Estrutura de lista

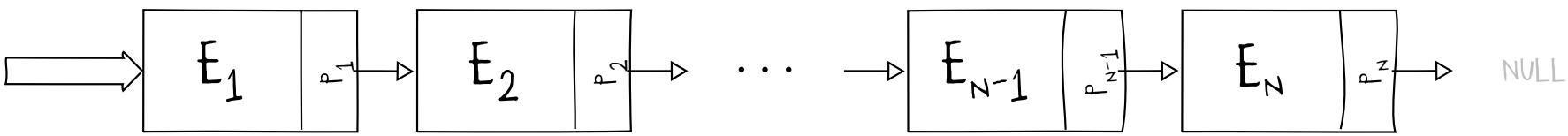
Estruturas de Dados

Bruno Prado

Departamento de Computação / UFS

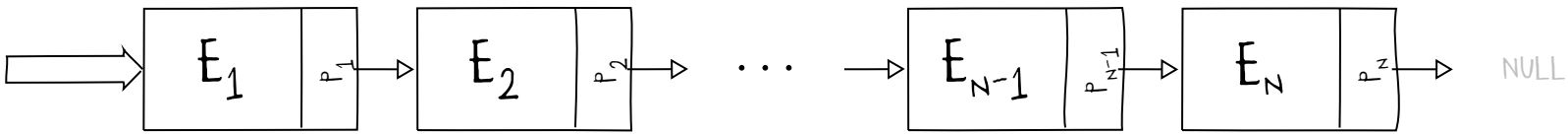
Introdução

- Estrutura de lista
  - São sequências de elementos E_i
  - Utiliza ponteiros P_i para referenciar o próximo elemento da sequência



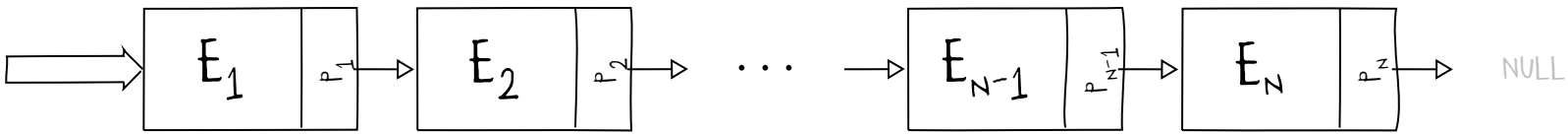
Introdução

- Lista encadeada
  - Armazenamento descontínuo em memória
  - Tempo de acesso sequencial



Introdução

- Lista encadeada
  - Armazenamento descontínuo em memória
  - Tempo de acesso sequencial



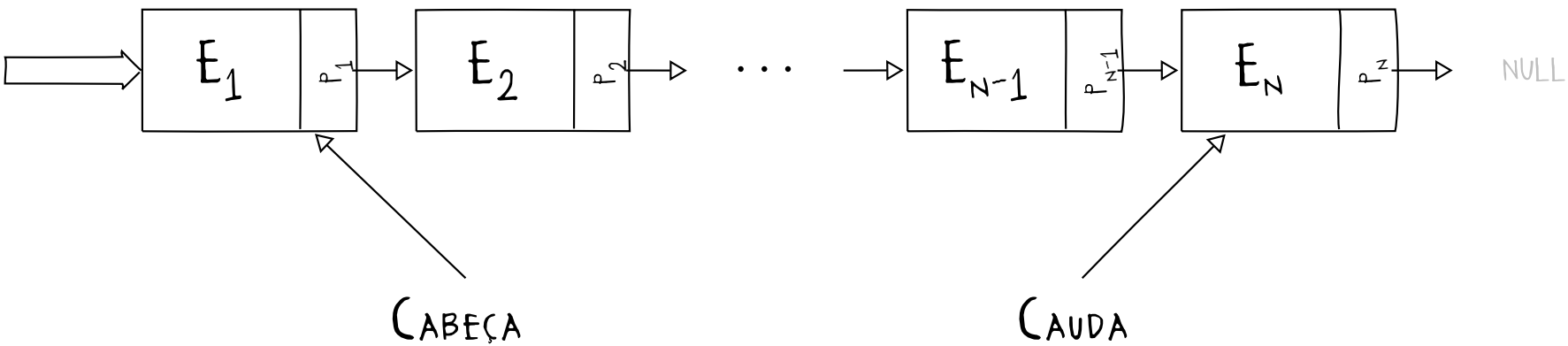
- Vetor
  - Armazenamento contínuo em memória
  - Tempo de acesso constante

