SIN5013 - Análise de Algoritmos e Estruturas de Dados

Estruturas de Dados / Listas / Árvores

Prof. Flávio Luiz Coutinho

Estruturas de Dados

Responsabilidades:

- armazenar informação
- organizar informação
- definir e implementar as operações de:
    - inserção
    - busca
    - percurso
    - remoção

Estruturas de Dados

Cada estrutura diferente implementa uma organização diferente da informação armazenada, visando vantagens específicas em algumas operações.

Estruturas de Dados

Exemplo: usar um vetor para representar uma coleção de valores inteiros positivos, implementar duas operações básicas sobre tal representação: inserção de um novo valor, e busca por um valor específico.

Estruturas de Dados

Abordagem 1 - organização/armazenamento da informação:

```
int v[100];

int livre = 0;
```

Estruturas de Dados

Abordagem 1 - inserção do valor x:

```
if(livre < 100) {
    v[livre] = x;
    livre++;
}
```

Estruturas de Dados

Abordagem 1 - busca pelo valor x:

```
int i;

for(i = 0; i < livre; i++) {

    if(v[i] == x) return TRUE;

}

return FALSE;
```

Estruturas de Dados

Abordagem 1 funciona? Sim…

Estruturas de Dados

Abordagem 1 funciona? Sim…

É a única forma de se representar uma coleção de inteiros? Não…

Estruturas de Dados

Abordagem 1 funciona? Sim…

É a única forma de se representar uma coleção de inteiros? Não…

Vamos dar uma olhada em outra alternativa:

Estruturas de Dados

Abordagem 2 - organização/armazenamento da informação:

```
int v[VALOR_MAXIMO + 1];

// inicializa o vetor, tal que v[i] = 0 para todo i.
```

Estruturas de Dados

Abordagem 2 - adição do valor x:

```
v[x]++;
```

Estruturas de Dados

Abordagem 2 - busca pelo valor x:

```
return v[x] > 0;
```

Estruturas de Dados

Abordagem 2 também funciona? Sim…

Estruturas de Dados

Abordagem 2 também funciona? Sim…

Qual das duas abordagens é melhor? Depende…

Estruturas de Dados

Abordagem 2 também funciona? Sim…

Qual das duas abordagens é melhor? Depende…

- Abordagem 1: Θ(n)
- Abordagem 2: **Θ(1)**

(complexidade de tempo da busca)

Estruturas de Dados

Abordagem 2 também funciona? Sim…

Qual das duas abordagens é melhor? Depende…

- Abordagem 1: Θ(n)
- Abordagem 2: **Θ(k)**, onde k é a quantidade de valores possíveis

(consumo de memória da estrutura)

Listas

- Conjunto de elementos armazenados é organizado em uma estrutura linear.

- Cada elemento possui um sucessor e um antecessor (com exceção das extremidades).

- Existe uma associação entre um elemento e sua posição (índice) dentro da lista.

Listas

- Fisicamente, os elementos podem se encontrar armazenados de forma sequencial, ou não. Ou seja, a ordem física de armazenamento dos elementos (disposição dos elementos em memória) não necessariamente corresponde à ordem lógica da estrutura.

- Duas implementações “essenciais”:

    - listas sequenciais (**array** como espaço de armazenamento)
    - listas ligadas (**nó**: informação e endereço do próximo nó)

Listas

Operações:

- inserir(lista, elemento, índice)
- buscar(lista, elemento)
- remove(lista, índice) / remover(lista, elemento)
- percorrer(lista)

- obter_elemento(lista, índice)

Listas

Operações:

- inserir(lista, elemento, índice) — O(n)
- buscar(lista, elemento) — O(n)
- remove(lista, índice) / remover(lista, elemento) — O(n)
- percorrer(lista) — Θ(n)

- obter_elemento(lista, índice) — Θ(1): sequencial

  O(n): ligada

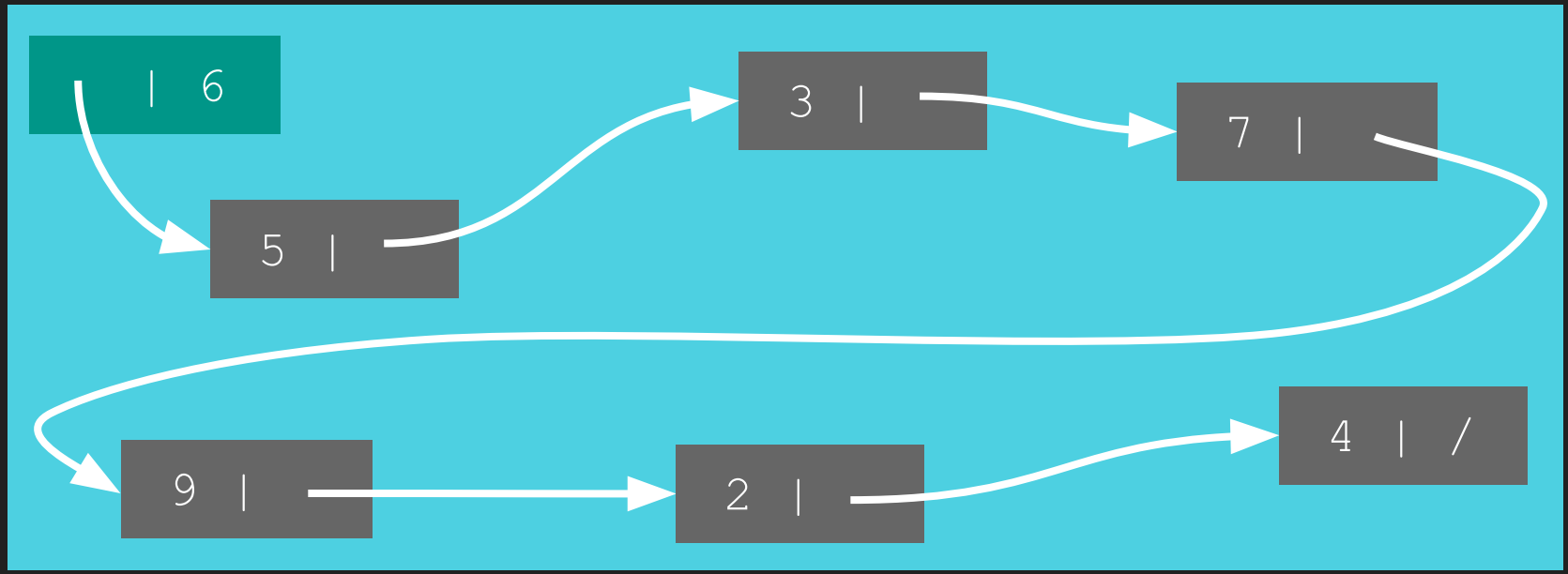
Lista sequencial

Lista: { 5, 3, 7, 9, 2, 4 }

Representação sequencial (ordem lógica == ordem física):

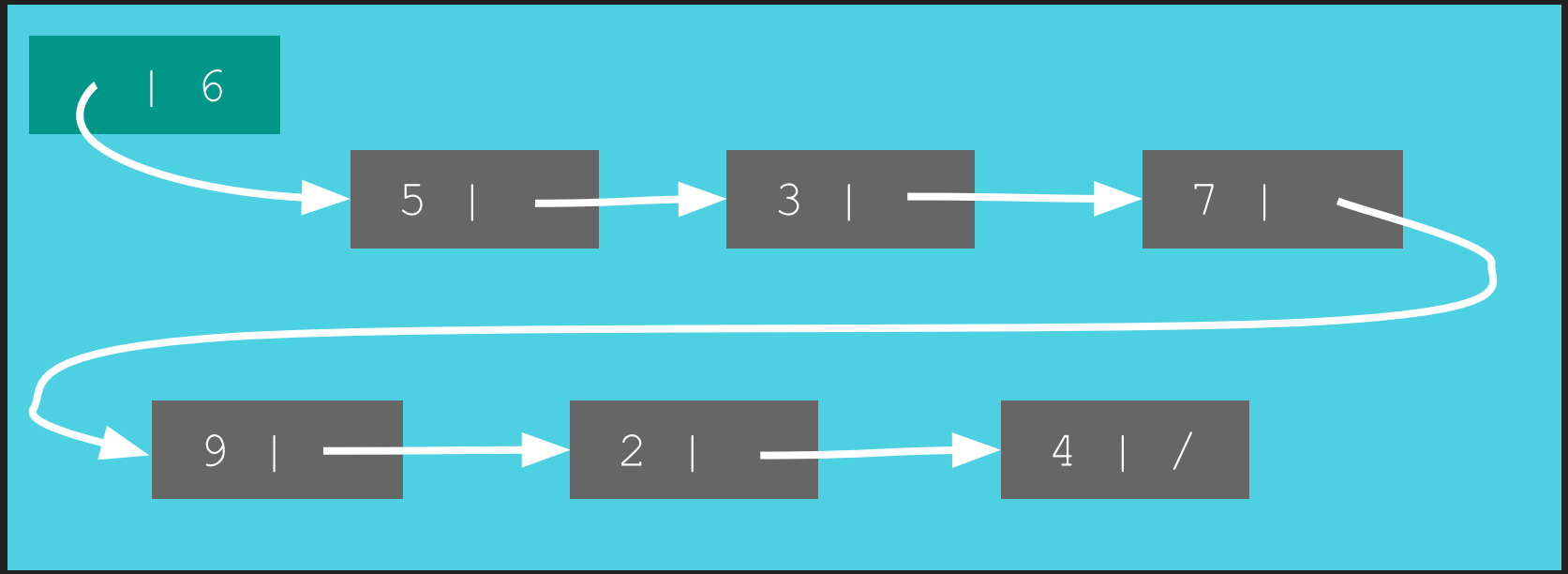
Lista ligada

Implementação de lista ligada.



