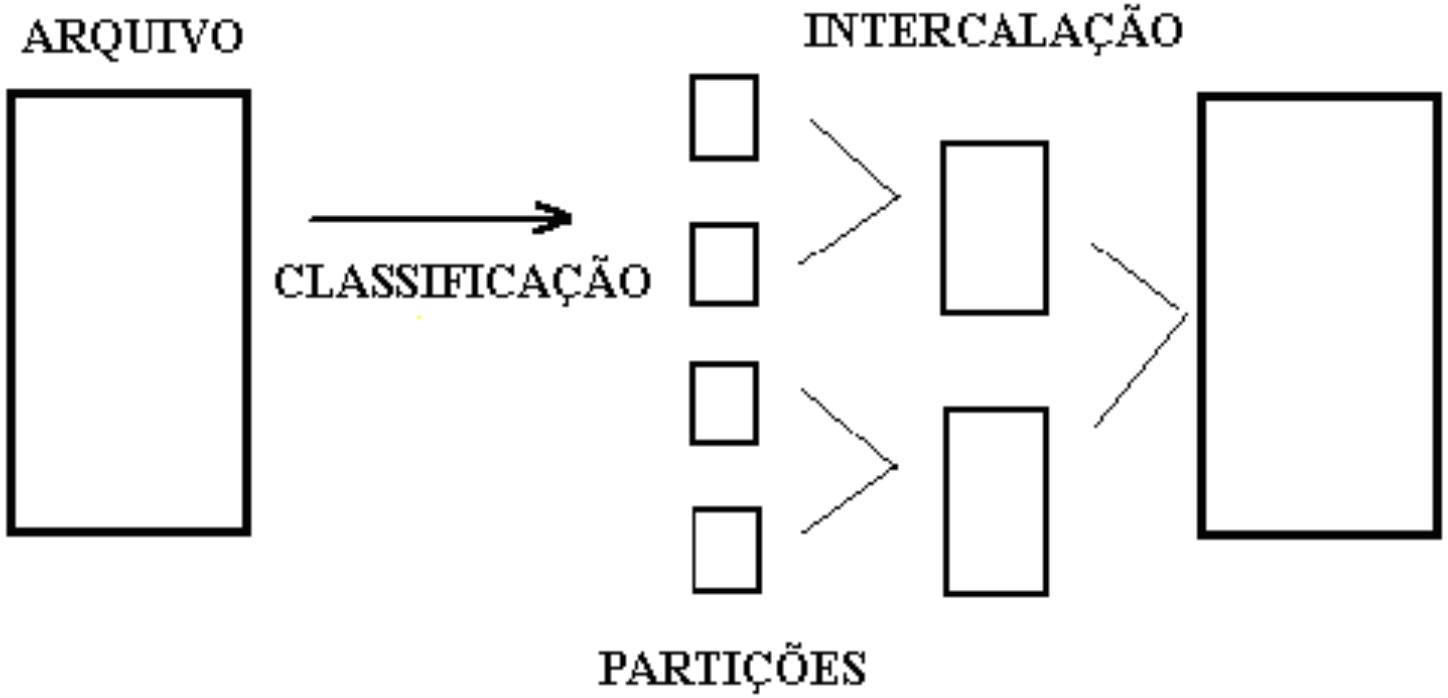
# Classificação Externa: Intercalação de Partições Classificadas

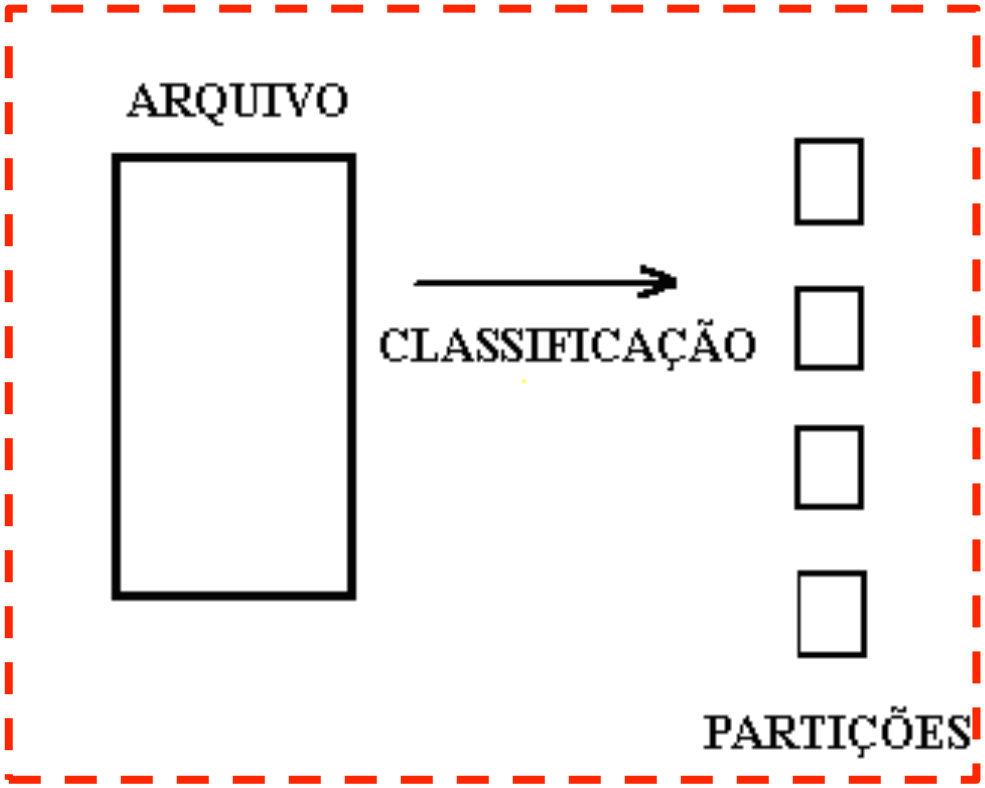
Vanessa Braganholo

Baseado no Material de: Inhaúma Neves Ferraz (IC/UFF)

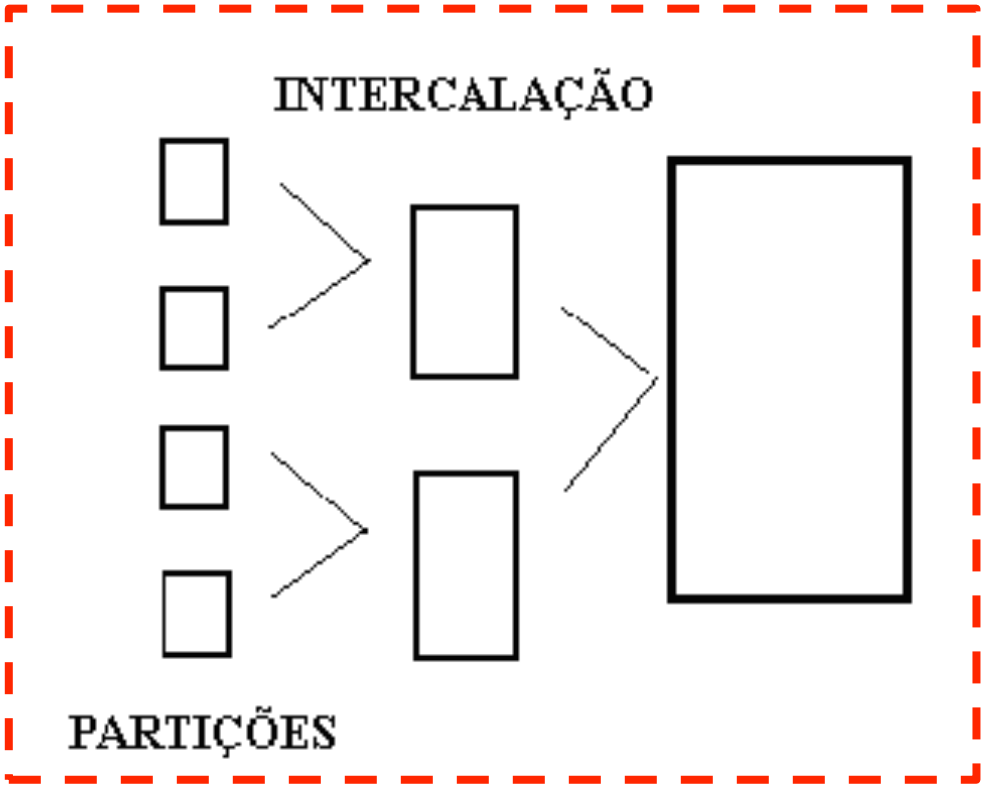
# Relembrando:
# Modelo da Classificação Externa



# Aula Passada: Etapa de Classificação



# Aula de Hoje: 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

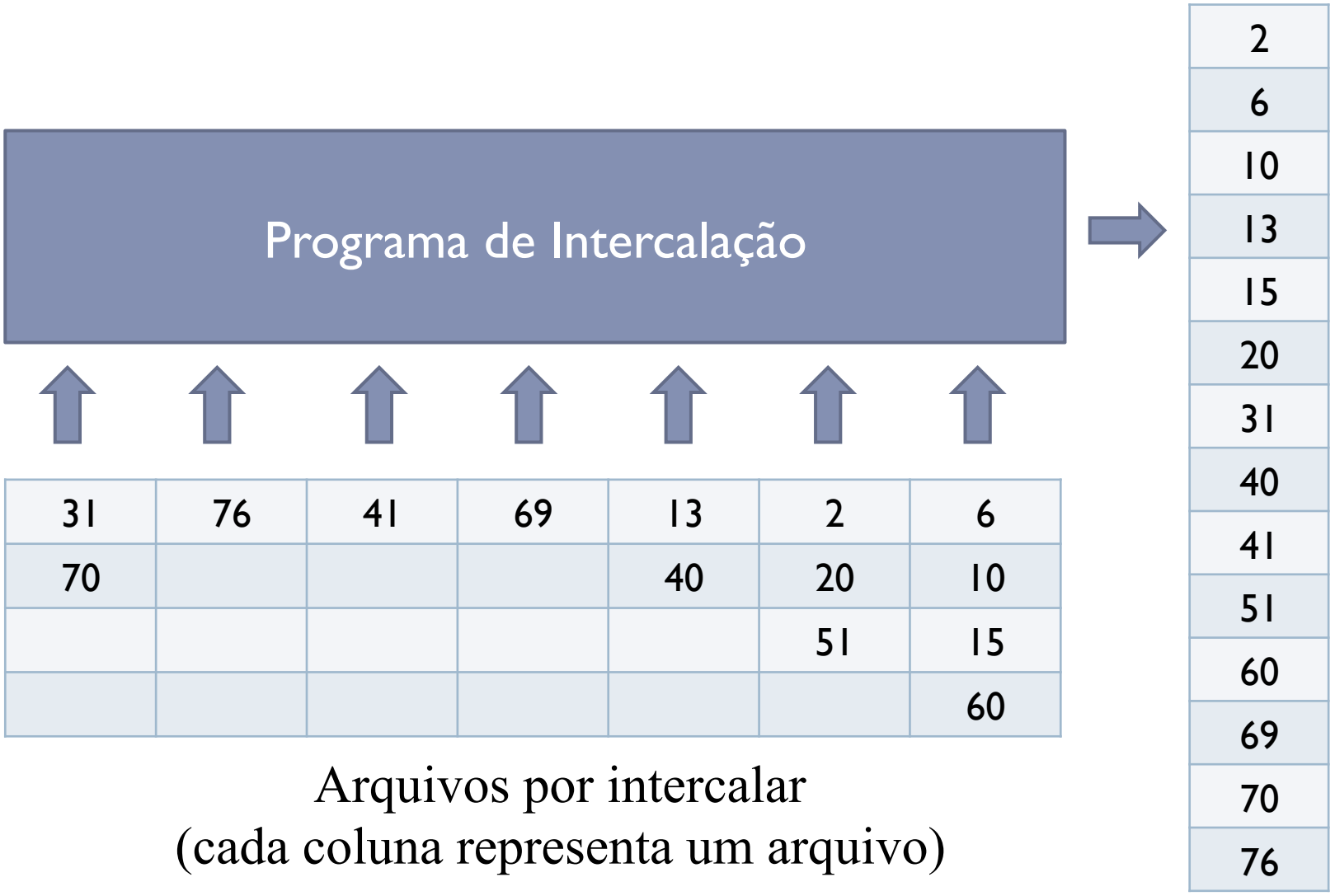
# Problema

- 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)



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 |

# 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

- 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 interno tem quatro componentes
  - Vencedor: valor da menor chave daquela sub-árvore
  - EndVencedor: ponteiro para o arquivo que tem aquela chave
  - Left: ponteiro para o filho da esquerda
  - Rigth: ponteiro para o filho da direita

# Exemplo

- Arquivos a serem ordenados
  - Cada coluna abaixo representa um arquivo com suas respectivas chaves

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

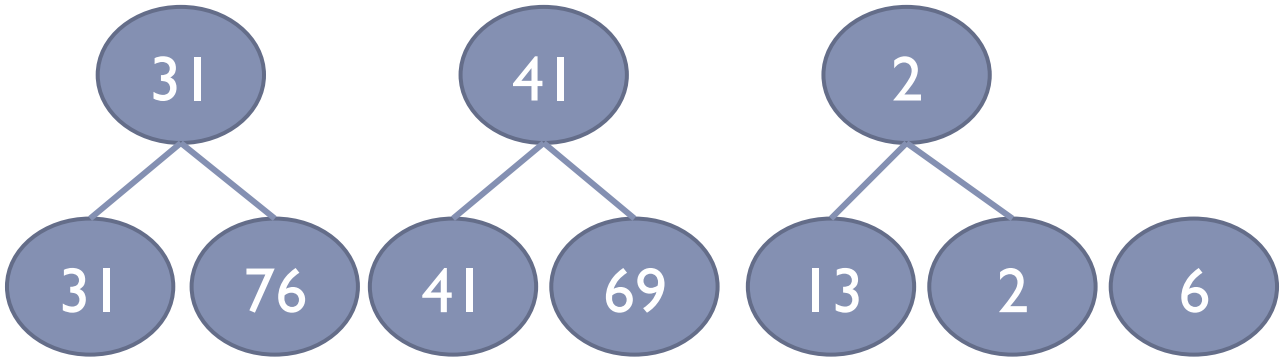
# Árvore Binária de Vencedores

- Colocar em memória o primeiro registro de cada arquivo
  - Cada registro é um nó folha da árvore (aqui usamos apenas as chaves para simplificar)


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

# Árvore Binária de Vencedores

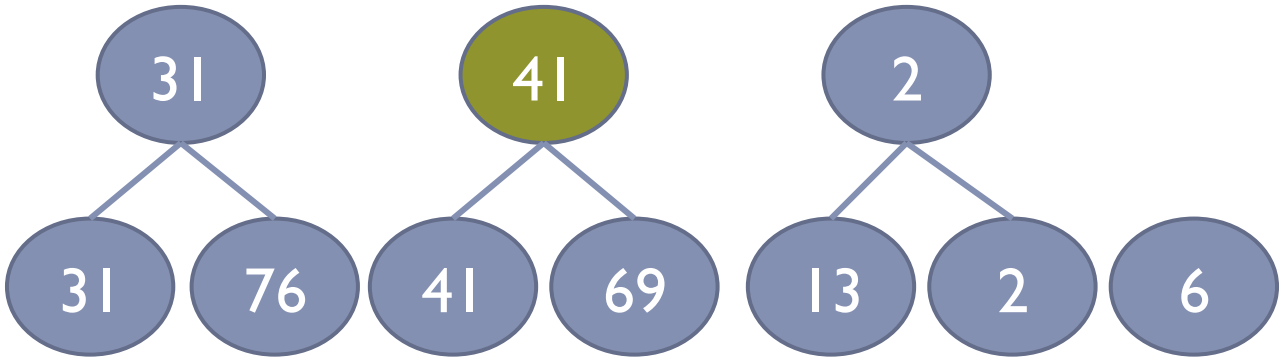
- Criar um nó raiz para cada 2 nós folha, com o menor dos dois valores




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

# Árvore Binária de Vencedores

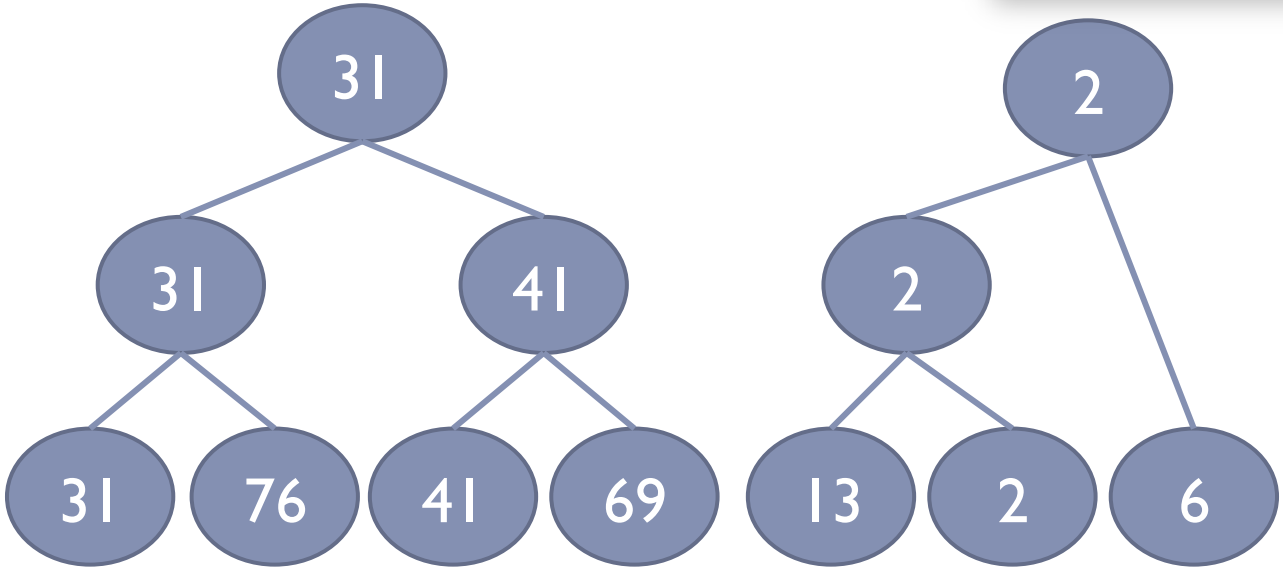
- Representação do nó interno 41
  - Vencedor: 41
  - EndVencedor: 3
  - Left: 41
  - Rigth: 69




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

# Árvore Binária de Vencedores

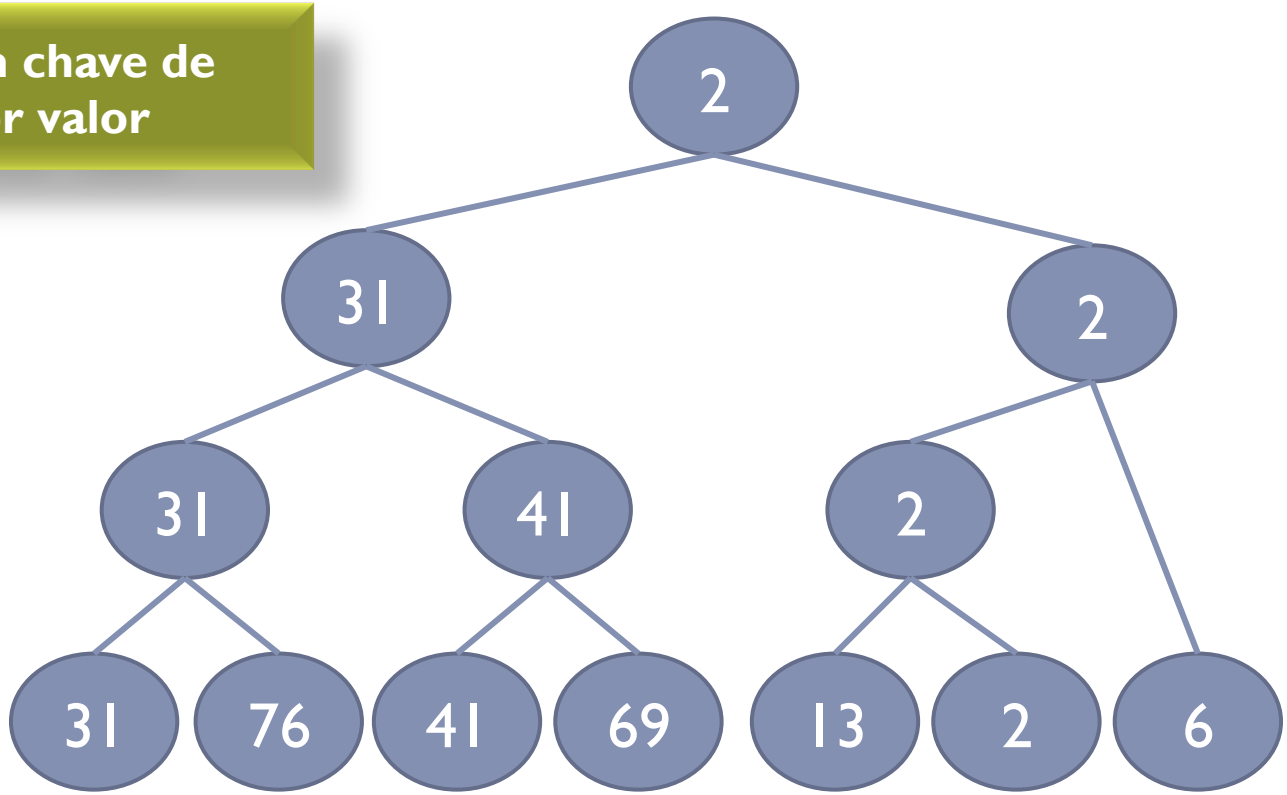
**Atenção: valores de chave se repetem em vários níveis**




