

PRA – Projeto de Arquivos

Classificação em memória
principal e secundária

Prof. Allan Rodrigo Leite

Programas usando classificação

- Relembrando um desafio de arquivo sequencial
 - Acesso não pode ser feito em posições aleatórias do arquivo
 - Exemplo: para ler o décimo registro é necessário ler os nove registros prévios
 - E se o arquivo conter 10^9 registros e for necessário ler o último registro?
- Programas podem gastar muito recurso computacional e tempo em atividades de classificação de registros
 - Ordenação, pesquisas, agrupamento de dados, dentre outros
 - Para simplificar, usaremos o termo classificação como sinônimo de ordenação

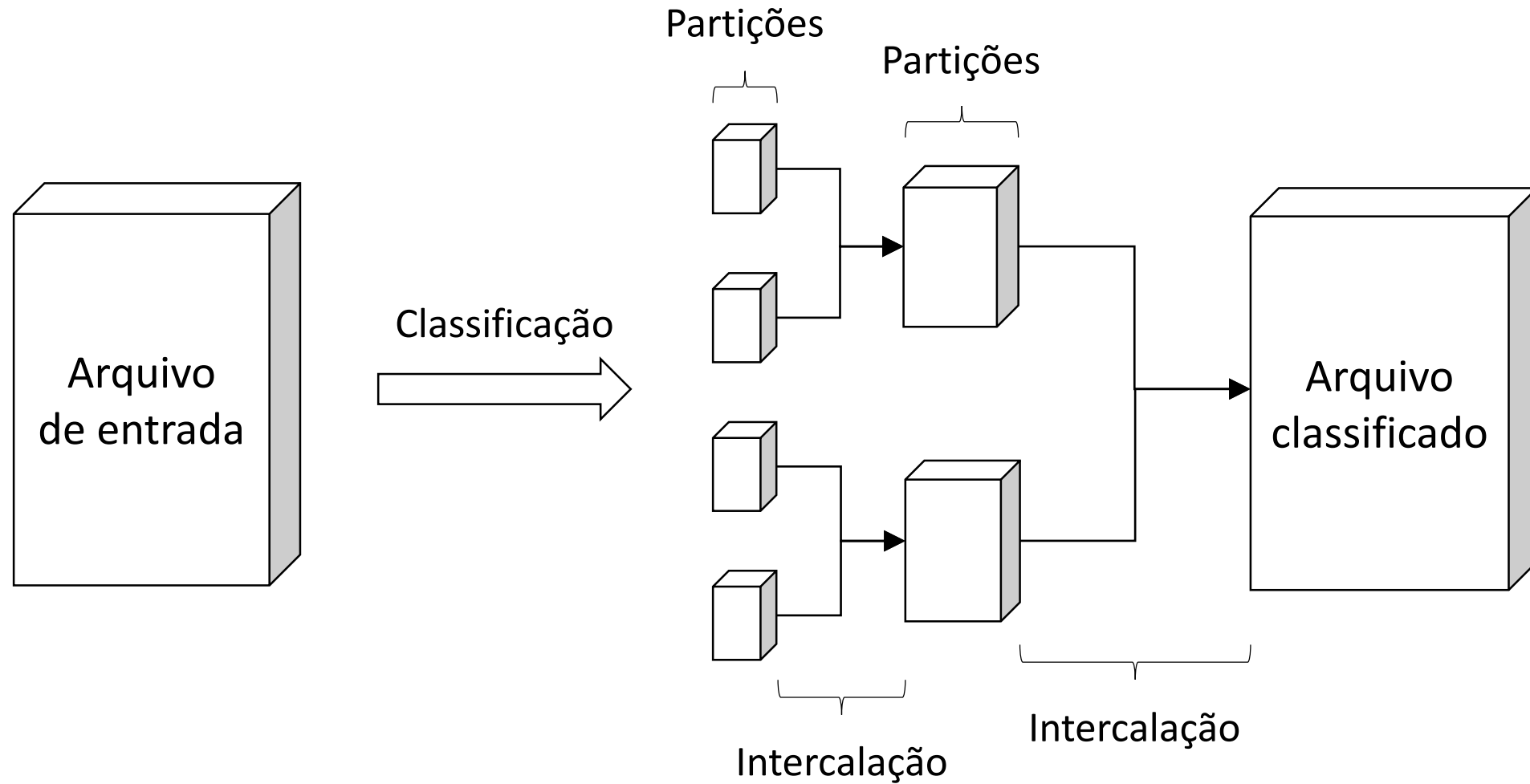
Tipos de classificação

- A classificação de registros de um arquivo pode ser de dois tipos
 - **Classificação interna:** utiliza exclusivamente a memória principal
 - Lista de registros pode ser mantida inteiramente em memória principal
 - **Classificação externa:** utiliza a memória secundária
 - Lista de registros é maior do que a capacidade de armazenamento da memória principal
- Na classificação externa é fundamental minimizar o número de operações de entrada e saída na memória secundária
 - Visto que esta memória possui um esforço computacional maior para acesso

Tipos de classificação

- Classificação externa
 - É possível realizar a classificação de um arquivo sobre ele mesmo ou utilizar um armazenamento auxiliar
 - A classificação sobre o próprio arquivo economiza espaço
 - Corre risco de inconsistência em caso de término anormal de processamento
- Estratégia da classificação externa
 - Dividir o arquivo em pequenas frações que são ordenadas e intercaladas em dois estágios: classificação e intercalação

Estratégia da classificação externa



Estratégia da classificação externa

- Definição

- Arquivo de entrada é um arquivo não classificado
- Arquivo de saída é um arquivo classificado
- Uma partição é uma sequencia classificada de n registros

- Procedimento da classificação

- Registros são lidos de arquivos de entrada e de partições
- Estes registros são classificados e gravados em arquivos de saída ou partições

Estágio da classificação

- Métodos
 - Classificação interna
 - Seleção com substituição
 - Seleção natural
- Considera-se que
 - Memória principal tem capacidade para armazenar M registros do arquivo
 - Arquivo não classificado possui N registros

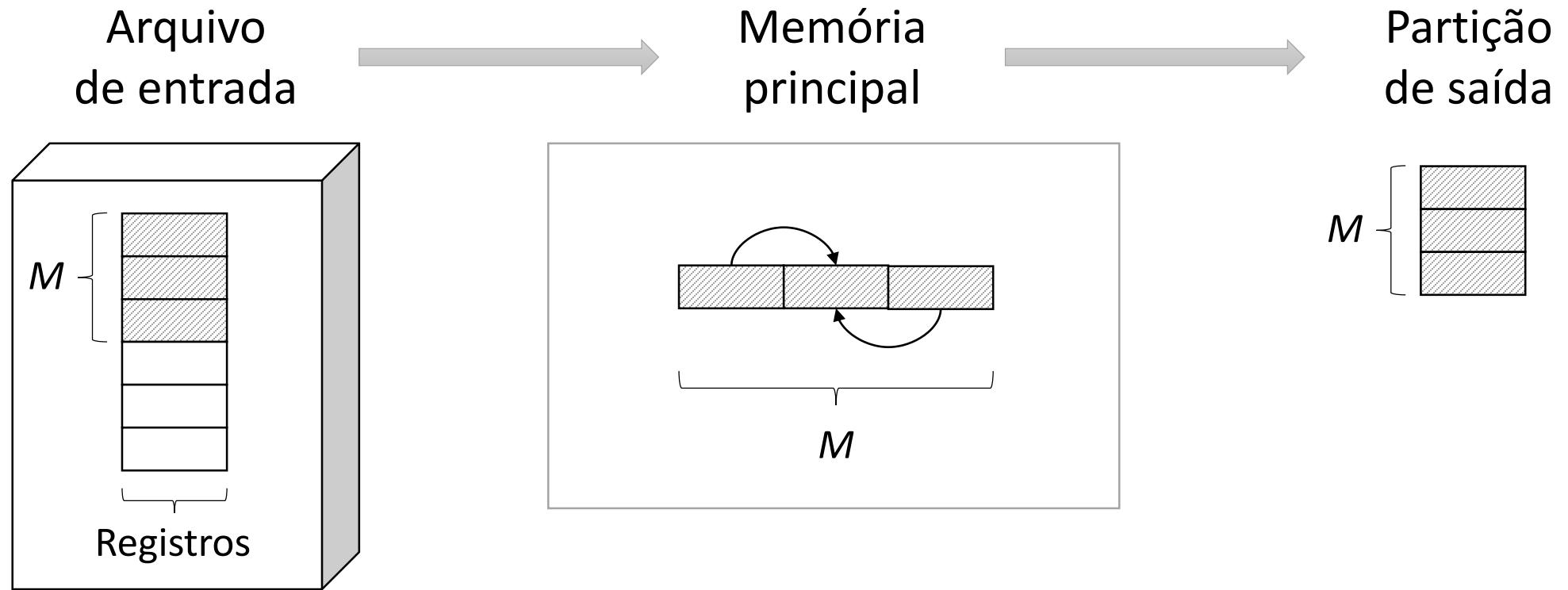
Classificação interna

- Critério fundamental de eficiência da classificação interna
 - Número de comparações entre chaves de registros
- Consiste na leitura de M registros para a memória
 - Classificação desses registros por qualquer processo de classificação interna
 - Gravação desses registros classificados em uma partição
- Todas as partições classificadas contêm M registros, com exceção da última que eventualmente possua menos registros

Classificação interna

- Técnicas utilizadas para classificação
 - Seleção: *bubble sort*, *selection sort*, *heap sort*
 - Dividir para conquistar: *merge sort*, *quick sort*
 - Inserção: *insertion sort*, *shell sort*
 - Linear: *bucket sort*, *radix sort*
- Complexidade computacional pode variar
 - <https://www.bigocheatsheet.com>

Classificação interna



Classificação interna

- Chaves do arquivo não classificado
 - Sequência de leitura da esquerda para direita: 29, 14, 76, ...
- Assumir que a capacidade de memória principal são 6 registros

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Classificação interna

Memória principal

$M = 6$ {

29	14	76	75	59	6
----	----	----	----	----	---



29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Classificação interna

Memória principal

Ordenado {

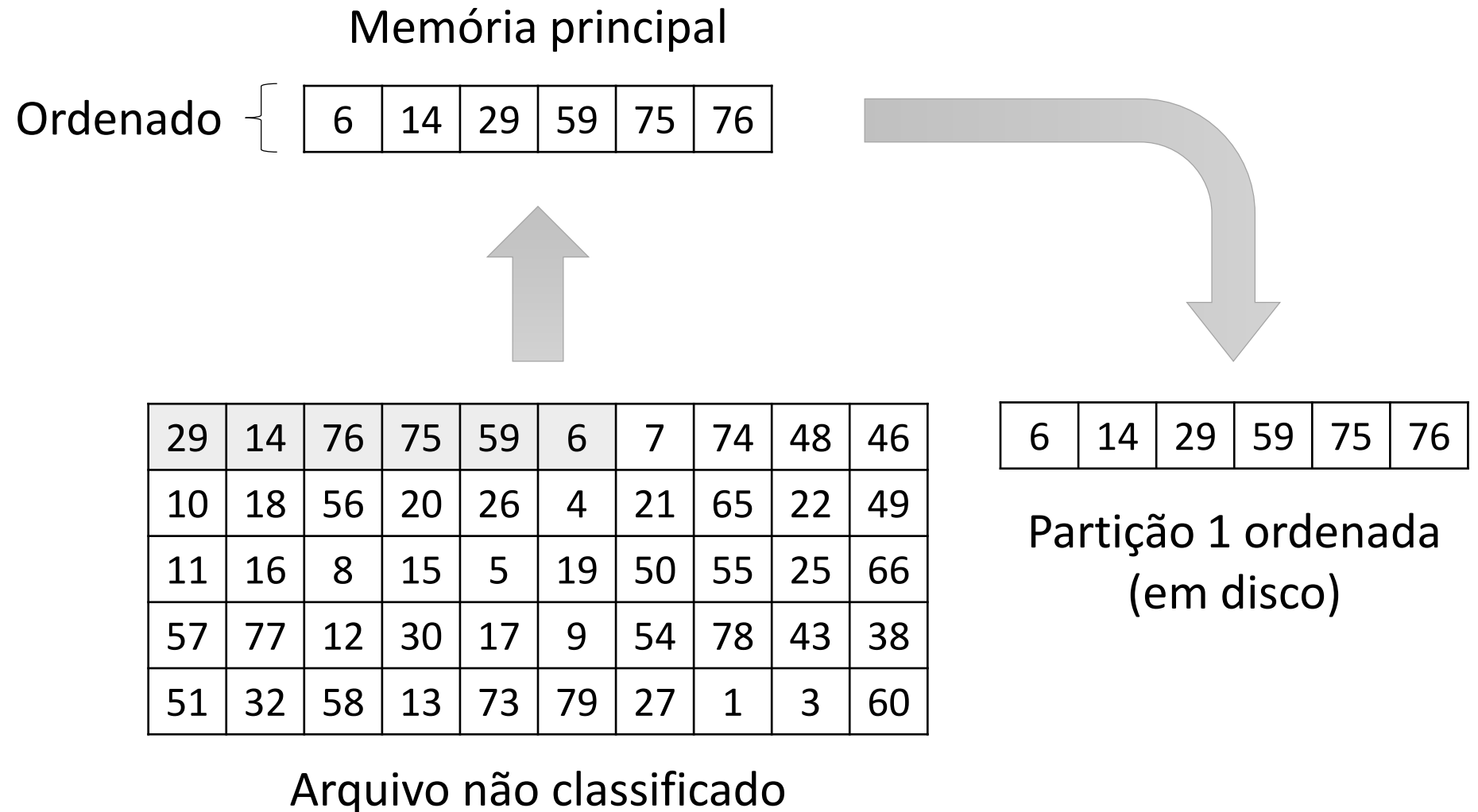
6	14	29	59	75	76
---	----	----	----	----	----