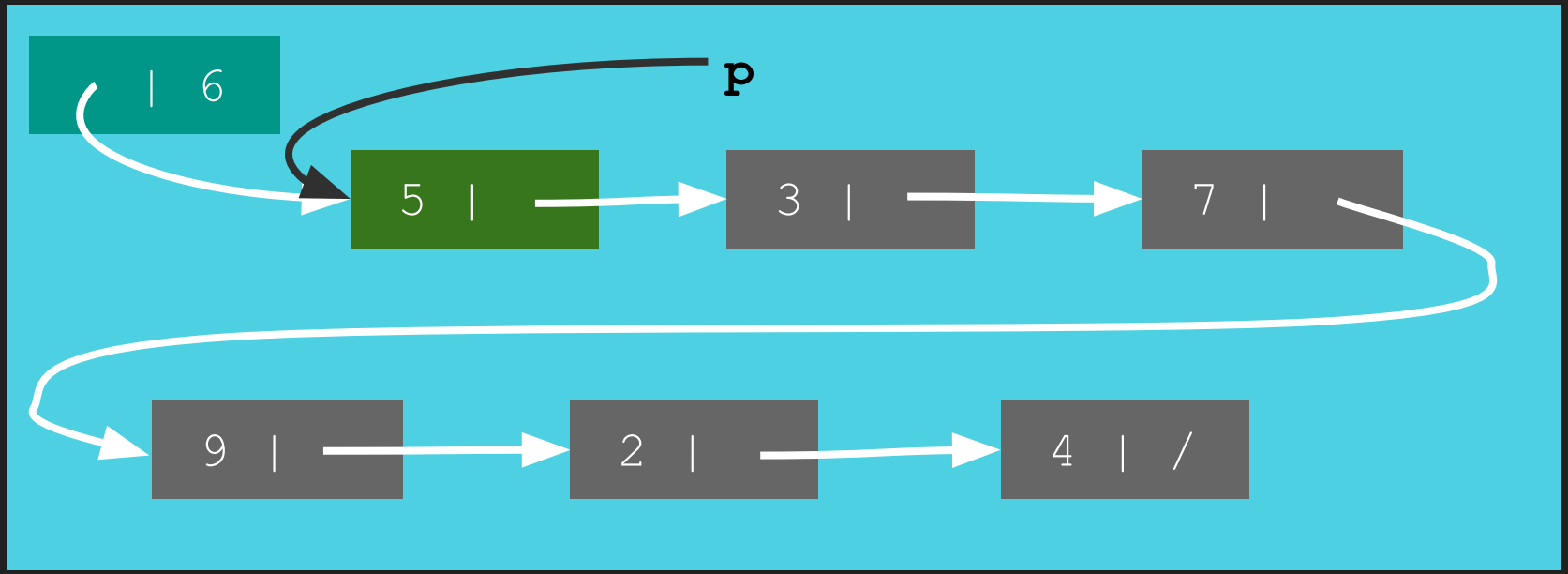
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



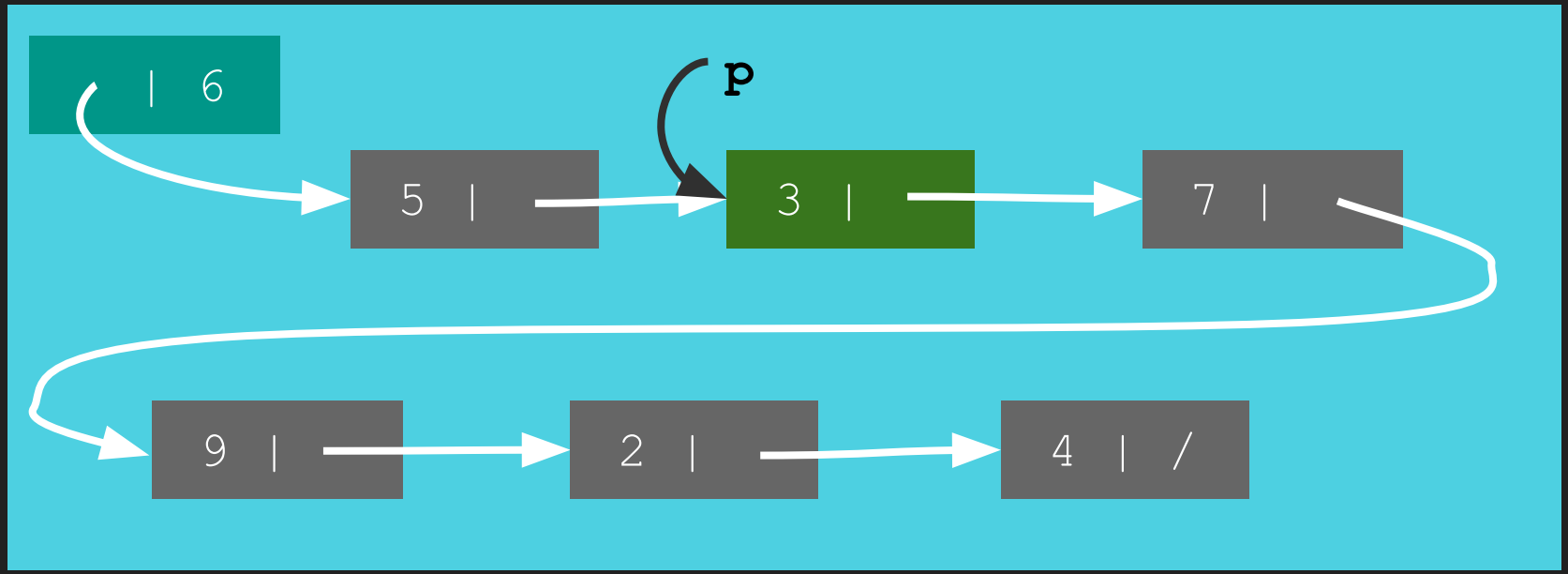
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



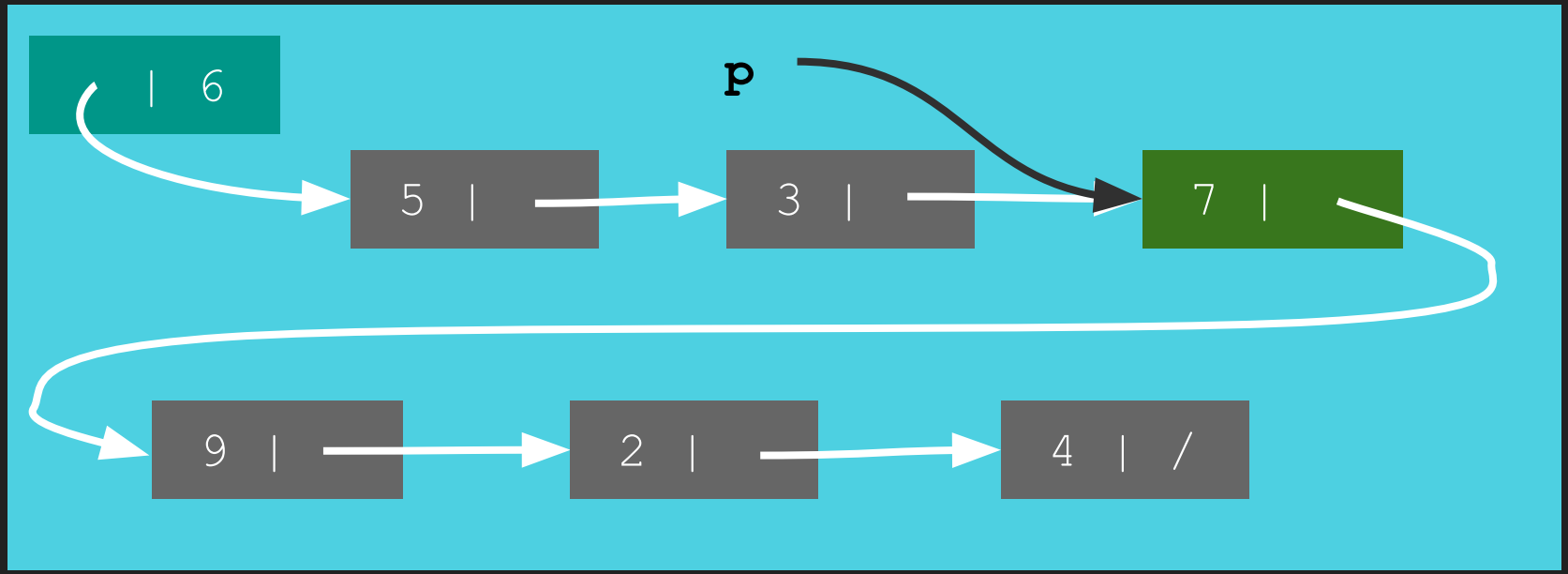
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