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

# Árvore Binária de Vencedores

**Raiz tem chave de menor valor**

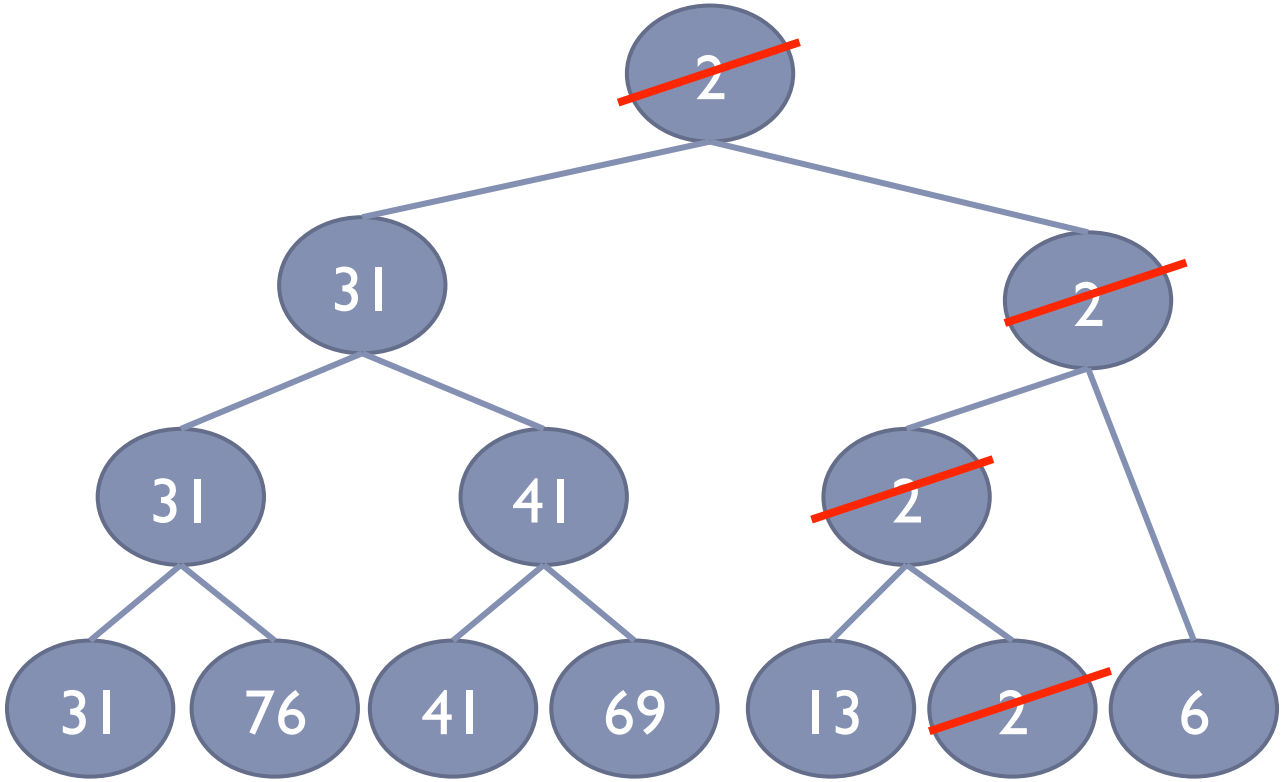



| | | | | | | |
|---|---|---|---|---|---|---|
| 31 | 76 | 41 | 69 | 13 | 2 | 6 |
| 70 | | | | 40 | 20 | 10 |
| | | | | | 51 | 15 |
| | | | | | | 60 |

# Uso da Árvore de Vencedores no algoritmo de Intercalação

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

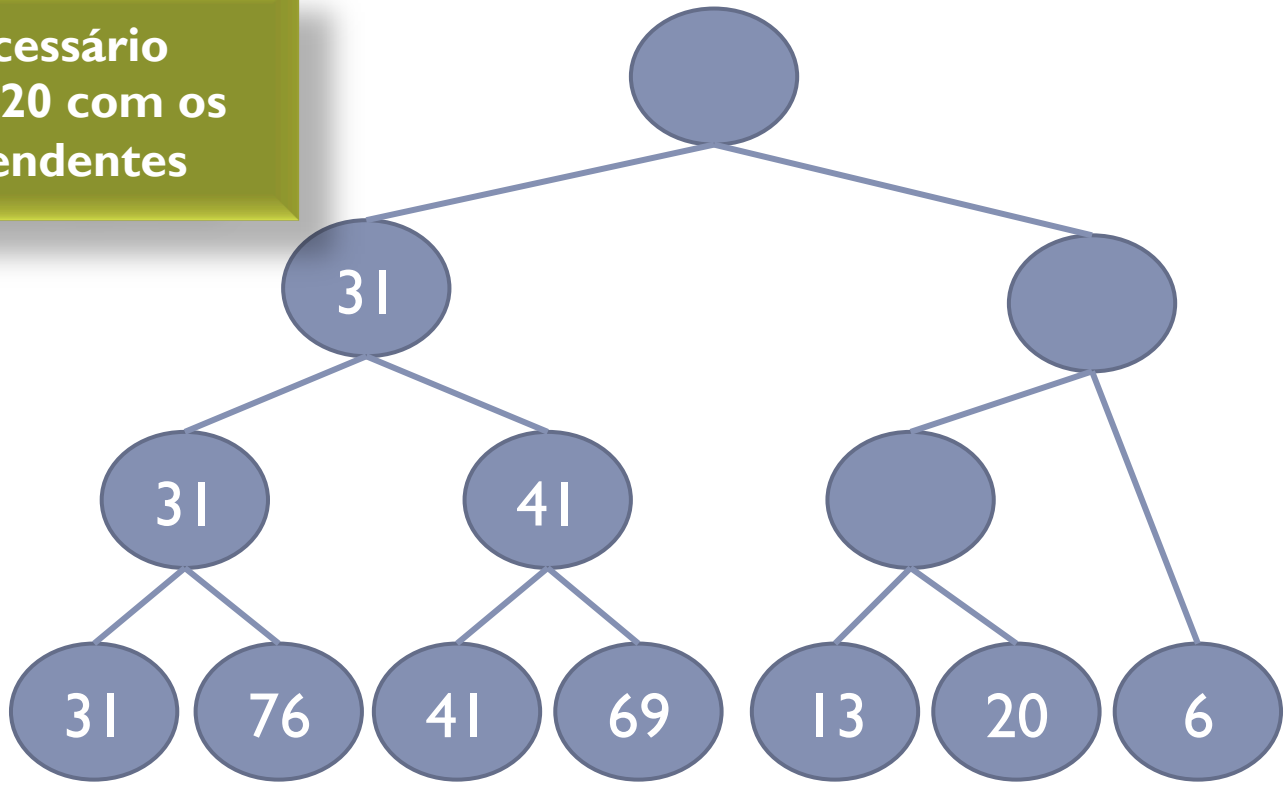
# Árvore Binária de Vencedores




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

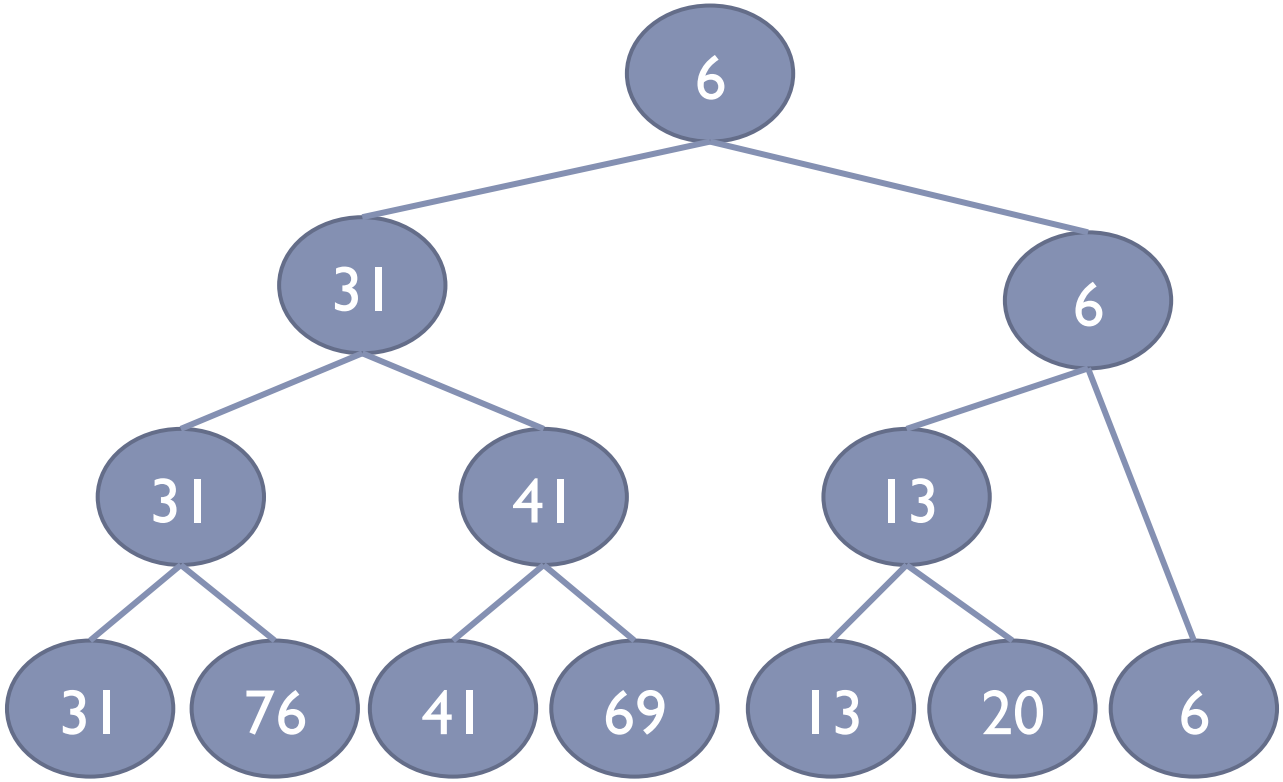
# Árvore Binária de Vencedores

**Só é necessário comparar 20 com os seus ascendentes**



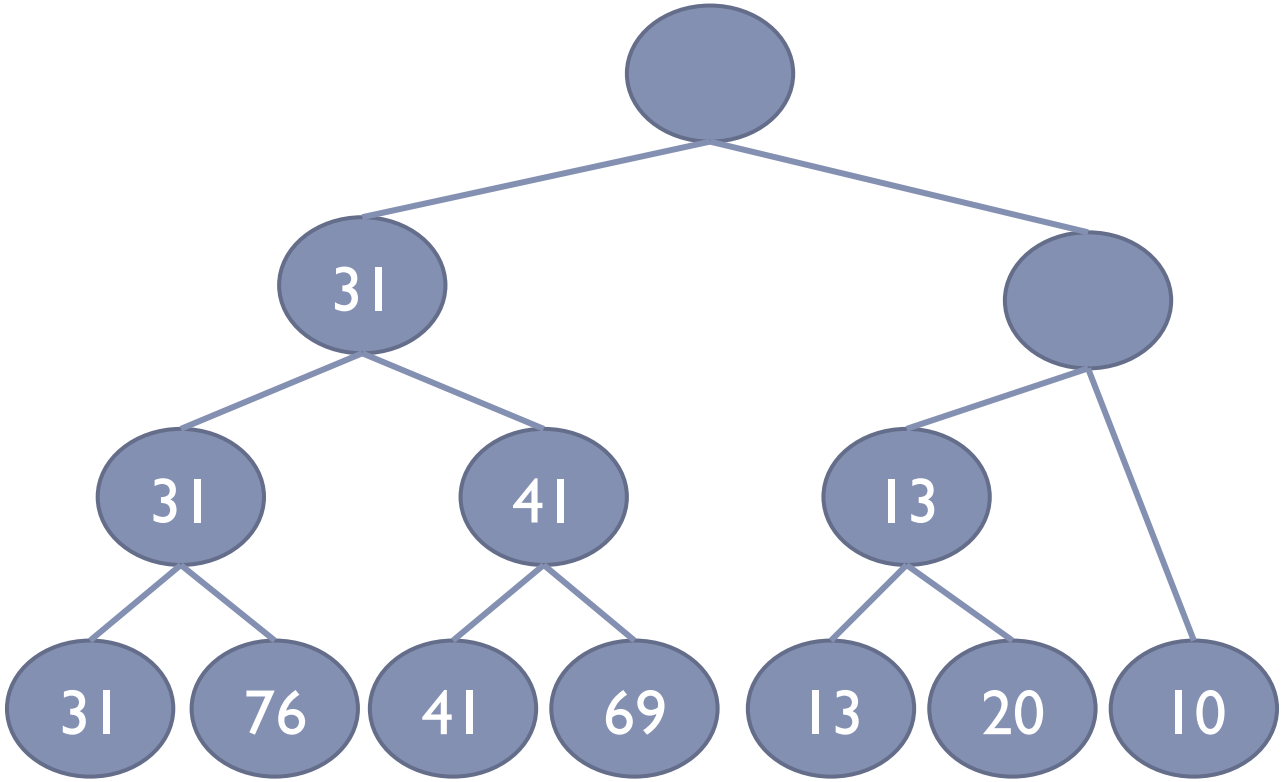

| 31 | 76 | 41 | 69 | 13 | 20 | 6 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | 40 | 51 | 10 |
| HV | | | | HV | HV | 15 |
| | | | | | | 60 |
| | | | | | | HV |

# Árvore Binária de Vencedores



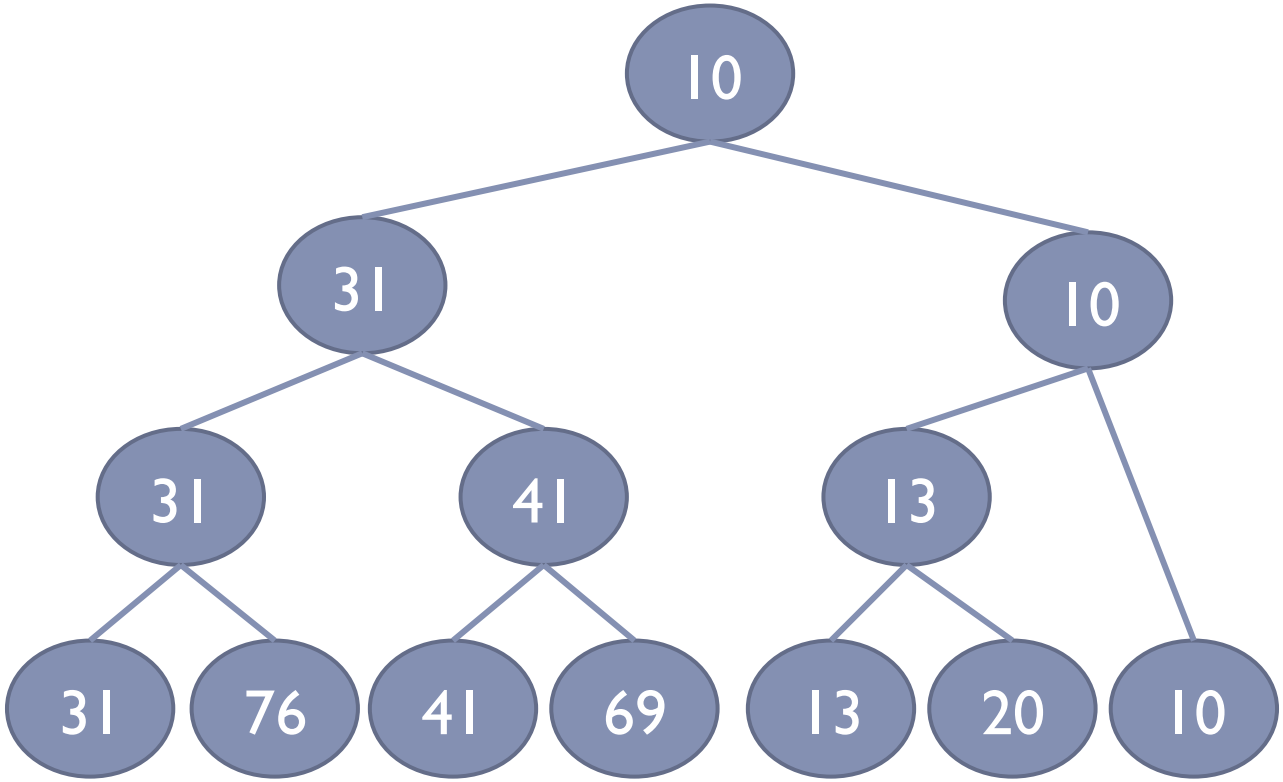

| 31 | 76 | 41 | 69 | 13 | 20 | 6 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | 40 | 51 | 10 |
| HV | | | | HV | HV | 15 |
| | | | | | | 60 |
| | | | | | | HV |

# Árvore Binária de Vencedores



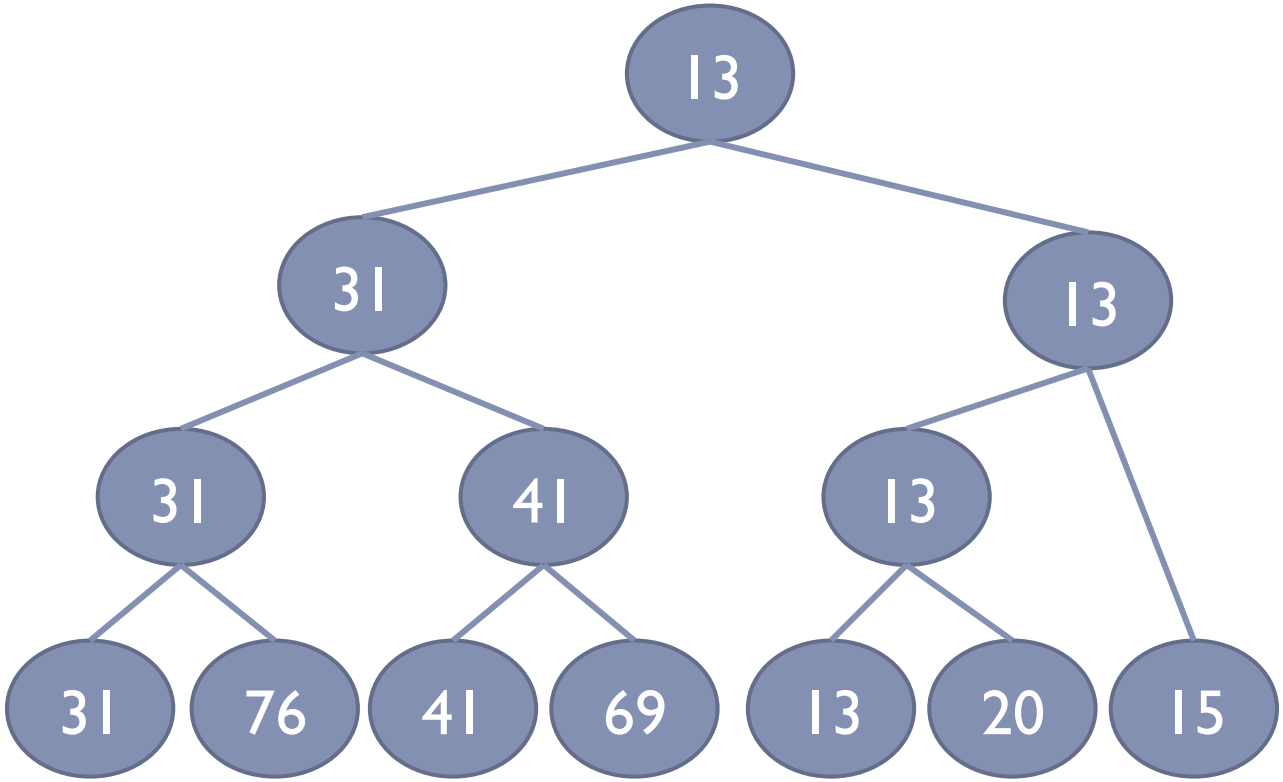

| 31 | 76 | 41 | 69 | 13 | 20 | 10 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | 40 | 51 | 15 |
| HV | | | | HV | HV | 60 |
| | | | | | | HV |

# Árvore Binária de Vencedores



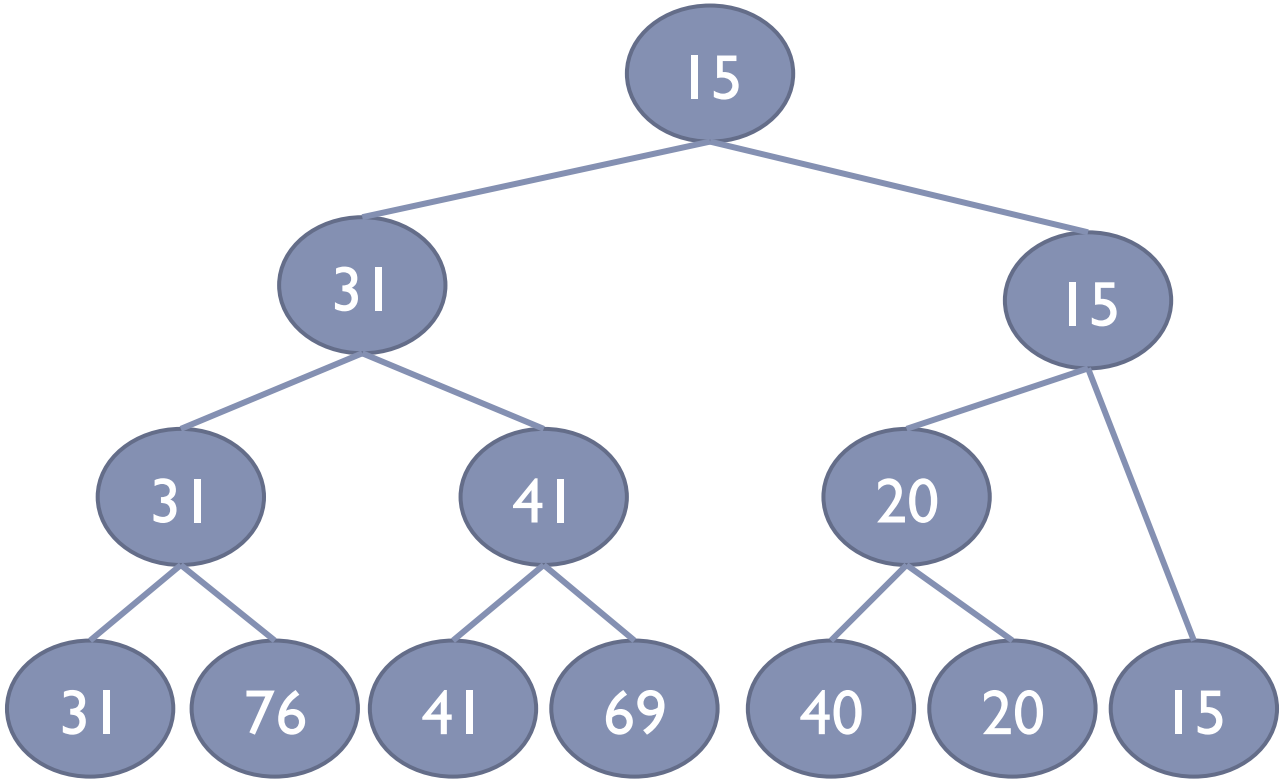

| 31 | 76 | 41 | 69 | 13 | 20 | 10 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | 40 | 51 | 15 |
| HV | | | | HV | HV | 60 |
| | | | | | | HV |

