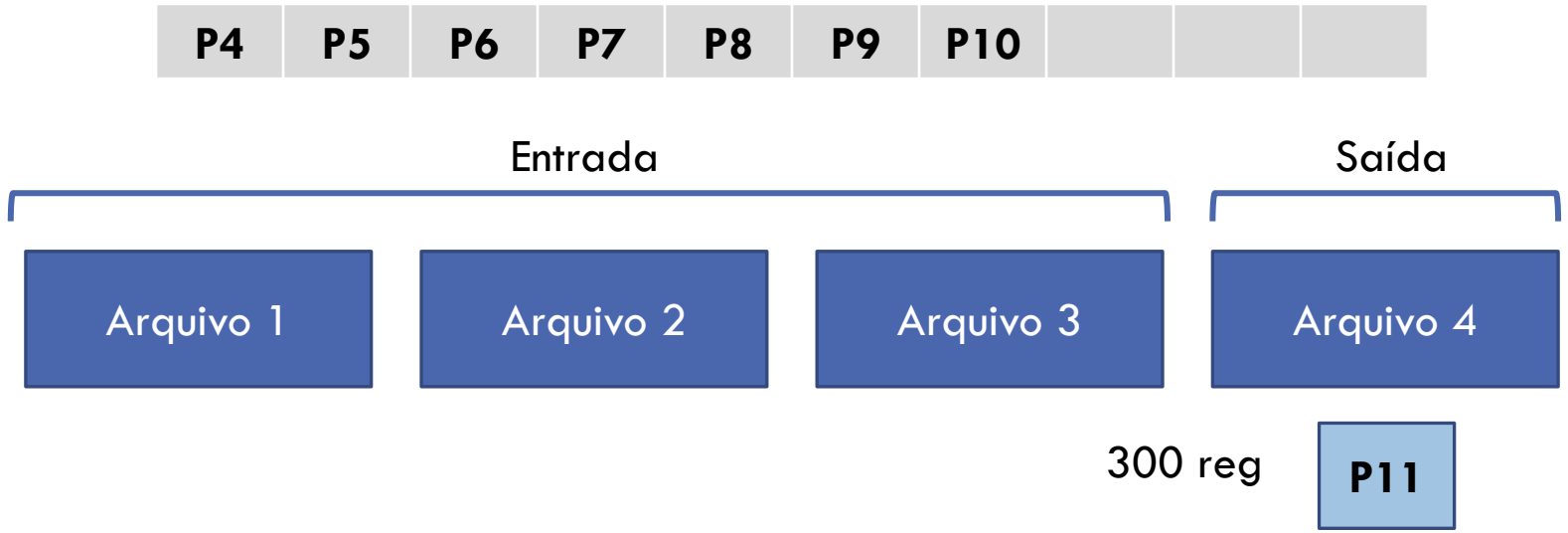
EXEMPLO (F = 4 E 10 PARTIÇÕES):



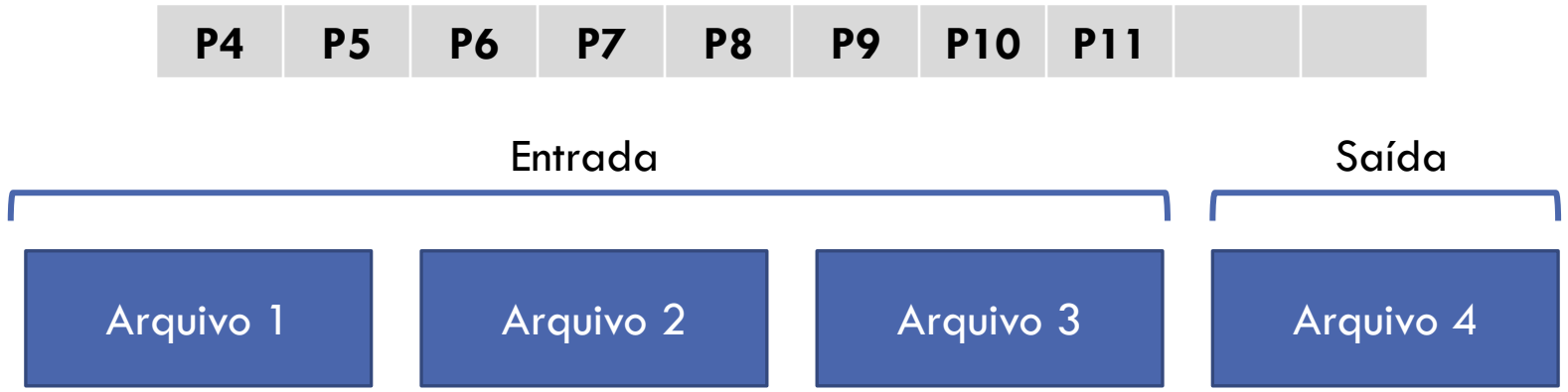
EXEMPLO (F = 4 E 10 PARTIÇÕES):