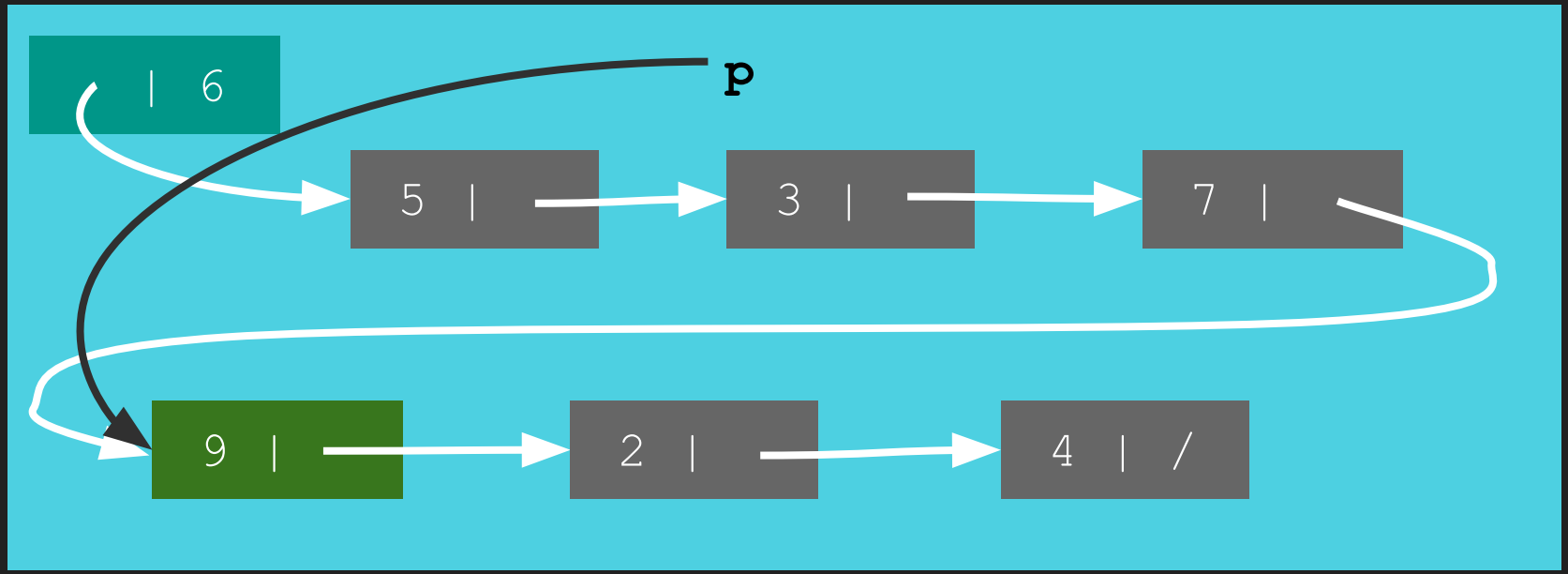
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



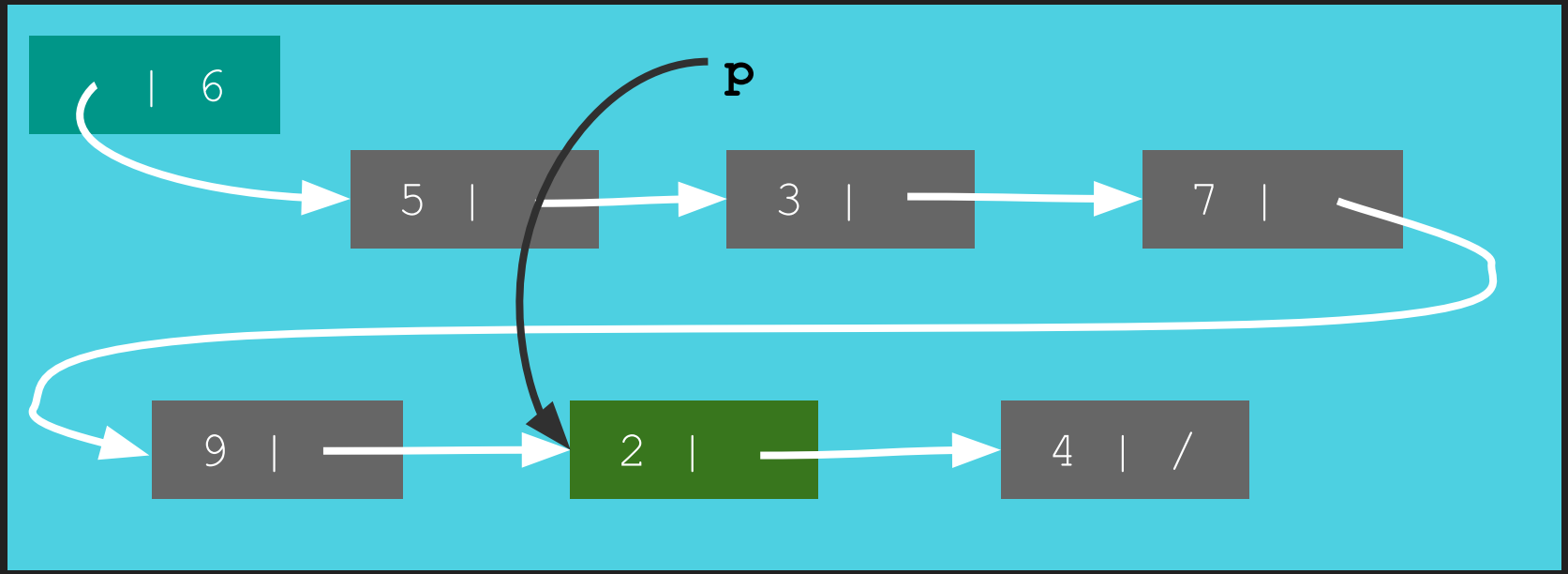
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



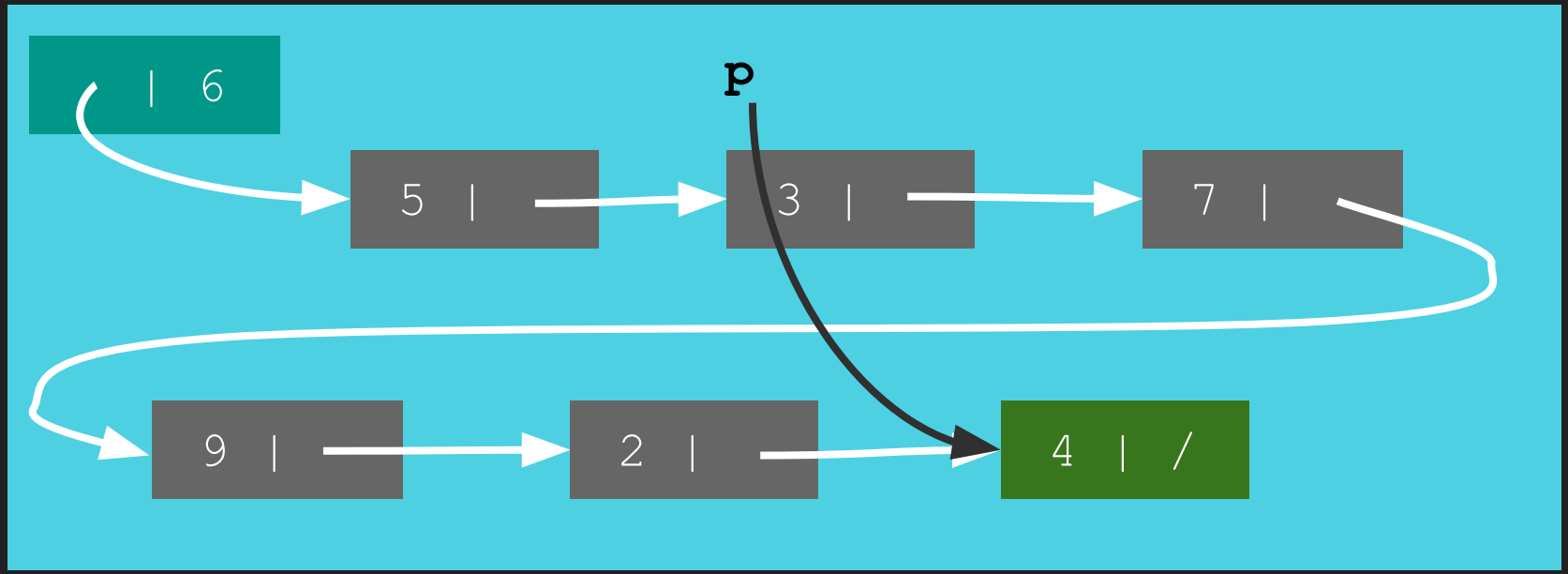
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