Introdução

- Operações principais
  - Busca
  - Inserção
  - Remoção
  - Modificação

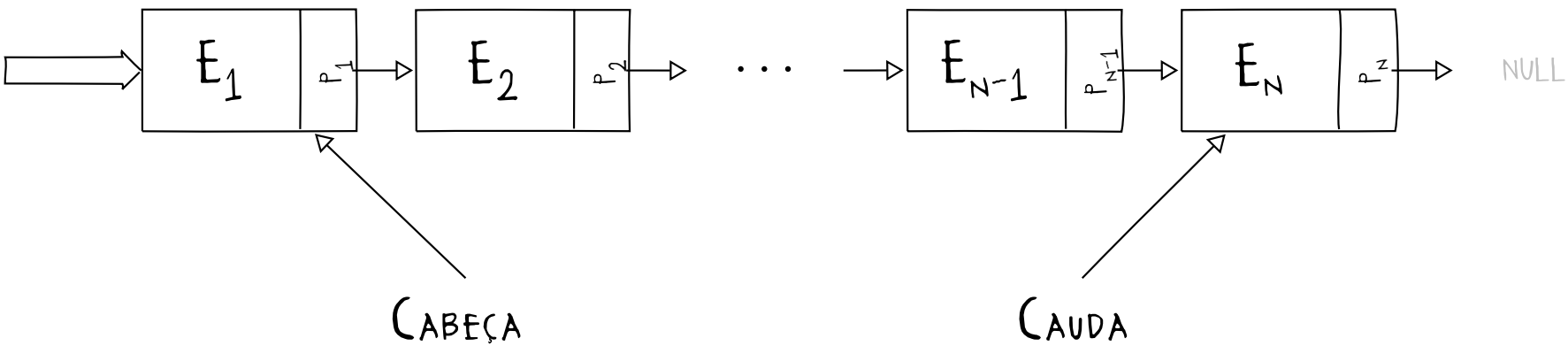
Lista encadeada

- Coleção de elementos
  - Cabeça: primeiro elemento da lista
  - Cauda: último elemento da lista



Lista encadeada

- Coleção de elementos
  - Cabeça: primeiro elemento da lista
  - Cauda: último elemento da lista



Cada elemento possui somente um ponteiro unidirecional para o próximo elemento da sequência

Lista encadeada

- Implementação em C
  - Definição do elemento

```c
// Padrão de tipos por tamanho
#include <stdint.h>
...
// Estrutura de elemento
typedef struct elemento {
    // Valor
    uint32_t E;
    // Ponteiro
    elemento* P;
} elemento;
```

Lista encadeada

- Implementação em C
  - Definição da lista

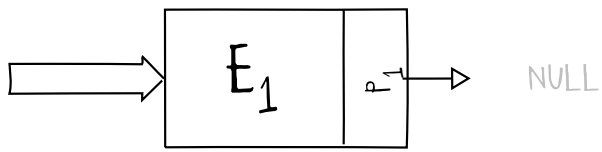
```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  ...
4  // Estrutura de lista
5  typedef struct lista {
6      // Ponteiro
7      elemento* L;
8  } lista;
```

Lista encadeada

- Inserção
  - A cabeça da lista é acessada
  - É feita a busca do ponteiro para o próximo elemento até que uma referência nula seja encontrada

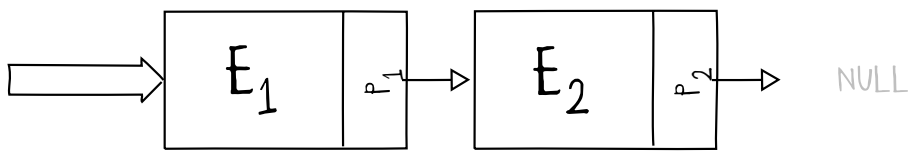
Lista encadeada

- Inserção
  - É feita a alocação dinâmica do elemento
  - O ponteiro é atualizado para referenciar o novo elemento inserido na lista