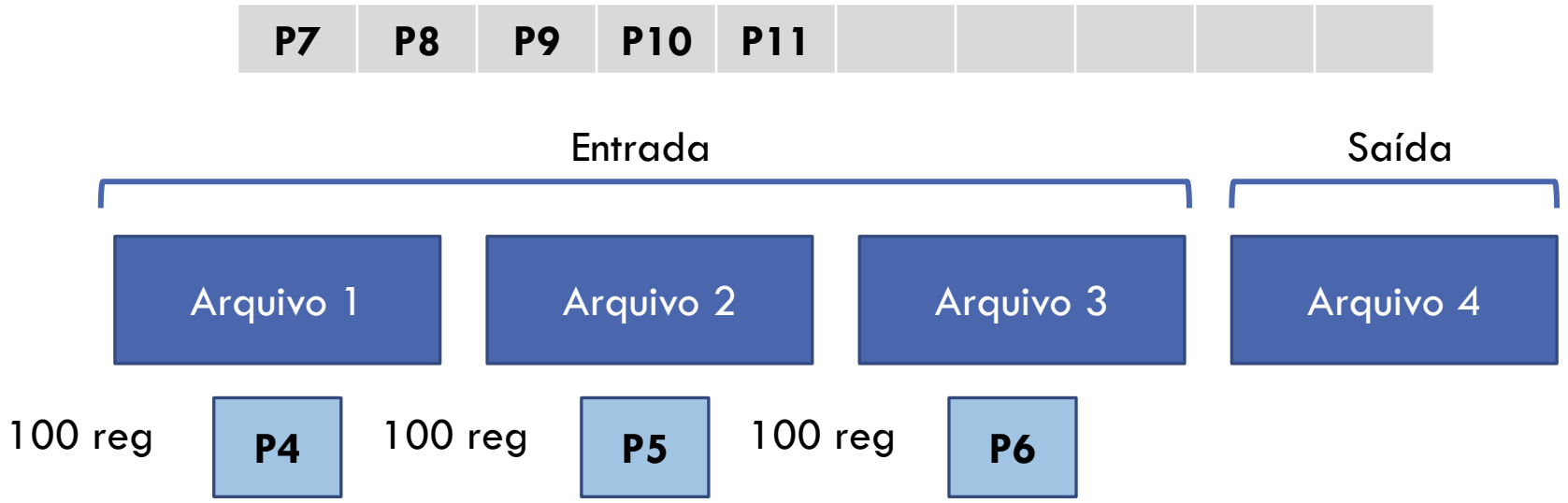
EXEMPLO (F = 4 E 10 PARTIÇÕES):