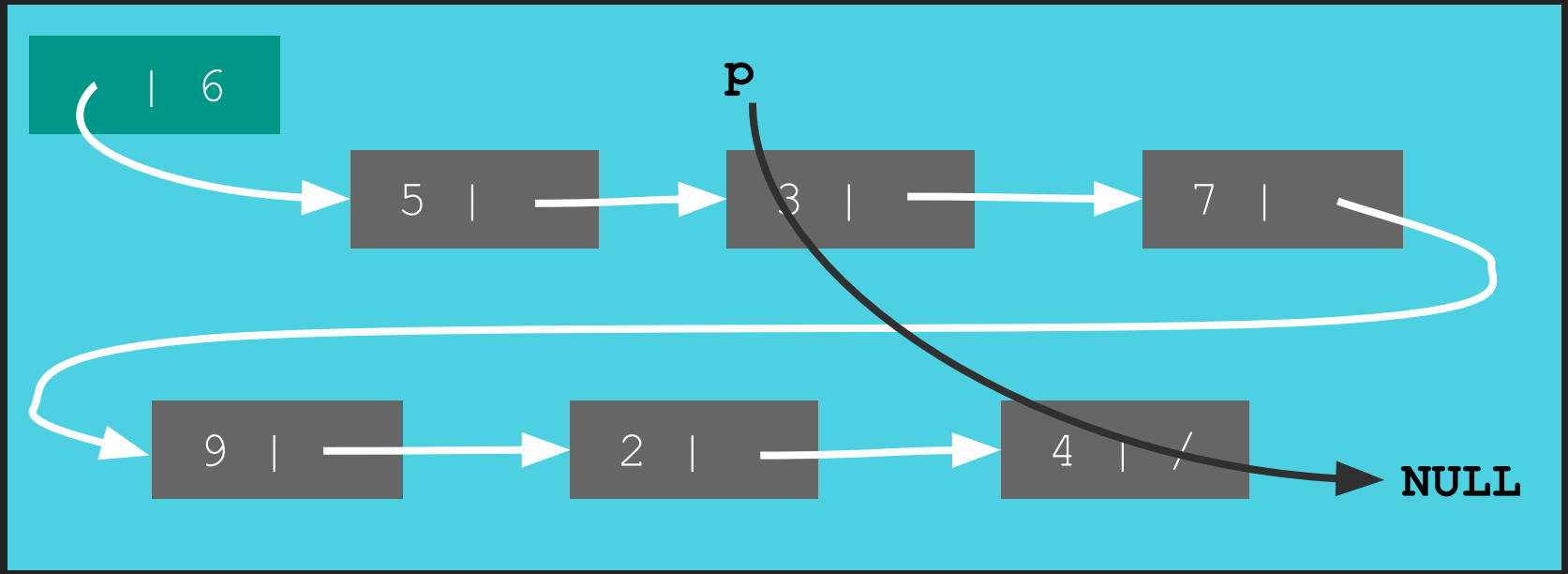
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



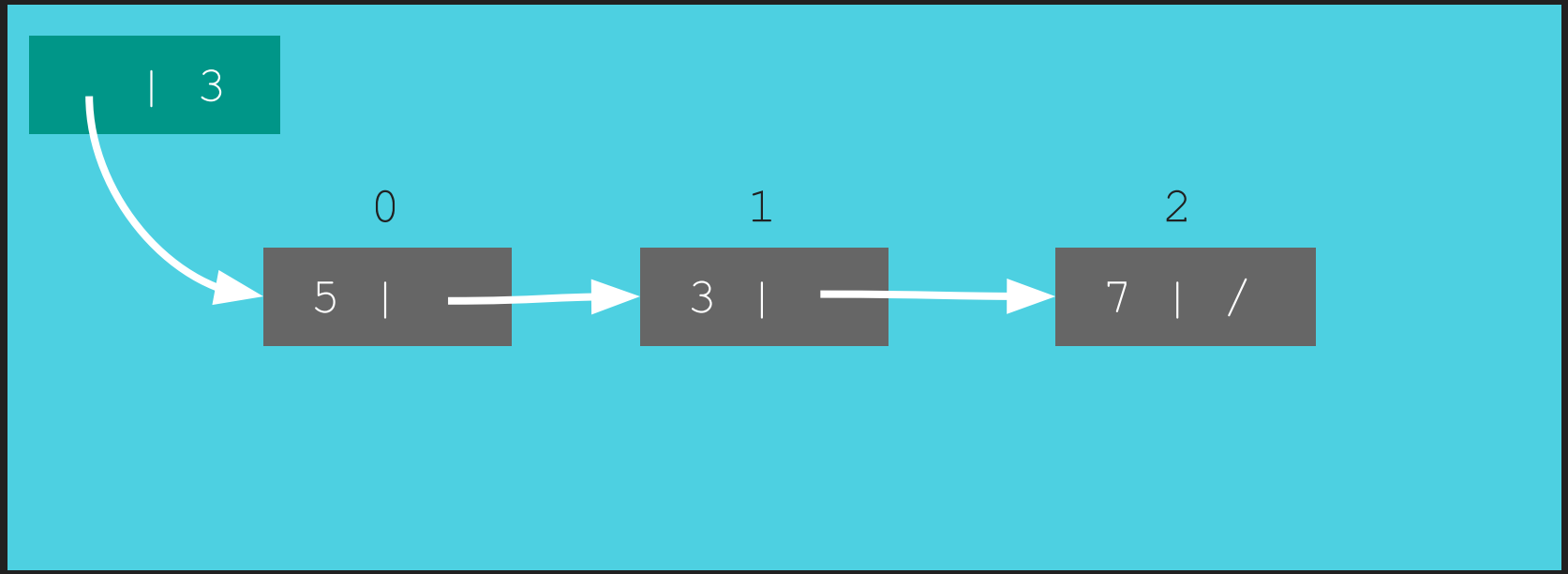
Listas ligadas

base da iteração (para impressão, busca sequencial, determinar tamanho, etc):



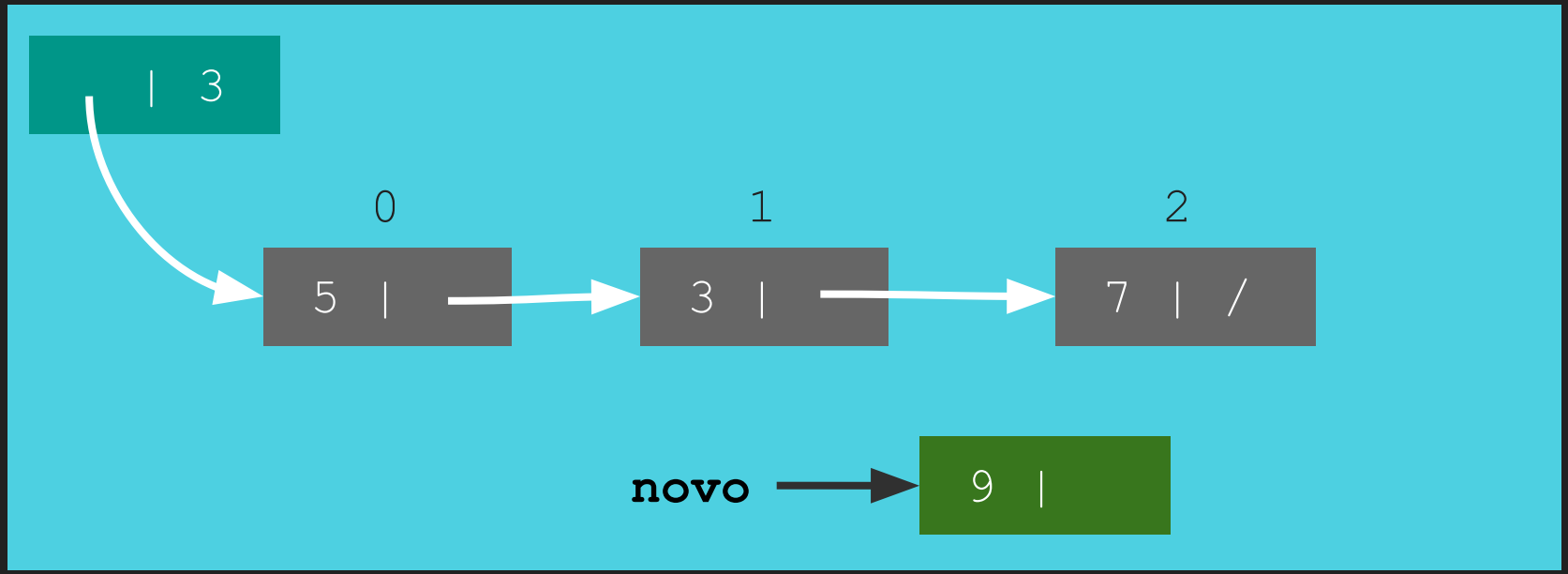
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