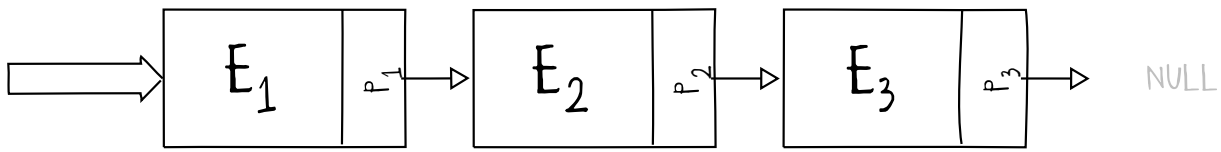
Lista encadeada

- Inserção
  - A cabeça da lista é acessada
  - É feita a busca do ponteiro para o próximo elemento até que uma referência nula seja encontrada



Lista encadeada

- Inserção
  - É feita a alocação dinâmica do elemento
  - O ponteiro é atualizado para referenciar o novo elemento inserido na lista

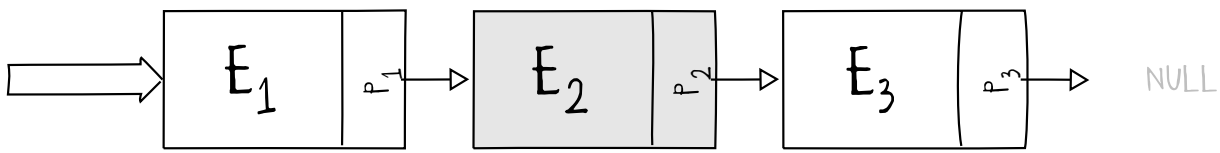


Lista encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Inserção: $\Theta(1)$

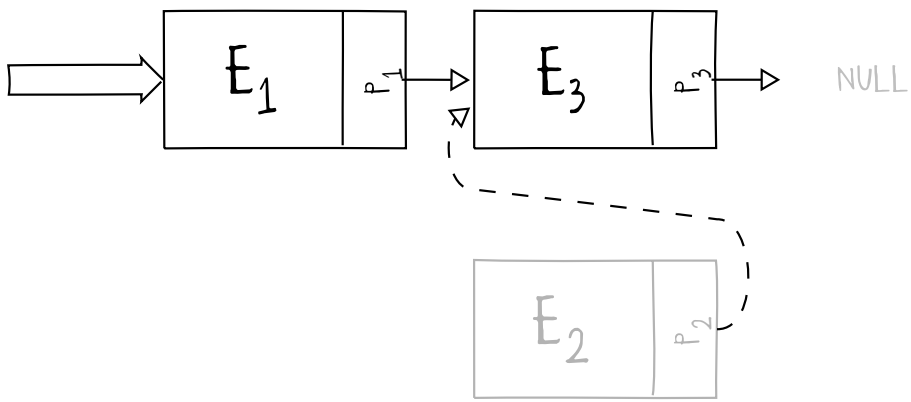
Lista encadeada

- Remoção
  - É feita a busca pelo valor do elemento E_2
  - O ponteiro P_1 é preparado para remoção



Lista encadeada

- Remoção
  - É removido da sequência o elemento E_2
  - O ponteiro P_1 é atualizado para referenciar o elemento E_3 que é o sucessor do elemento removido

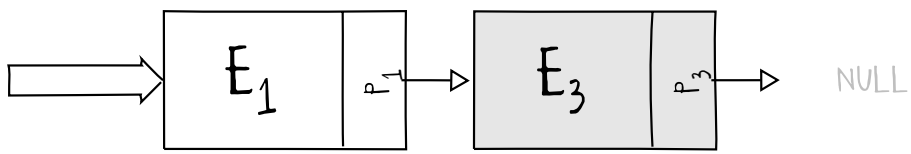


Lista encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Remoção: $\Theta(1)$