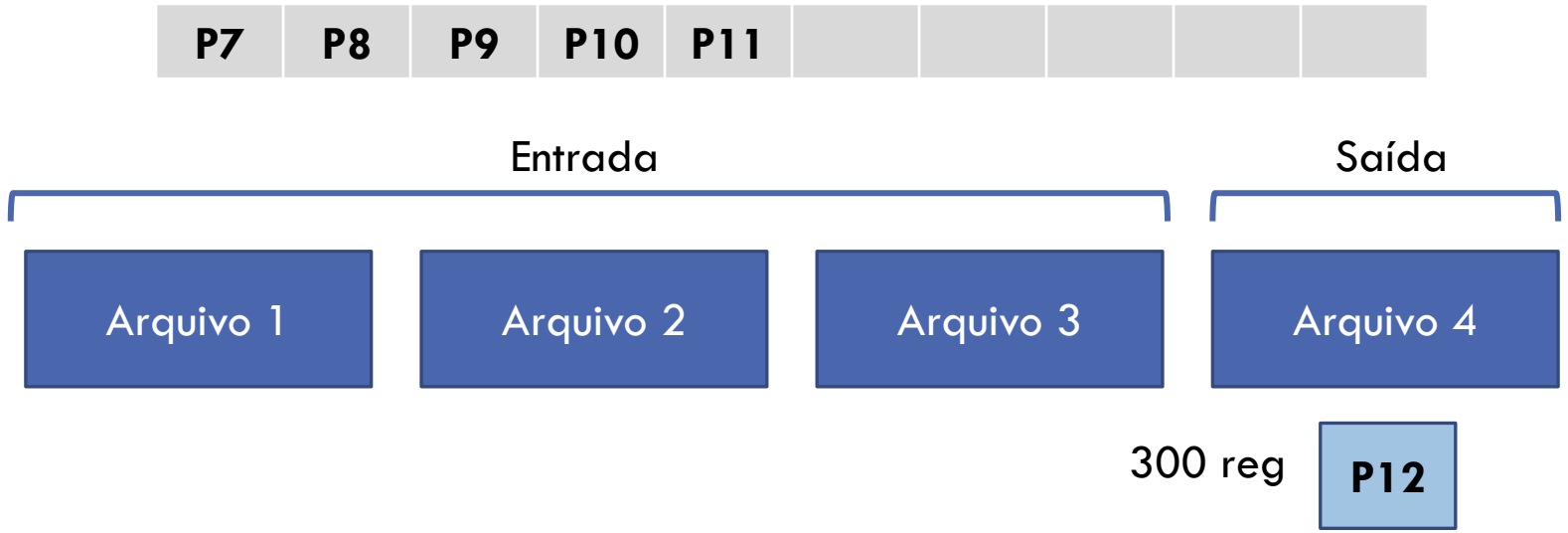
EXEMPLO (F = 4 E 10 PARTIÇÕES):



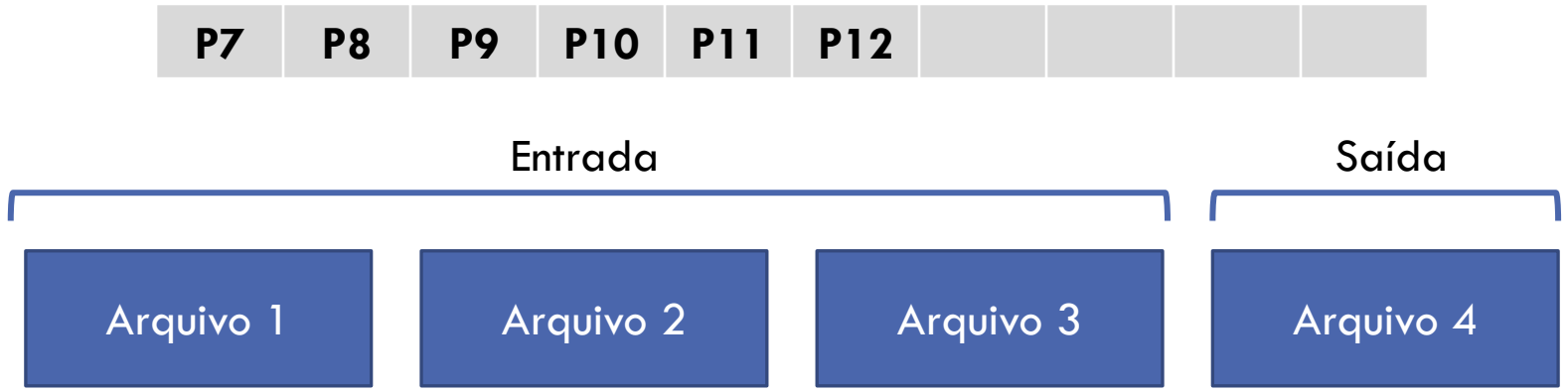
EXEMPLO (F = 4 E 10 PARTIÇÕES):