29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Classificação interna



Seleção com substituição

- Objetivo: aproveitar uma possível classificação parcial do arquivo de entrada
- Algoritmo
 1. Ler M registros do arquivo para a memória
 2. Selecionar no vetor em memória o registro r com menor chave
 3. Gravar o registro r na partição de saída
 4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
 5. Se a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
 6. Caso existam em memória registros **não congelados**, voltar ao passo 2
 7. Caso contrário:
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Seleção com substituição

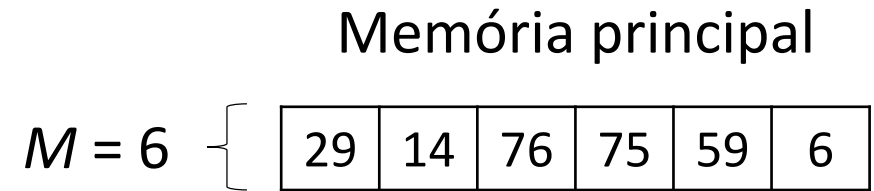
- Chaves do arquivo não classificado
 - Sequência de leitura da esquerda para direita: 29, 14, 76, ...
- Assumir que a capacidade de memória principal são 6 registros

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2



29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Memória principal

$r = 6$ {

29	14	76	75	59	6
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Memória principal

$r = 6$

29	14	76	75	59	6
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. **Caso existam em memória registros não congelados, voltar ao passo 2**
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Memória principal

$r = 7$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Memória principal

$r = 7$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. **Caso existam em memória registros não congelados, voltar ao passo 2**
7. Caso contrário
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave
3. Gravar o registro r na partição de saída
4. Substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Caso a chave deste último for menor do que a chave recém gravada, considera-lo **congelado** e ignora-lo no restante do processamento
6. Caso existam em memória registros **não congelados**, voltar ao passo 2
7. **Caso contrário**
 - Fechar a partição de saída
 - Descongelar os registros congelados
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7	14	29	46	48	59	74	75	76
---	---	----	----	----	----	----	----	----	----

Memória principal

$r = 76$

10	18	4	26	56	20
----	----	---	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção com substituição

- Tamanho das partições geradas
 - Em média, o tamanho das partições obtidas pelo processo de seleção com substituição é de $2 \times M$

Seleção natural

- Desvantagem da seleção com substituição
 - No final da partição grande parte do espaço em memória principal está ocupado com registros congelados
- Seleção natural
 - Reserva-se um espaço de memória secundária (o reservatório) para abrigar os registros congelados num processo de substituição
 - A formação de uma partição se encerra quando o reservatório estiver cheio ou quando terminarem os registros de entrada
 - Para a memória com capacidade de M registros, supõe-se um reservatório comportando n registros
 - Considerando $M = n$, o comprimento médio das partições é de $M \times e$, onde $e = 2,718$

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Seleção natural

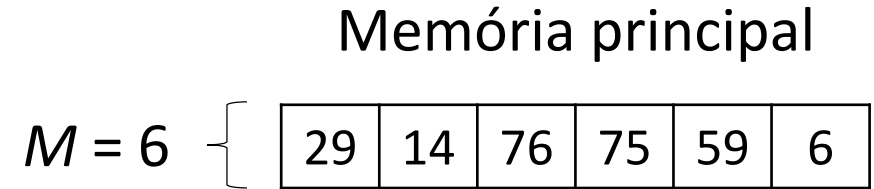
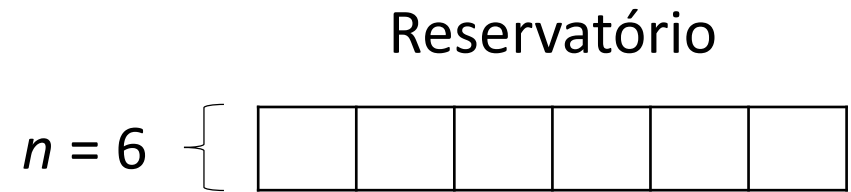
- Chaves do arquivo não classificado
 - Sequência de leitura da esquerda para direita: 29, 14, 76, ...
- Assumir que a capacidade de memória e reservatório são 6 registros
 - $M = 6$ e $n = 6$

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2



29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Reservatório

--	--	--	--	--	--

Memória principal

$r = 6$	{	29	14	76	75	59	6
---------	---	----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Reservatório

--	--	--	--	--	--

Memória principal

$r = 6$

29	14	76	75	59	6
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Reservatório

--	--	--	--	--	--

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Reservatório

--	--	--	--	--	--

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. **Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2**
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Reservatório

--	--	--	--	--	--

Memória principal

$r = 6$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6

Reservatório

--	--	--	--	--	--

Memória principal

$r = 7$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Reservatório

--	--	--	--	--	--

Memória principal

$r = 7$

29	14	76	75	59	7
----	----	----	----	----	---

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Reservatório

--	--	--	--	--	--

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Reservatório

--	--	--	--	--	--

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7
---	---

Reservatório

--	--	--	--	--	--

Memória principal

$r = 7$

29	14	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Seleção natural

1. Ler M registros do arquivo para a memória
2. Selecionar no vetor em memória o registro r com menor chave ainda não gravado na partição de saída
3. Gravar o registro r na partição de saída
4. Caso o reservatório não estiver cheio, substituir no vetor em memória o registro r pelo próximo registro do arquivo de entrada
5. Enquanto a chave deste último for menor do que a chave recém gravada, gravá-lo no reservatório e substituir, no vetor em memória, o registro r pelo próximo registro do arquivo de entrada
6. Caso ainda exista espaço livre no reservatório ou exista registros na memória ainda não gravados na partição de saída, voltar ao passo 2
7. Caso contrário
 - Fechar a partição de saída
 - Copiar os registros do reservatório para o vetor em memória
 - Esvaziar o reservatório
 - Abrir nova partição de saída
 - Voltar ao passo 2

Partição de saída

6	7	14	29	46	48	56	59	74	75	76
---	---	----	----	----	----	----	----	----	----	----

Reservatório

10	18	20	26	4	21
----	----	----	----	---	----

Memória principal

56	48	76	75	59	74
----	----	----	----	----	----

29	14	76	75	59	6	7	74	48	46
10	18	56	20	26	4	21	65	22	49
11	16	8	15	5	19	50	55	25	66
57	77	12	30	17	9	54	78	43	38
51	32	58	13	73	79	27	1	3	60

Arquivo não classificado

Intercalação de partições classificadas

- Objetivo da intercalação
 - Transformar um conjunto de partições classificadas em um único arquivo
 - Deve conter todos os registros de todas as partições originais do conjunto
 - O arquivo gerado deve estar classificado seguindo o mesmo critério de classificação das partições iniciais
- Considere a existência de R partições geradas pelo processo de geração de partições anterior
 - A intercalação vai exigir uma série de etapas cujos registros são lidos de um conjunto de partições e gravados em outra

Intercalação de partições classificadas

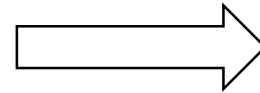
- Algoritmo básico
 - Basta ter em memória um registro para cada um dos arquivos a intercalar
- Considera-se cada arquivo como uma pilha
 - Topo da pilha: registro em memória
 - Para cada iteração do algoritmo, o topo da pilha contendo a menor chave é gravado no arquivo de saída e é substituído pelo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

Intercalação de partições classificadas

- Algoritmo básico

Intercalação						
31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

Cada coluna representa uma
partição



2	3	6	7	10	13
14	15	20	22	31	40
41	51	55	60	60	69
70	73	76	77	79	80

Arquivo classificado

Intercalação de partições classificadas

- Algoritmo básico
 - E se a quantidade de arquivos a intercalar for muito grande?
 - Operação de busca da menor chave precisa ser repetida várias vezes até esgotar as pilhas que representam as partições
- Abordagem para otimização do processo
 - Árvore binária de vencedores

Árvore binária de vencedores

- Estrutura da árvore
 - Nós folha são as chaves que estão no topo das pilhas das partições a intercalar
 - Cada nó da árvore representa a menor chave entre seus dois filhos
 - A raiz representa o nó da árvore com a menor chave
- Cada nó da árvore contém
 - Vencedor: valor da menor chave da subárvore iniciada pelo nó atual
 - Endereço do vencedor: ponteiro para a partição que detém aquela chave
 - Esquerda: ponteiro para o nó filho à esquerda
 - Direita: ponteiro para o nó filho à direita

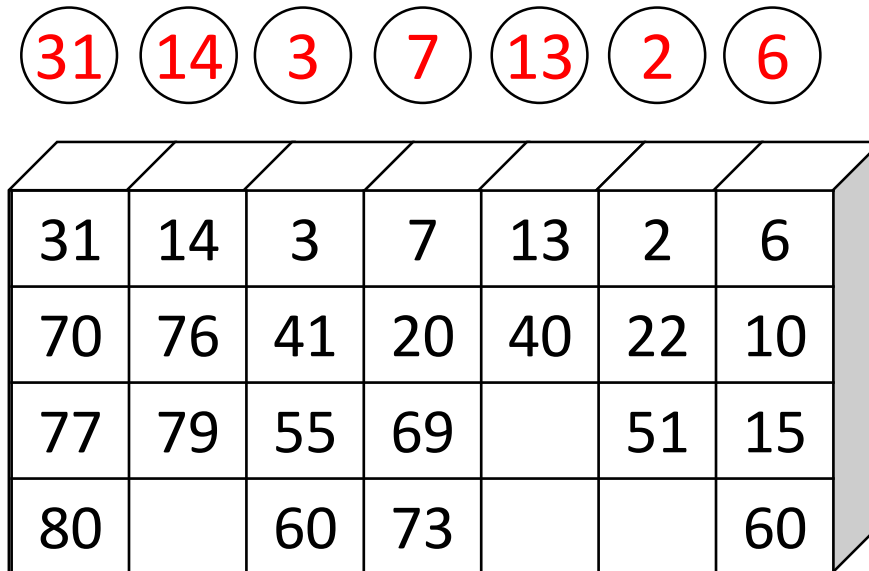
Árvore binária de vencedores

- Exemplo
 - Cada coluna abaixo representa uma partição com suas respectivas chaves

31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

Árvore binária de vencedores

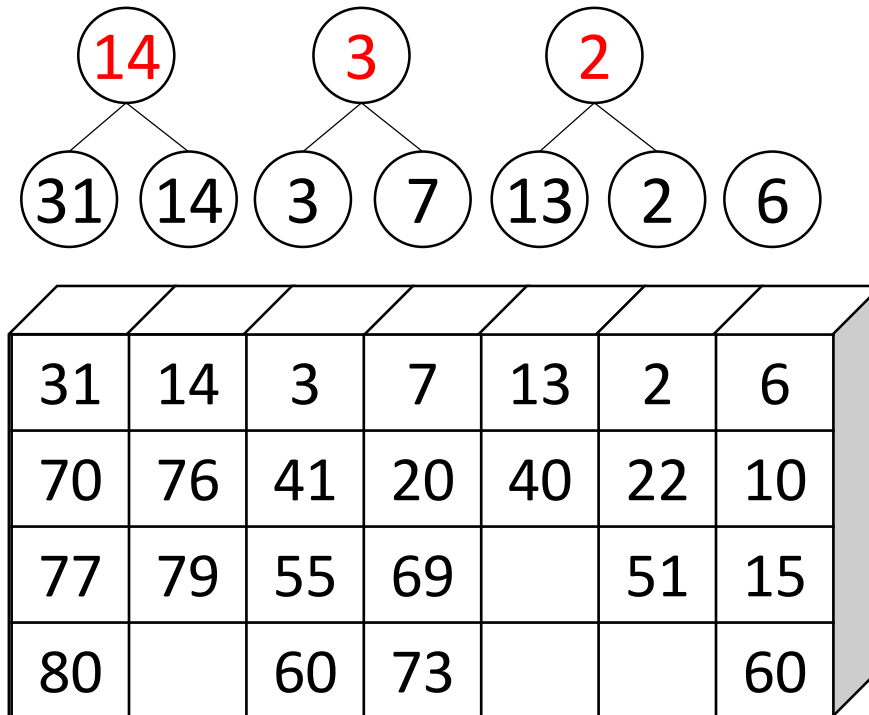
1. Colocar em memória o primeiro registro de cada partição
 - Cada registro é um nó folha da árvore



31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

Árvore binária de vencedores

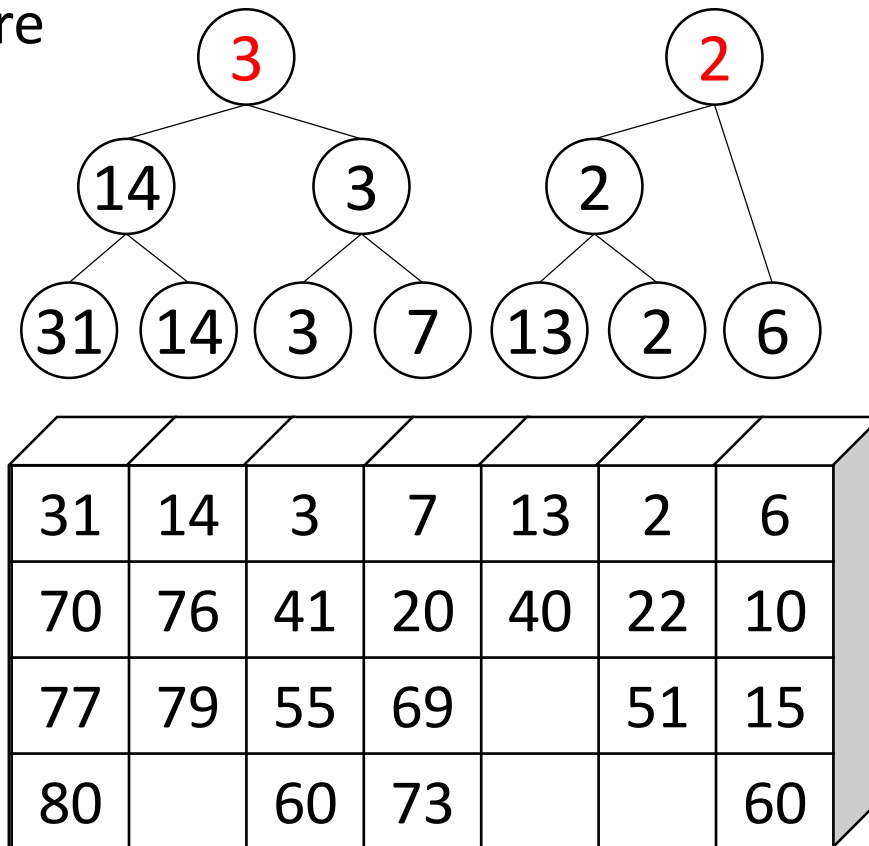
2. Criar um nó raiz para par de nós folha com o menor dos dois valores