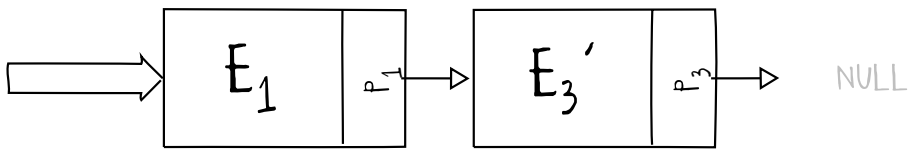
Lista encadeada

- Modificação
  - É feita a busca pelo valor do elemento E_3
  - O valor do ponteiro P_1 é armazenado



Lista encadeada

- Modificação
  - A estrutura do elemento E_3 é acessado através de sua referência e o seu valor é atualizado

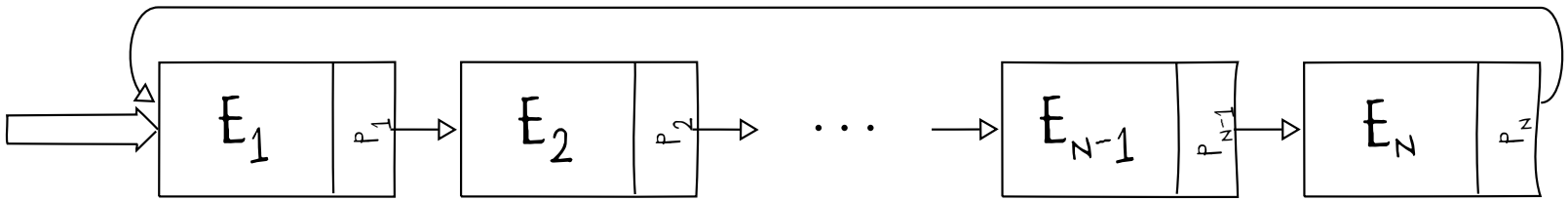


Lista encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Modificação: $\Theta(1)$

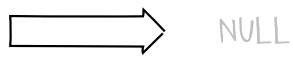
Lista circular

- É uma lista encadeada cíclica
  - As operações são equivalentes as realizadas nas listas encadeadas, mantendo o ponteiro do último elemento apontando para o primeiro



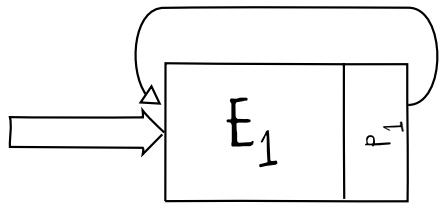
Lista circular

- Inserção
  - Como não existem elementos em uma lista vazia, não existe a referência circular



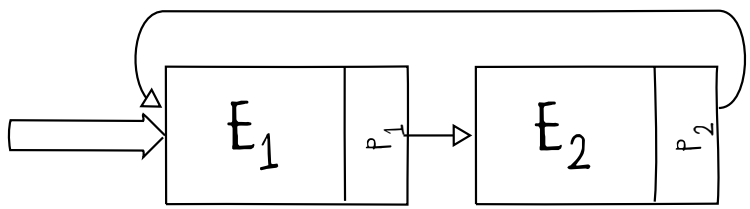
Lista circular

- Inserção
  - É feita a alocação dinâmica do elemento
  - Os ponteiros são atualizados para referenciar o novo elemento inserido na lista



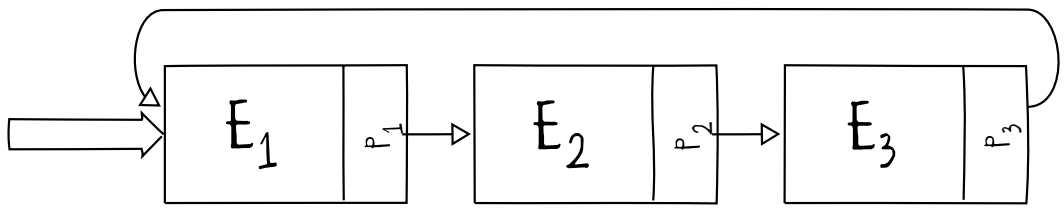
Lista circular

- Inserção
  - É feita a alocação dinâmica do elemento
  - O novo elemento é apontado e faz referência ao primeiro elemento da lista