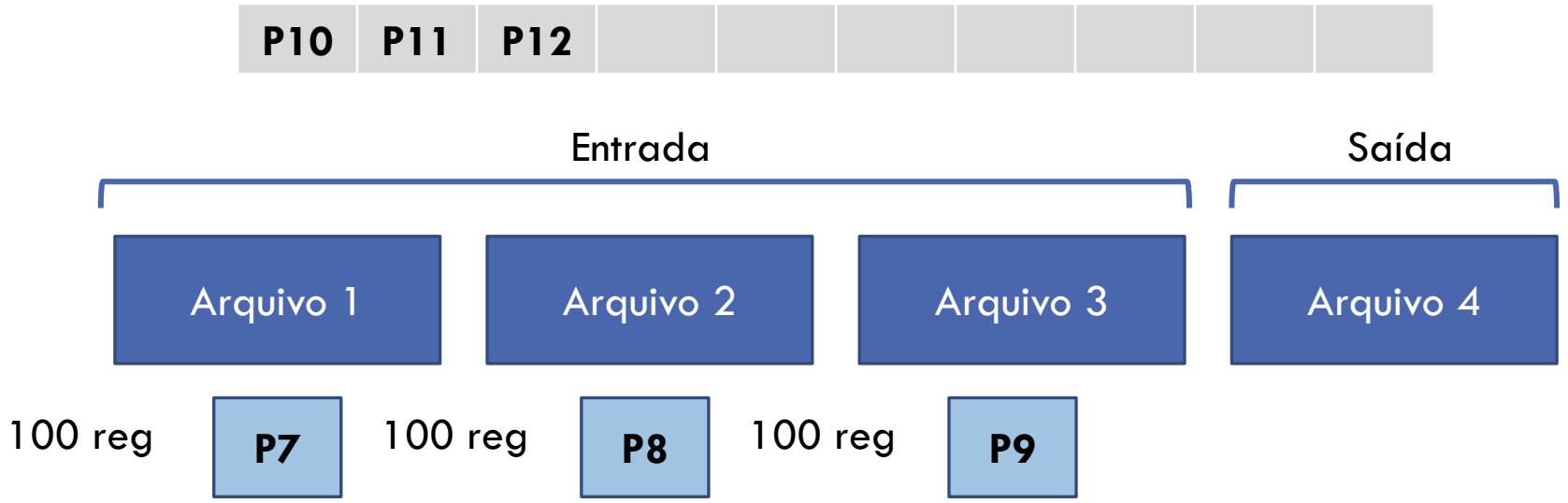
EXEMPLO (F = 4 E 10 PARTIÇÕES):