# Árvore Binária de Vencedores



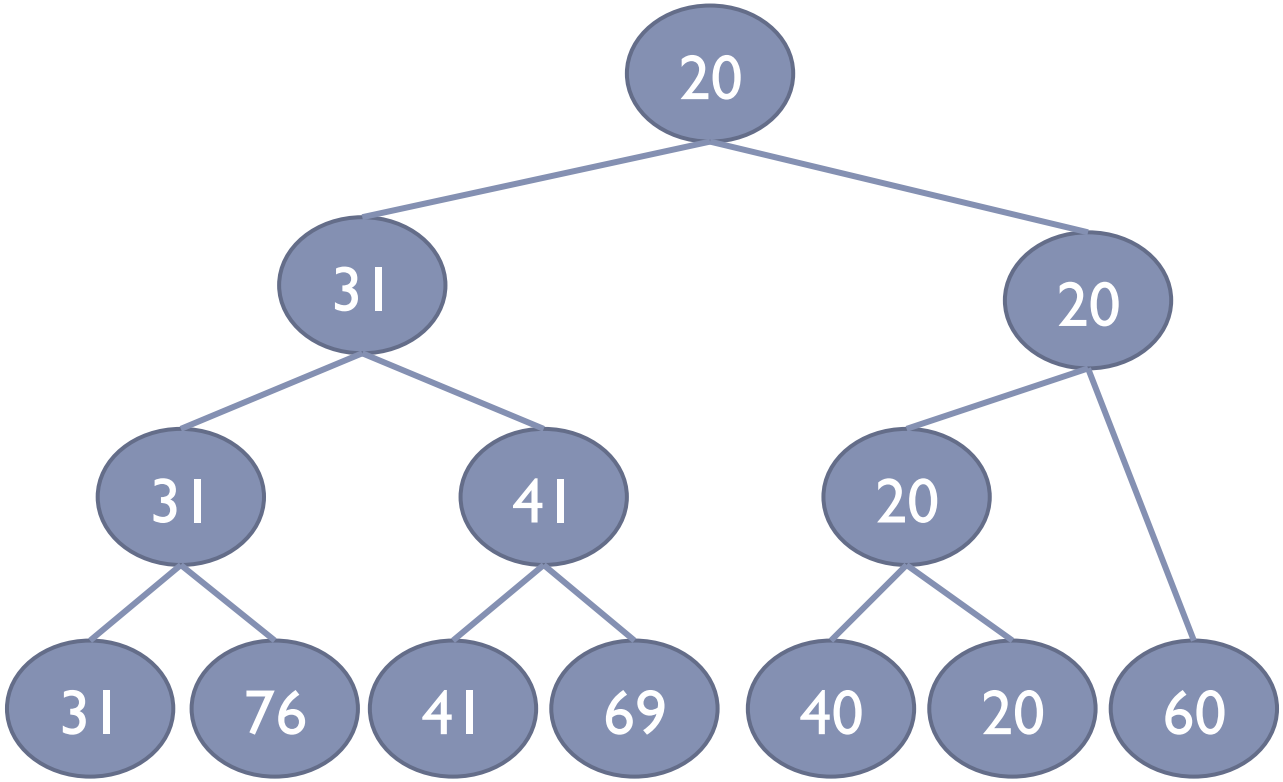

| 31 | 76 | 41 | 69 | 13 | 20 | 15 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | 40 | 51 | 60 |
| HV | | | | HV | HV | HV |
| | | | | | | |

# Árvore Binária de Vencedores



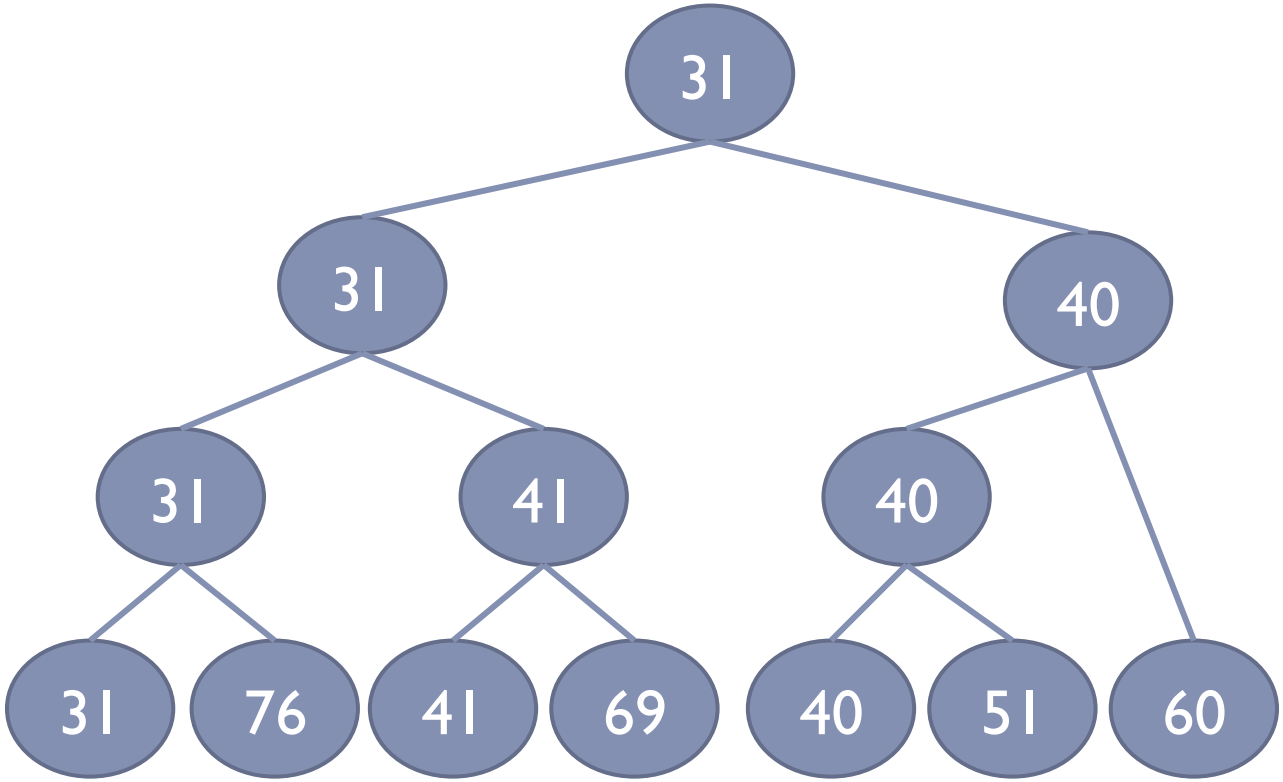

| 31 | 76 | 41 | 69 | 40 | 20 | 15 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | HV | 51 | 60 |
| HV | | | | | HV | HV |
| | | | | | | |

# Árvore Binária de Vencedores



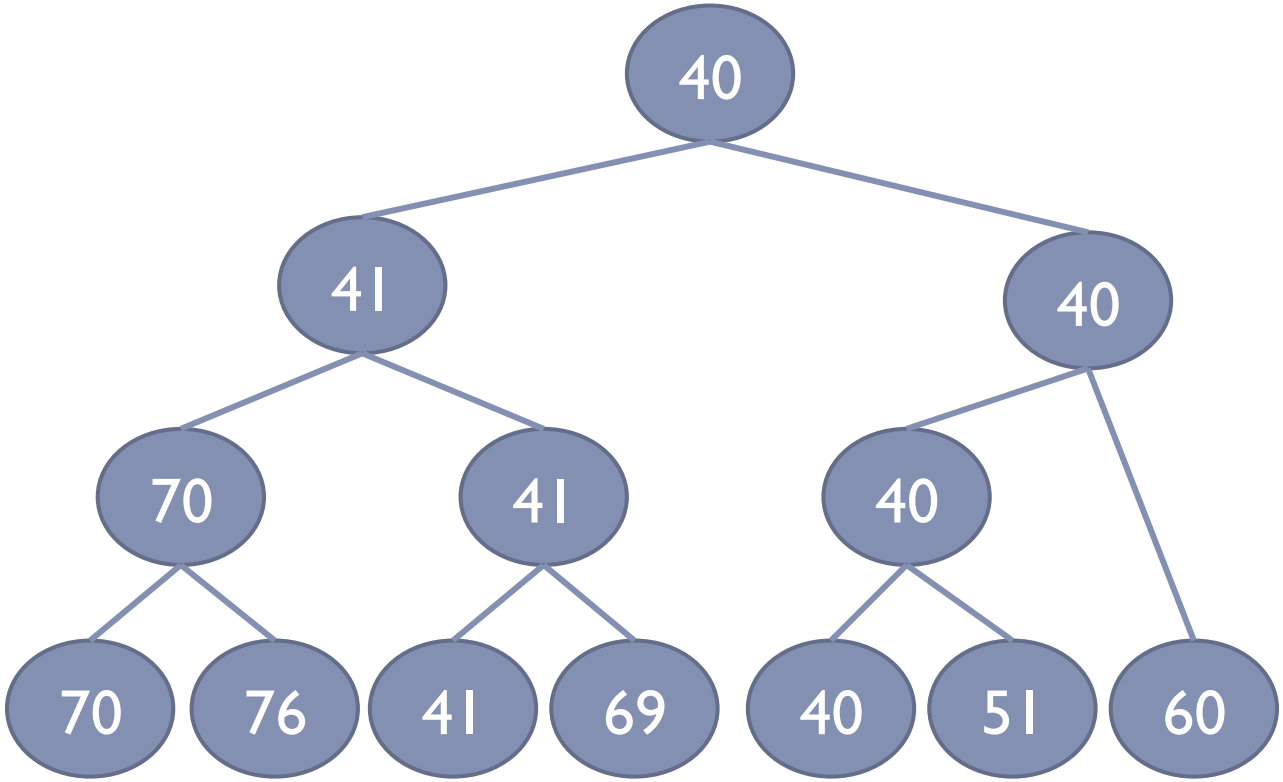

| 31 | 76 | 41 | 69 | 40 | 20 | 60 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | HV | 51 | HV |
| HV | | | | | HV | |
| | | | | | | |

# Árvore Binária de Vencedores



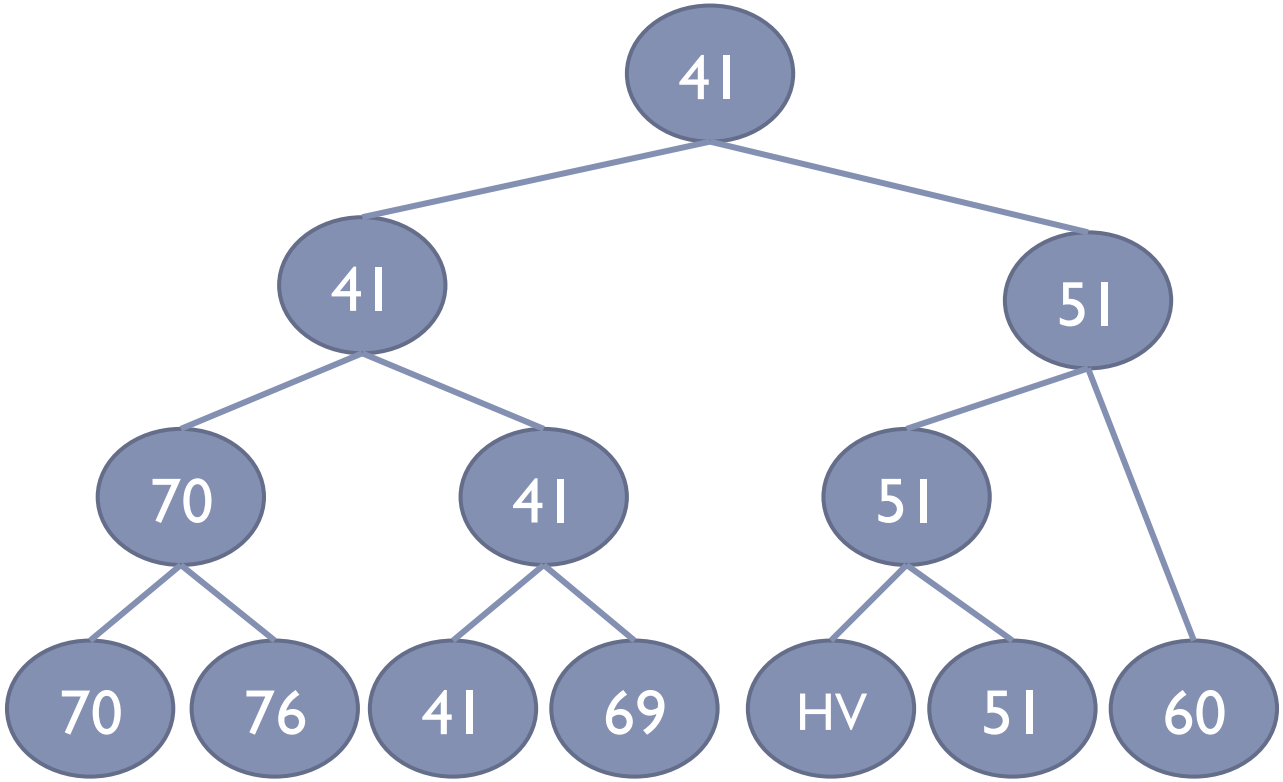

| 31 | 76 | 41 | 69 | 40 | 51 | 60 |
|---|---|---|---|---|---|---|
| 70 | HV | HV | HV | HV | HV | HV |
| HV | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



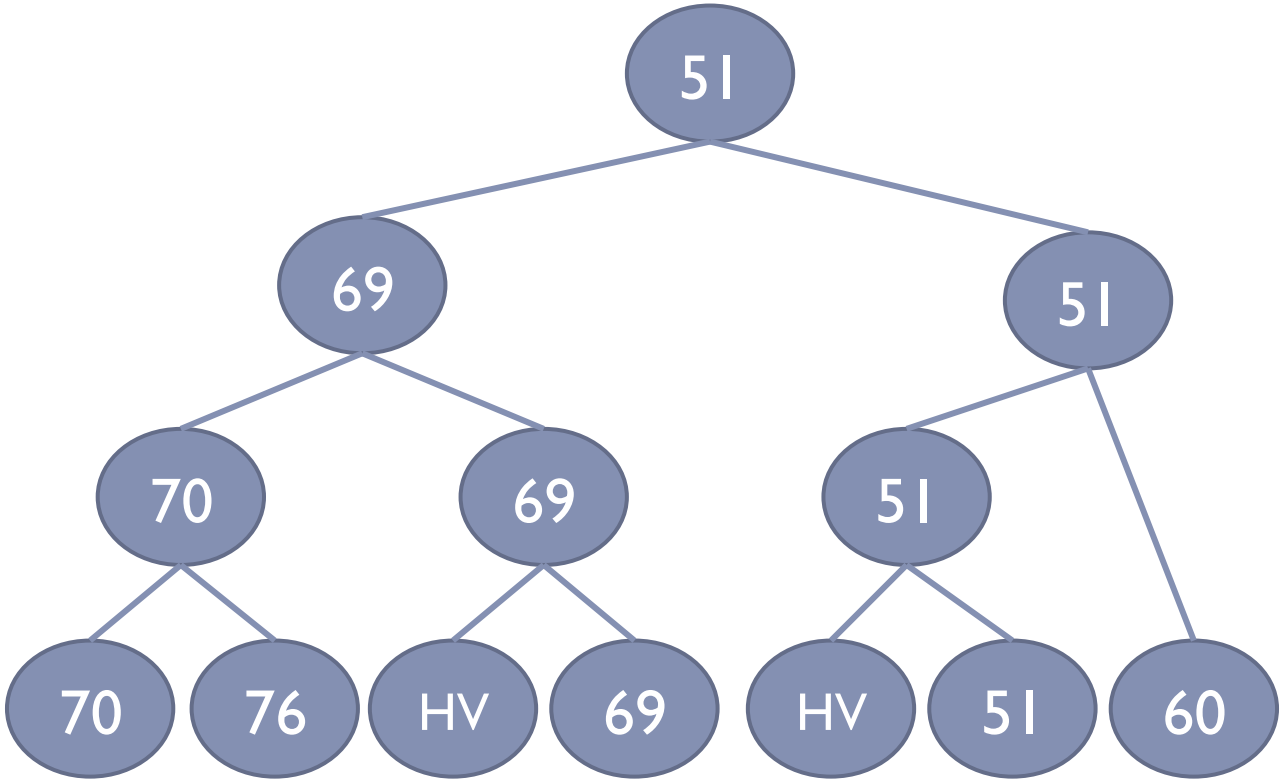

| 70 | 76 | 41 | 69 | 40 | 51 | 60 |
|---|---|---|---|---|---|---|
| HV | HV | HV | HV | HV | HV | HV |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



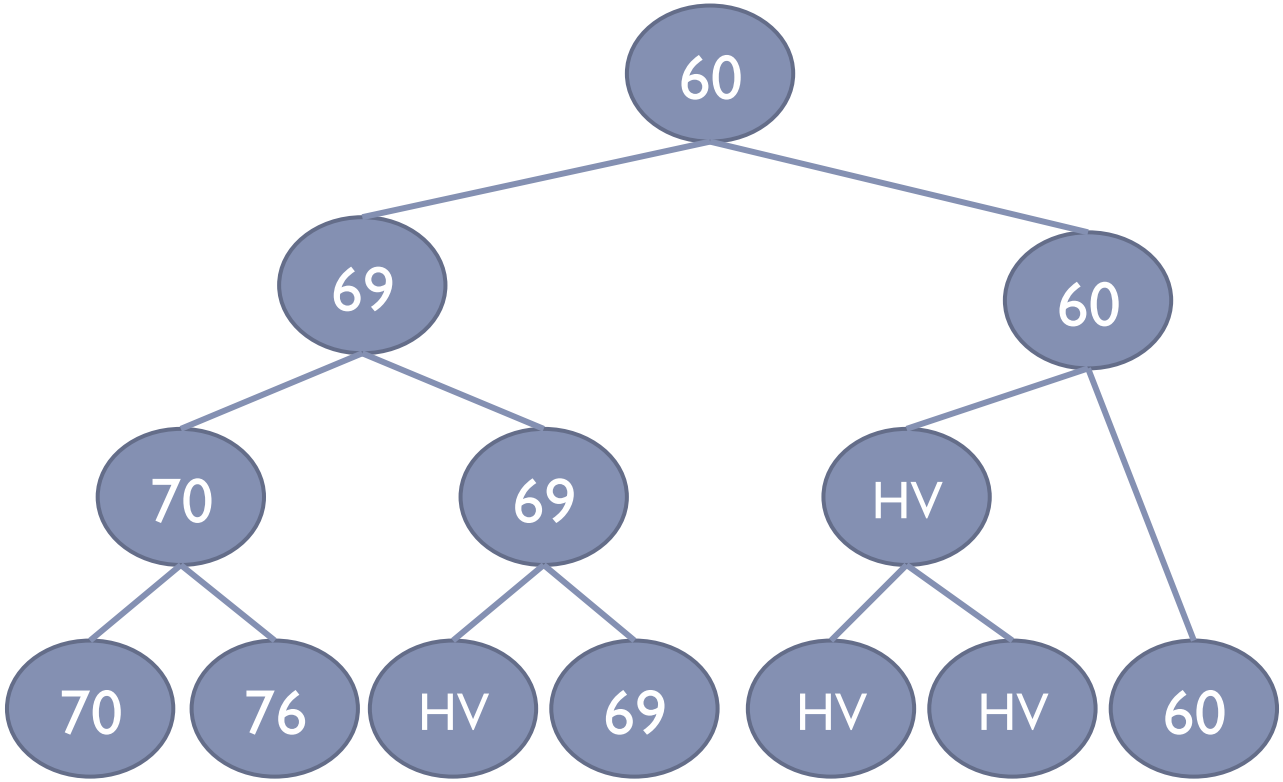

| 70 | 76 | 41 | 69 | HV | 51 | 60 |
|---|---|---|---|---|---|---|
| HV | HV | HV | HV | | HV | HV |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



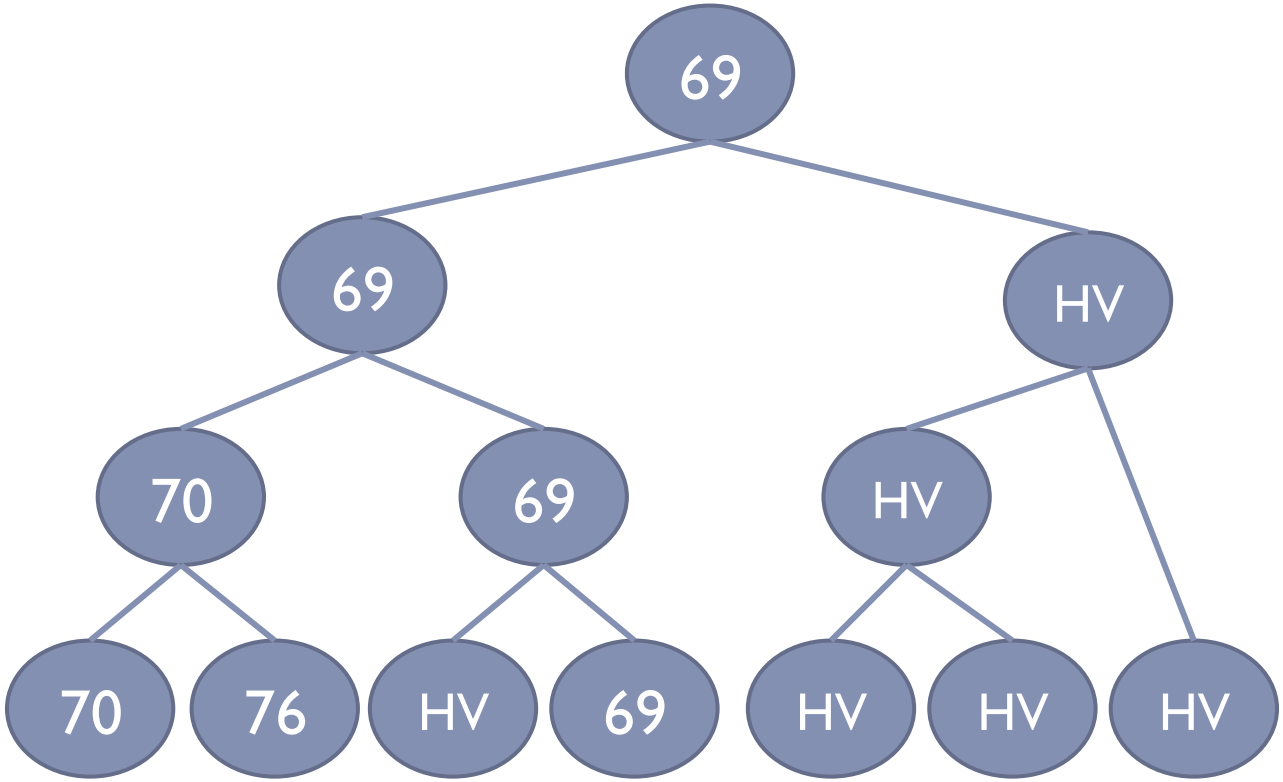

| | | | | | | |
|---|---|---|---|---|---|---|
| 70 | 76 | HV | 69 | HV | 51 | 60 |
| HV | HV | | HV | | HV | HV |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



