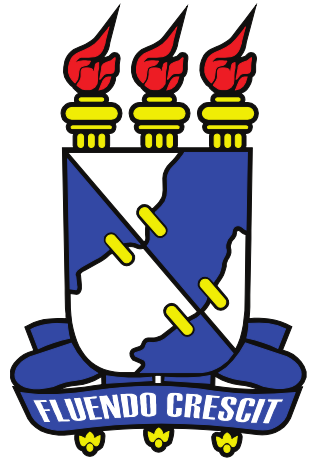


Estrutura de Dados I

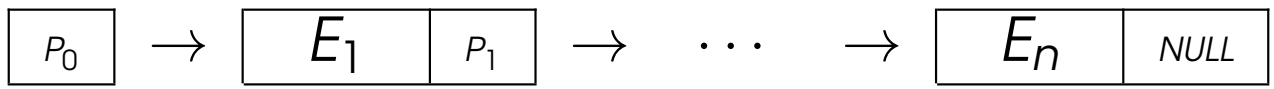
Estrutura de lista

Bruno Prado

Departamento de Computação / UFS

Introdução

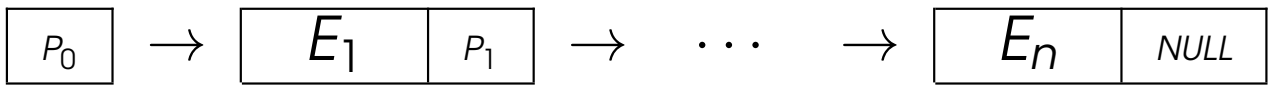
- Estrutura de lista
  - São sequências de elementos E_i
  - Utiliza ponteiros P_i para referenciar o próximo elemento da sequência



Lista

Introdução

- Lista x Vetor
  - Armazenamento descontínuo em memória
  - Tempo de acesso sequencial



Lista

  - Armazenamento contínuo em memória
  - Tempo de acesso constante

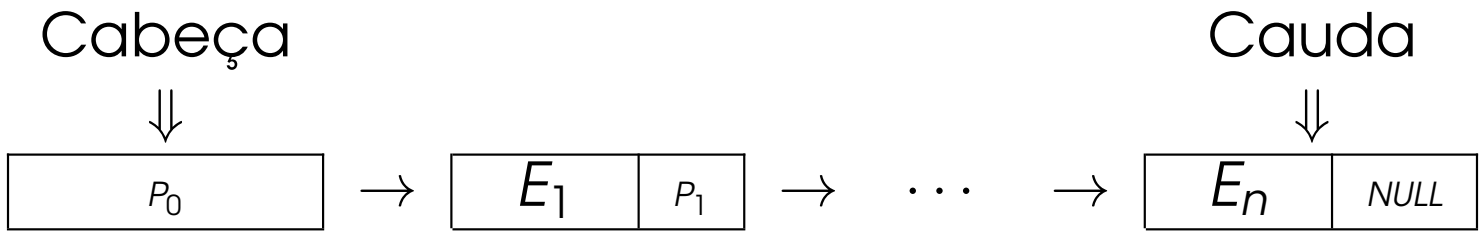
Vetor

Introdução

- Operações principais
  - Busca
  - Inserção
  - Remoção
  - Modificação

Lista Encadeada

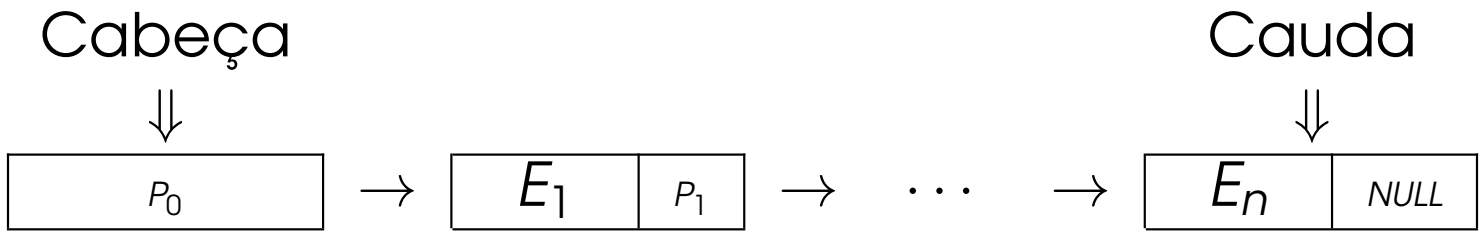
- Coleção de elementos
  - A cabeça da lista é um ponteiro para o primeiro elemento da lista encadeada
  - O último elemento da lista é denotado por cauda