Lista circular

- Inserção
  - É feita a alocação dinâmica do elemento
  - O novo elemento é apontado e faz referência ao primeiro elemento da lista

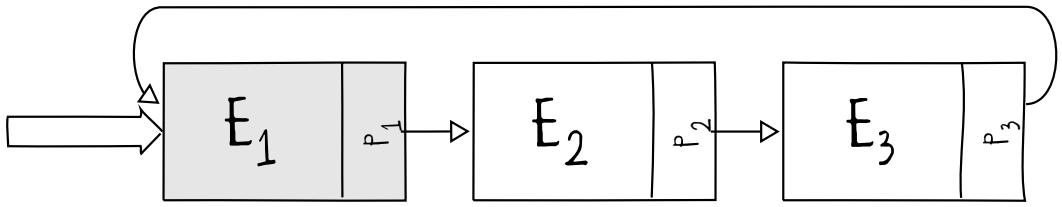


Lista circular

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Inserção: $\Theta(1)$

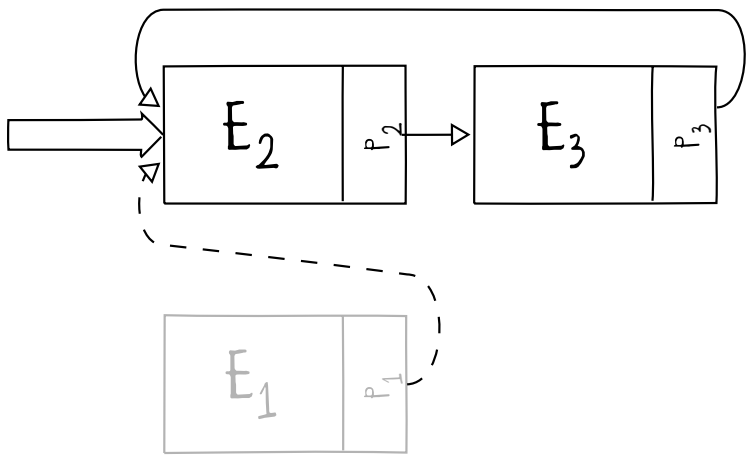
Lista circular

- Remoção
  - É feita a busca pelo valor do elemento E_1
  - O ponteiro da lista é preparado para remoção



Lista circular

- Remoção
  - É removido da sequência o elemento E_1
  - O ponteiro da lista é atualizado para referenciar o elemento E_2 e assim como o último elemento E_3

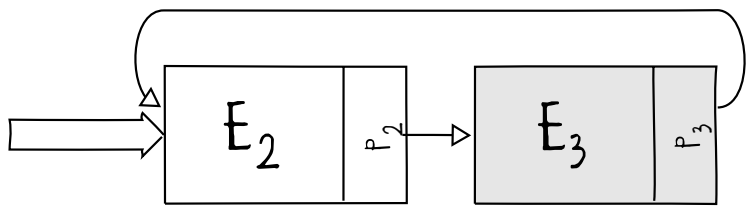


Lista circular

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Remoção: $\Theta(1)$

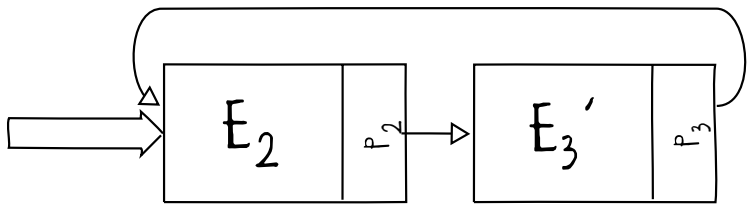
Lista circular

- Modificação
  - É feita a busca pelo valor do elemento E_3
  - O valor do ponteiro P_2 é armazenado



Lista circular

- Modificação
  - A estrutura do elemento E_3 é acessado através de sua referência e o seu valor é atualizado