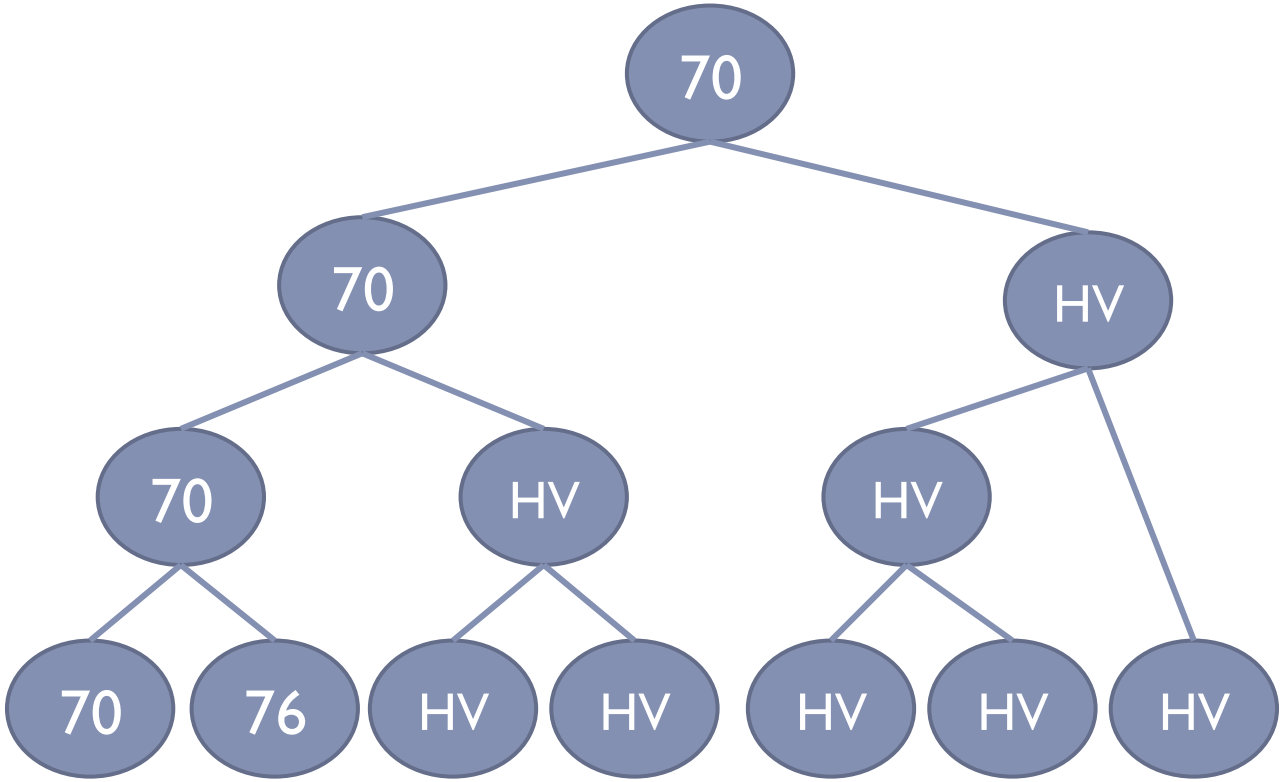

| | | | | | | |
|---|---|---|---|---|---|---|
| 70 | 76 | HV | 69 | HV | HV | 60 |
| HV | HV | | HV | | | HV |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



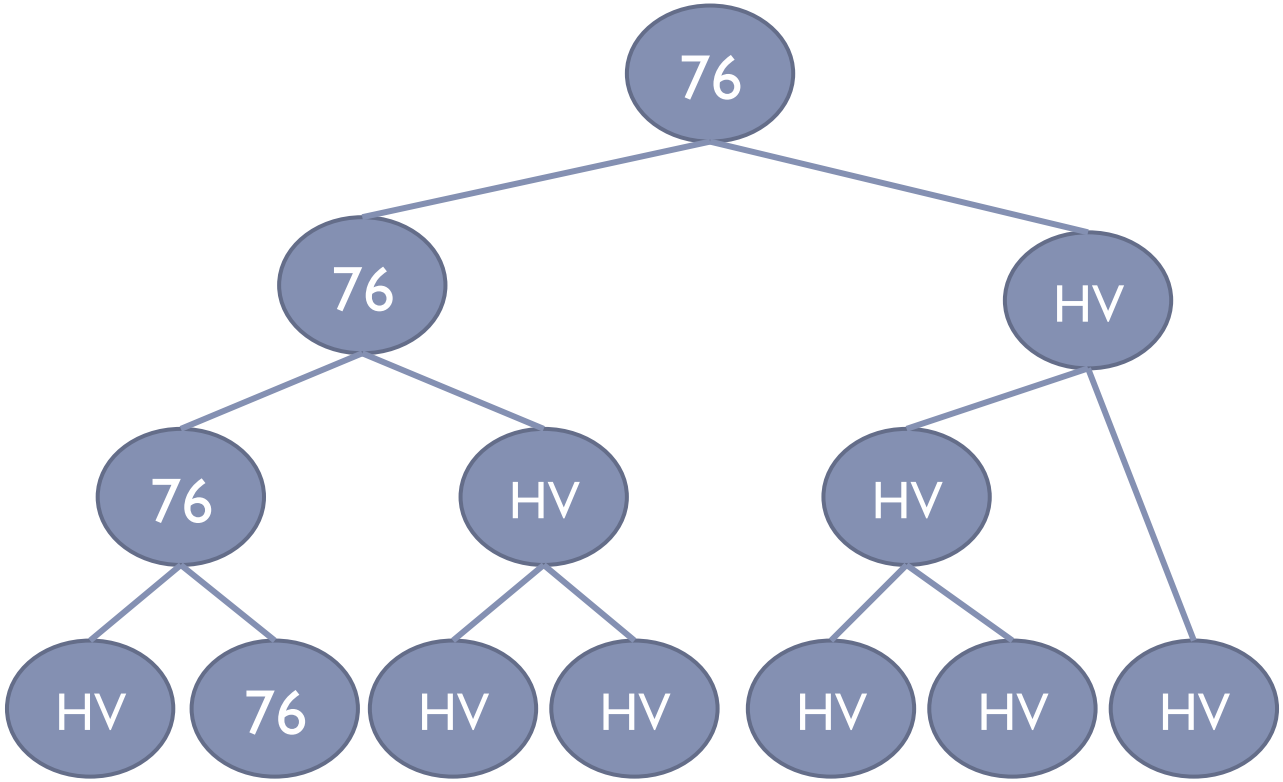

| | | | | | | |
|---|---|---|---|---|---|---|
| 70 | 76 | HV | 69 | HV | HV | HV |
| HV | HV | | HV | | | |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores



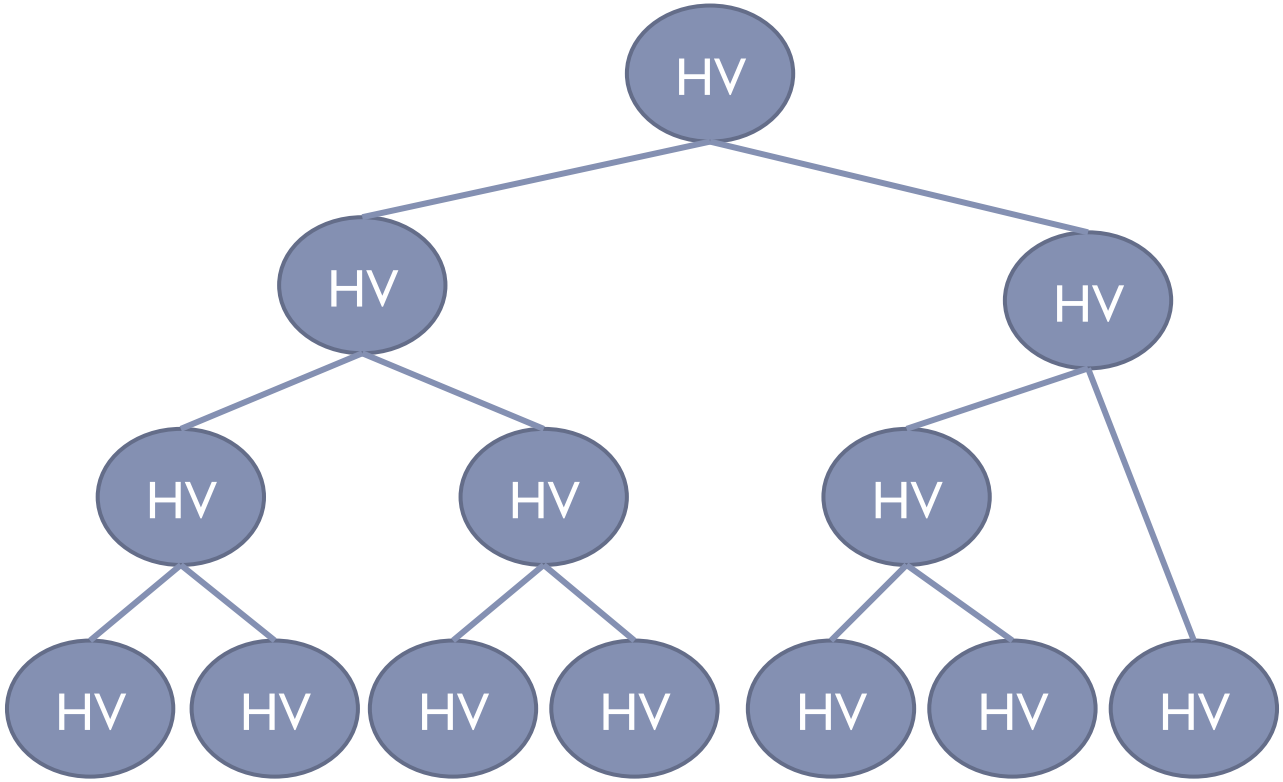

| 70 | 76 | HV | HV | HV | HV | HV |
|---|---|---|---|---|---|---|
| HV | HV | | | | | |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores




| | | | | | | |
|---|---|---|---|---|---|---|
| HV | 76 | HV | HV | HV | HV | HV |
| | HV | | | | | |
| | | | | | | |
| | | | | | | |

# Árvore Binária de Vencedores




| HV | HV | HV | HV | HV | HV | HV |
|---|---|---|---|---|---|---|
| | | | | | | |
| | | | | | | |
| | | | | | | |

# 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
- Simular a execução do algoritmo de intercalação

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

# Problema

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

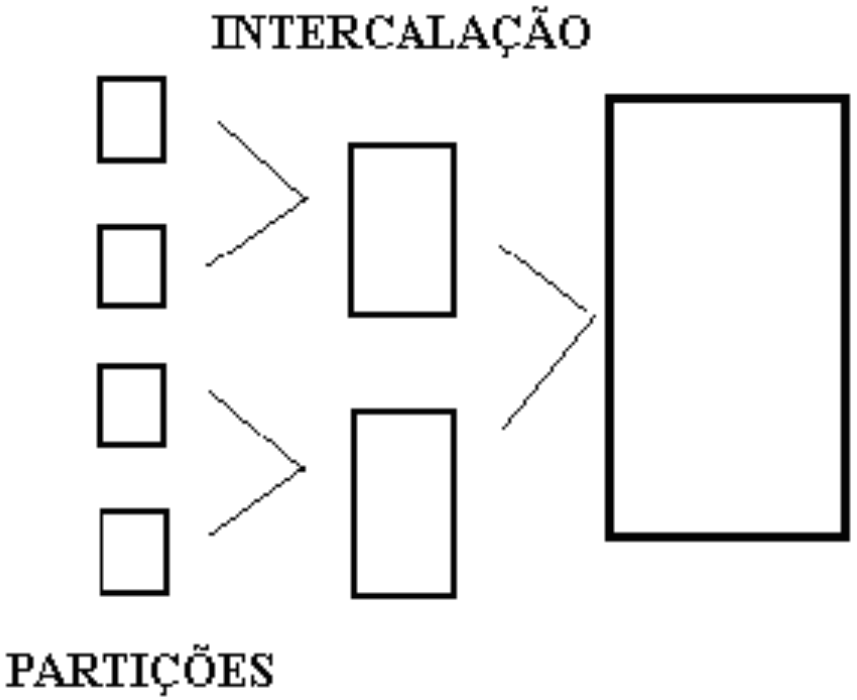
  (i) O número de arquivos a intercalar pode gerar uma árvore de vencedores maior do que a capacidade da memória

  (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

# Solução

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

# Etapa de Intercalação



# Estágio de Intercalação

Estratégias de distribuição e intercalação:

- Intercalação balanceada de N caminhos
- Intercalação ótima

# 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

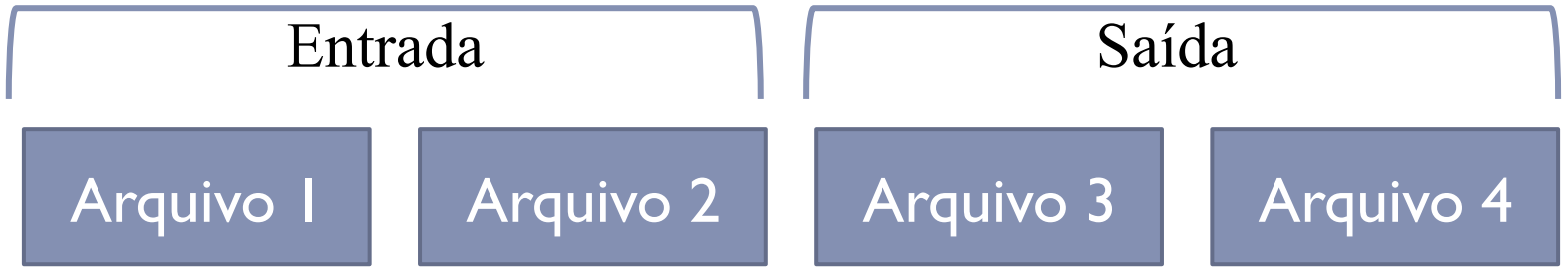
# Intercalação balanceada de N caminhos

# Intercalação Balanceada de N caminhos

- Primeiro passo: determinar o número de arquivos (*F*) que o algoritmo irá manipular
  - Metade dos arquivos (*F*/2) será usada para leitura (entrada)
  - A outra metade (*F*/2), para escrita (saída)

# Exemplo

- Número de arquivos $F = 4$

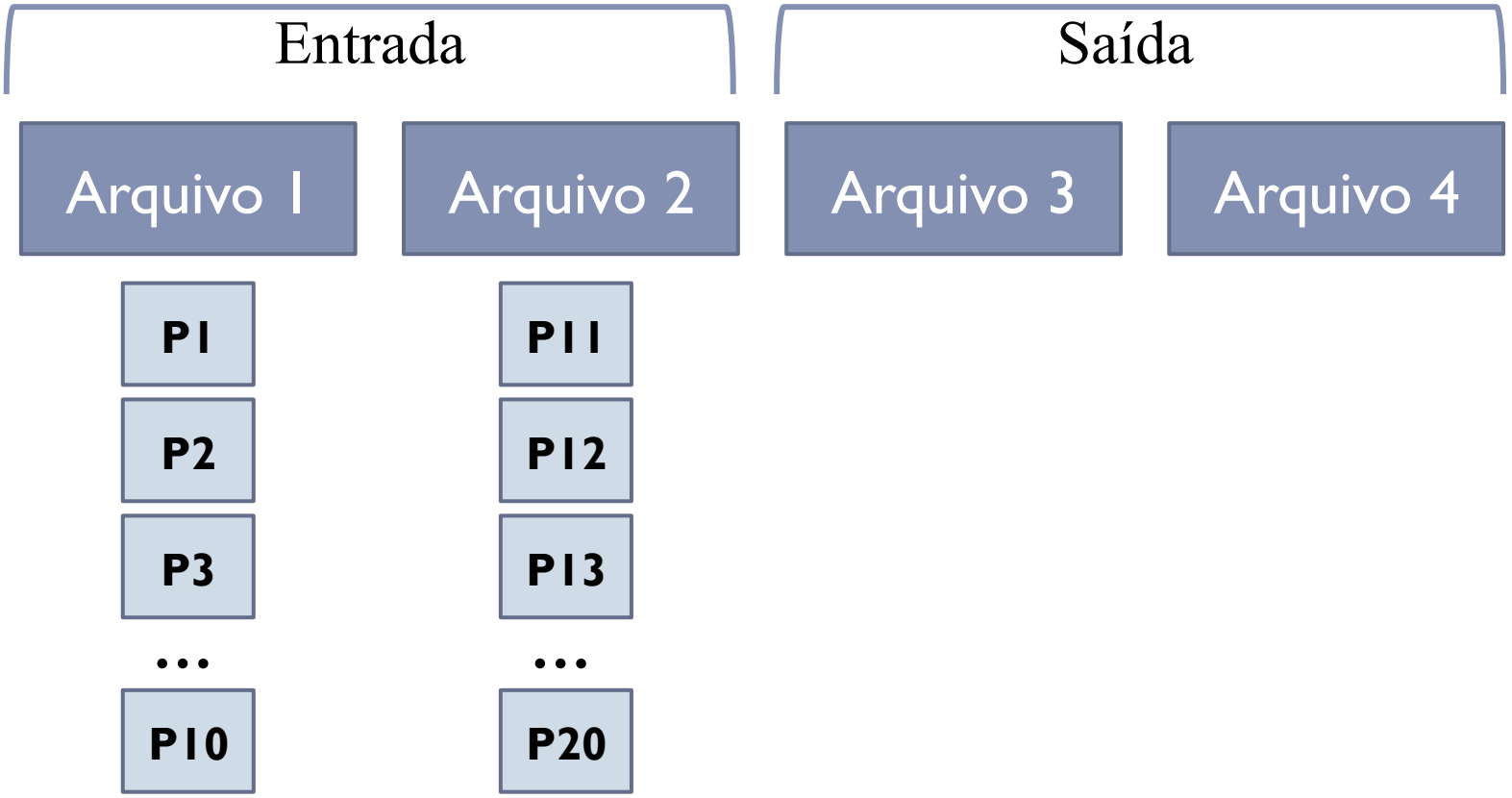


# Intercalação Balanceada de N caminhos

- Passo 2: Distribuir todas as partições, tão equilibradamente quanto possível, nos *F/2* arquivos de entrada
  - Atenção: aqui fazemos apenas uma "fila" para ver que "variável de arquivo" vai processar cada partição

# Exemplo

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20

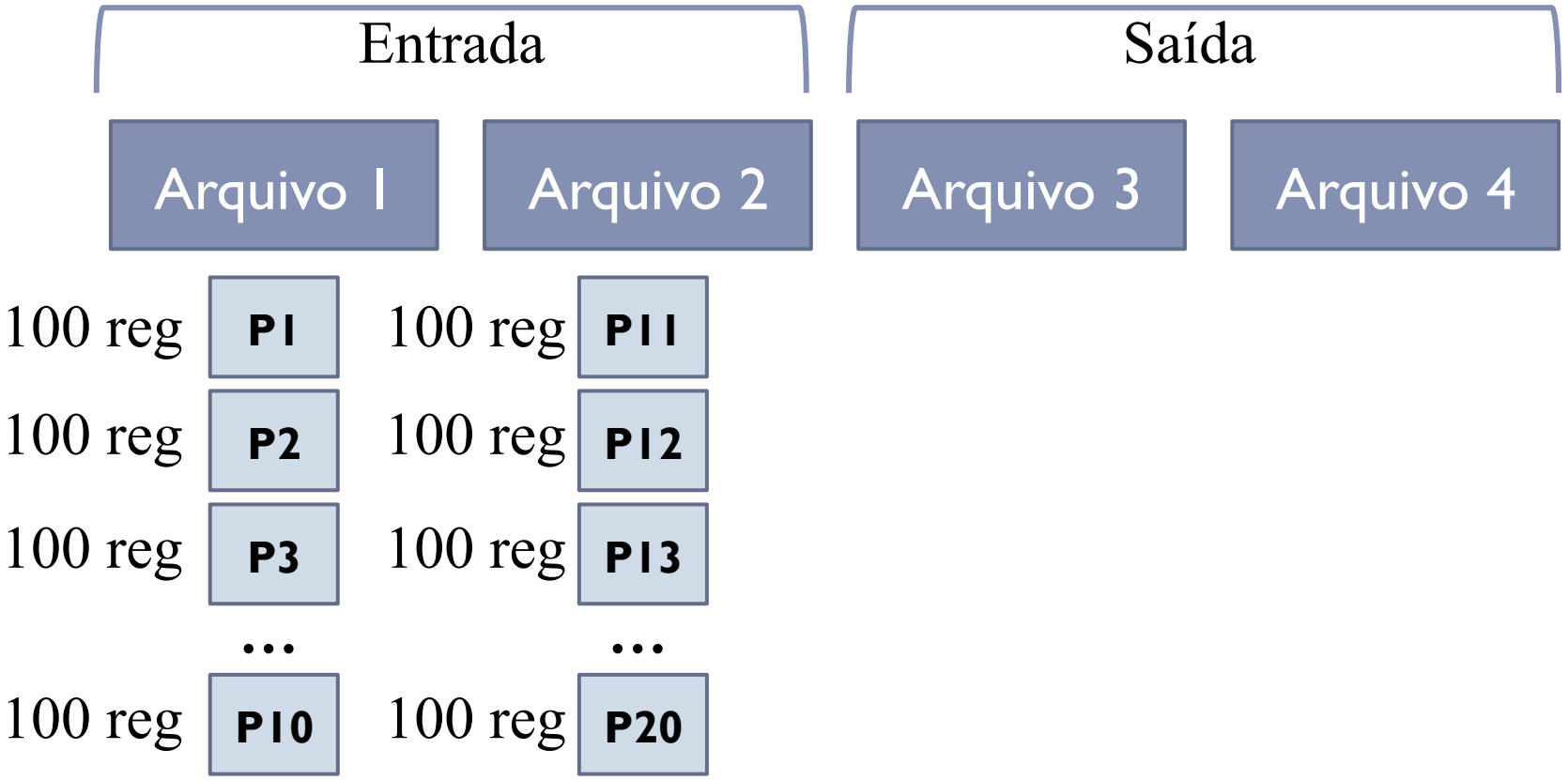


# Intercalação Balanceada de N caminhos

- Início da fase de intercalação: intercalar as primeiras F/2 partições, gravando o resultado em um dos F/2 arquivos de saída