Árvore binária de vencedores

3. Repetir o passo 2

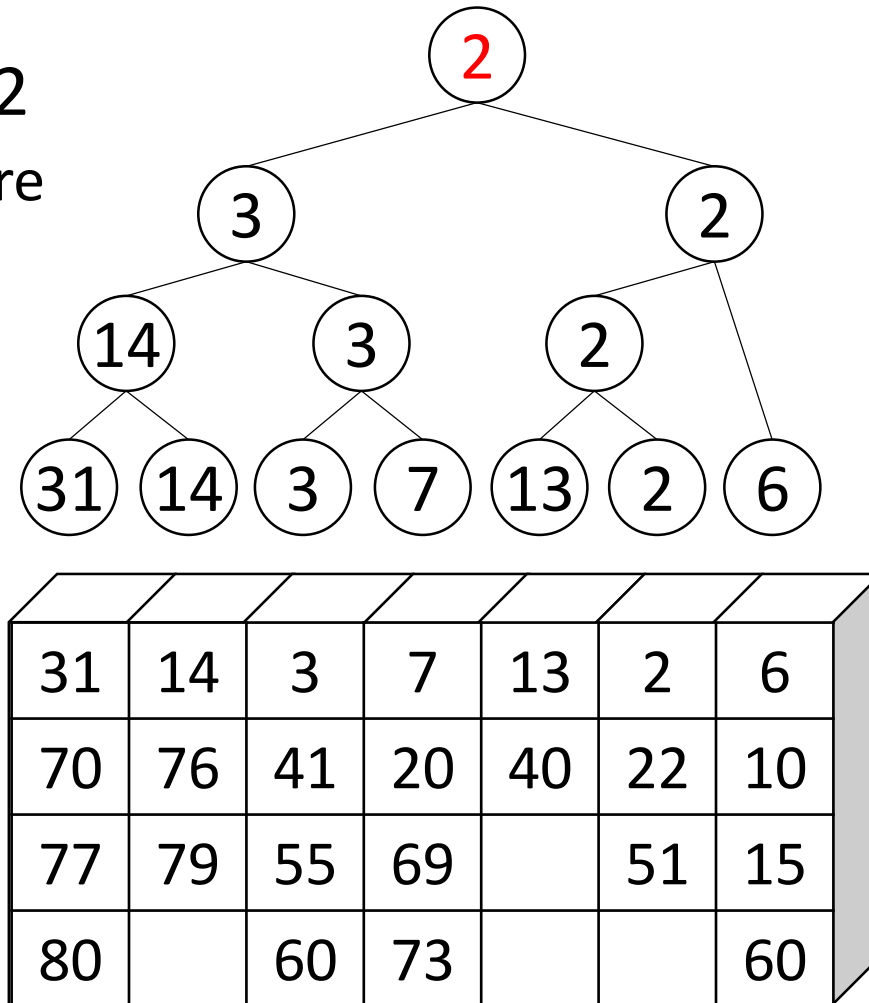
- Até a raiz da árvore



Árvore binária de vencedores

3. Repetir o passo 2

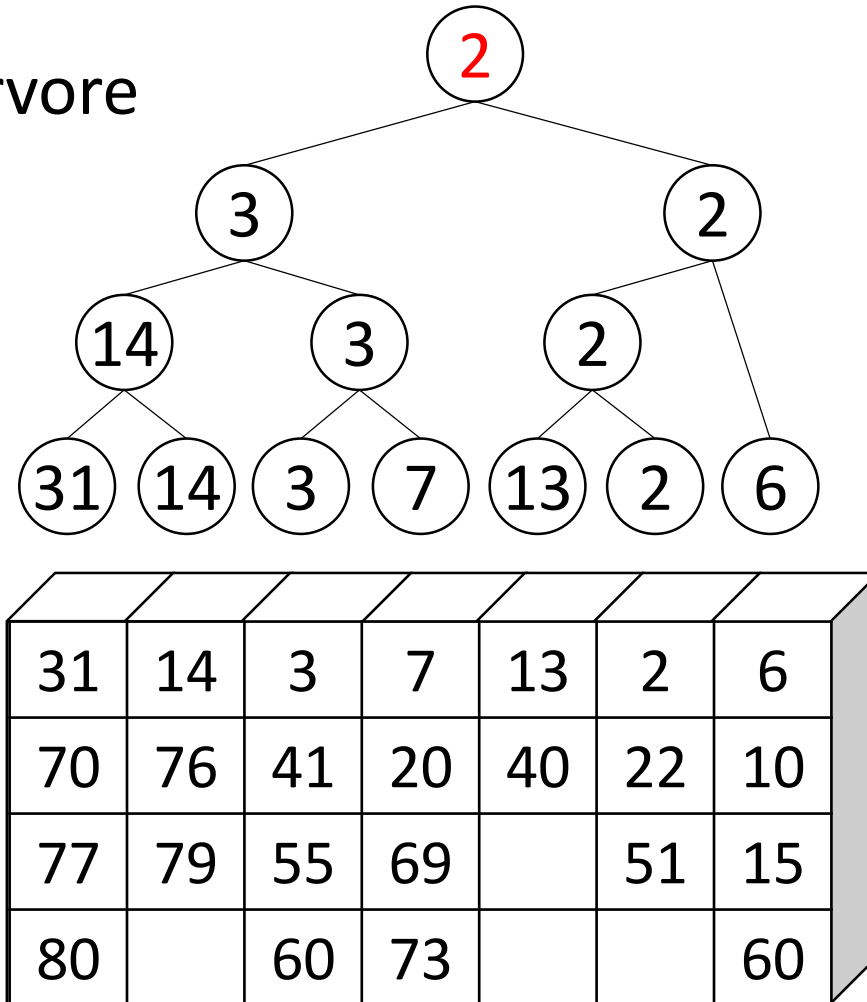
- Até a raiz da árvore



Árvore binária de vencedores

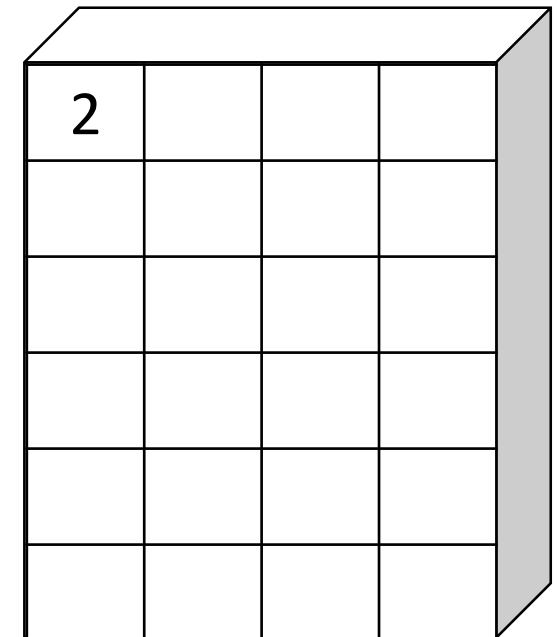
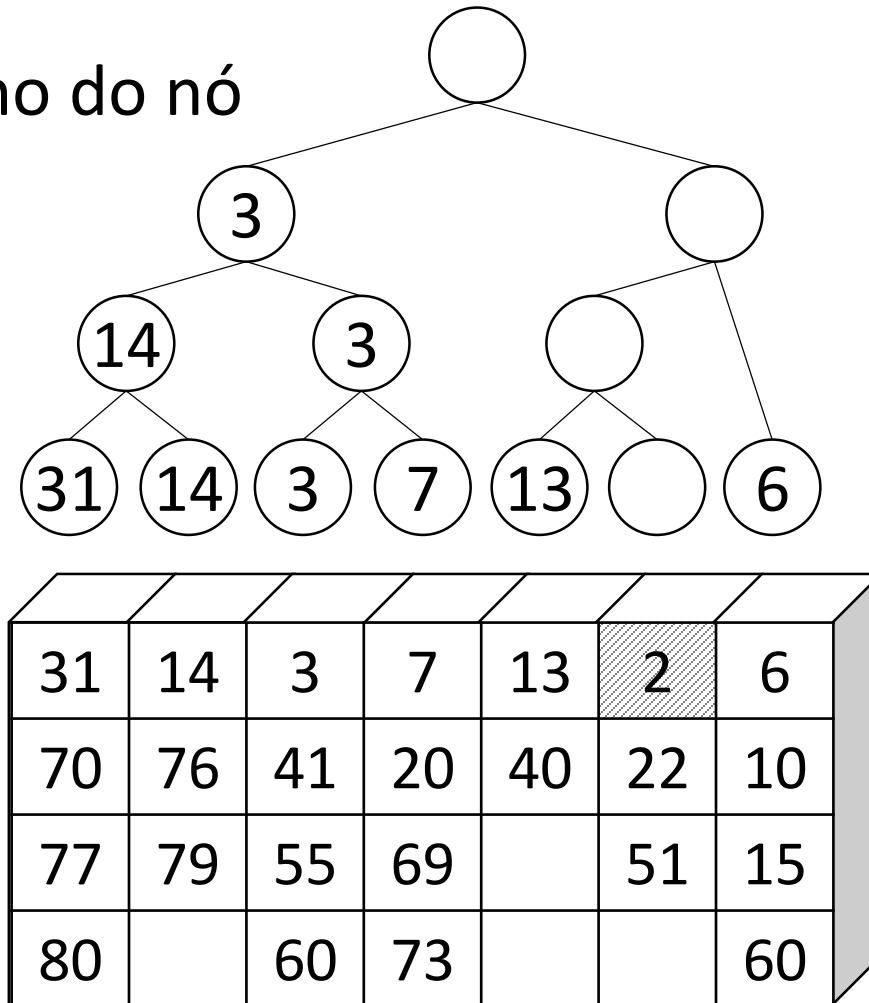
4. Retirar raiz da árvore