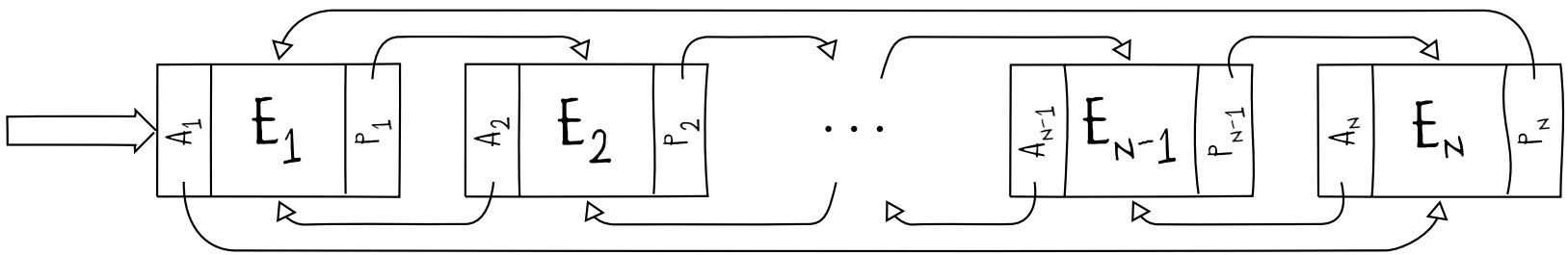


Lista circular

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Modificação: $\Theta(1)$

Lista duplamente encadeada

- Lista encadeada com dois ponteiros
  - Ponteiros do elemento referenciam o elemento anterior e próximo da sequência
  - Navegação em ambas direções



Lista duplamente encadeada

- Implementação em C
  - Definição do elemento

```
1  // Padrão de tipos por tamanho
2  #include <stdint.h>
3  ...
4  // Estrutura de elemento
5  typedef struct elemento {
6      // Valor
7      uint32_t E;
8      // Ponteiro para anterior
9      elemento* A;
10     // Ponteiro para próximo
11     elemento* P;
12 } elemento;
```

Lista duplamente encadeada

- Implementação em C
  - Definição da lista

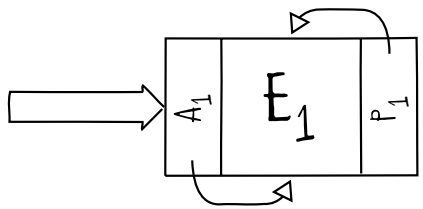
```
1 // Padrão de tipos por tamanho
2 #include <stdint.h>
3 ...
4 // Estrutura de lista
5 typedef struct lista {
6     // Ponteiro
7     elemento* L;
8 } lista;
```

Lista duplamente encadeada

- Inserção
  - A cabeça da lista é acessada
  - É feita a busca do ponteiro para o próximo elemento até que uma referência nula seja encontrada

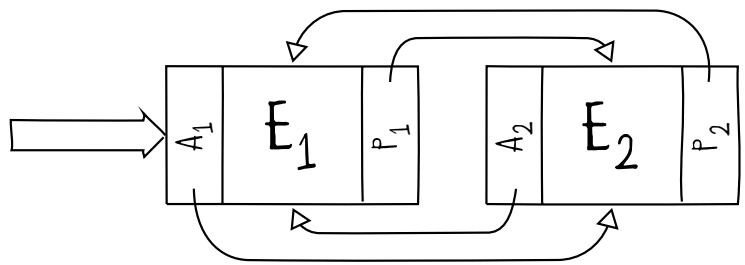
Lista duplamente encadeada

- Inserção
  - É feita a alocação dinâmica do elemento
  - Os ponteiros são atualizados para referenciar o novo elemento inserido na lista