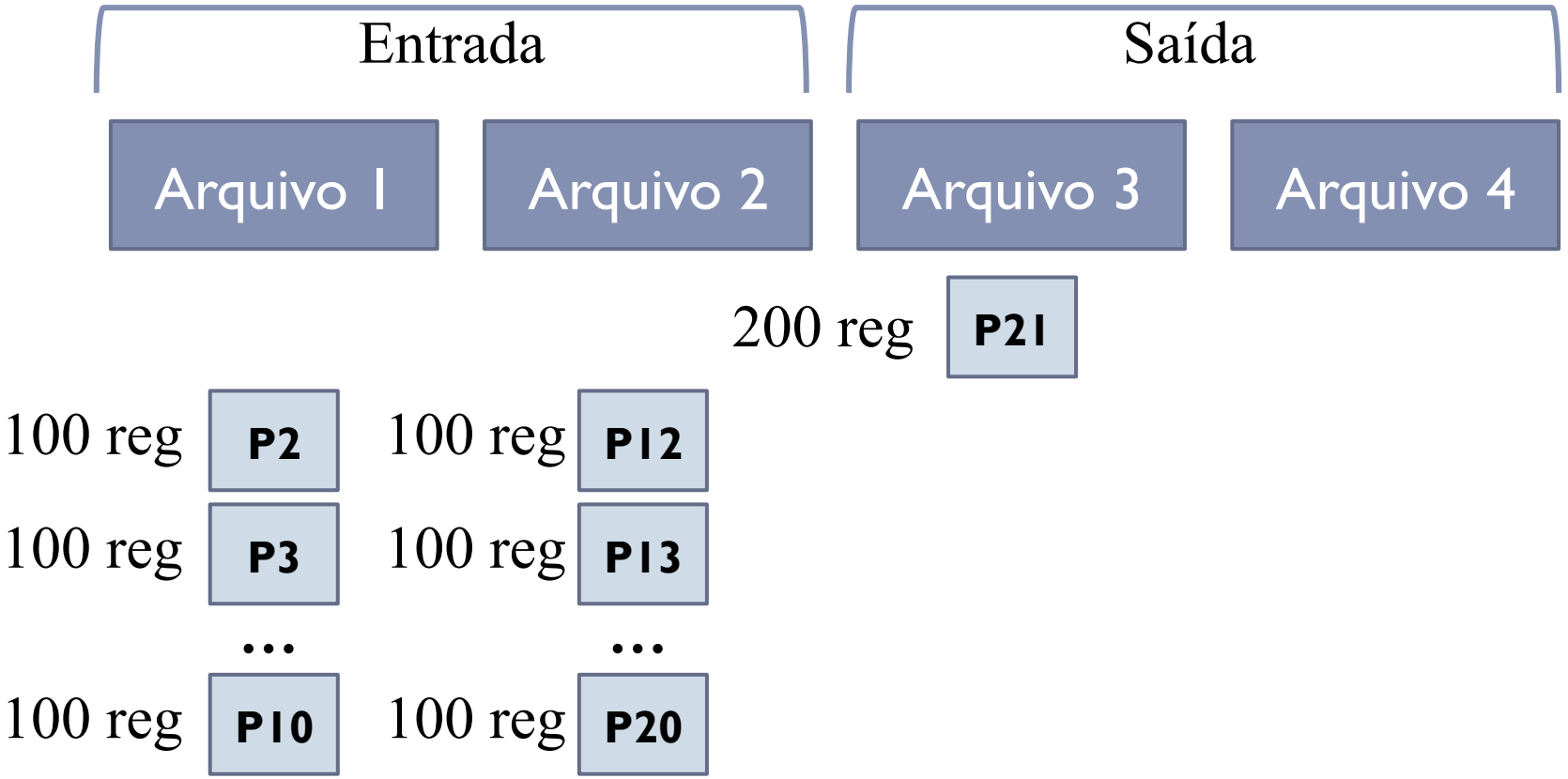
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



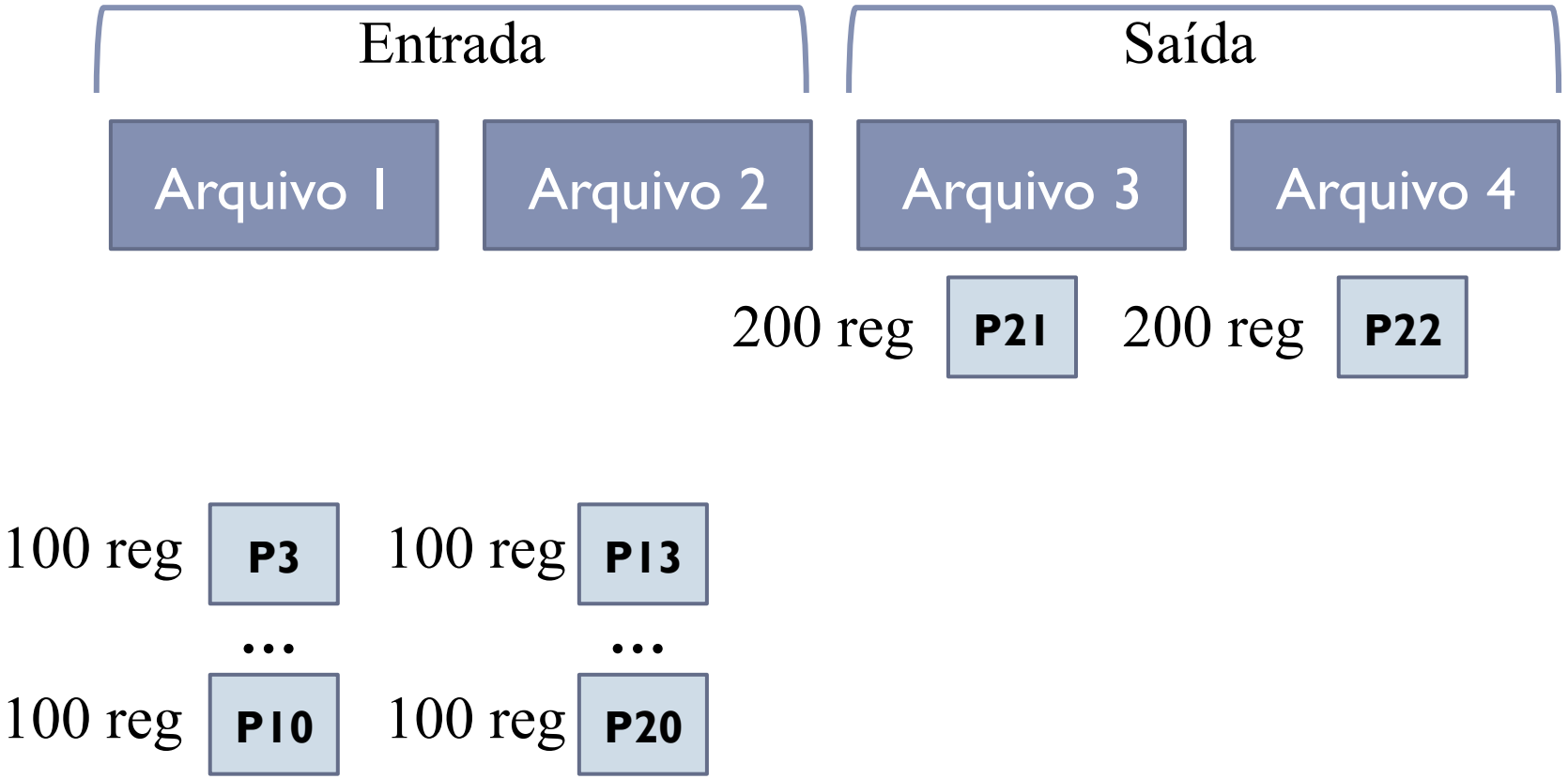
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



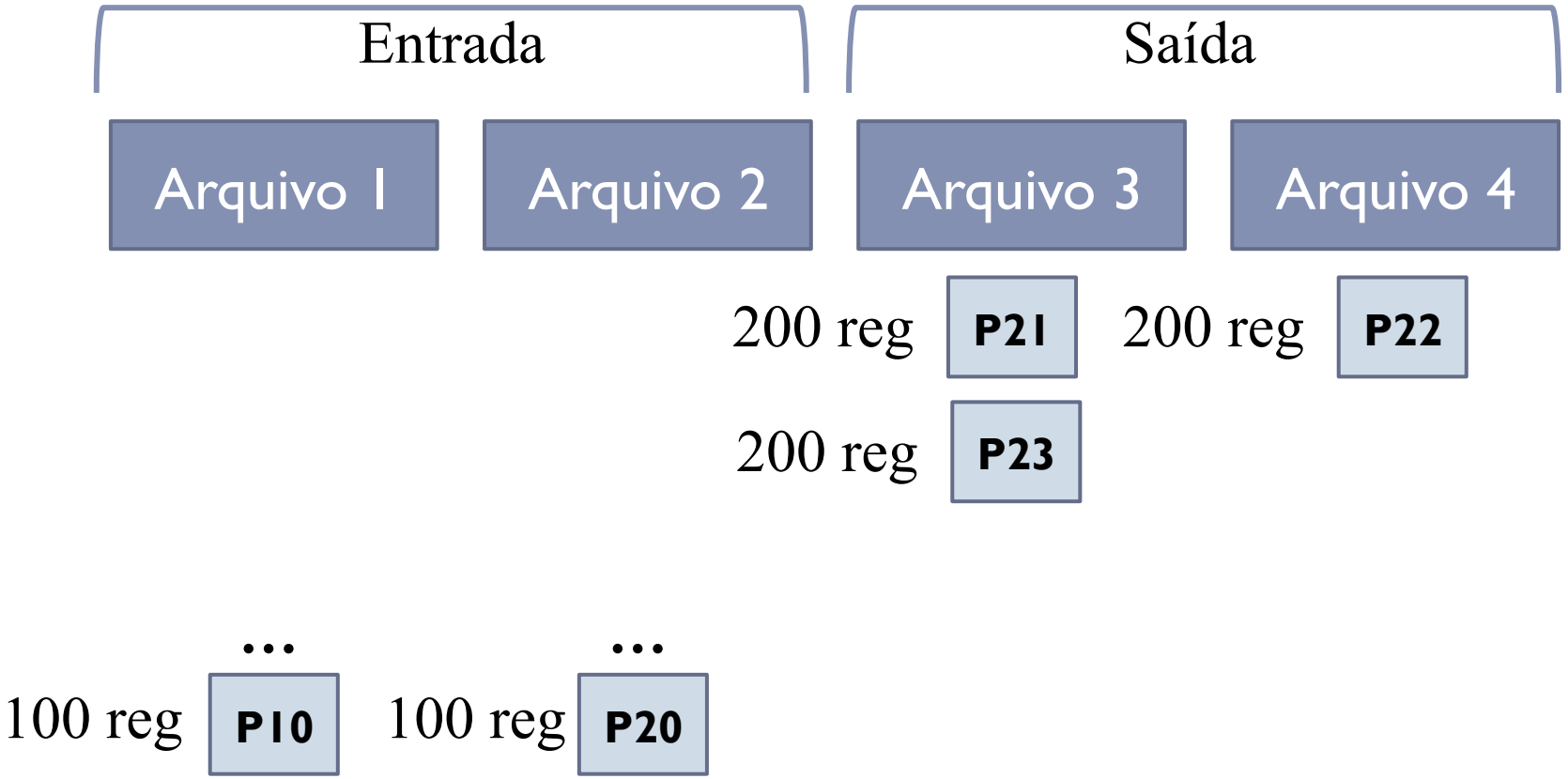
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



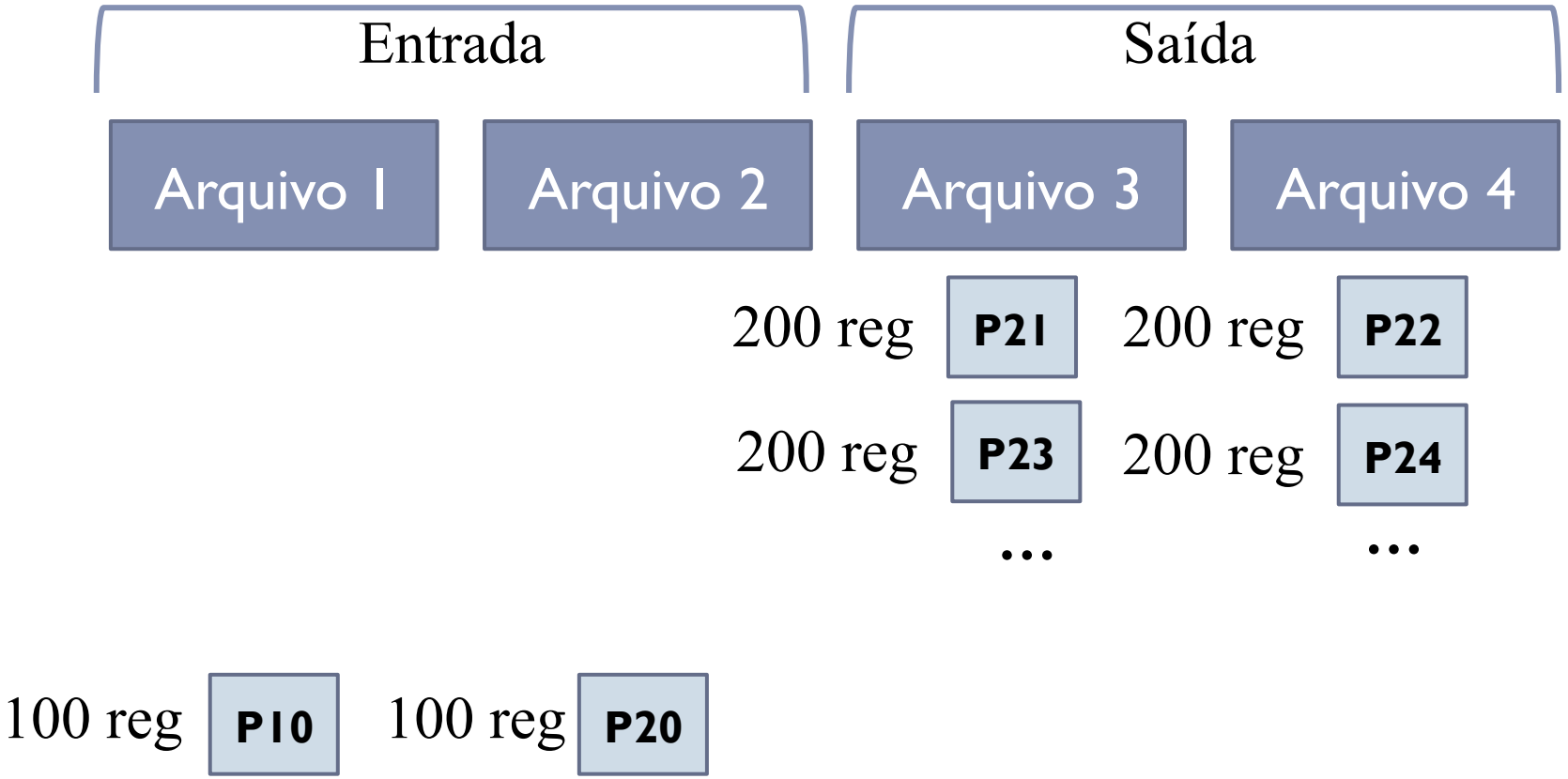
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



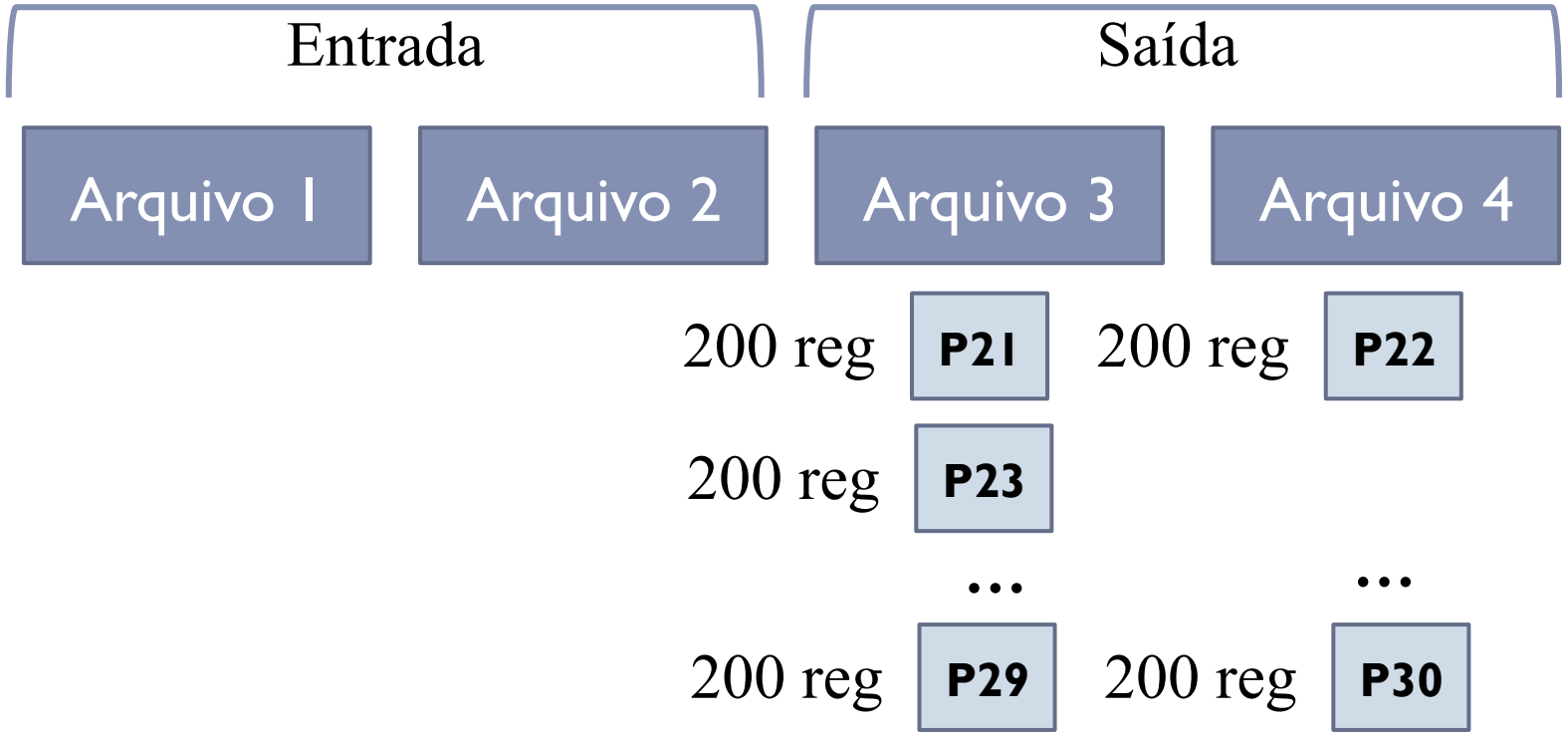
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



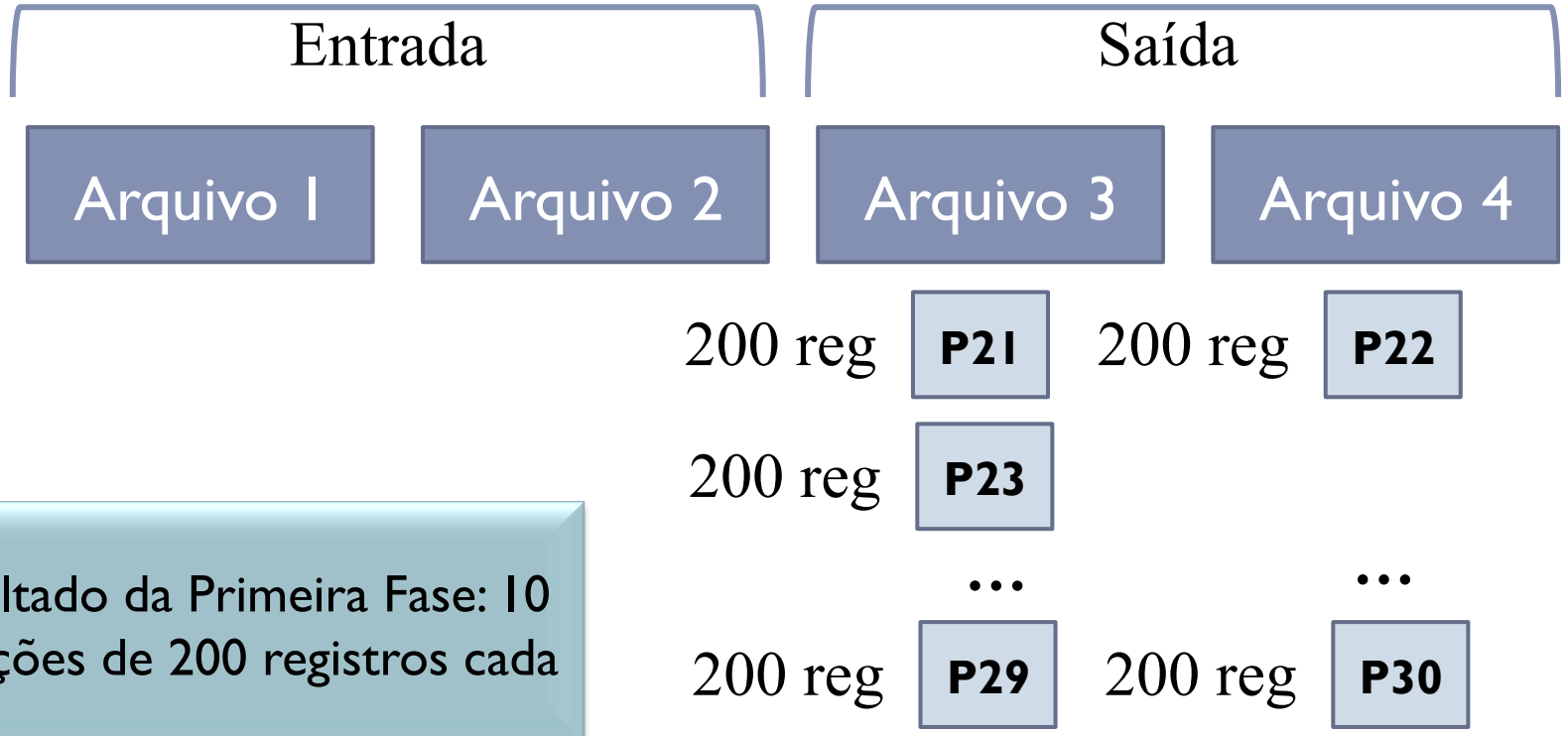
# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



# Exemplo: Fase 1

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



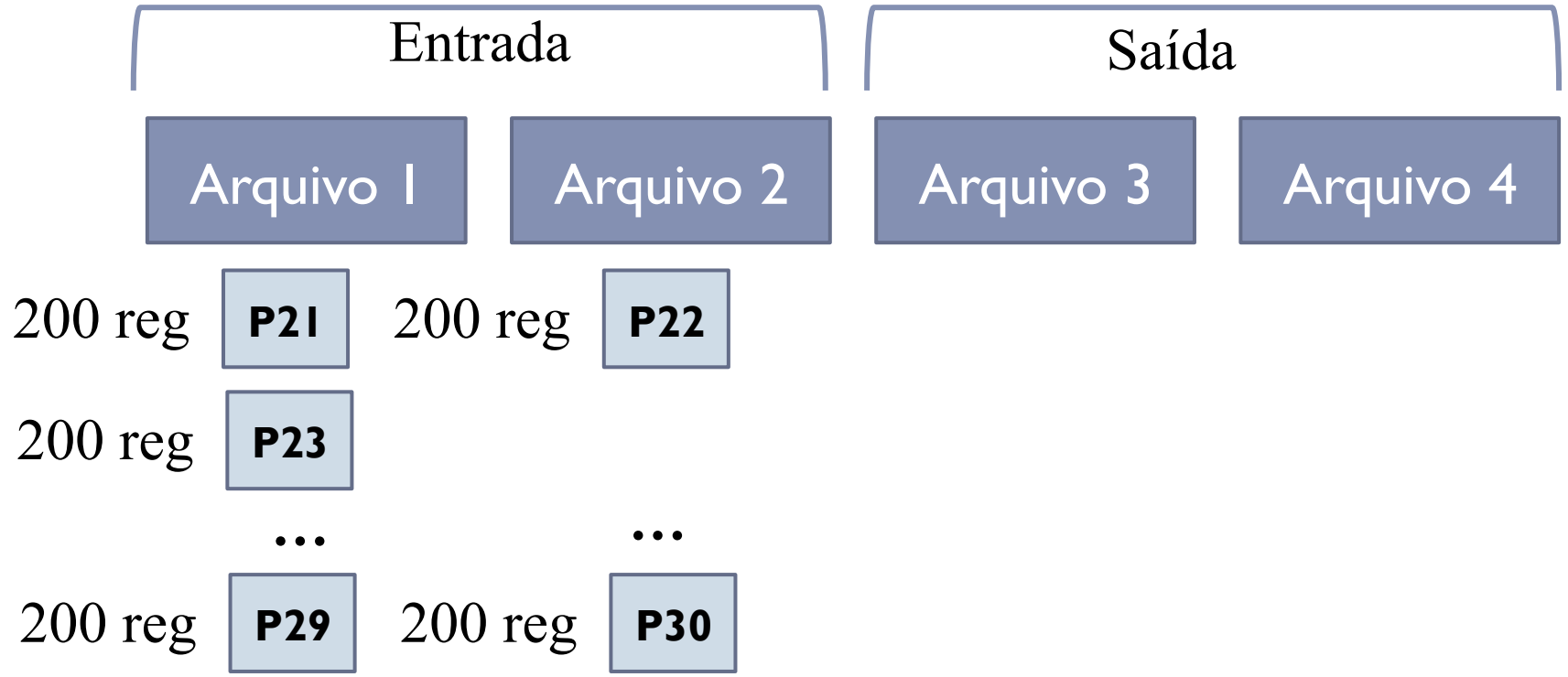

Resultado da Primeira Fase: 10 partições de 200 registros cada

# Intercalação Balanceada de N caminhos

- No final de cada fase, o conjunto de partições de saída torna-se o conjunto de entrada