- Inserir no arquivo classificado



Árvore binária de vencedores

5. Atualizar caminho do nó retirado

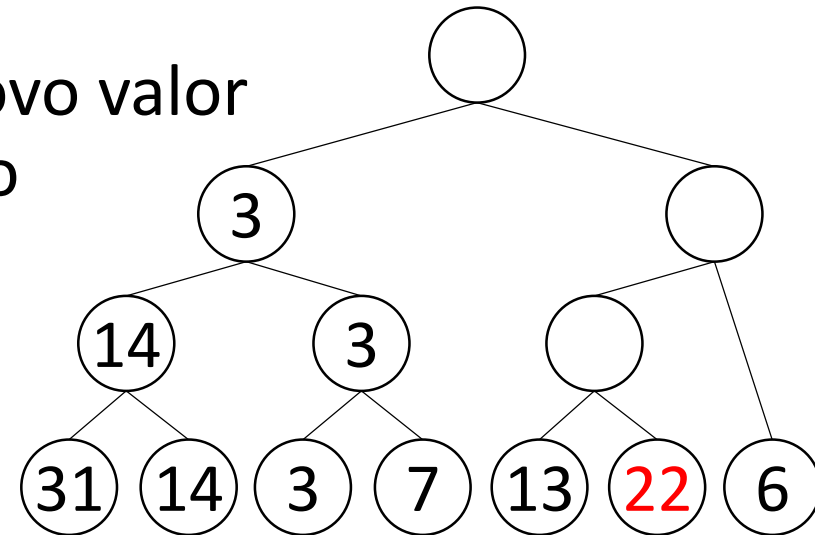


Arquivo
classificado

Árvore binária de vencedores

6. Adicionar um novo valor
ao nó folha vazio

- Caso necessário



31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

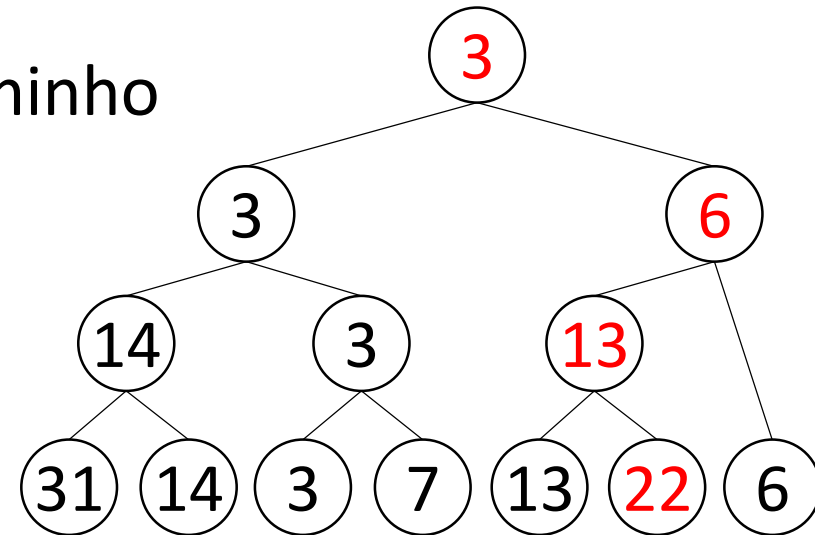
2			

Arquivo
classificado

Árvore binária de vencedores

7. Completar o caminho até o nó raiz

- Voltar ao passo 4



31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

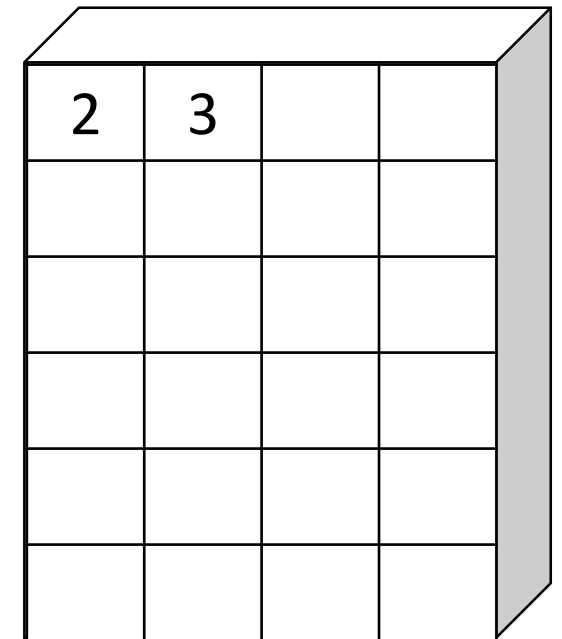
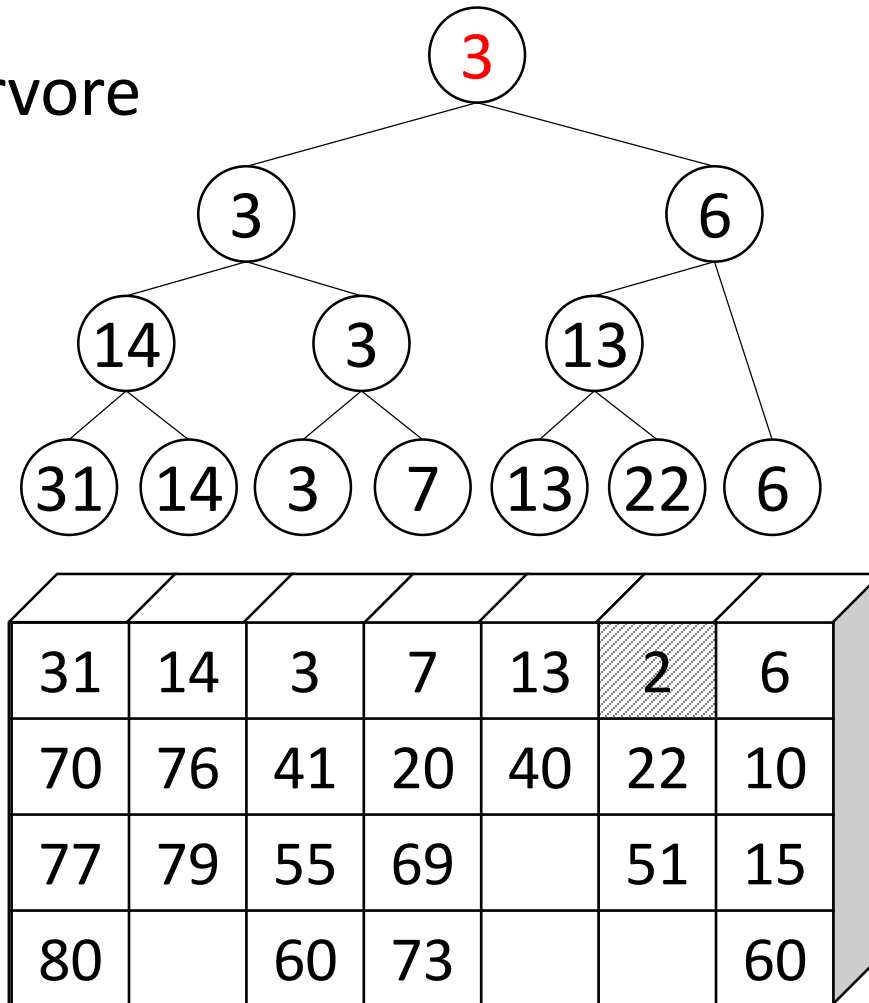
2			

Arquivo
classificado

Árvore binária de vencedores

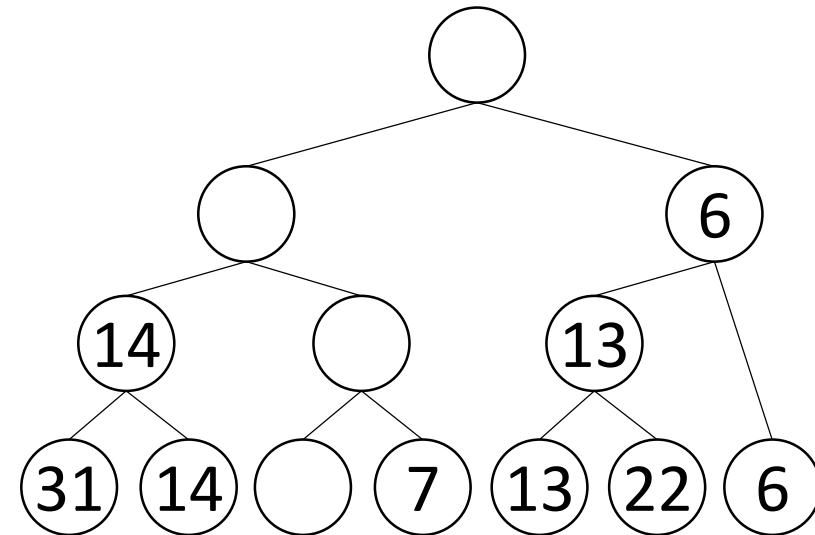
4. Retirar raiz da árvore

- Inserir no arquivo classificado



Arquivo
classificado

Árvore binária de vencedores

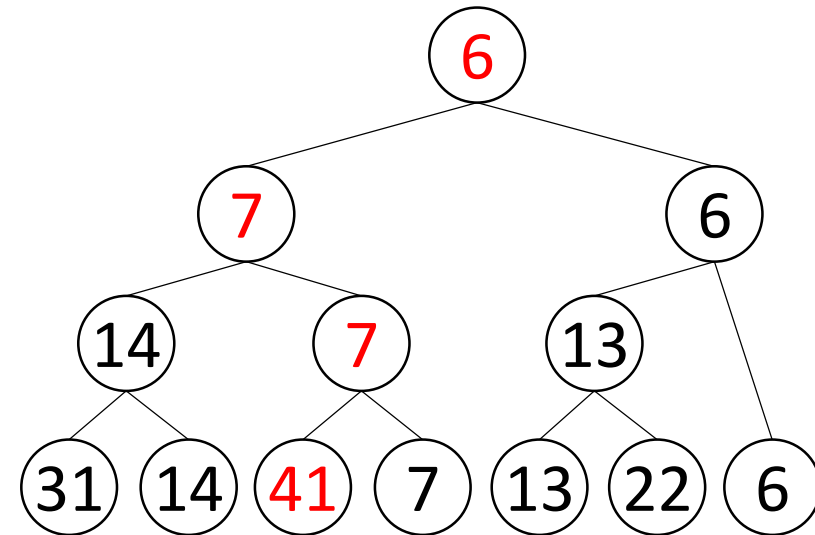


31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

2	3		

Arquivo
classificado

Árvore binária de vencedores

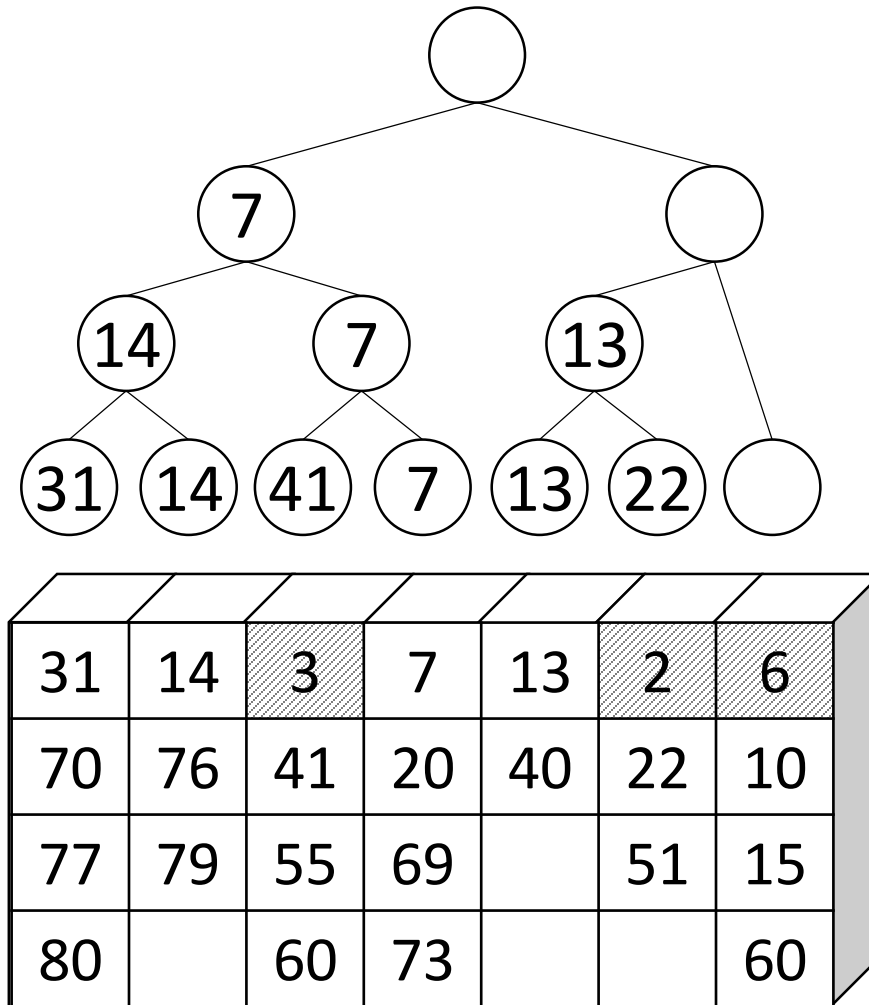


31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

2	3		

Arquivo
classificado

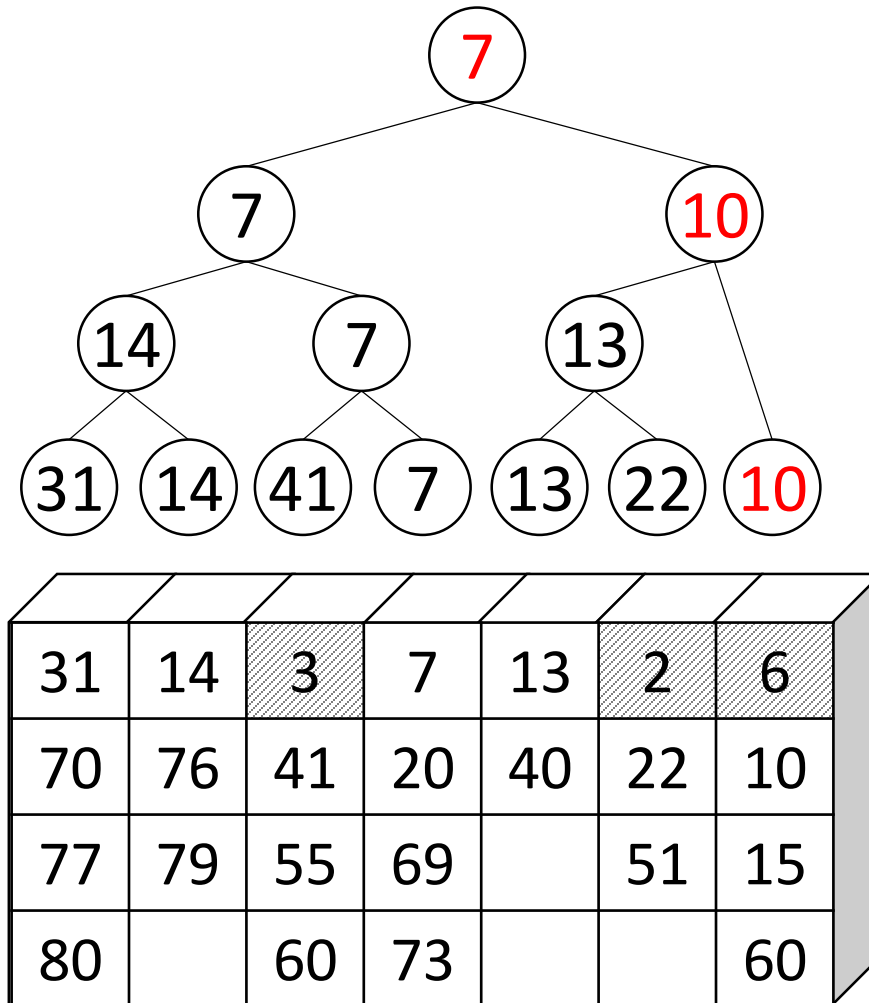
Árvore binária de vencedores



2	3	6	

Arquivo
classificado

Árvore binária de vencedores

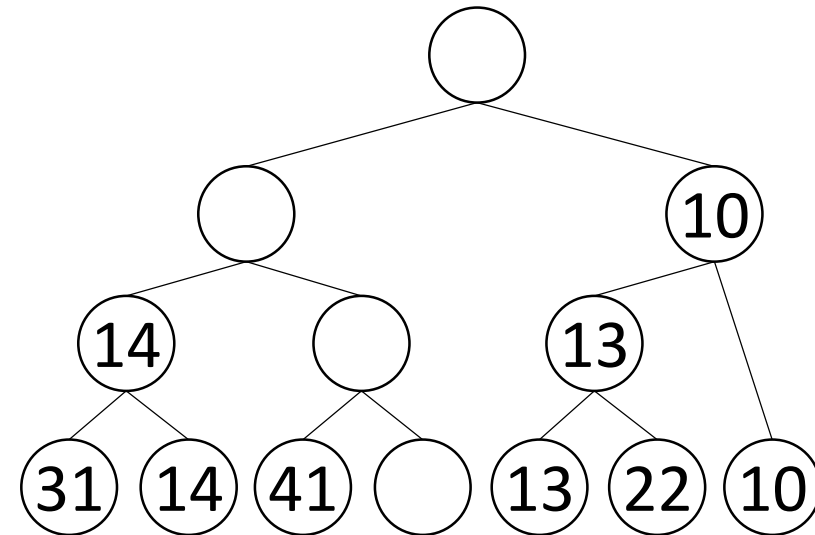


A 4x4 grid representing a tournament bracket. The first row contains the values 2, 3, 6, and an empty cell. The remaining rows are empty.