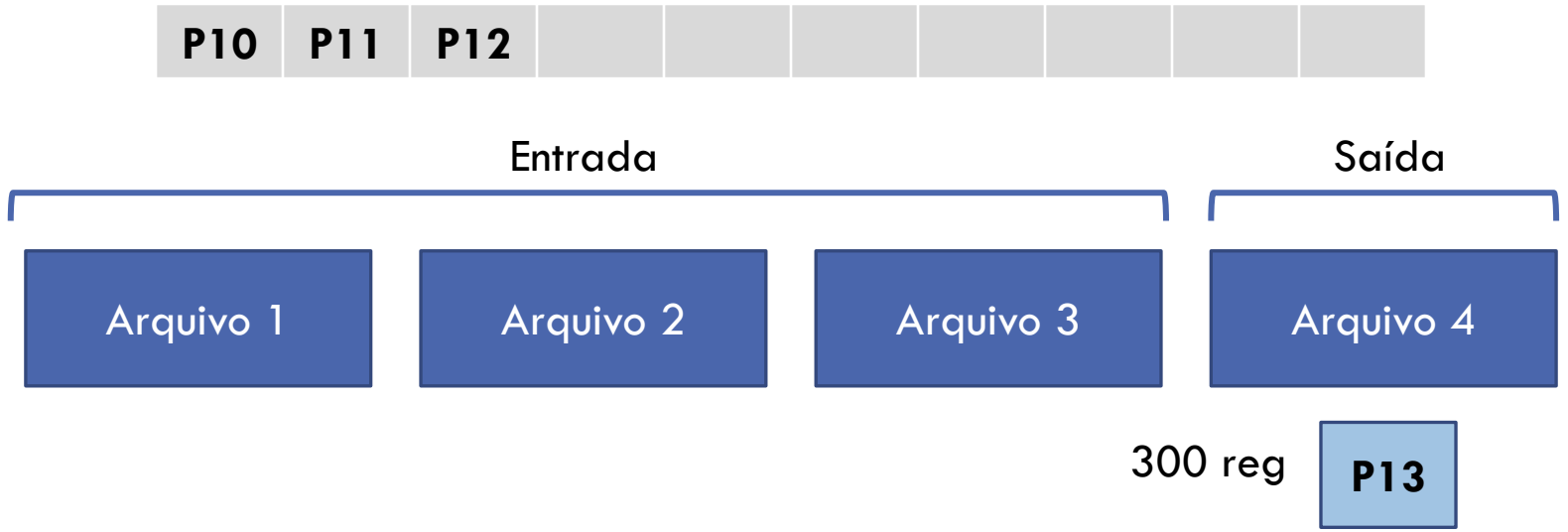
EXEMPLO (F = 4 E 10 PARTIÇÕES):



EXEMPLO (F = 4 E 10 PARTIÇÕES):

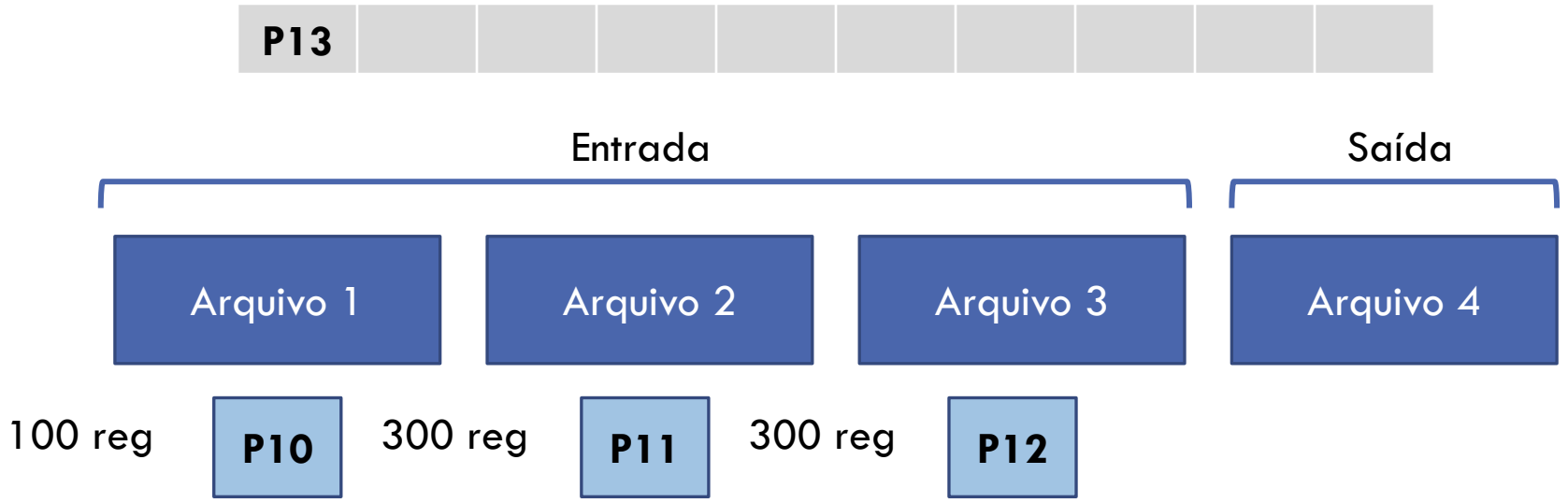


EXEMPLO (F = 4 E 10 PARTIÇÕES):