# Exemplo: Fase 2

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 10 (200 registros cada)



# Intercalação Balanceada de N caminhos

- A intercalação termina quando, em uma fase, grava-se apenas uma partição

# Número de Fases: 5

- Fase 1: 20 partições com 100 registros cada
- Fase 2: 10 partições com 200 registros cada
- Fase 3: 5 partições com 400 registros cada
- Fase 4: 2 partições com 800 registros cada + 1 partição com 400 registros cada
- Fase 5: 1 partição com 1600 registros + 1 partição com 400 registros

- Resultado da Fase 5: 1 partição com 2000 registros

# Número de Fases: 5

- Fase 1: 20 partições com 100 registros cada
- Fase 2: 10 partições com 200 registros cada
- Fase 3: 5 partições com 400 registros cada
- Fase 4: 2 partições com 800 registros cada + **1 partição com 400 registros cada**
- Fase 5: 1 partição com 1600 registros + **1 partição com 400 registros**

Pode ocorrer que partições sejam copiadas de um arquivo para outro sem qualquer processamento

# Número de Passos

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

- Número de registros lidos
  - Fase 1: 20 partições com 100 registros cada = 2000
  - Fase 2: 10 partições com 200 registros cada = 2000
  - Fase 3: 5 partições com 400 registros cada = 2000
  - Fase 4: 2 partições com 800 registros cada + 1 partição com 400 registros cada = 2000
  - Fase 5: 1 partição com 1600 registros + 1 partição com 400 registros = 2000

2000 * 5 = 10000 registros lidos

# Número de Passos

$$\text{Número de passos} = \frac{10000}{\text{No. total de registros no arquivo classificado}}$$

- Número total de registros no arquivo classificado: 2000

# Número de Passos

$$\text{Número de passos} = \frac{10000}{2000} = 5$$

- Portanto: número de passos = número de fases

# Intercalação Balanceada de N caminhos

- O balanceamento do processo baseia-se em colocar nos arquivos de entrada aproximadamente o mesmo número de registros

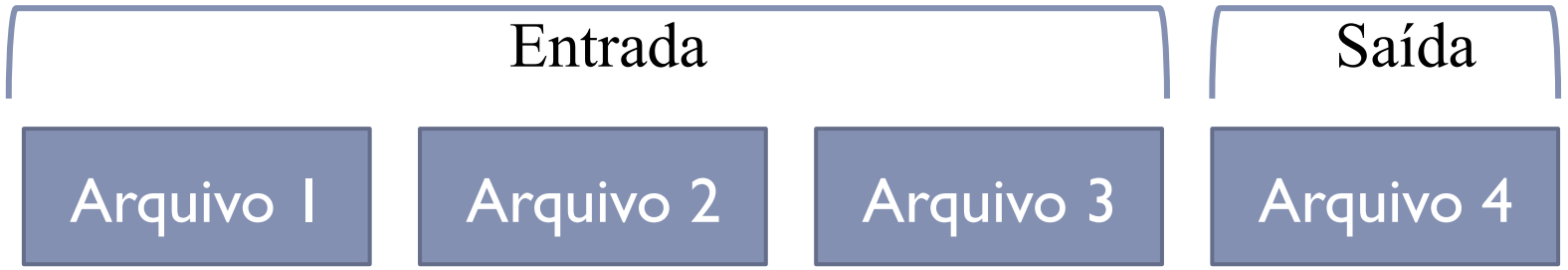
# Intercalação Ótima

# Intercalação Ótima

- F arquivos
  - F – 1 para entrada
  - 1 para saída

# Exemplo

- Número de arquivos *F* = 4

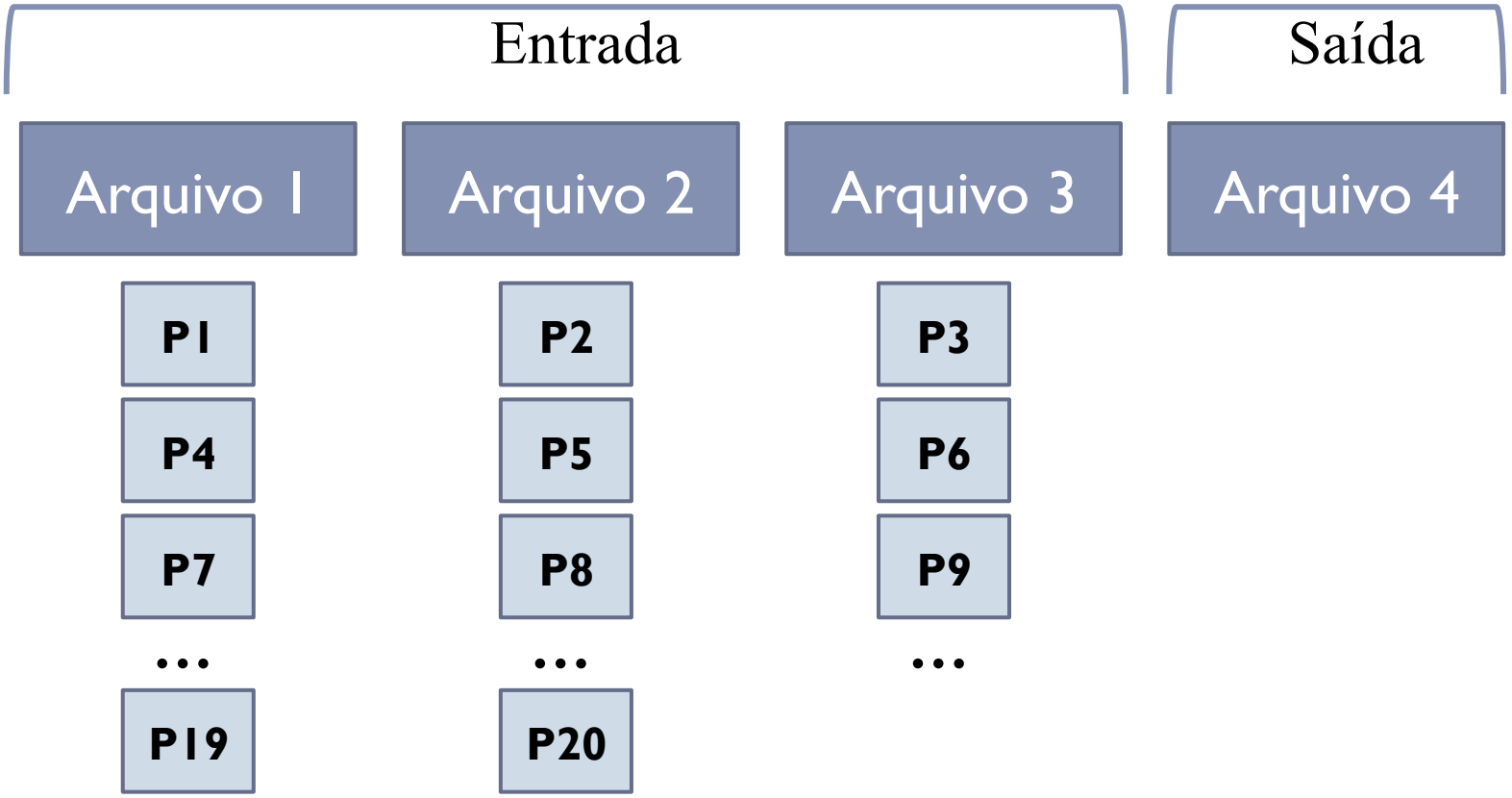


# Intercalação Ótima

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

# Exemplo

- Número de arquivos $F = 4$
- Partições a serem intercaladas = 20

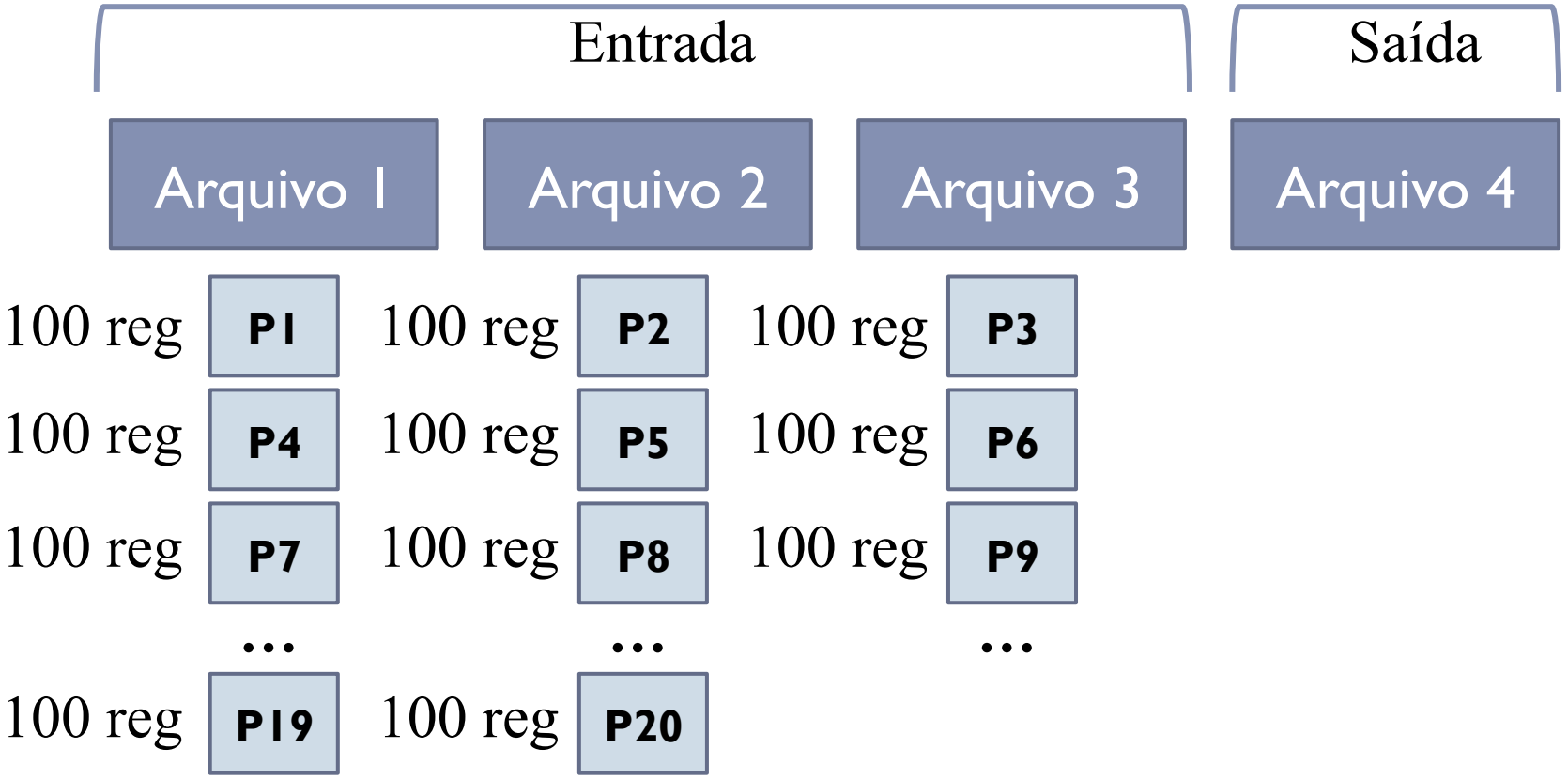


# Intercalação Ótima

- 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

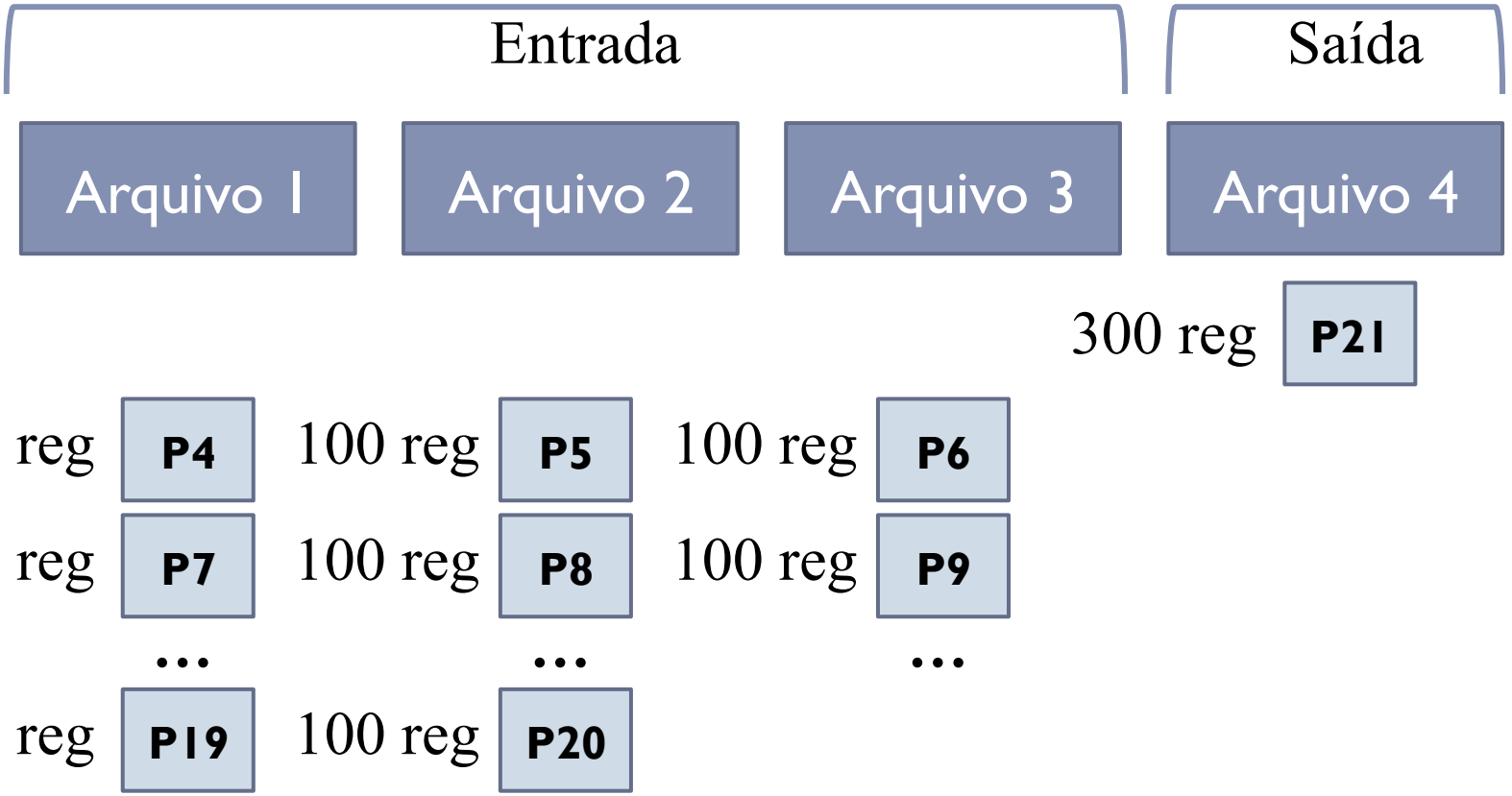
# Exemplo

- Número de arquivos $F = 4$
- Partições a serem intercaladas = 20 (100 registros cada)



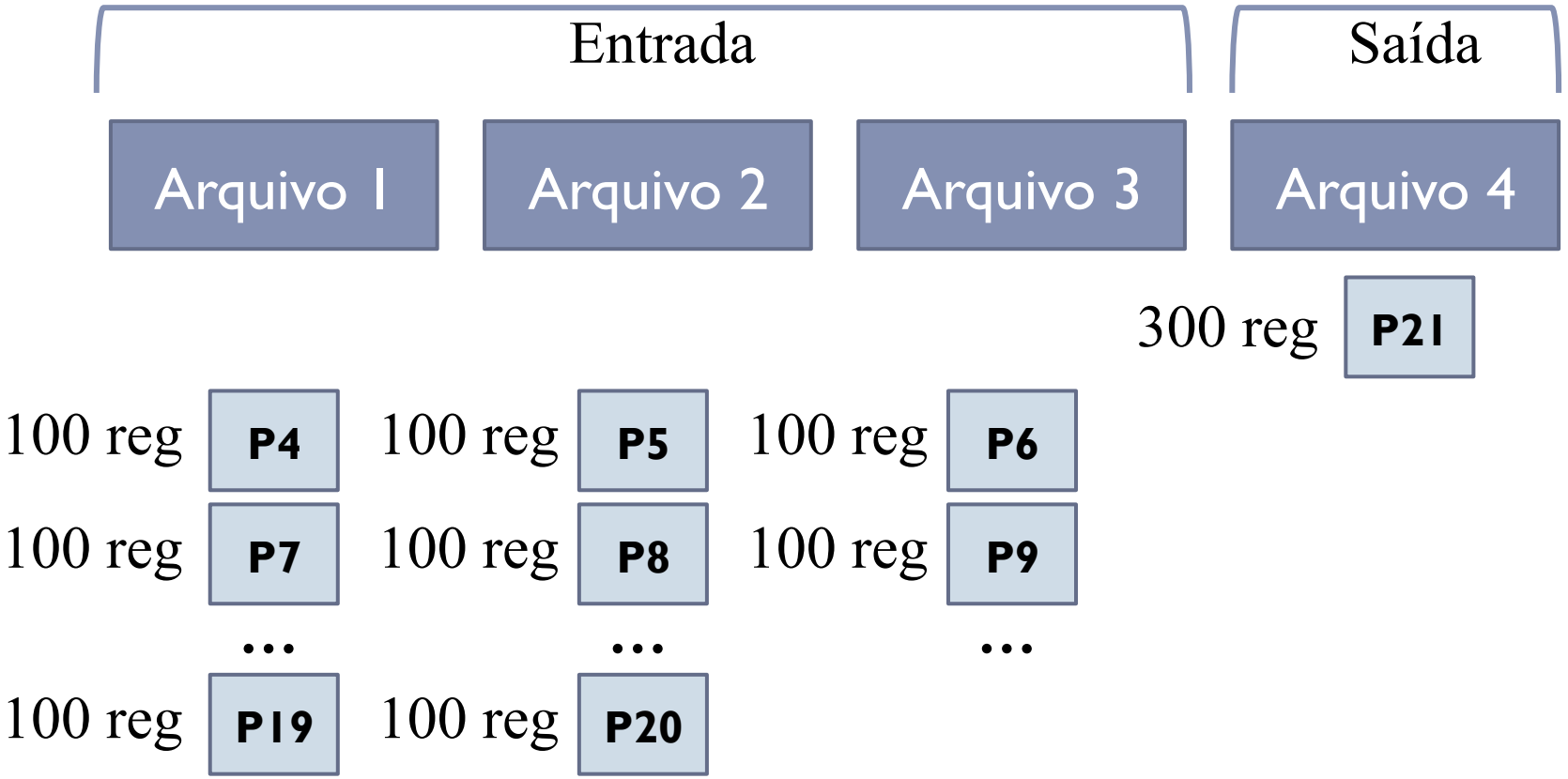
# Exemplo

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



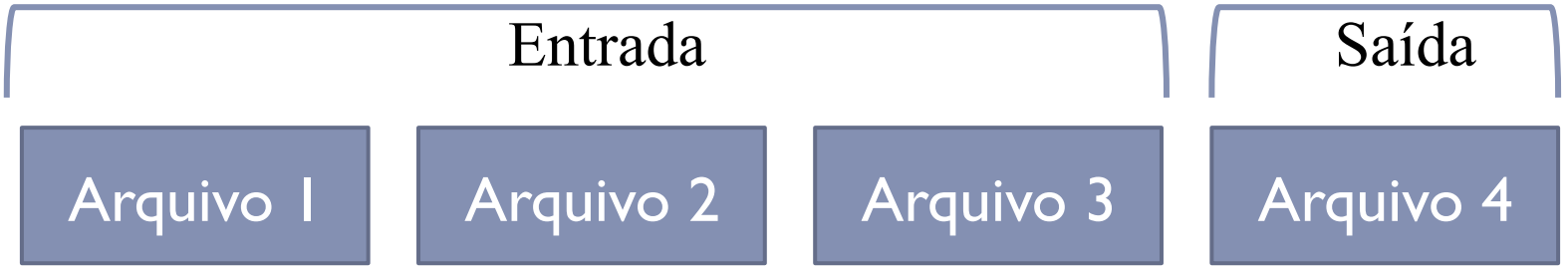
# Exemplo

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



# Exemplo

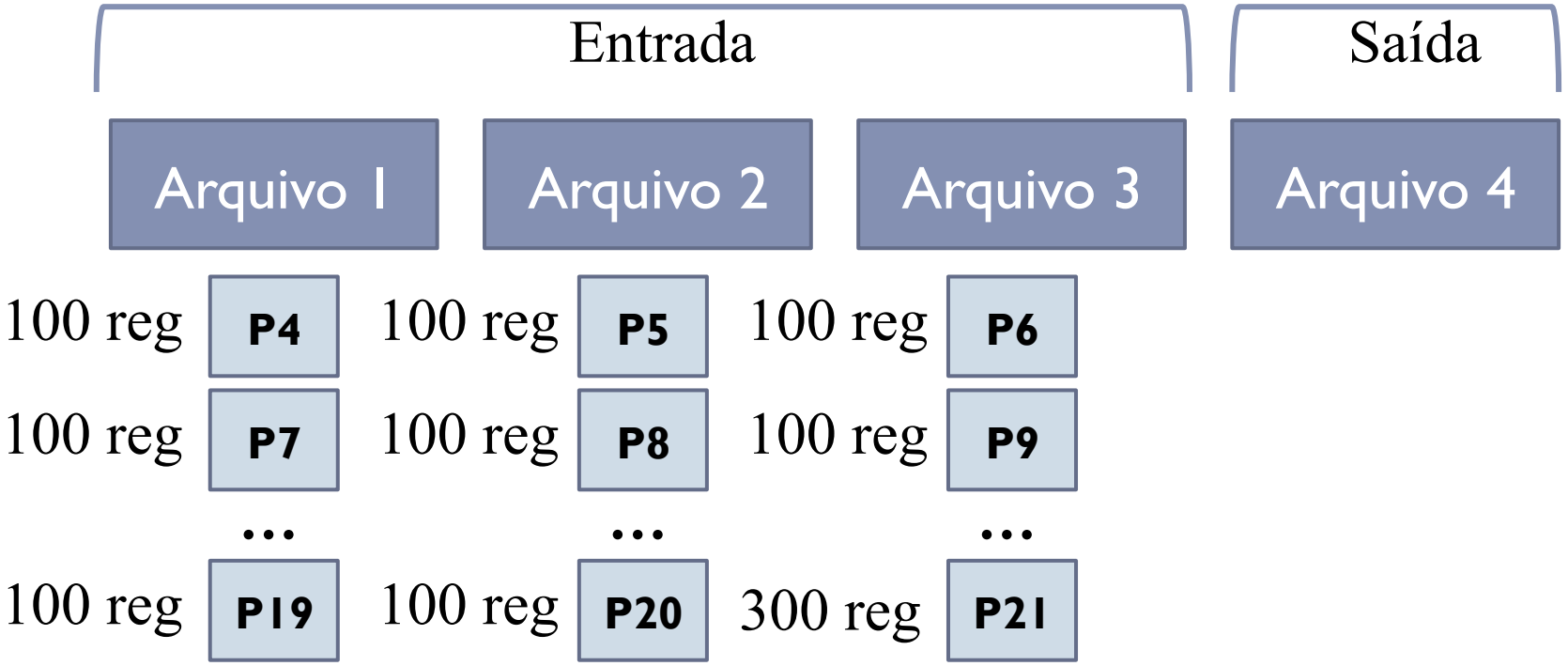
- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)