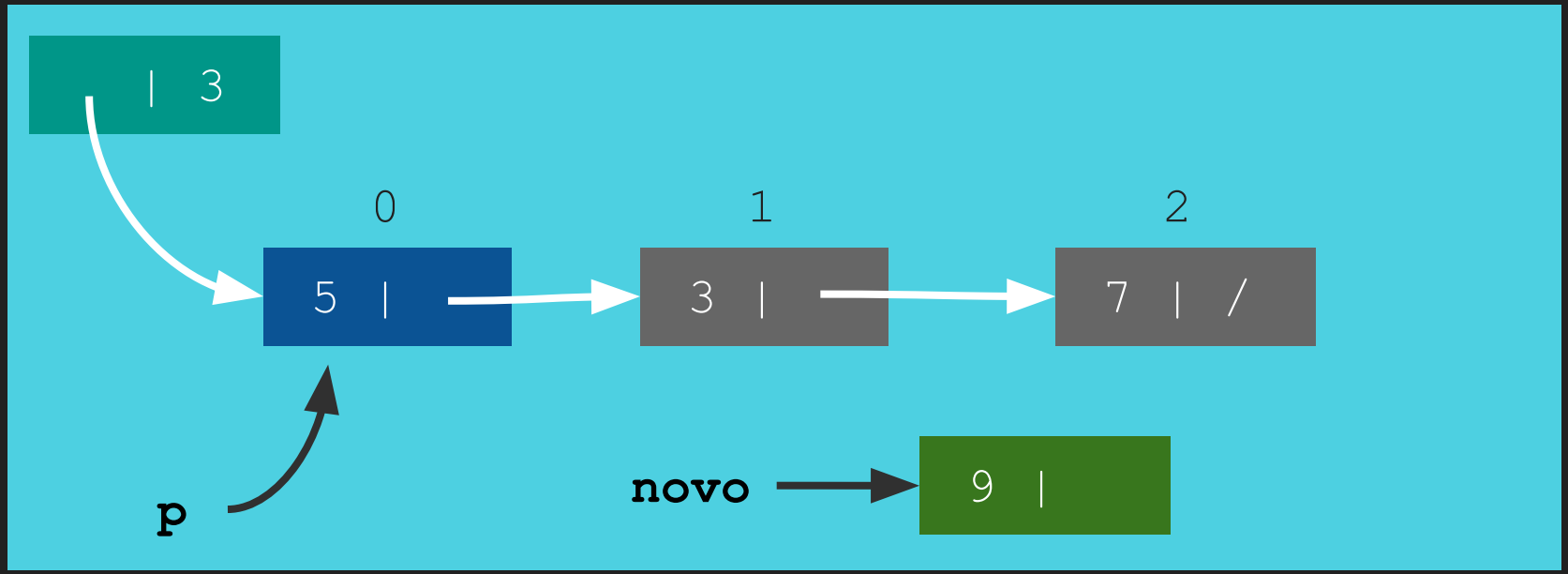
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



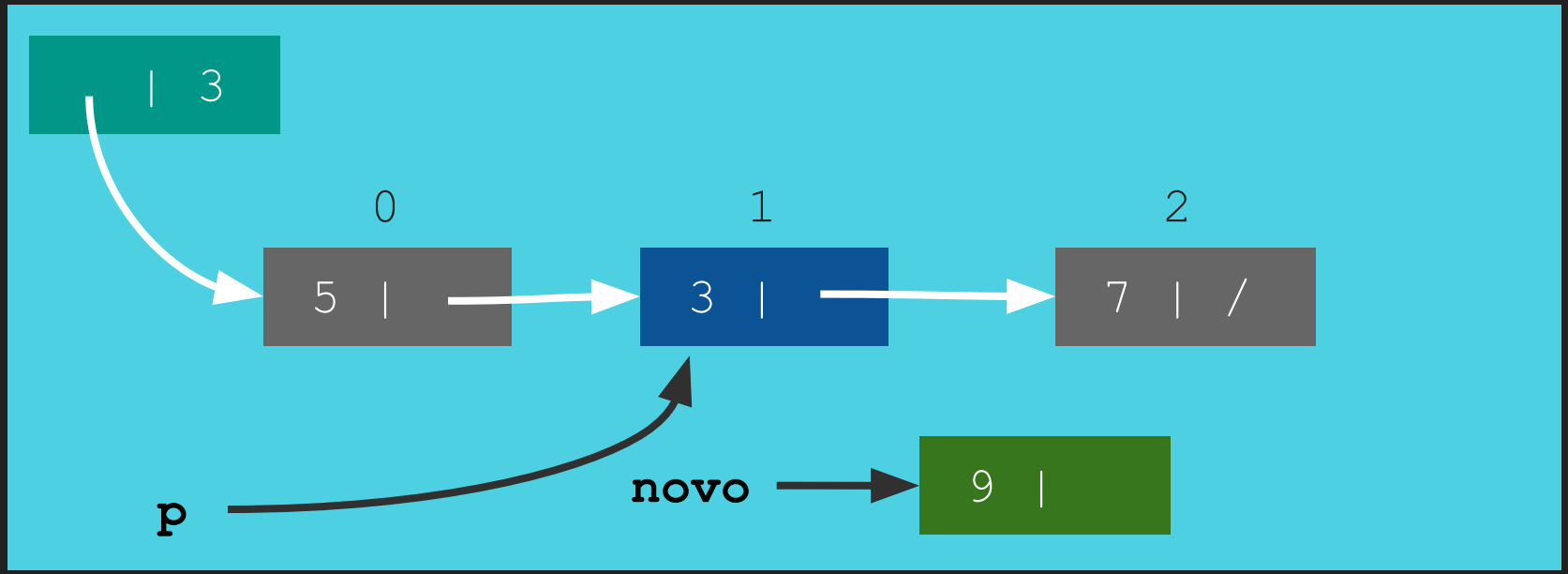
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



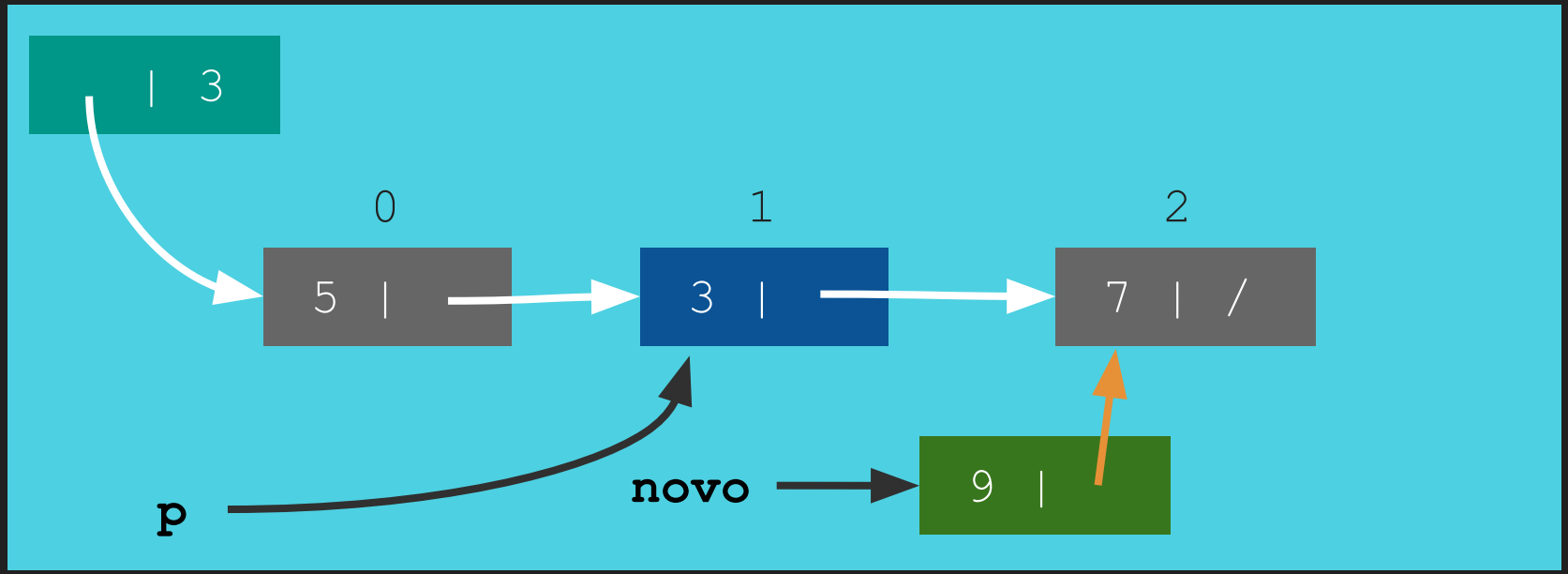
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