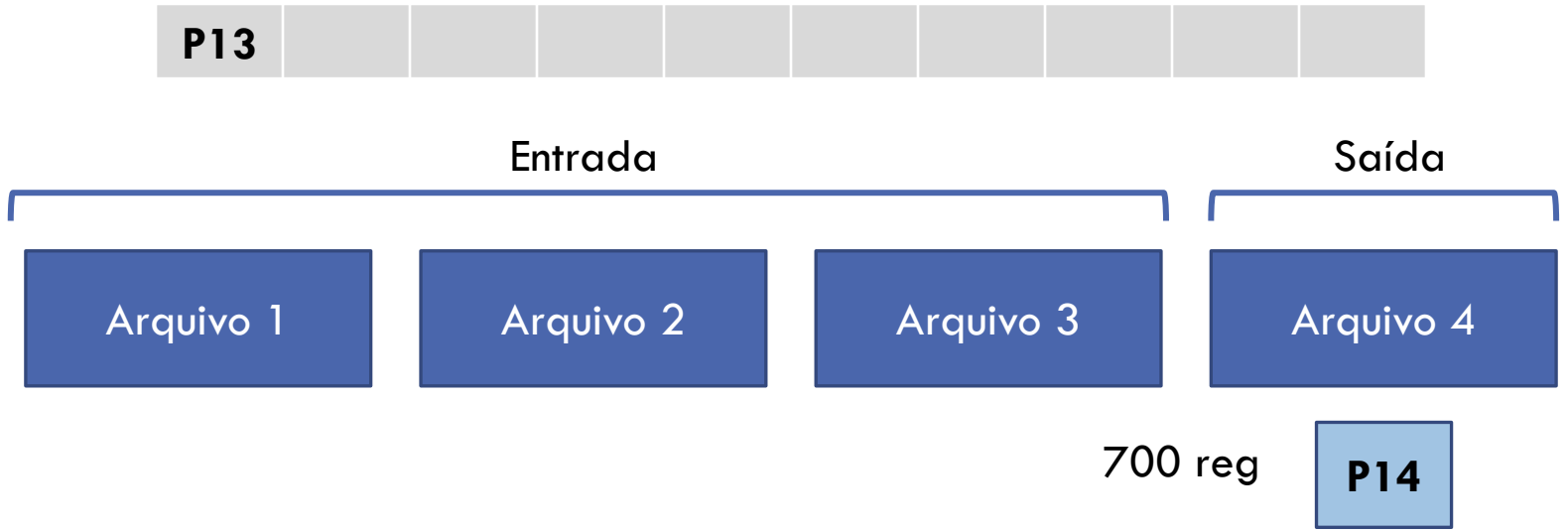


EXEMPLO (F = 4 E 10 PARTIÇÕES):

EXEMPLO (F = 4 E 10 PARTIÇÕES):