| | | |
|---|---|---|
| 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 |

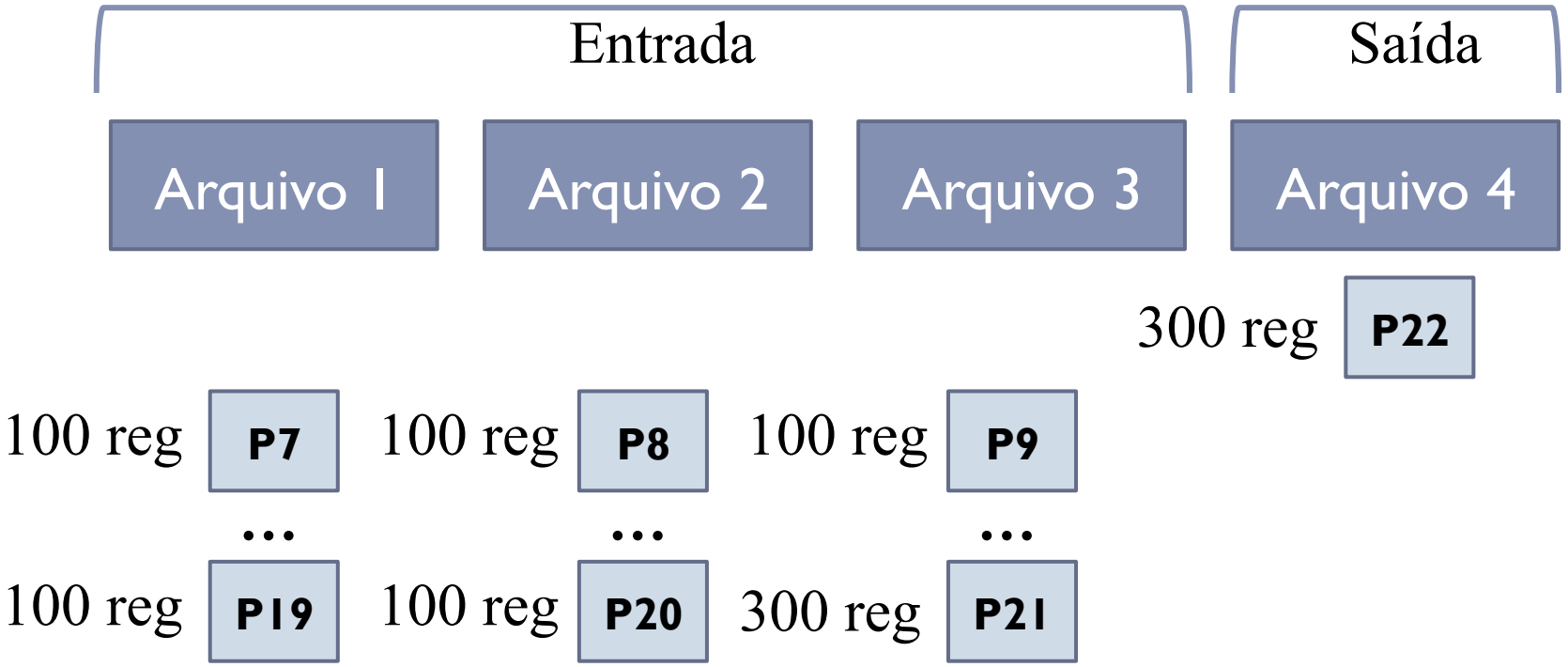
# Exemplo

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (100 registros cada)



# Exemplo

- Número de arquivos *F* = 4
- Partições a serem intercaladas = 20 (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 |

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

# 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 |
| 10 | 28:900 | 29:1100 | ---------- | 30:2000 | 2000 |
| | | | | TOTAL | 6300 |

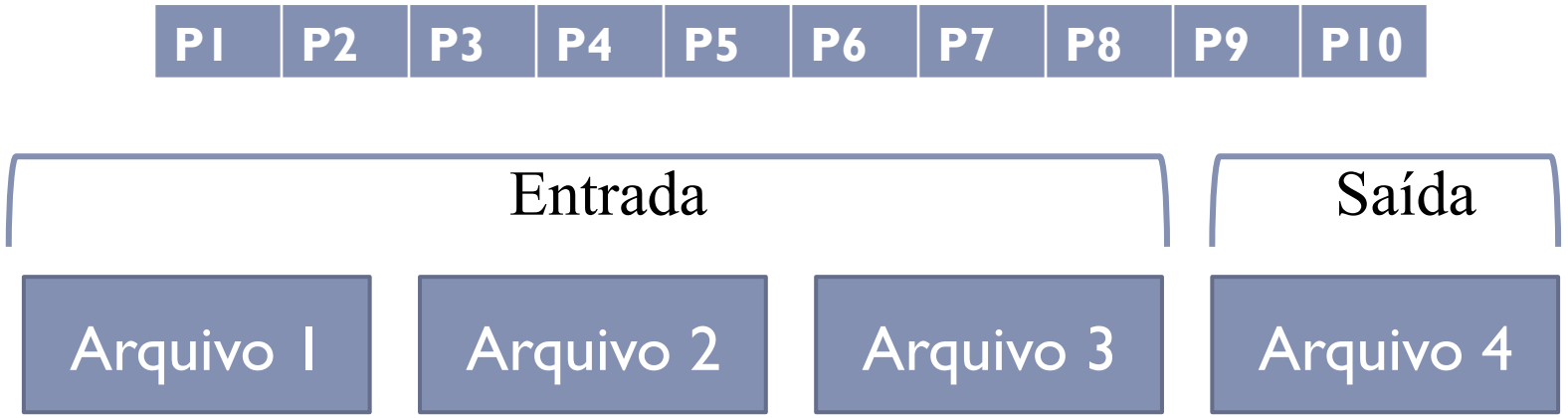
Notar que o conceito de **Fase** desse algoritmo é diferente do usado no algoritmo anterior. Usando esse conceito o algoritmo anterior teria 21 fases.

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

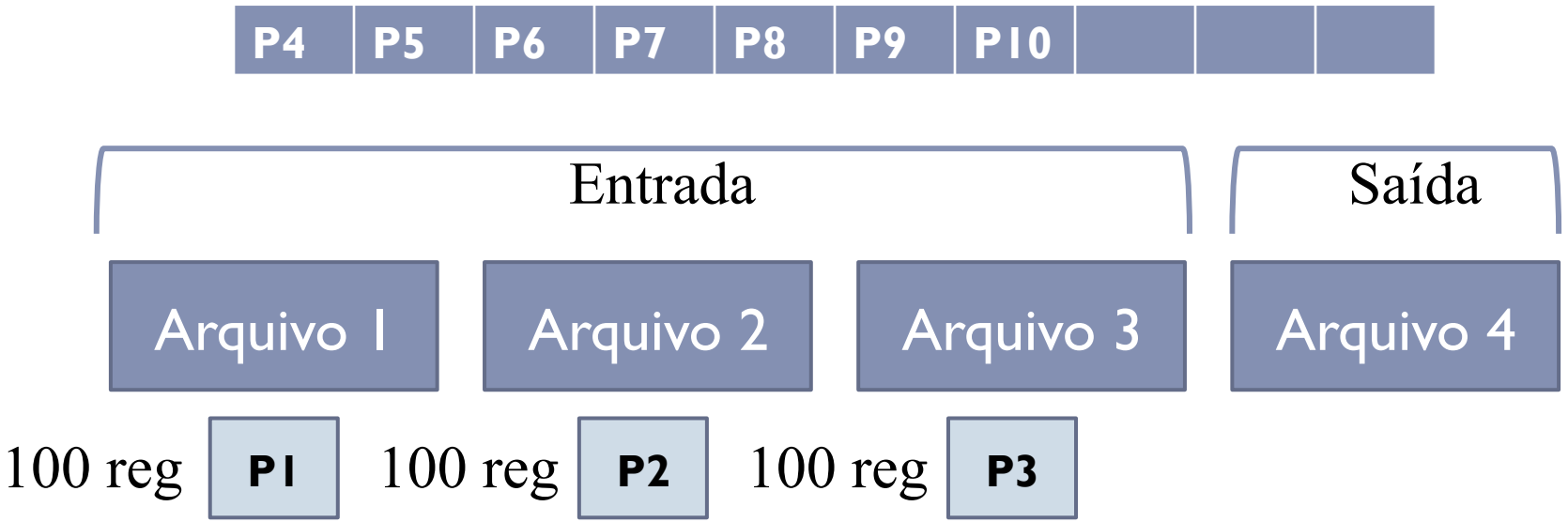
# Possível forma de Implementação

- Usar uma lista que contém os nomes dos arquivos a ordenar
- A cada passo do algoritmo, retirar os 3 primeiros itens da lista, intercalá-los, colocar o arquivo resultante no final da lista
- O algoritmo pára quando a lista tiver apenas 1 arquivo (que será o resultante)

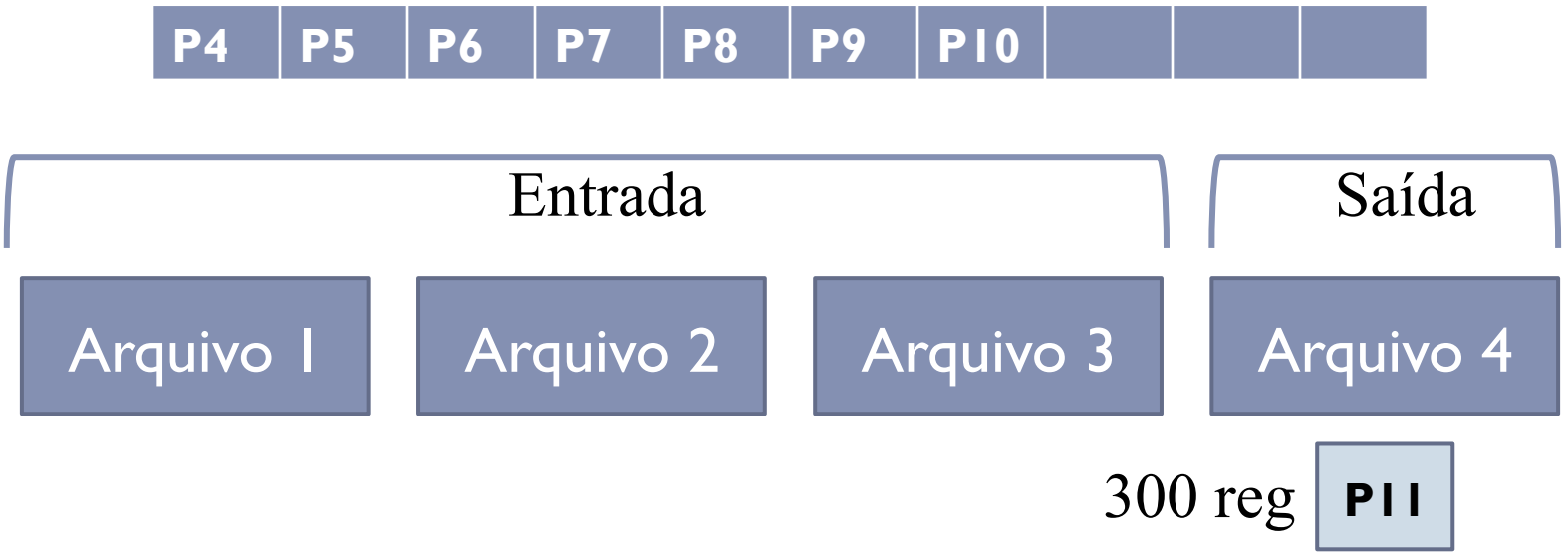
# Exemplo (F = 4 e 10 partições):



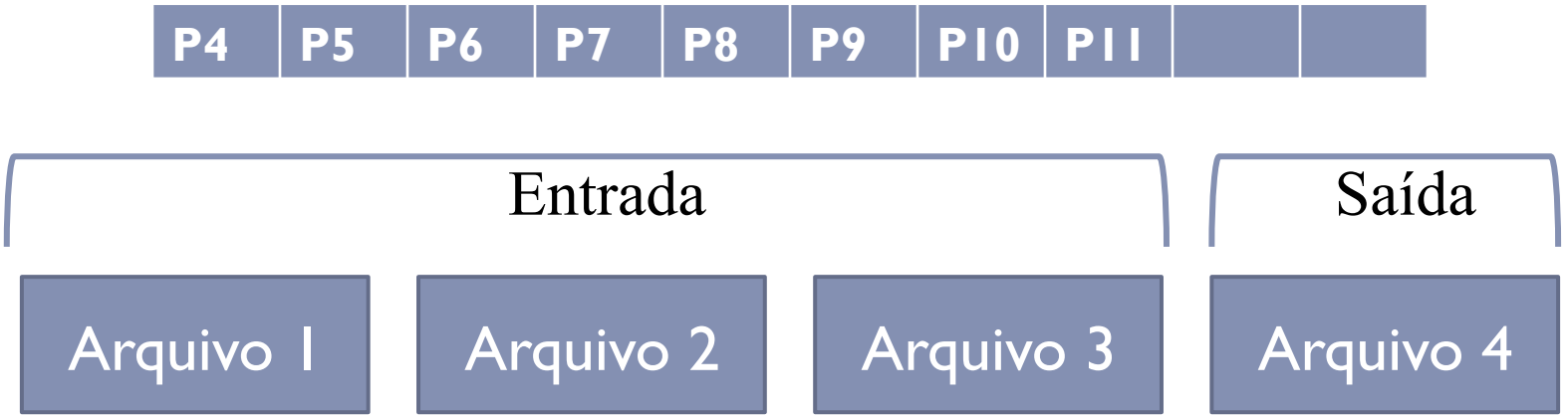
# Exemplo (F = 4 e 10 partições):



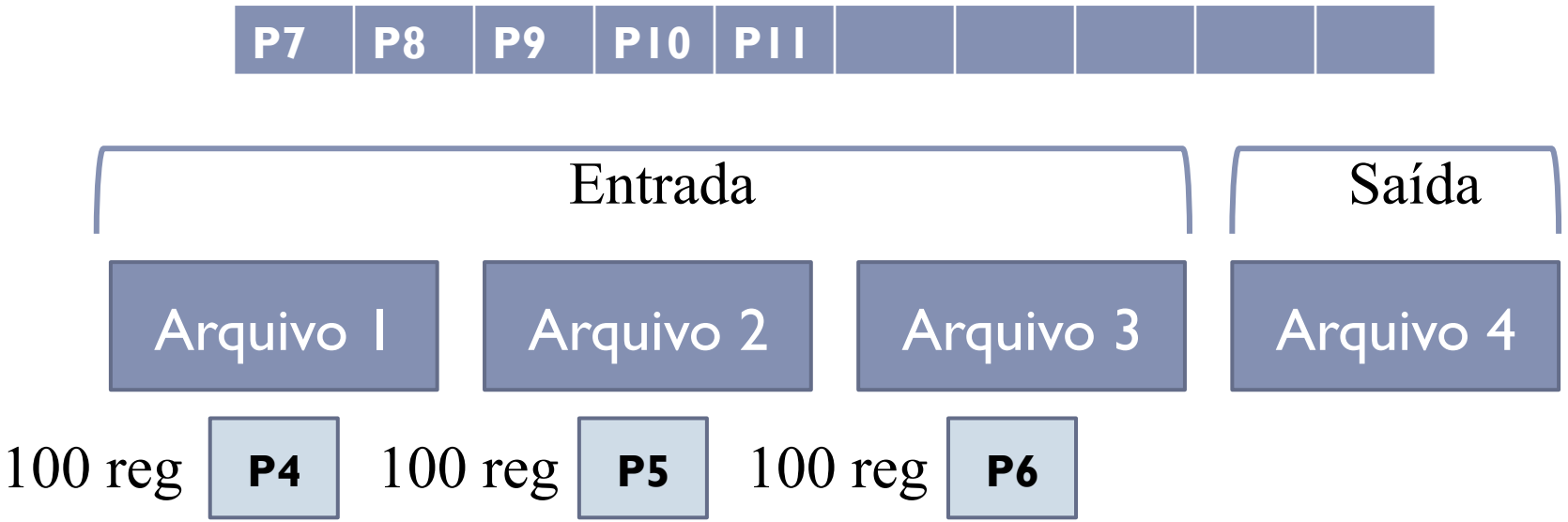
# Exemplo (F = 4 e 10 partições):



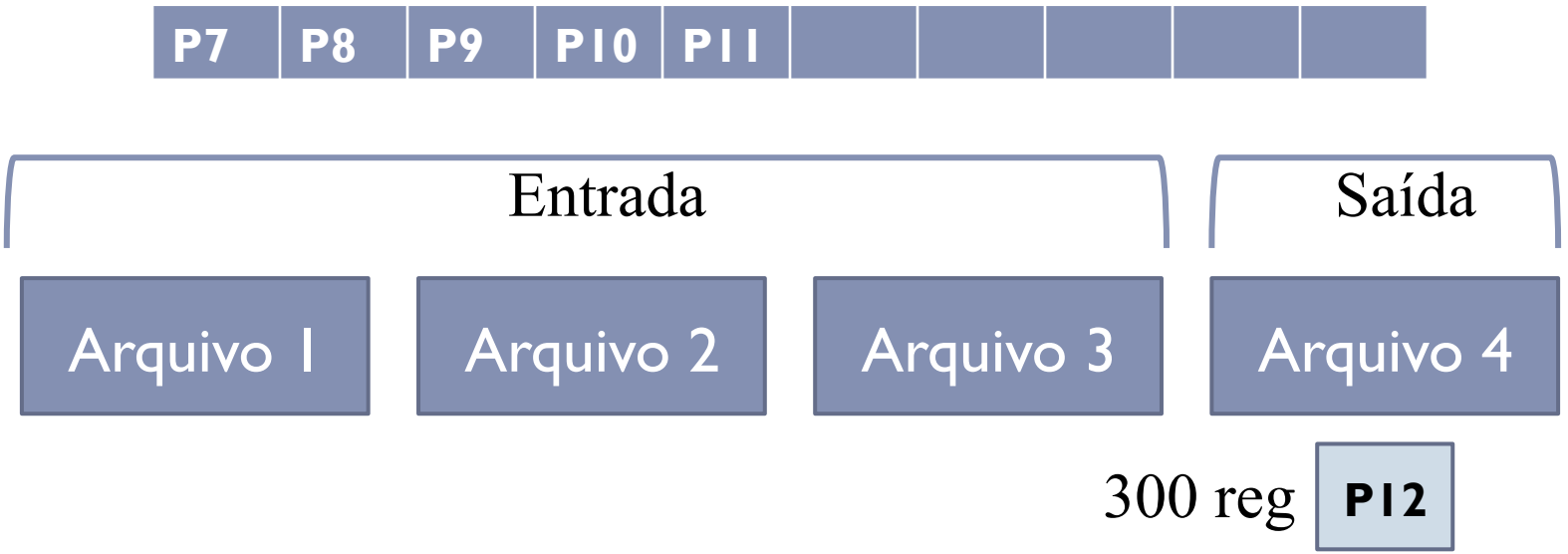
# Exemplo (F = 4 e 10 partições):



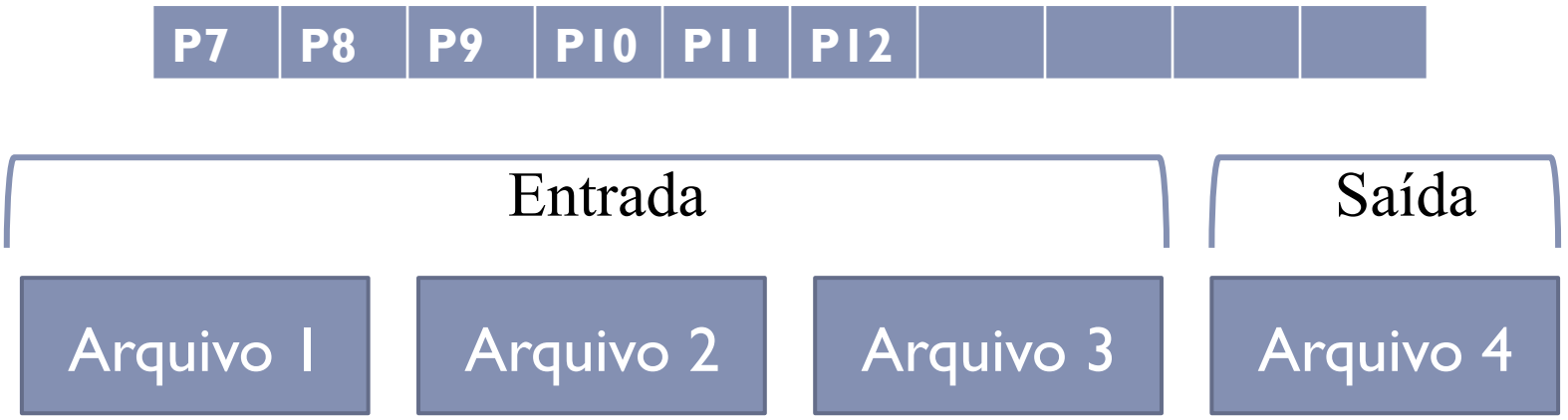
# Exemplo (F = 4 e 10 partições):