2	3	6	

Arquivo
classificado

Árvore binária de vencedores

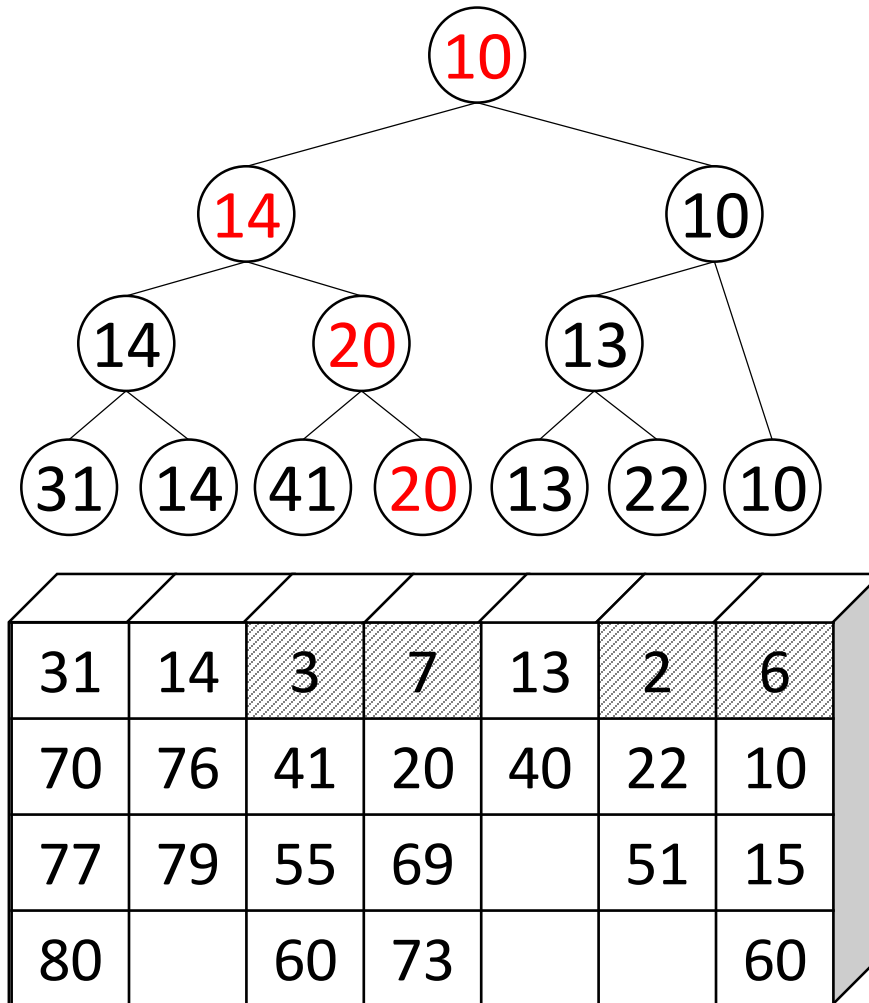


31	14	3	7	13	2	6
70	76	41	20	40	22	10
77	79	55	69		51	15
80		60	73			60

2	3	6	7

Arquivo
classificado

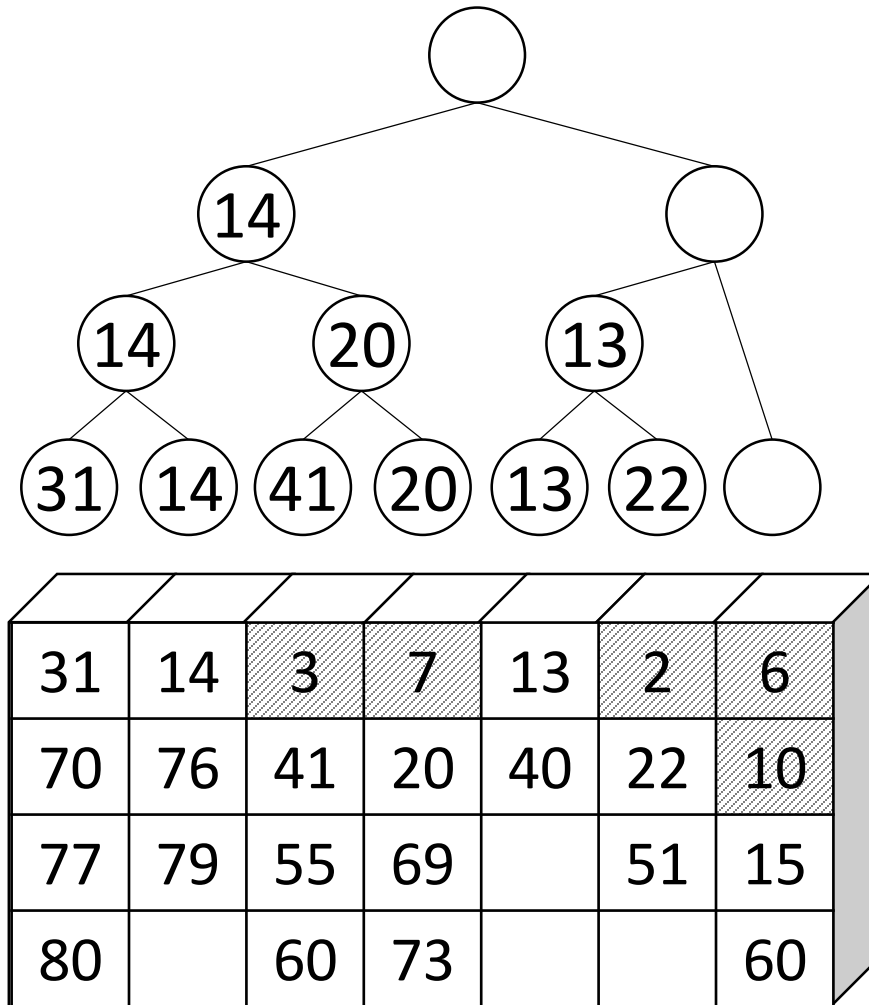
Árvore binária de vencedores



2	3	6	7

Arquivo
classificado

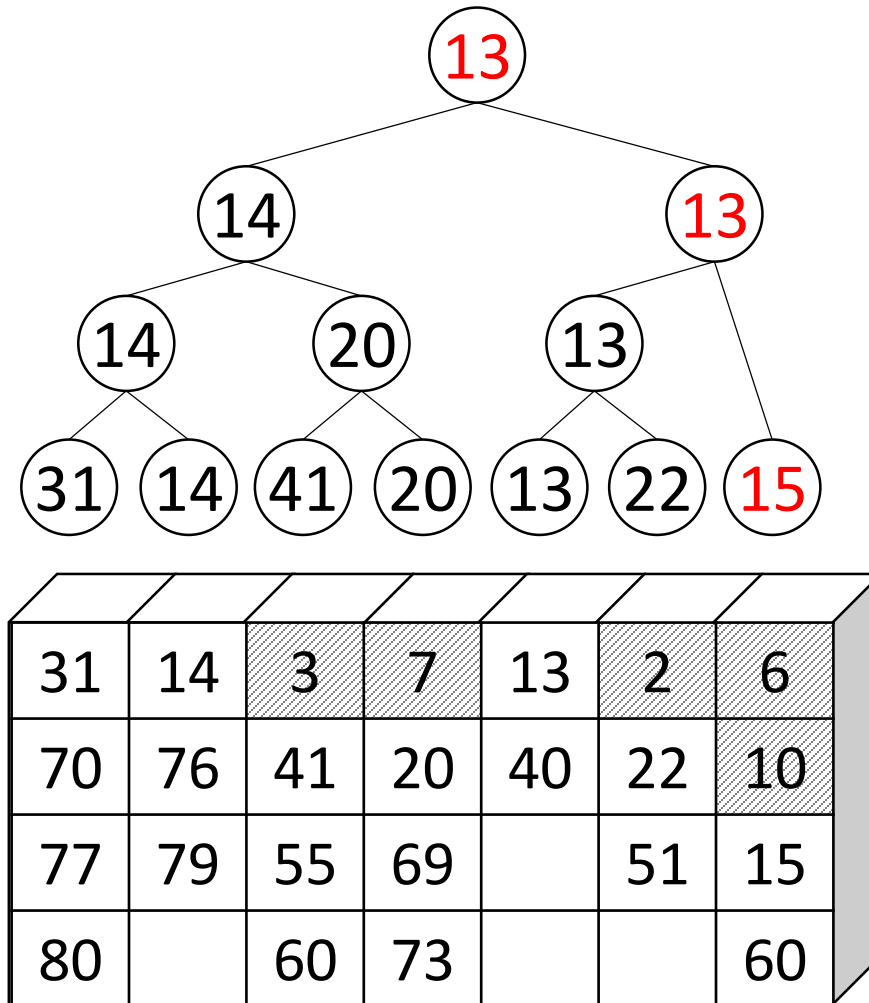
Árvore binária de vencedores



2	3	6	7
10			

Arquivo
classificado

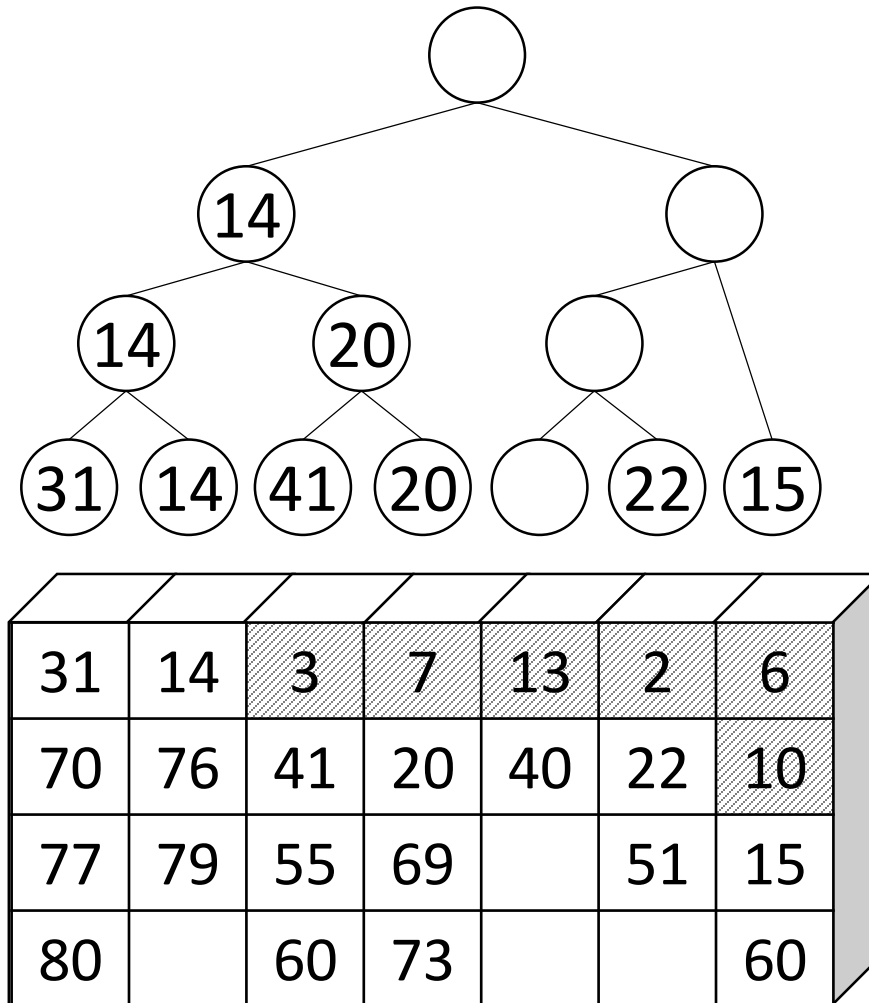
Árvore binária de vencedores



2	3	6	7
10			

Arquivo
classificado

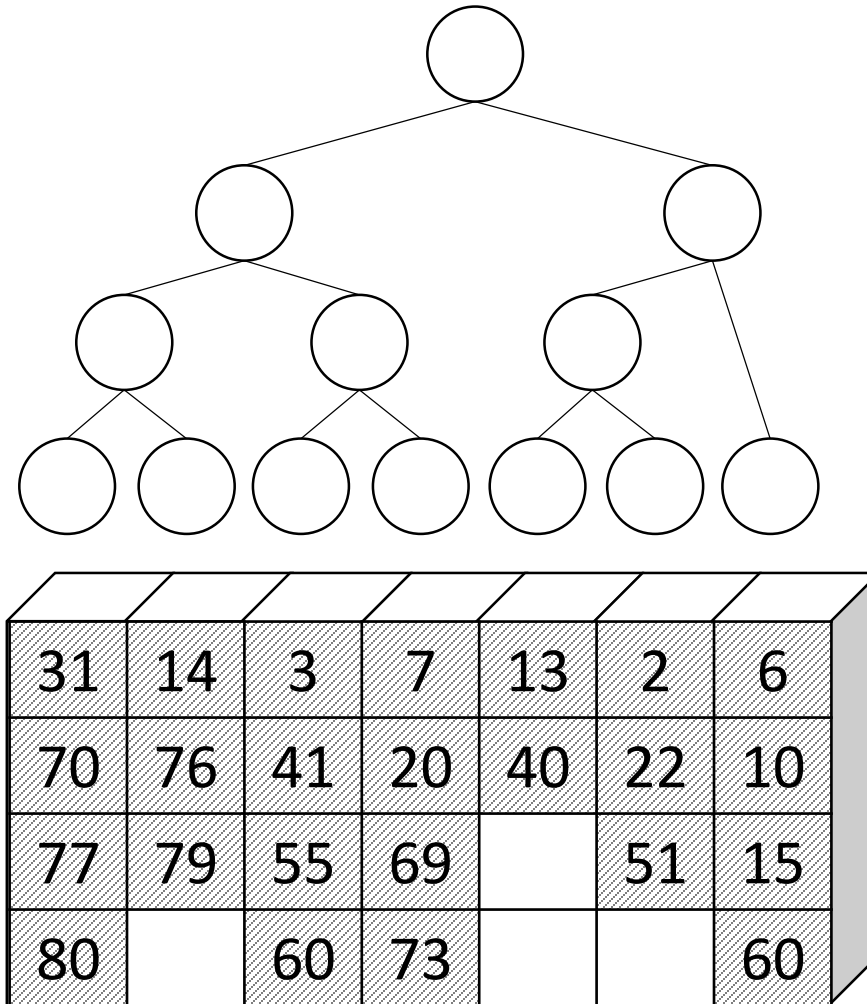
Árvore binária de vencedores



2	3	6	7
10	13		

Arquivo
classificado

Árvore binária de vencedores



2	3	6	7
10	13	14	15
20	22	31	40
41	51	55	60
60	69	70	73
76	77	79	80

Arquivo
classificado

Árvore binária de vencedores

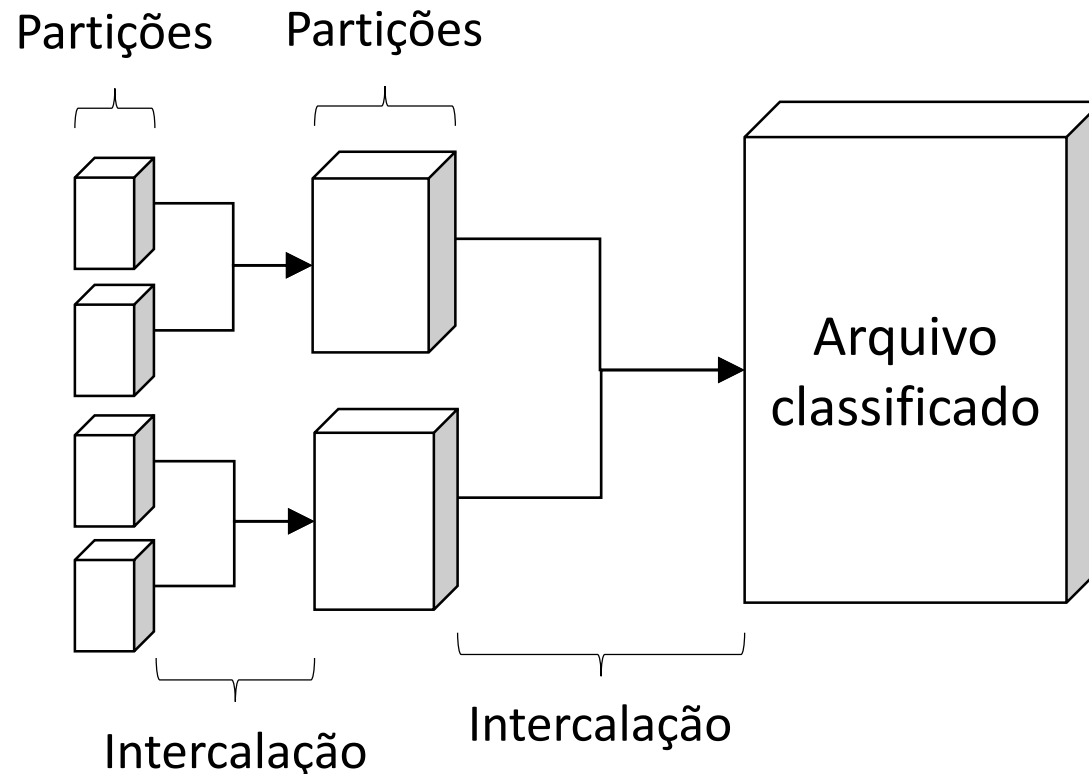
- Complexidade algorítmica
 - Montagem da árvore: $O(n)$
 - Cada iteração requer $\log n$ comparações (n é o número de partições a comparar)
 - Número de iterações: número total de registros a serem ordenados

Árvore binária de vencedores

- Nem sempre é possível intercalar todas as partições de uma só vez e obter o arquivo classificado
 - O número de arquivos a intercalar pode gerar uma árvore de vencedores maior do que a capacidade da memória
 - Sistemas operacionais estabelecem número máximo de arquivos abertos simultaneamente (`ulimit -Hn`)