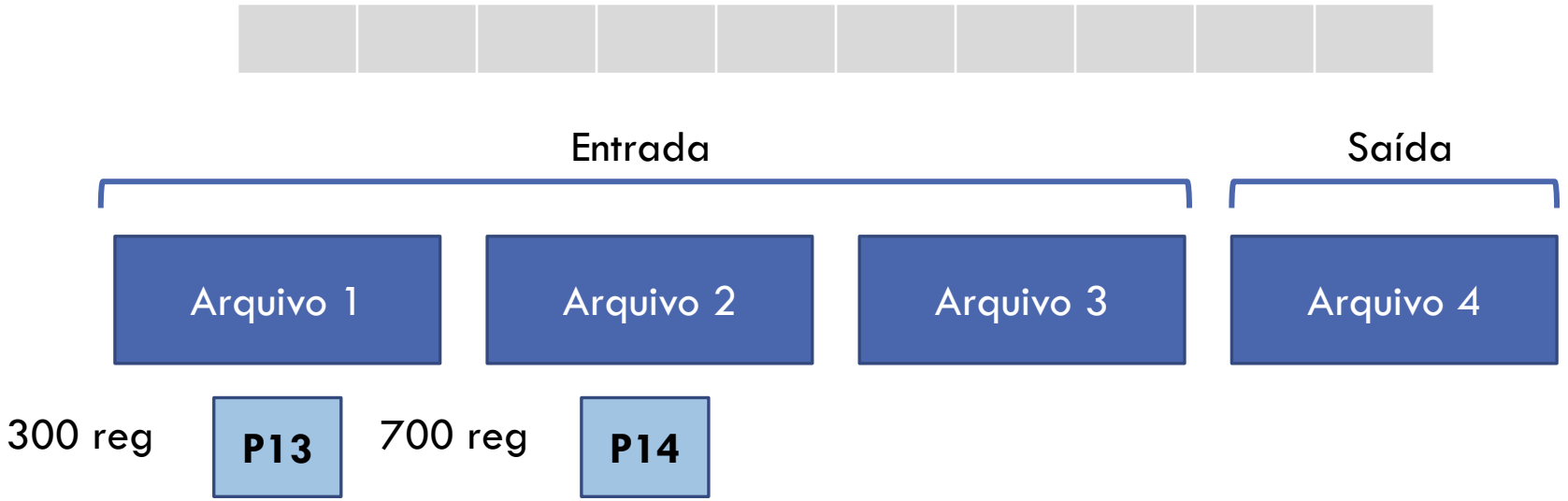


EXEMPLO (F = 4 E 10 PARTIÇÕES):

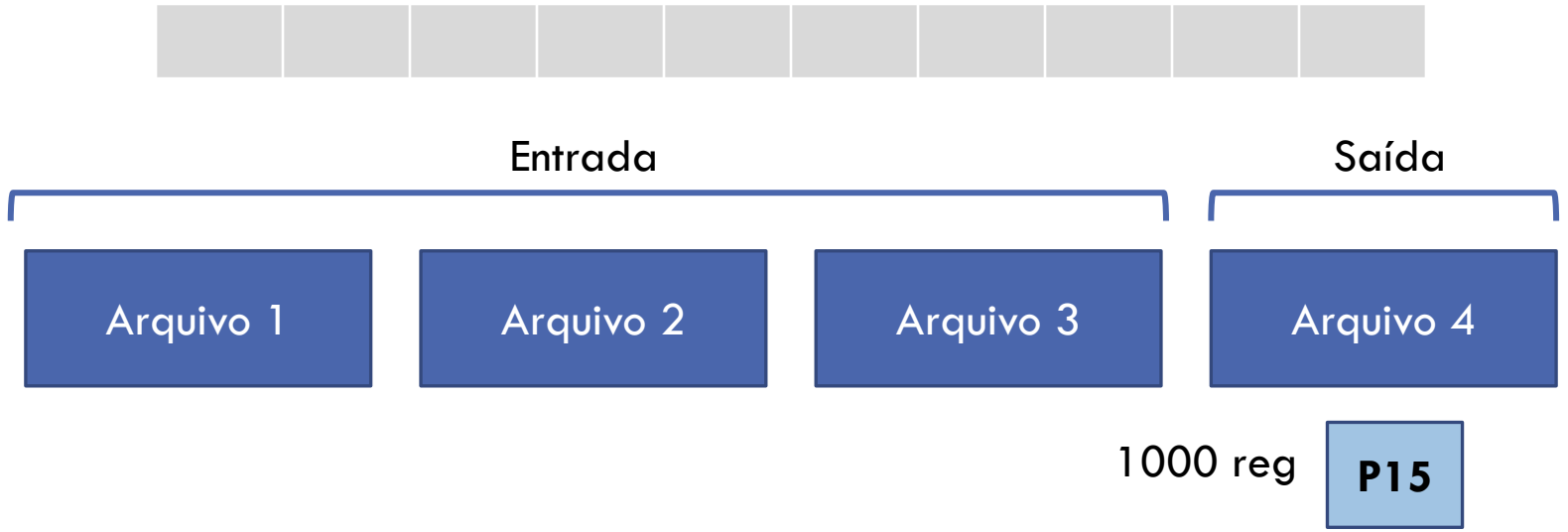


EXEMPLO (F = 4 E 10 PARTIÇÕES):

EXEMPLO (F = 4 E 10 PARTIÇÕES):



EXEMPLO (F = 4 E 10 PARTIÇÕES):



EXEMPLO (F = 4 E 10 PARTIÇÕES):

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

ESCOLHA DO MELHOR MÉTODO

Seleção com Substituição ou Seleção Natural?