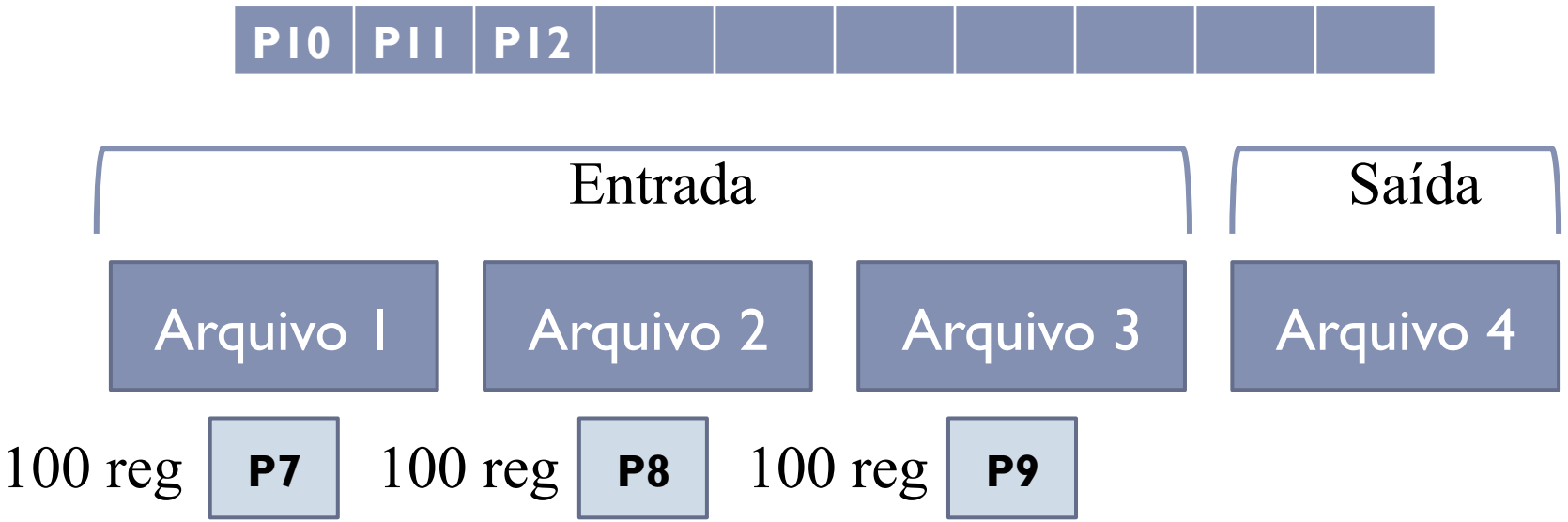
# Exemplo (F = 4 e 10 partições):



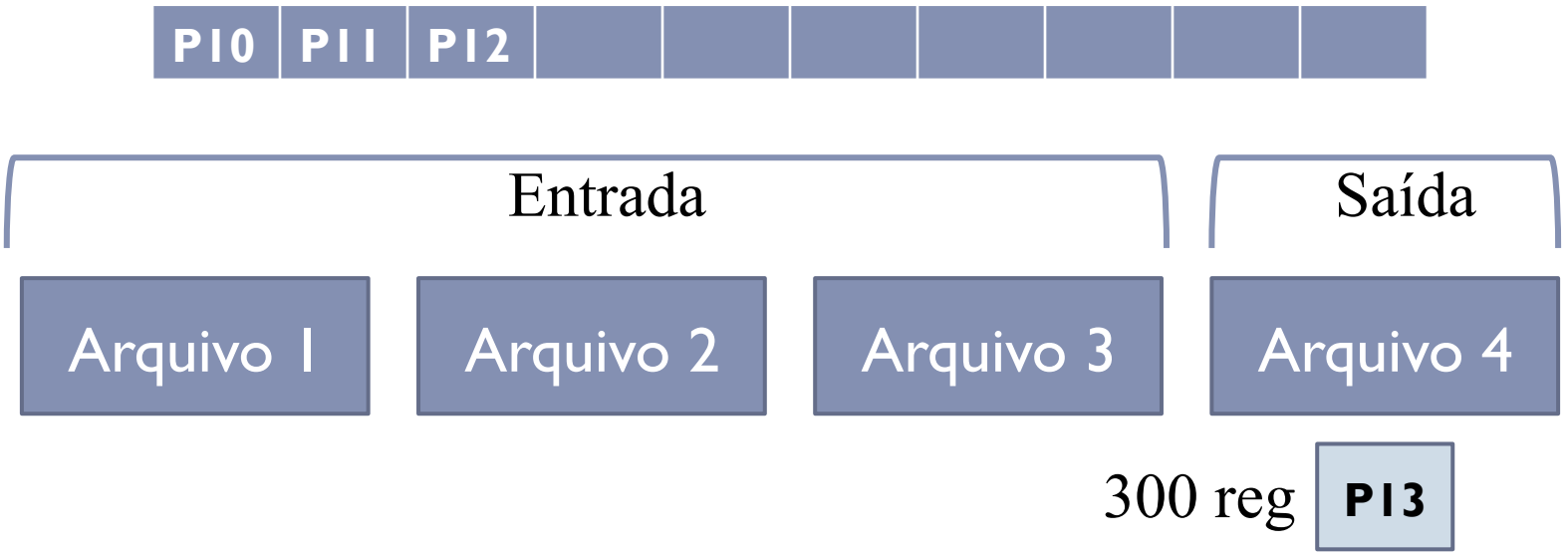
# Exemplo (F = 4 e 10 partições):



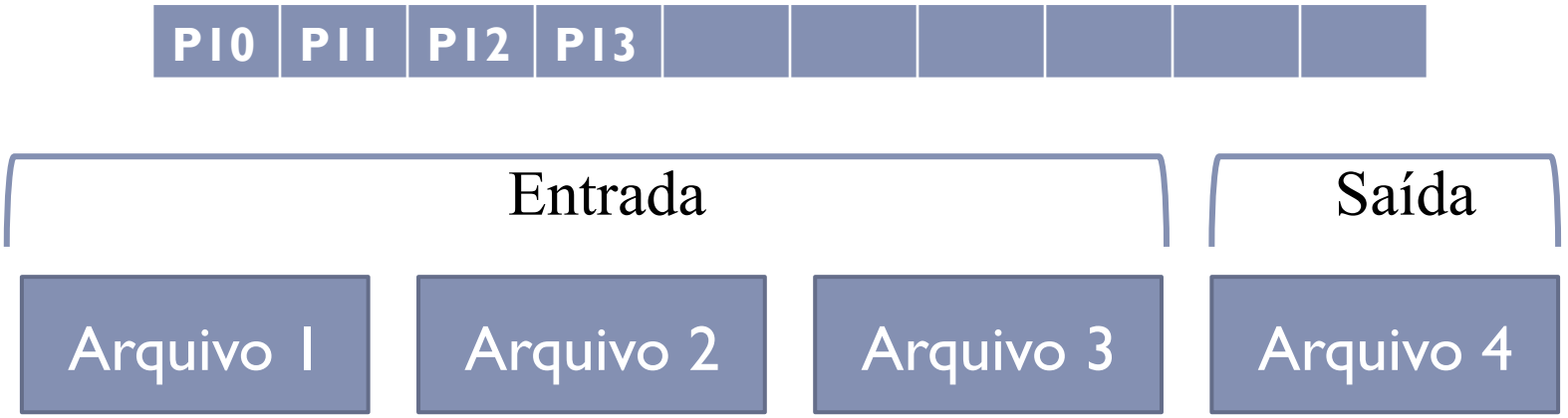
# Exemplo (F = 4 e 10 partições):



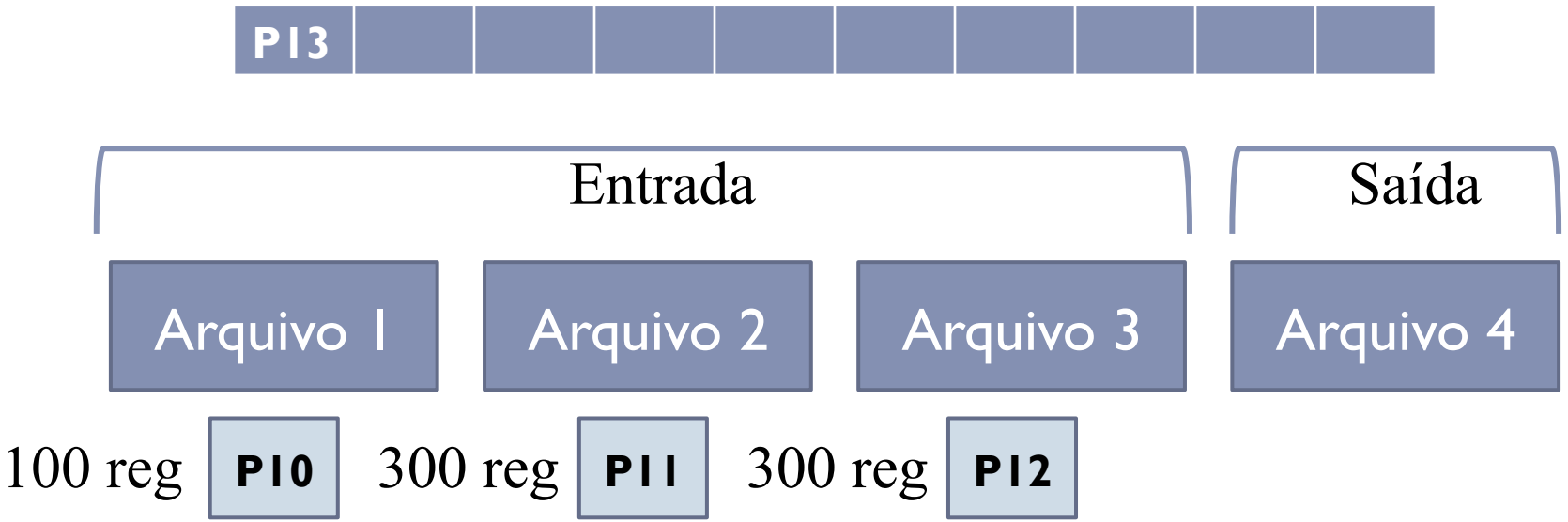
# Exemplo (F = 4 e 10 partições):



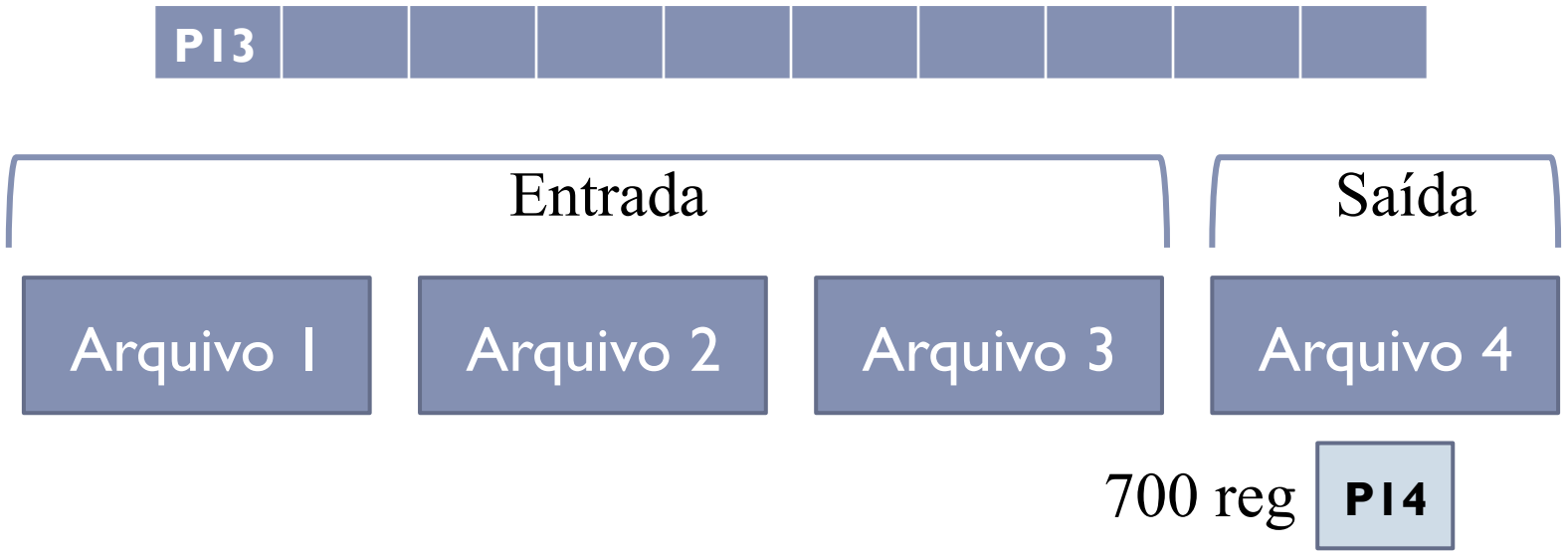
# Exemplo (F = 4 e 10 partições):



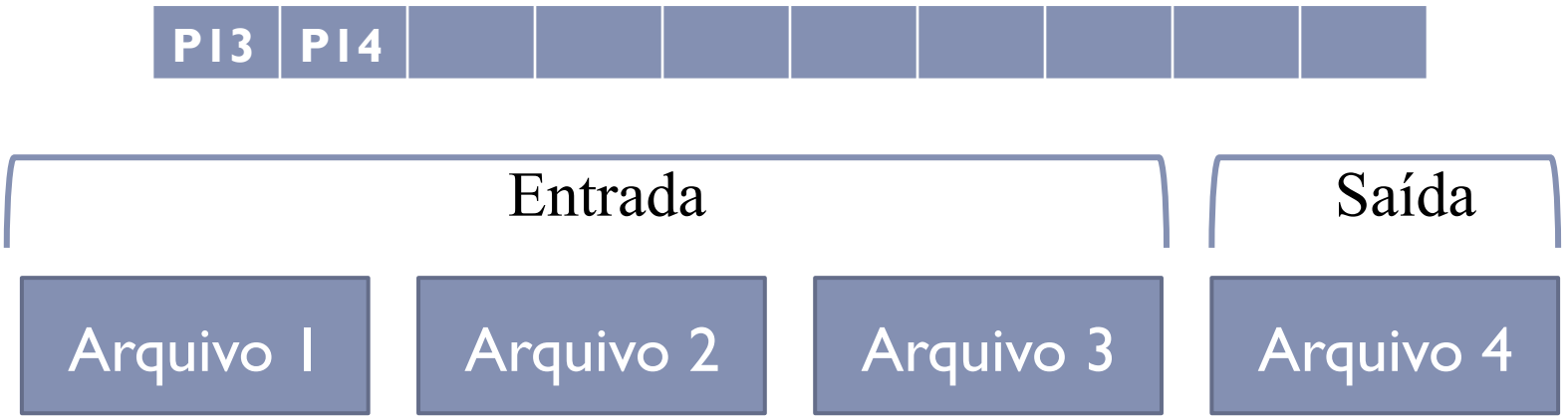
# Exemplo (F = 4 e 10 partições):



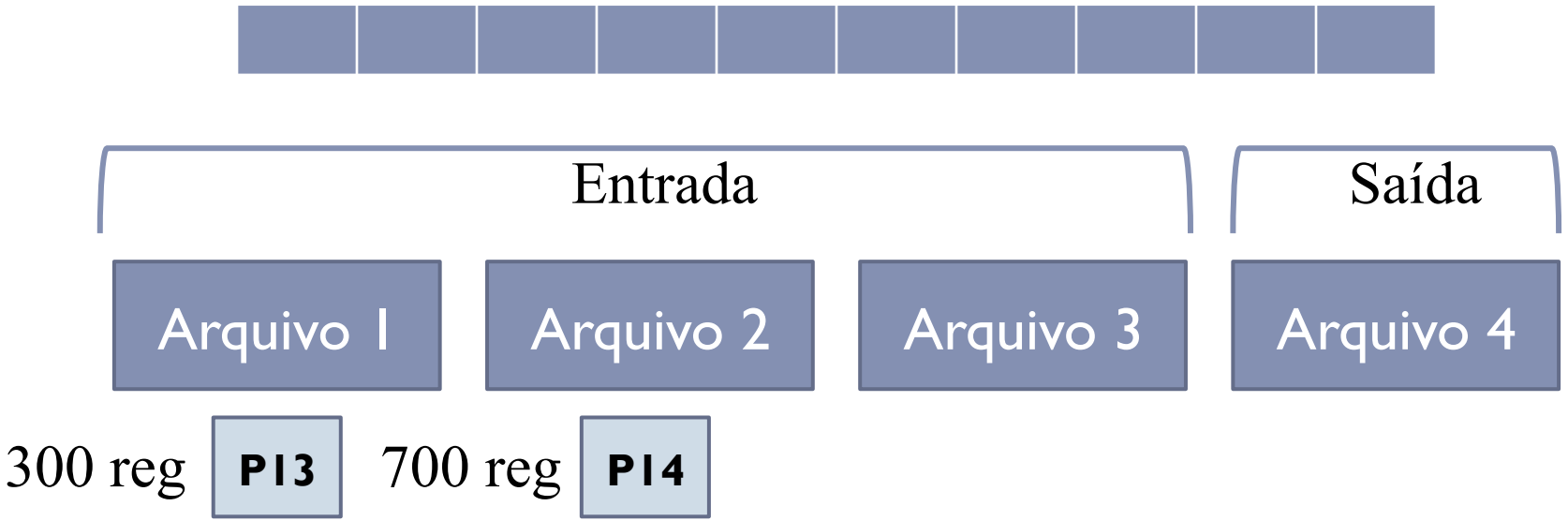
# Exemplo (F = 4 e 10 partições):



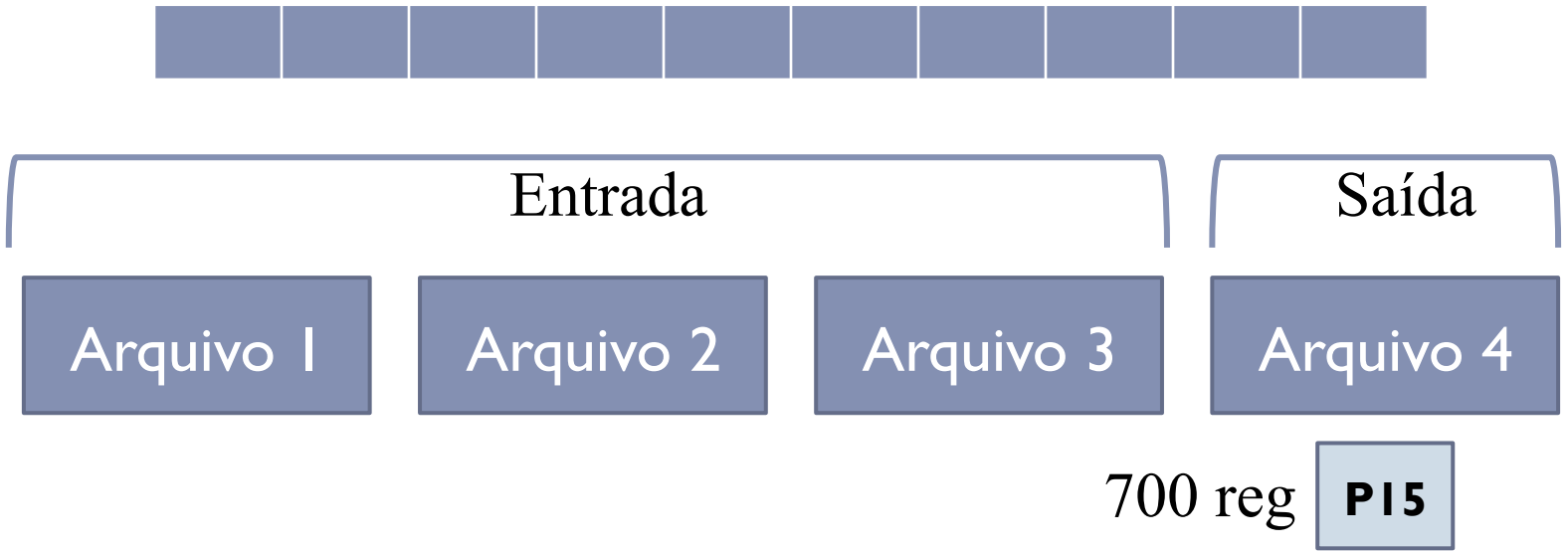
# Exemplo (F = 4 e 10 partições):



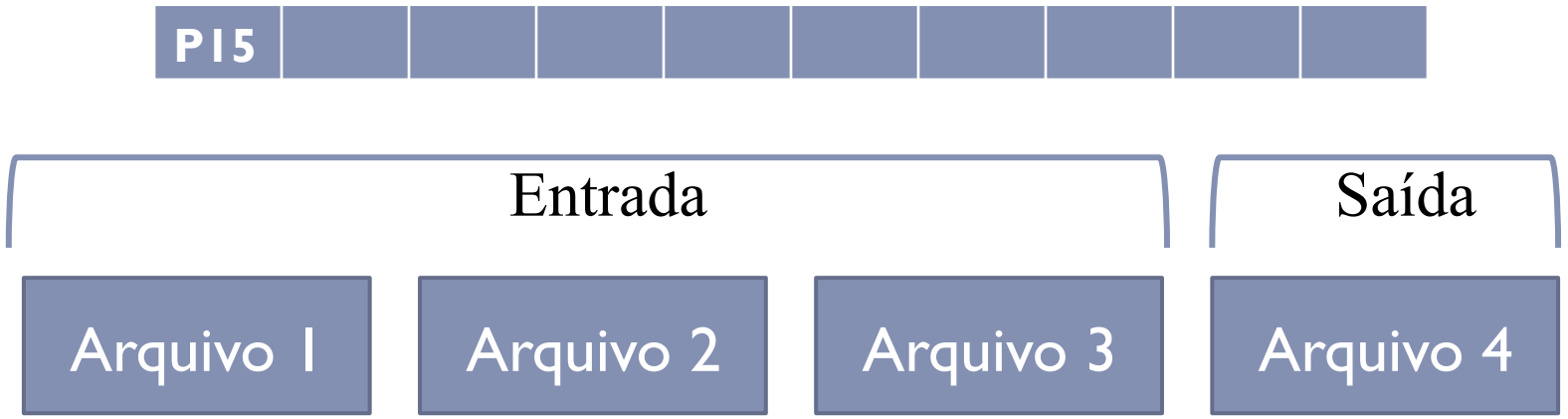
# Exemplo (F = 4 e 10 partições):



# 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**

# Implementação

- Problema: fazer intercalação de N partições ordenadas para gerar um único arquivo ordenado
- Restrição: Sistema Operacional pode lidar com apenas 4 arquivos ao mesmo tempo
- Entrada:
  - Lista com nomes dos arquivos a intercalar
- Saída:
  - Arquivo resultante ordenado, chamado "saida.dat"

# Exercício

- Implementar o algoritmo de intercalação utilizando árvore binária de vencedores
  - Entrada:
    - Lista com os nomes dos arquivos de entrada
    - Nome do arquivo de saída
  - Estrutura dos arquivos: Clientes (CodCliente, Nome, DataNascimento)