- Esses números podem ser bem menor do que o número de partições a serem intercaladas

Estágio de intercalação

- A intercalação vai exigir uma série de fases onde os registros são lidos de um conjunto de partições e gravados em outras partições



Estágio de intercalação

- Estratégias de distribuição e intercalação
 - Intercalação balanceada de N caminhos
 - Intercalação ótima
- Medida de eficiência
 - A eficiência do estágio de intercalação é dada pelo número de passos
 - Representa o número médio de vezes que um registro é lido ou gravado durante o estágio de intercalação

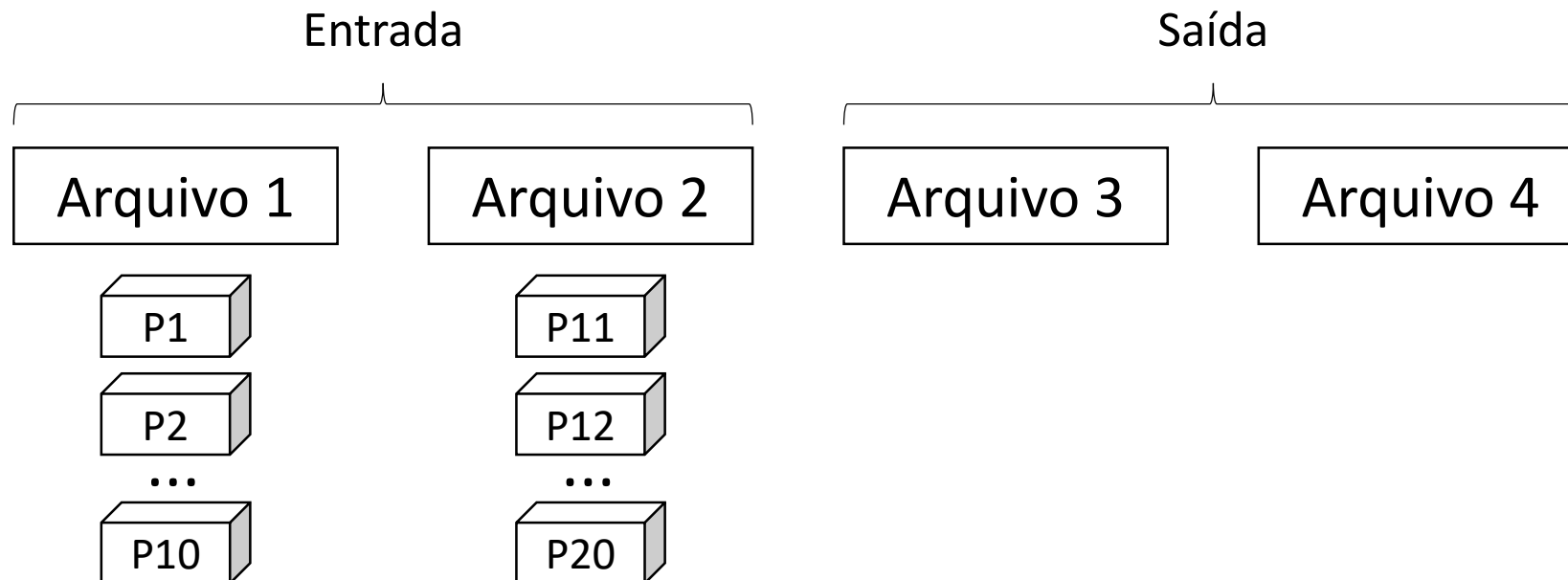
$$\text{Número de passos} = \frac{\text{Total de registros lidos}}{\text{Total de registros contidos no arquivo classificado}}$$

Intercalação balanceada de N caminhos

1. Determinar o número de arquivos F que o algoritmo irá manipular
 - A primeira metade das partições será usada para leitura (entrada)
 - A segunda metade para escrita (saída)
2. Distribuir todas as partições o mais igualitária possível entre os arquivos de entrada
3. Intercalar duas partições gravando o resultado em uma nova partição em um dos arquivos de saída

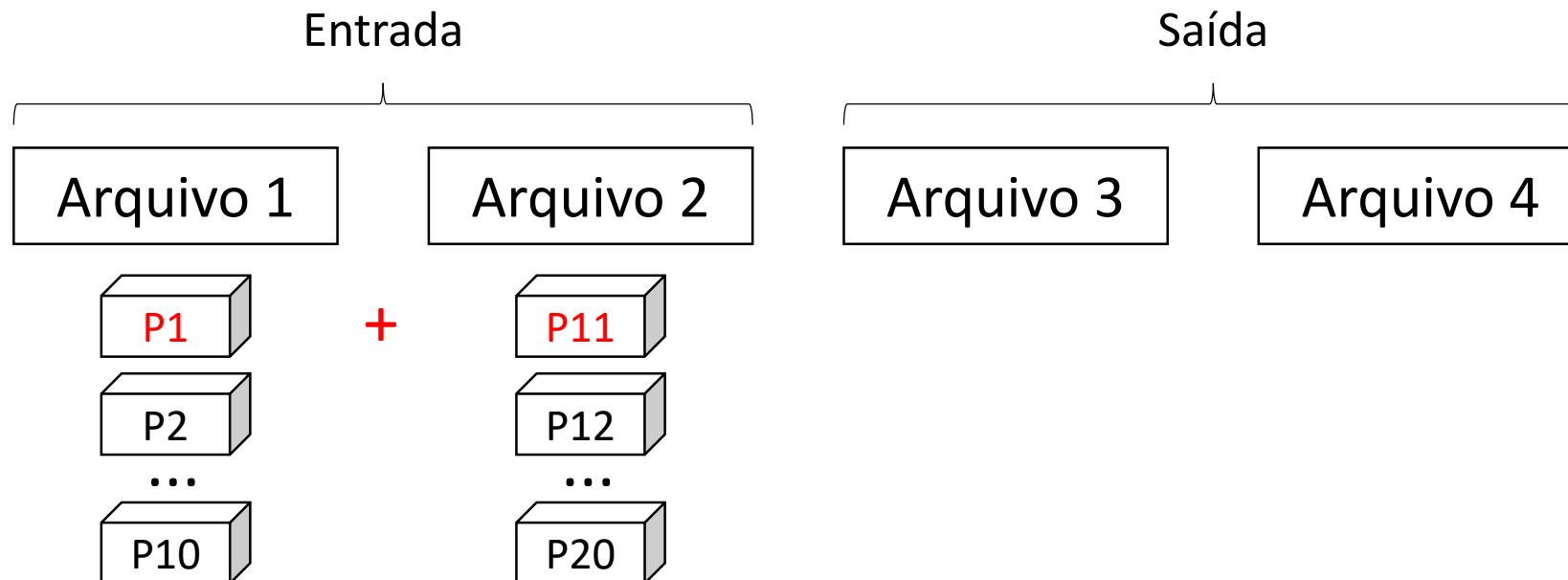
Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



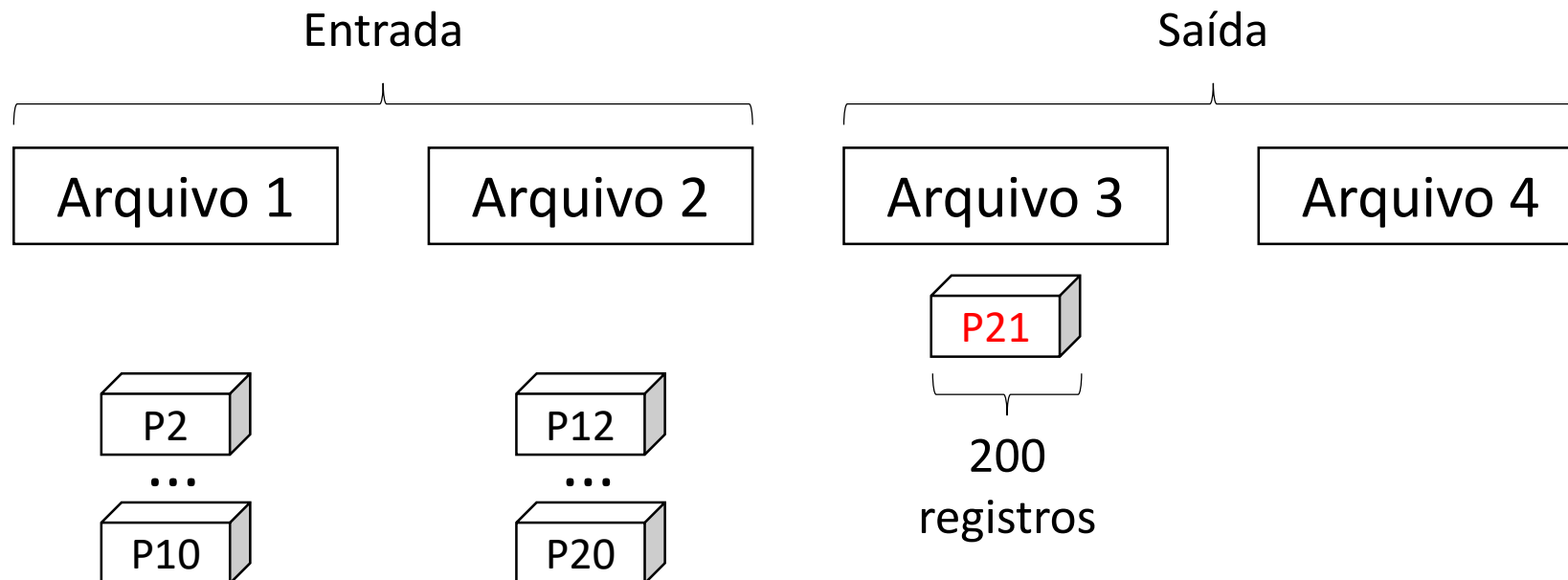
Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