Depende do cenário:

- Usar a estimativa de tamanho das partições para decidir
- Verificar se há restrições quanto ao número de arquivos que podem ser manipulados ao mesmo tempo para decidir entre Intercalação Ótima ou Árvore de Vencedores

CENÁRIO 1

Arquivo com 400.000 registros de cliente

Restrições

- Limite de memória: 20.000 registros (M = 20.000)
- Apenas 10 arquivos podem ser manipulados ao mesmo tempo

CENÁRIO 1 – GERAÇÃO DE PARTIÇÕES

Seleção com Substituição

- Partições de tamanho médio 2 * M = 40.000
- Número de partições geradas = 400.000 / 40.000 = 10
- Custo de I/O = 400.000

Seleção Natural

- Partições de tamanho médio e * M = 2,718 * 20.000 = 54.360
- Número de partições geradas = 400.000 / 54.360 = 8
- Custo de I/O = 400.000 + custo do reservatório 8-1 vezes = 400.000 + 20.000 * 7 = 540.000

CENÁRIO 1 – INTERCALAÇÃO

Seleção com Substituição

- 10 partições, 2 fases
- FASE 1 = 40.000 * 9 = 360.000
- FASE 2 = 360.000 + 40.000 = 400.000
- TOTAL = 760.000

Seleção Natural

- 8 partições, 1 fase
- FASE 1 = 400.000

CENÁRIO 1 - TOTAIS

Seleção com Substituição

- Geração + Intercalação = 400.000 + 760.000 = 1.160.000

Seleção Natural

- Geração + Intercalação = 540.000 + 400.000 = 940.000

Método escolhido para geração das partições: **Seleção Natural**

CENÁRIO 2

Arquivo com 400.000 registros de cliente

Restrições

- Limite de memória: 10.000 registros (M = 10.000)
- Apenas 10 arquivos podem ser manipulados ao mesmo tempo

CENÁRIO 2 – GERAÇÃO DE PARTIÇÕES

Seleção com Substituição

- Partições de tamanho médio 2 * M = 20.000
- Número de partições geradas = 400.000 / 20.000 = 20
- Custo de I/O = 400.000

Seleção Natural

- Partições de tamanho médio e * M = 2,718 * 10.000 = 27.180
- Número de partições geradas = 400.000 / 27180 = 15
- Custo de I/O = 400.000 + custo do reservatório 15-1 vezes = 400.000 + 10.000 * 14 = 540.000

CENÁRIO 2 – INTERCALAÇÃO

Seleção com Substituição

- 20 partições, 3 fases
- FASE 1 = 20.000 * 9 = 180.000
- FASE 2 = 20.000 * 9 = 180.000
- FASE 3 = 180.000 + 180.000 + 20.000 + 20.000 = 400.000
- TOTAL = 760.000

Seleção Natural

- 15 partições, 2 fases
- FASE 1 = 27.180 * 9 = 244.620
- FASE 2 = 27.180 * 6 + 244.620 = 407.700
- TOTAL = 652.320

CENÁRIO 2 - TOTAIS

Seleção com Substituição

- Geração + Intercalação = 400.000 + 760.000 = 1.160.000

Seleção Natural

- Geração + Intercalação = 540.000 + 652.320 = 1.192.320

Método escolhido para geração das partições: **Seleção com Substituição**

EXERCÍCIO

Implementar Intercalação Ótima

Usar o arquivo fornecido em aula, que já contém funções auxiliares e os casos de testes automatizados

REFERÊNCIA

Ferraz, I. N. Programação com Arquivos. Editora Manole Ltda. Barueri, 2003.