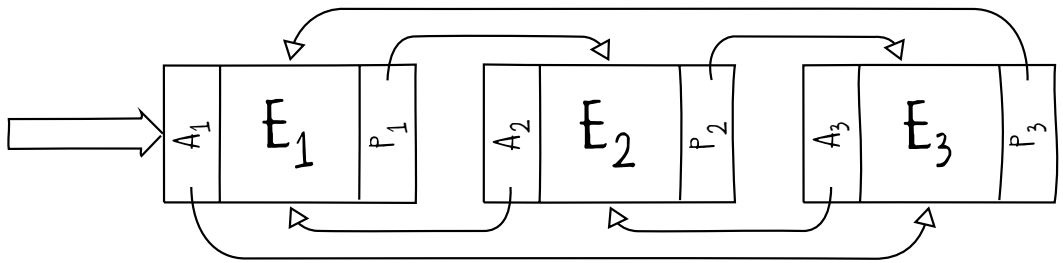
Lista duplamente encadeada

- Inserção
  - É feita a alocação dinâmica do elemento
  - Os ponteiros são atualizados para referenciar o novo elemento inserido na lista



Lista duplamente encadeada

- Inserção
  - É feita a alocação dinâmica do elemento
  - Os ponteiros são atualizados para referenciar o novo elemento inserido na lista

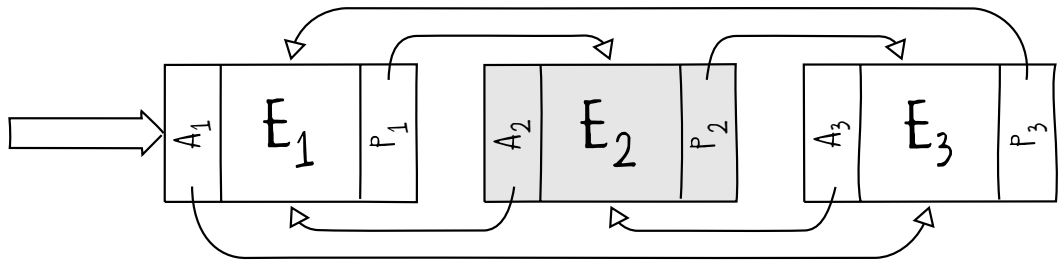


Lista duplamente encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Inserção: $\Theta(1)$

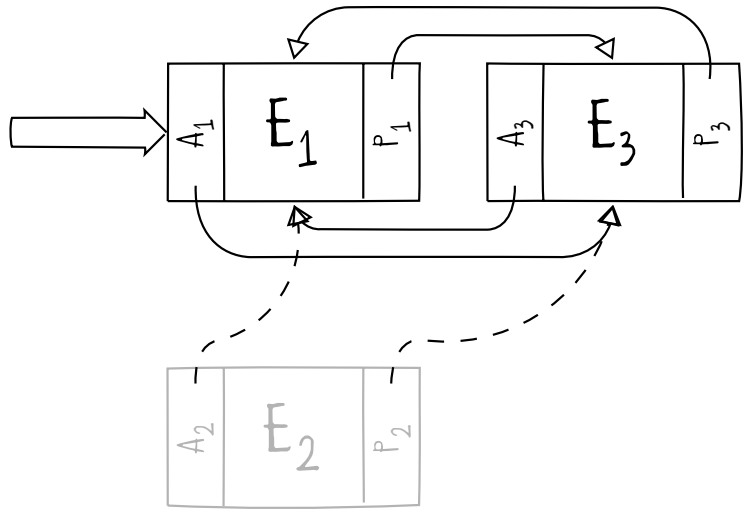
Lista duplamente encadeada

- Remoção
  - É feita a busca pelo valor do elemento E_2
  - Os ponteiros P_1 e A_3 são preparados para remoção



Lista duplamente encadeada

- Remoção
  - O elemento E_2 é removido da sequência
  - Os ponteiros P_1 e A_3 são atualizados

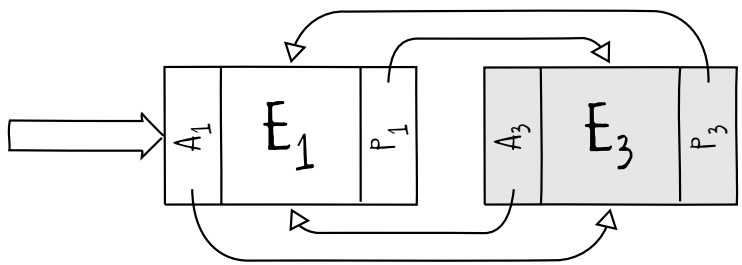


Lista duplamente encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Remoção: $\Theta(1)$