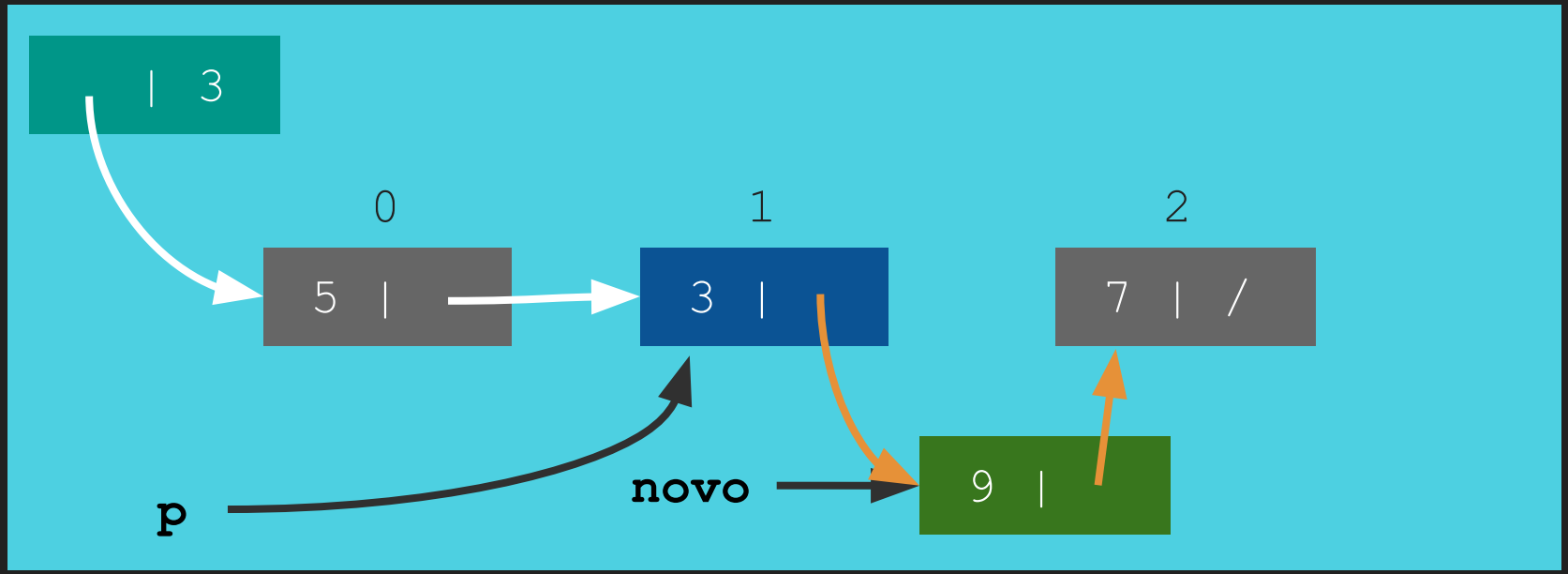
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



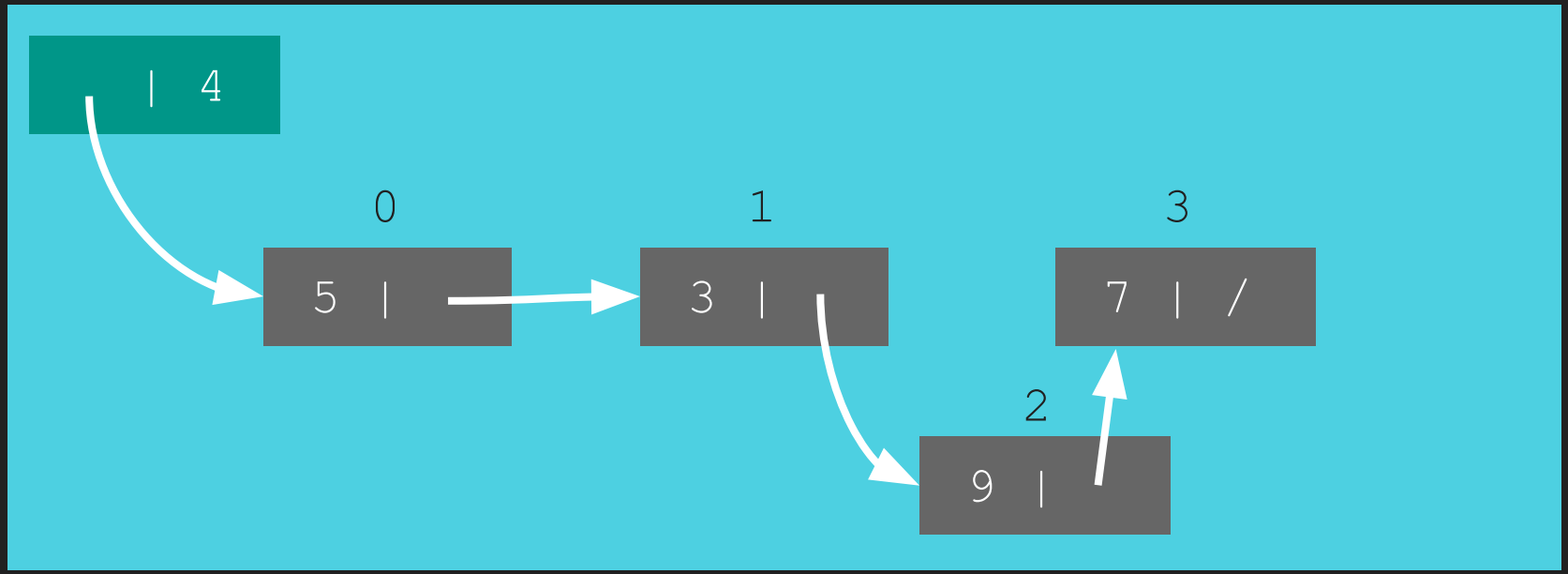
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