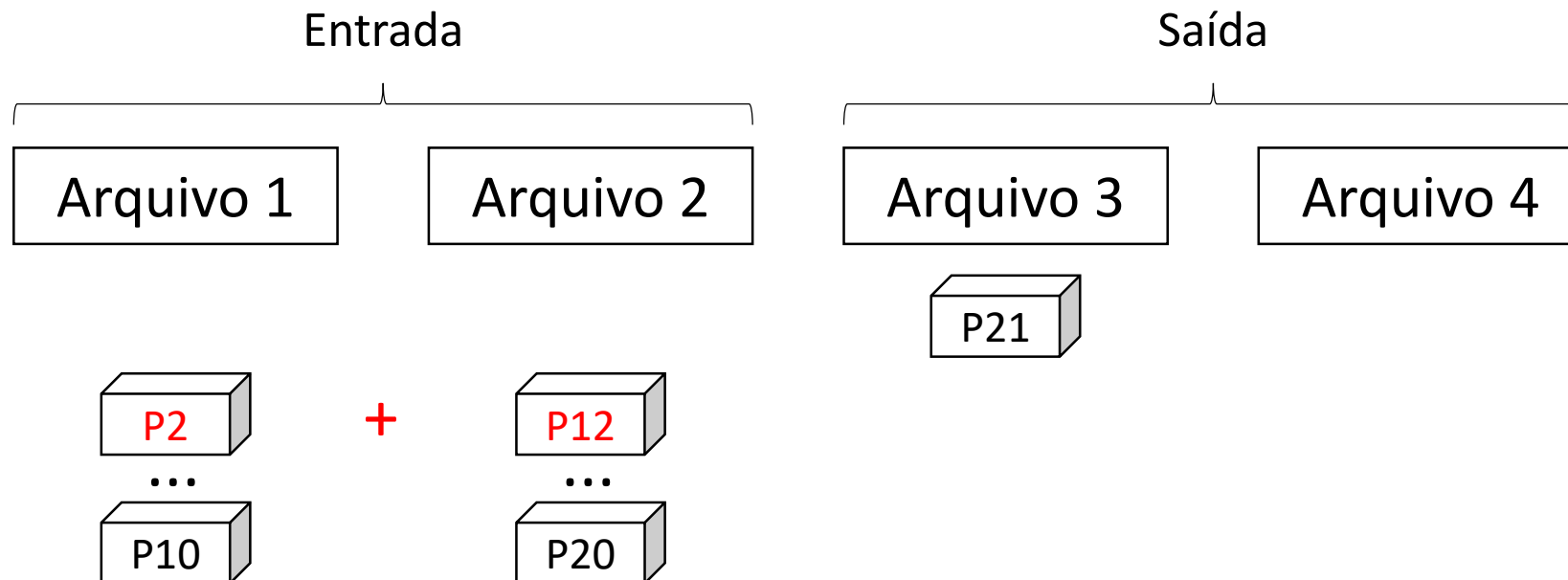
Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



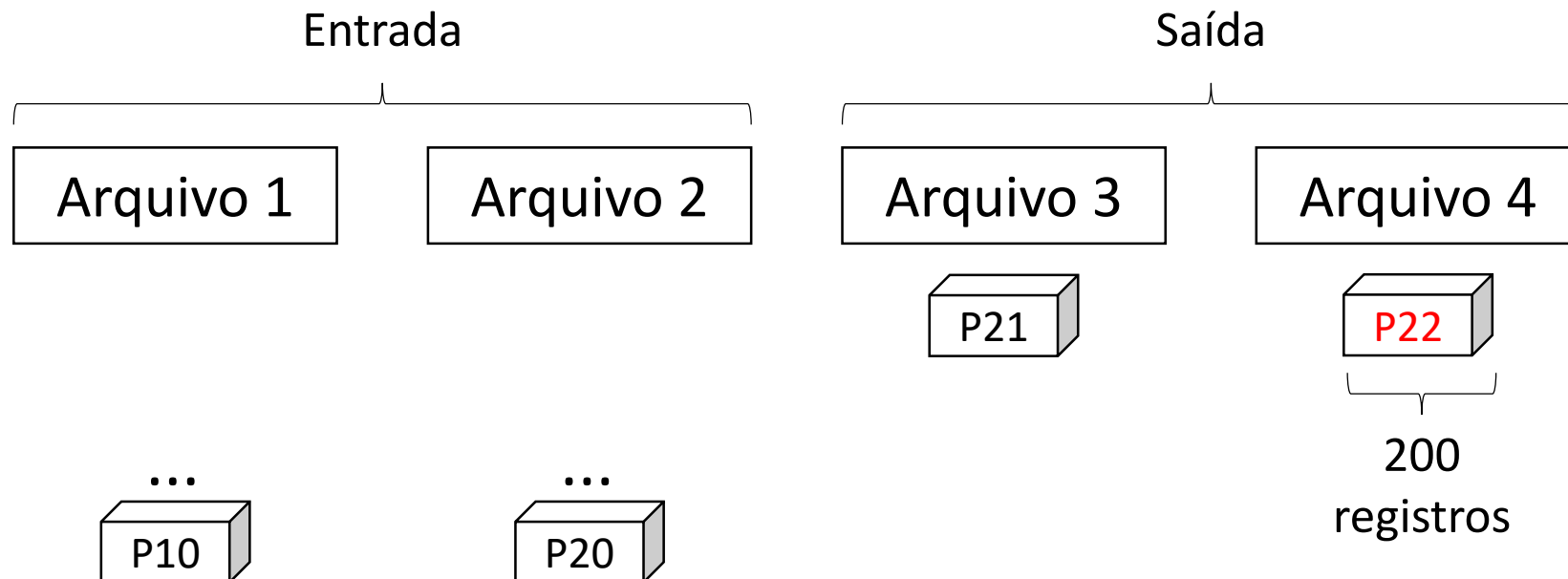
Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



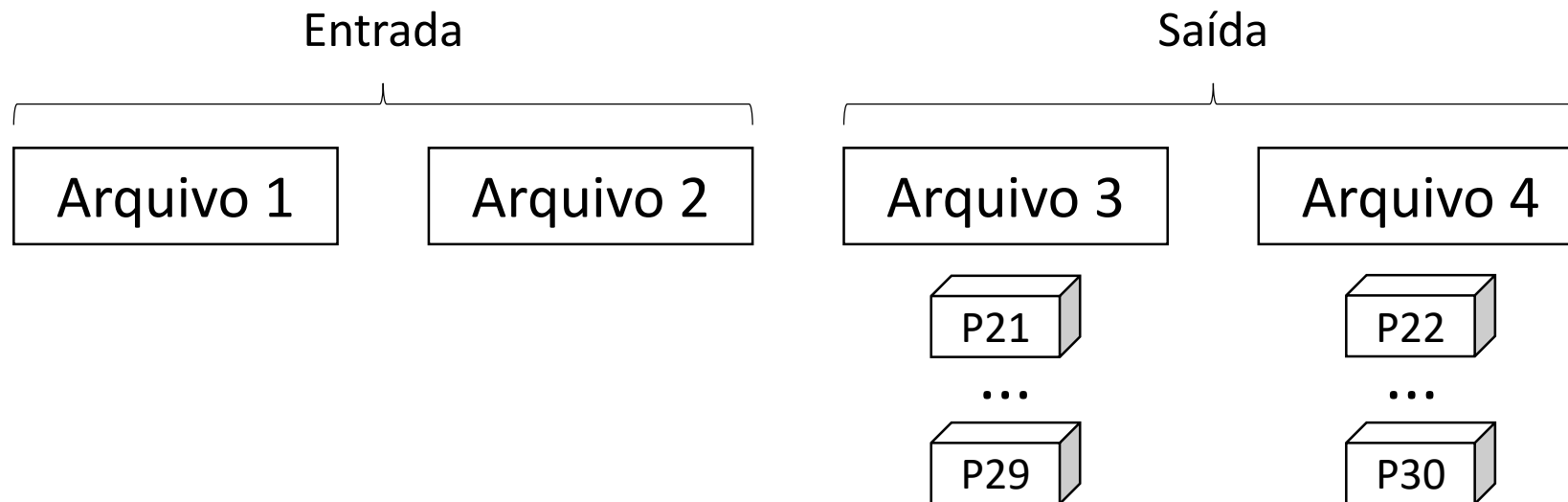
Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



Intercalação balanceada de N caminhos

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



Intercalação balanceada de N caminhos

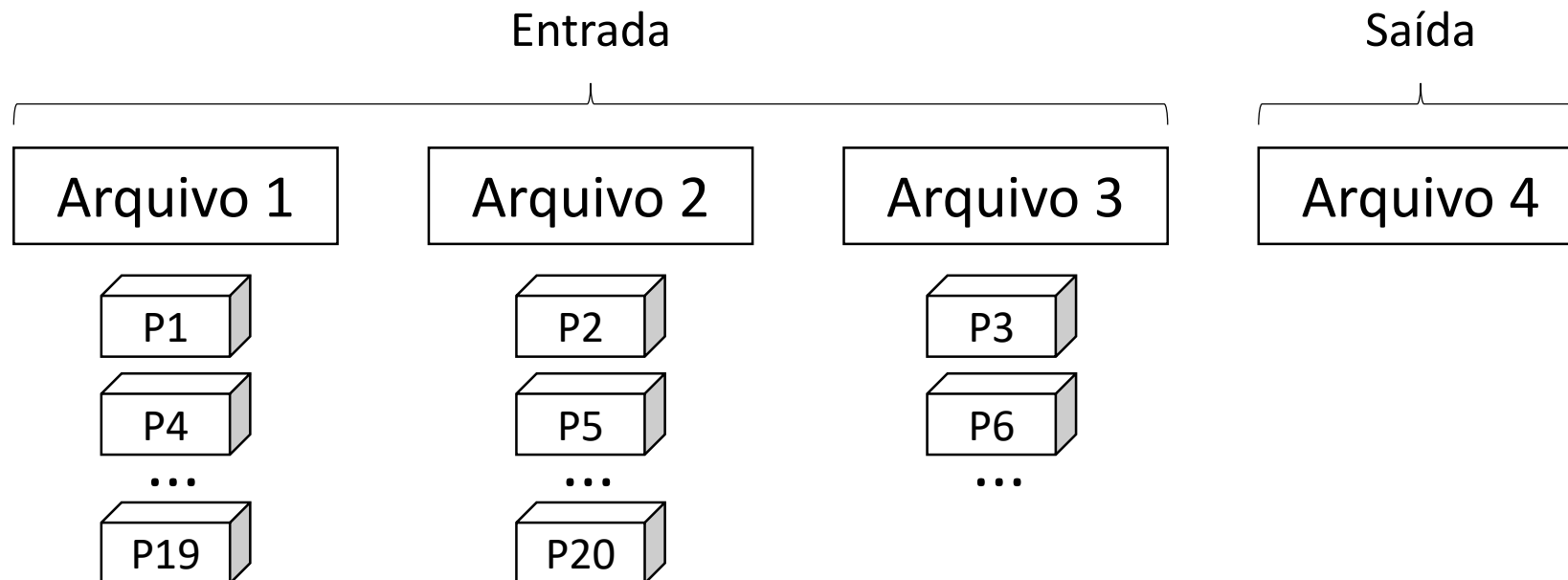
- Ao encerrar a primeira rodada, o conjunto de partições de saída torna-se o conjunto de entrada para a rodada seguinte
- A intercalação termina quando grava-se apenas uma partição na rodada atual
- Exemplo
 - Rodada 1: 20 partições com 100 registros cada
 - Rodada 2: 10 partições com 200 registros cada
 - Rodada 3: 5 partições com 400 registros cada
 - Rodada 4: 2 partições com 800 registros cada e 1 partição com 400 registros cada
 - Rodada 5: 1 partição com 1600 registros + 1 partição com 400 registros
 - Resultado final: 1 partição com 2000 registros

Intercalação ótima

1. Determinar o número de arquivos F que o algoritmo irá manipular
 - $F - 1$ para a entrada
 - 1 para a saída
2. Durante cada rodada do algoritmo, $F - 1$ partições são intercaladas e gravadas em um único arquivo de saída
3. Do conjunto inicial de partições, removem-se as partições intercaladas e a agrega-se a algum arquivo de entrada a partição gerada na intercalação
 - Algoritmo termina quando este conjunto tiver apenas uma partição

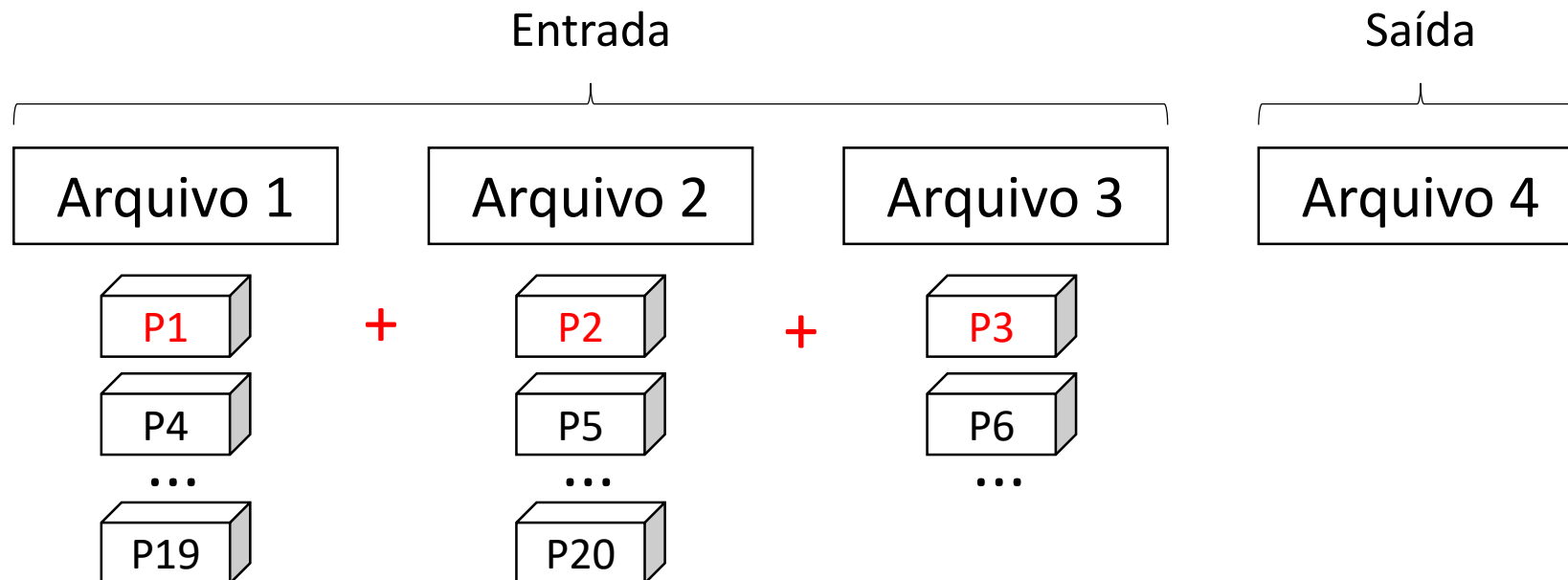
Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