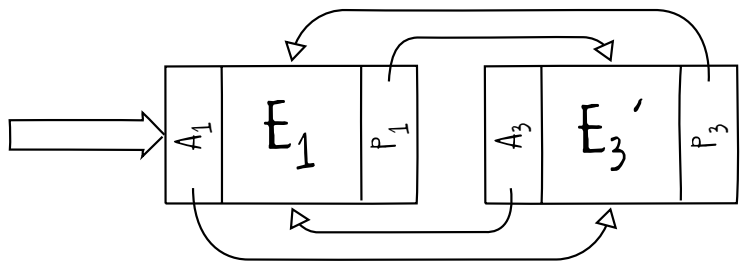
Lista duplamente encadeada

- Modificação
  - É feita a busca pelo valor do elemento E_3
  - O valor do ponteiro A_1 ou P_1 é armazenado



Lista duplamente encadeada

- Modificação
  - A estrutura do elemento E_3 é acessado através de sua referência e o seu valor é atualizado



Lista duplamente encadeada

- Análise de complexidade
  - Busca: $\Omega(1)$ e $O(n)$
  - Modificação: $\Theta(1)$

Aplicações

- Vantagens
  ✓ Permite alocação descontínua e incremental de memória, sem necessidade de realocação
  ✓ Realização de operações de inserção, de remoção e de modificação em tempo constante

Aplicações

- Vantagens
  ✓ Permite alocação descontínua e incremental de memória, sem necessidade de realocação
  ✓ Realização de operações de inserção, de remoção e de modificação em tempo constante
- Desvantagens
  ✗ Necessidade de busca sequencial para realização das operações sobre elementos
  ✗ O espaço utilizado pelos ponteiros por ser maior que o dado armazenado, como o tipo caractere

Exercício

- A empresa de tecnologia Poxim Tech está desenvolvendo uma rede social para os melhores amigos, com a ideia de unir as pessoas como se estivessem de mãos dadas através de um círculo de pessoas que interagem com os vizinhos
  - Os nomes dos usuários desta rede são compostos exclusivamente por letras com até 50 caracteres
  - Quando um usuário é adicionado ele sempre será amigo do último e do primeiro usuário da rede social
  - Caso seja removido da rede social, os amigos do usuário passam a ser amigos entre si
  - É possível buscar uma determinada pessoa através do seu nome e mostrar os nomes de seus amigos

Exercício

- Formato do arquivo de entrada
  - Adicionar pessoa: *ADD name*
  - Remover pessoa: *REMOVE name*
  - Mostrar amigos: *SHOW name*

```
1  ADD␣Jose␣da␣Silva
2  SHOW␣Jose␣da␣Silva
3  ADD␣Jose␣da␣Silva
4  ADD␣Joao␣dos␣Santos
5  ADD␣Maria␣da␣Penha
6  REMOVE␣Joao␣dos␣Santos
7  REMOVE␣Maria␣da␣Silva
8  ADD␣Alan␣Turing
9  SHOW␣Maria␣da␣Penha
10 SHOW␣Bruno␣Prado
```

Exercício

- Formato do arquivo de saída
  - São exibidos os resultados de cada operação realizada, informando o resultado de cada execução

```
1  [SUCCESS]ADD=Jose_da_Silva
2  [SUCCESS]SHOW=Jose_da_Silva<-Jose_da_Silva->Jose_da_Silva
3  [FAILURE]ADD=Jose_da_Silva
4  [SUCCESS]ADD=Joao_dos_Santos
5  [SUCCESS]ADD=Maria_da_Penha
6  [SUCCESS]REMOVE=Joao_dos_Santos
7  [FAILURE]REMOVE=Maria_da_Silva
8  [SUCCESS]ADD=Alan_Turing
9  [SUCCESS]SHOW=Jose_da_Silva<-Maria_da_Penha->Alan_Turing
10 [FAILURE]SHOW=?<-Bruno_Prado->?
```