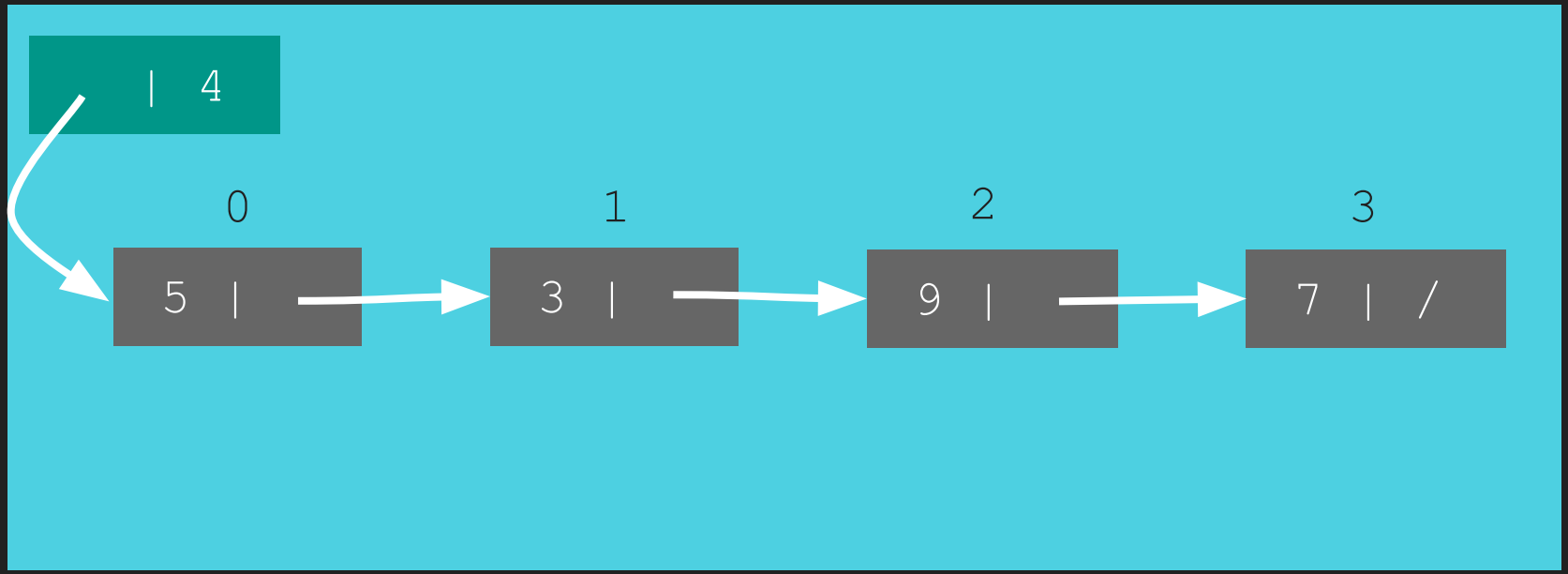
Listas ligadas

Inserção de um elemento no i-ésimo índice (valor 9, no índice 2):



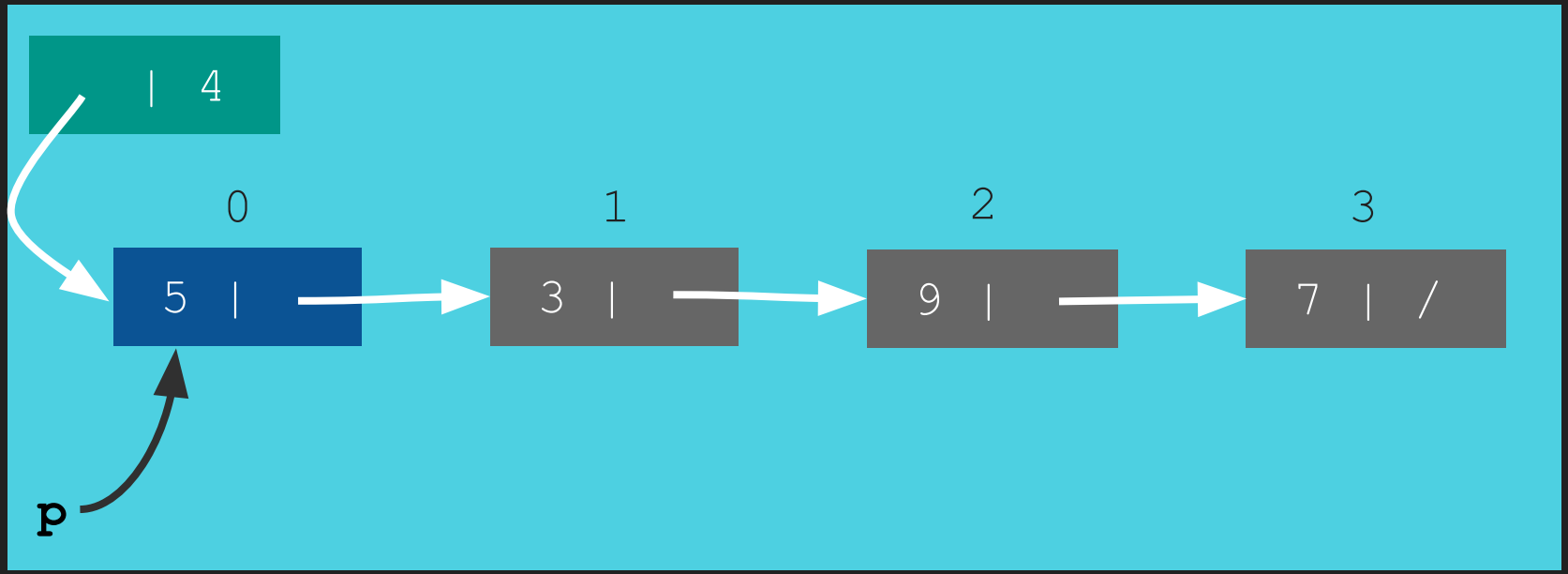
Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



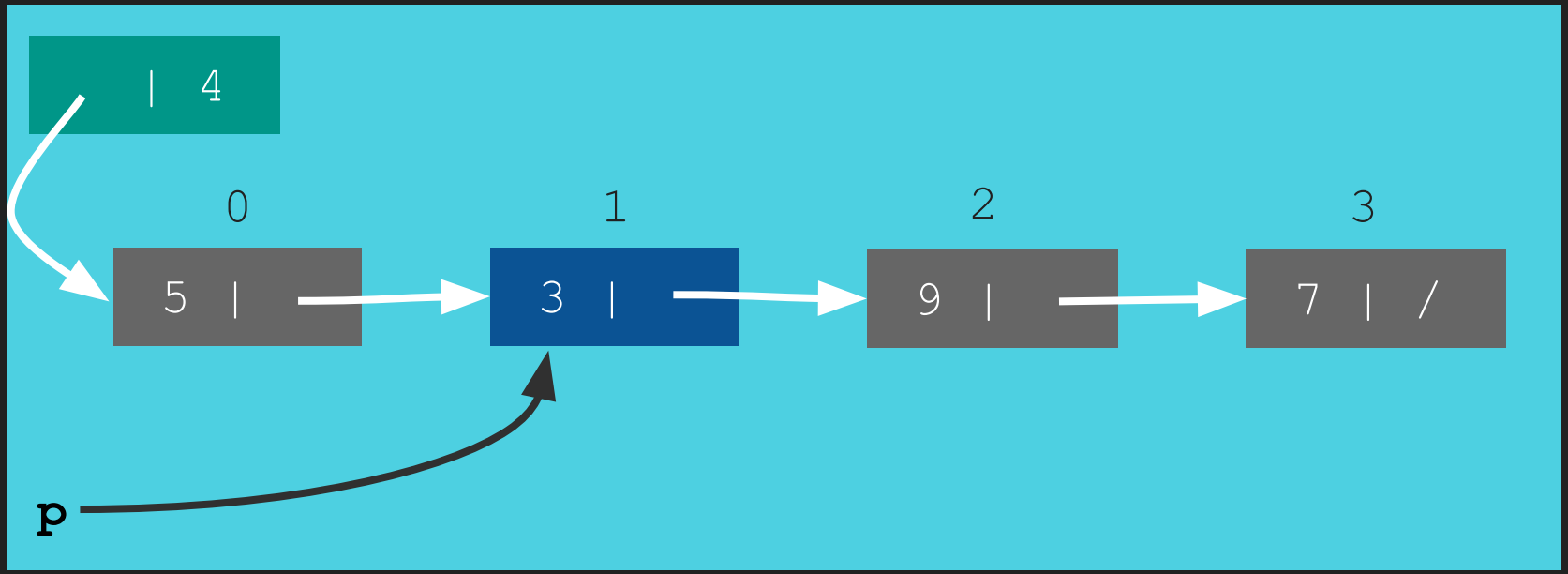
Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