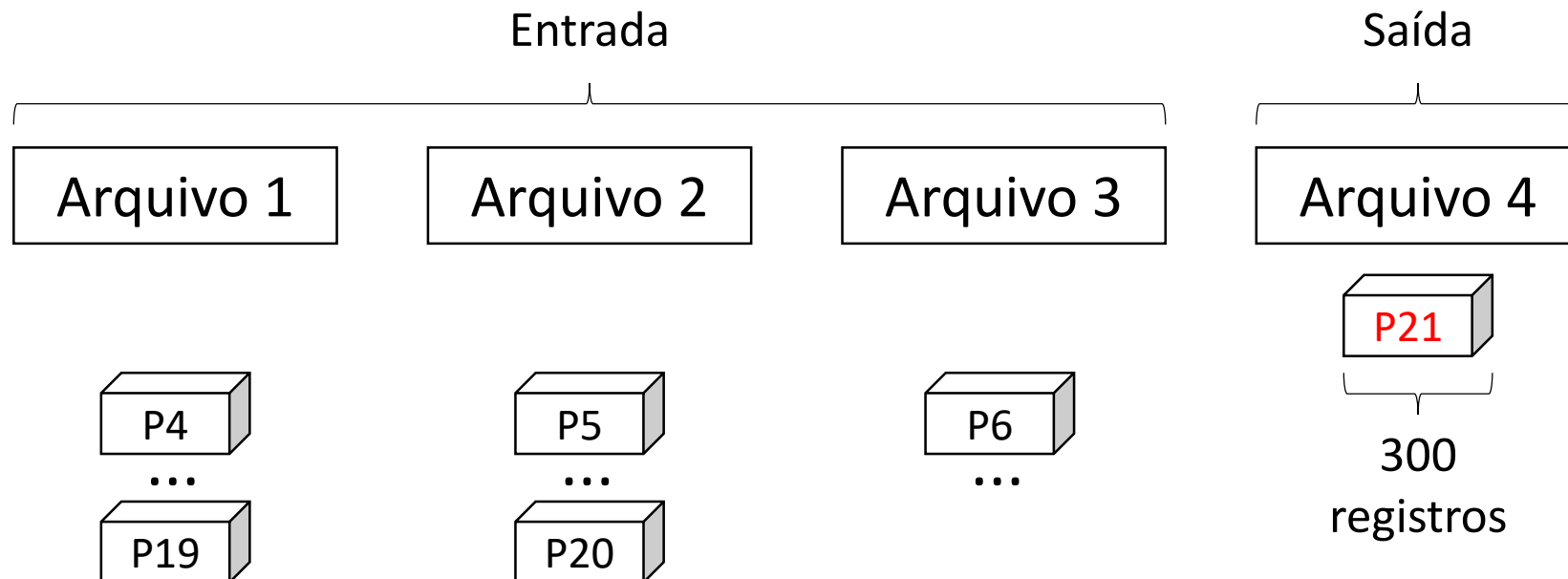
Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



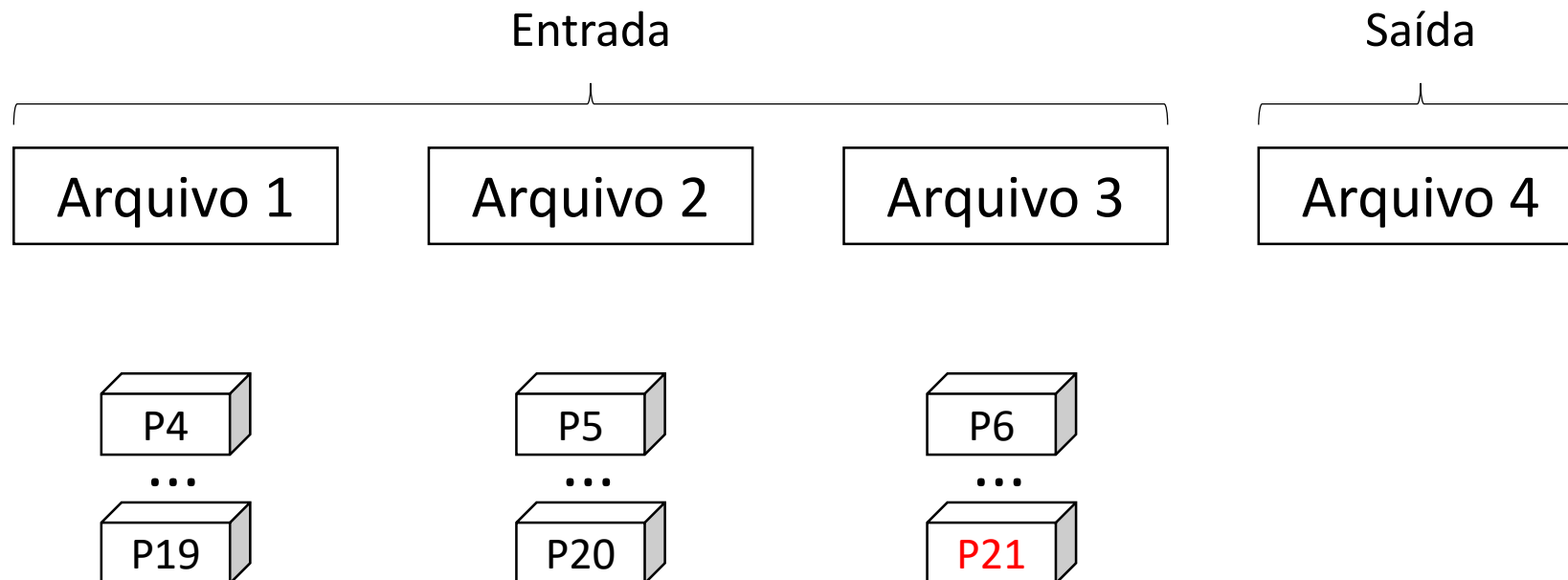
Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



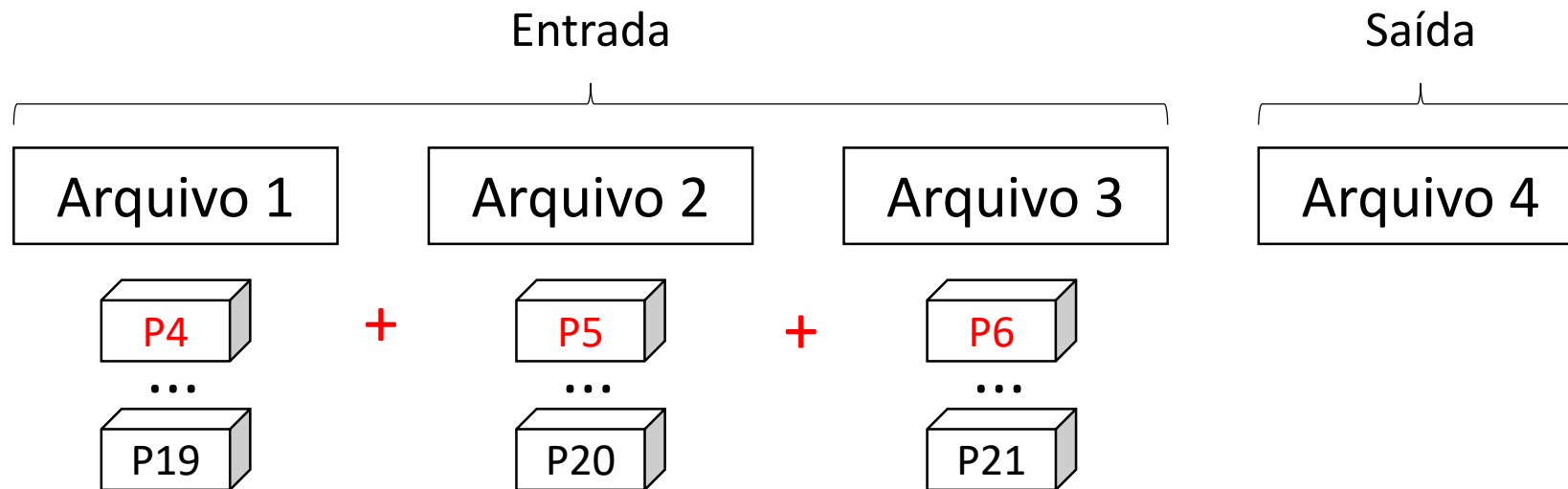
Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



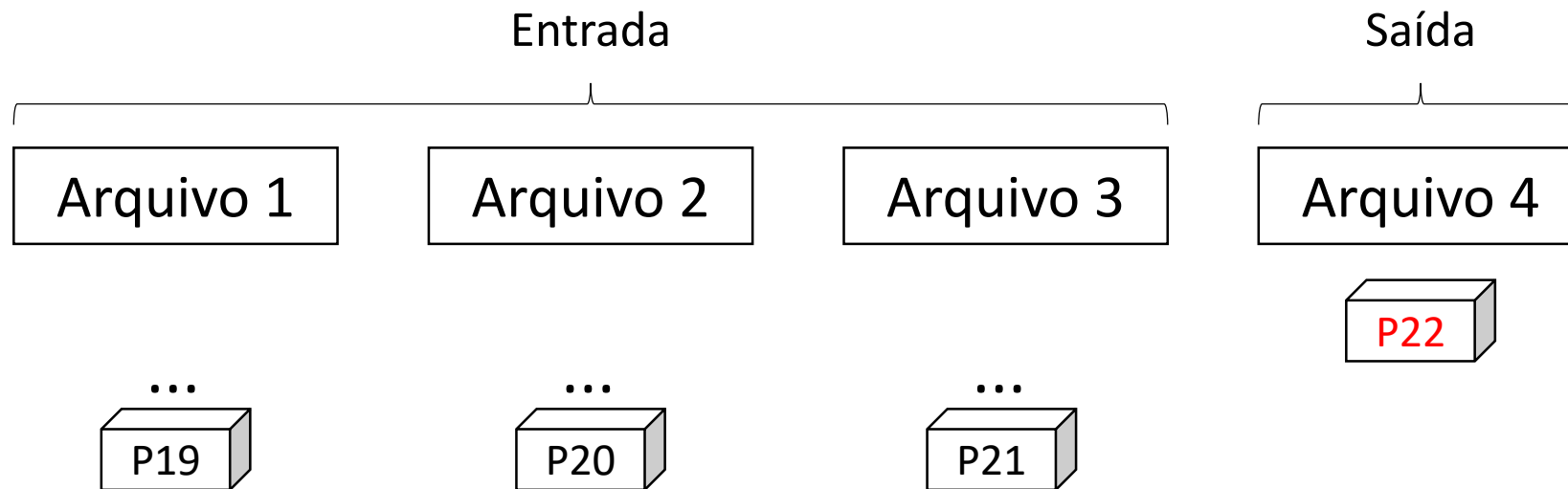
Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



Intercalação ótima

- Exemplo
 - Número de arquivos: 4
 - Número de partições a serem intercaladas: 20
 - Número de registros por partição: 100



Intercalação ótima

- Possível estratégia de implementação
 - Usar uma lista que contém os nomes dos arquivos a ordenar
 - A cada rodada do algoritmo
 - Retirar os 3 primeiros itens da lista
 - Intercalar os itens retirados
 - Colocar o arquivo resultante no final da lista
 - O algoritmo encerra quando a lista tiver apenas um arquivo
 - Arquivo classificado resultante

Exercícios

1. Implementar na linguagem C o algoritmo *insertion-sort* para classificação interna de um vetor contendo M registros
 - Os valores para cada posição do vetor pode ser atribuído aleatoriamente
 - Utilize a include `<time.h>` e as funções `srand` e `rand`
2. Gerar partições classificadas segundo o método de seleção com substituição para a seguinte situação
 - Assuma que a memória possui capacidade para $M = 7$ registros simultaneamente
 - As partições devem ser representadas por vetores dinâmicos
 - Arquivo a ser ordenado

30	14	15	75	32	6	5	81	48	41	87	18
56	20	26	4	21	65	22	49	11	16	8	12
44	9	7	81	23	19	1	78	13	16	51	8

Exercícios

3. Gerar partições classificadas segundo o método de seleção com substituição para a seguinte situação
- Assumir que a memória possui capacidade para $M = 7$ registros simultaneamente
 - As partições devem ser representadas por vetores dinâmicos
 - Arquivo a ordenar

30	14	15	75	32	6	5	81	48	41	87	18
56	20	26	4	21	65	22	49	11	16	8	12
44	9	7	81	23	19	1	78	13	16	51	8

4. Implementar na linguagem em C o algoritmo de seleção natural
- Utilize os mesmos dados do exercício 2

Exercícios

5. Implementar na linguagem C o algoritmo de intercalação balanceada de N caminhos utilizando árvore binária de vencedores
 - Utilize os vetores dinâmicos das partições gerados pelo exercício 3
 - Considere 4 arquivos para intercalação das partições
 - As partições devem ser distribuídas igualmente nos 2 arquivos de entrada
6. Implementar na linguagem C o algoritmo de intercalação ótima utilizando árvore binária de vencedores
 - Utilize os vetores dinâmicos das partições gerados pelo exercício 3
 - Considere 4 arquivos para intercalação das partições
 - Utilize uma lista contendo os vetores dinâmicos das partições a serem intercaladas

PRA – Projeto de Arquivos

Classificação em memória
principal e secundária

Prof. Allan Rodrigo Leite