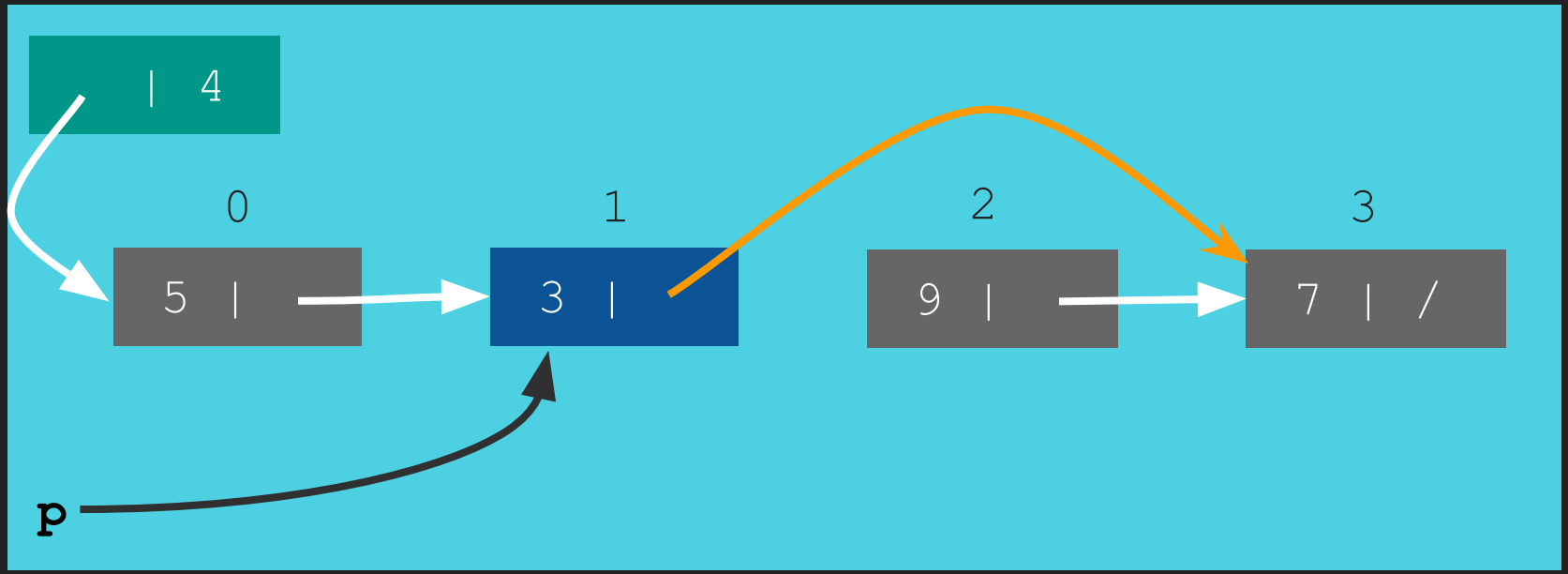
Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



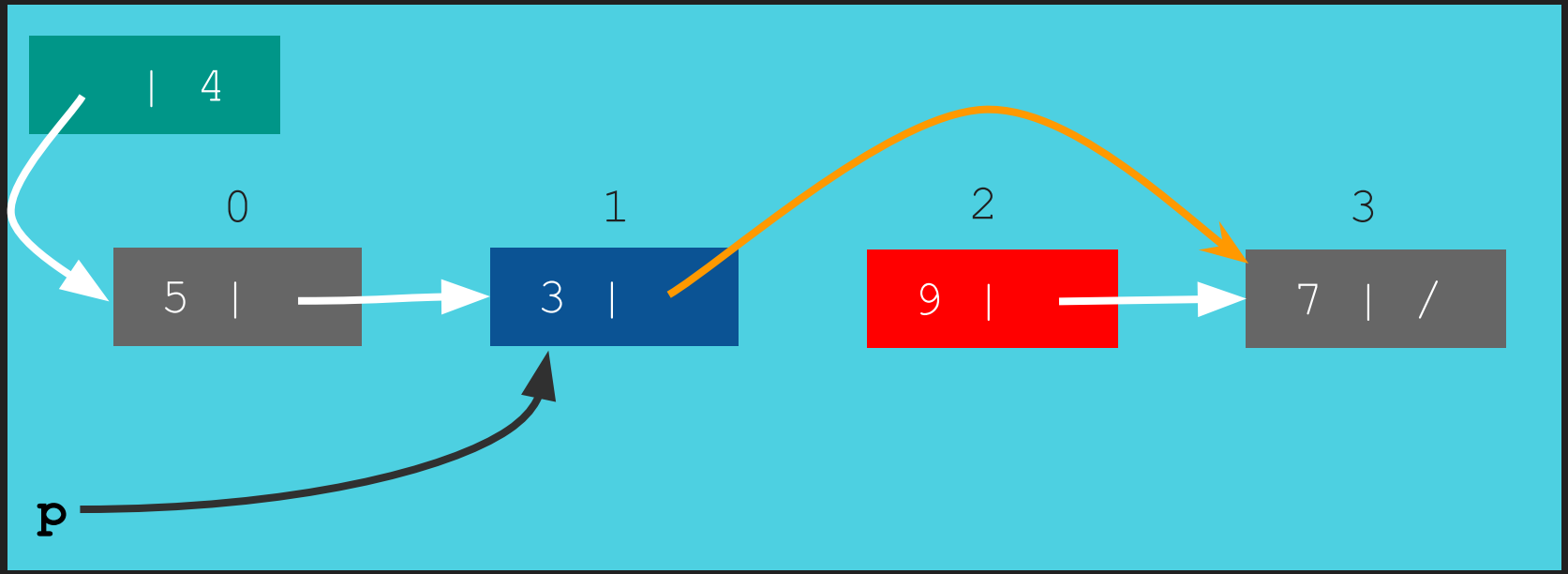
Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



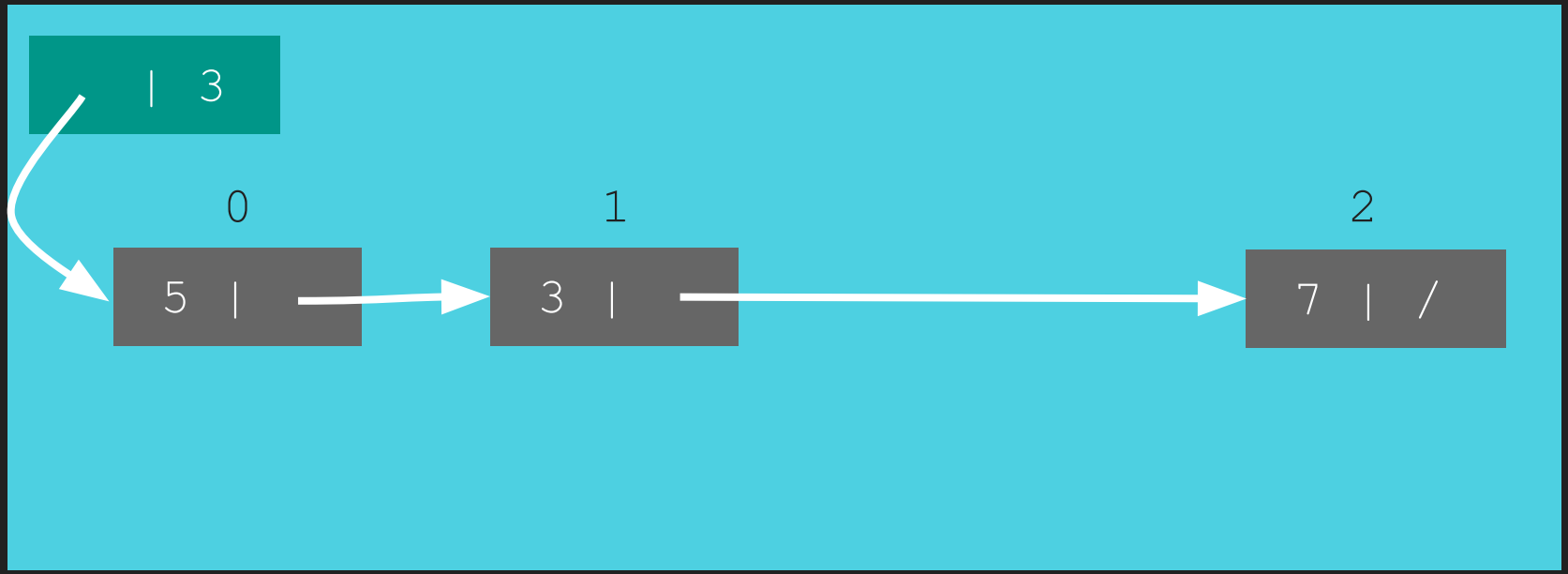
Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



Listas ligadas

Remoção de um elemento (valor 9, que se encontra no índice 2):



Listas

Considerações:

- Função obter_elemento: ineficiente em listas ligadas.
- Melhor uso da memória em listas ligadas.
- Listas ligadas não “deslocam” elementos na memória para realizar inserções e remoções (mas isso não reduz a complexidade assintótica das operações).
- A representação por listas ligadas pode ser melhor para implementar alguns conjuntos de operaçoes além das essenciais, por exemplo: fusão de duas listas em uma, divisão de uma lista em duas sublistas.

Listas

Considerações:

- Listas ordenadas: posição dos elementos na lista é definida em função do seu valor, de modo a manter a lista ordenada.

- Consequências:

    - para listas sequenciais: busca com complexidade $\Theta(\lg n)$.
    - para listas ligadas: não há benefício.

Listas

Considerações:

- Pilhas e filas podem ser vistas como listas mais restritas: inserções/remoções feitas apenas nas extremidades.

- Tanto a representação por lista sequencial, quanto por lista ligada pode ser adequada para implementar tais estruturas e suas operações de forma eficiente, com inserções e remoções com complexidade $\Theta(1)$, mas adaptações podem ser necessárias.