Lista Encadeada

Lista Encadeada

- Coleção de elementos
  - A cabeça da lista é um ponteiro para o primeiro elemento da lista encadeada
  - O último elemento da lista é denotado por cauda



Lista Encadeada

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

Lista Encadeada

- Implementação em C
  - Definição das estruturas

```
// Estrutura de elemento
typedef struct elemento {
        // Valor
        int E;
        // Ponteiro
        elemento* P;
} elemento;
```

Lista Encadeada

- Implementação em C
  - Definição das estruturas

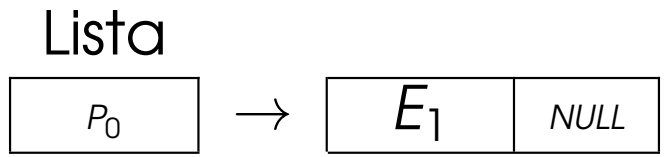
```
// Estrutura de lista
typedef struct lista {
        // Ponteiro
        elemento* P;
} lista;
```

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

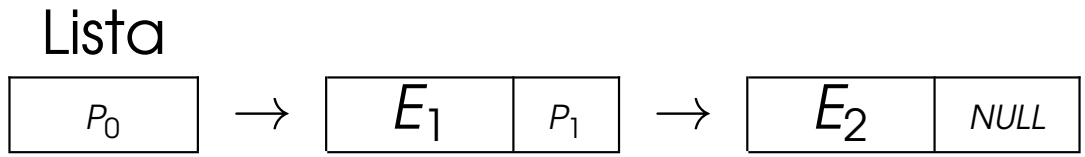
Lista Encadeada

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



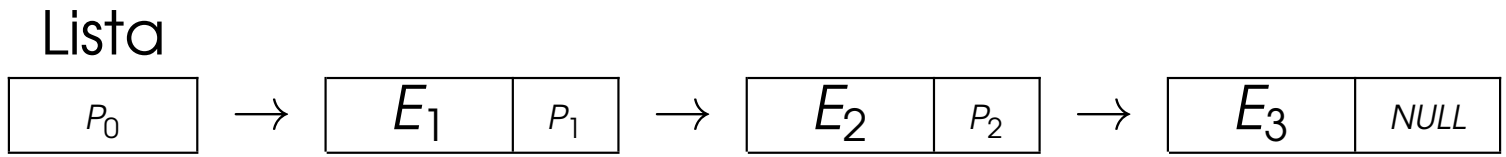
Lista Encadeada

- 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

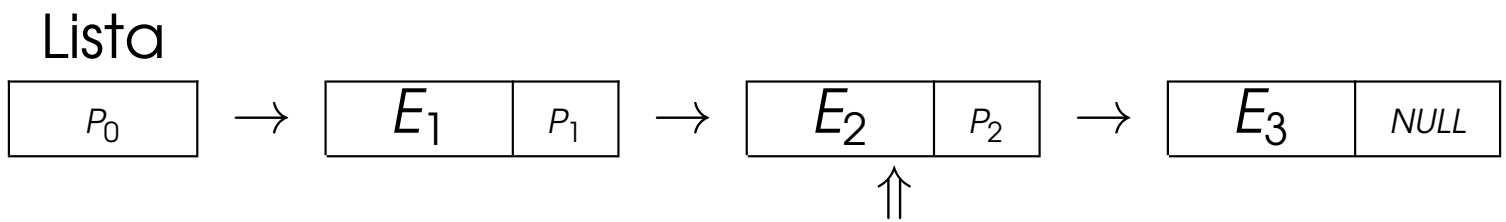


Lista Encadeada

- Análise de complexidade
  - Busca $O(n)$
  - Inserção $O(1)$

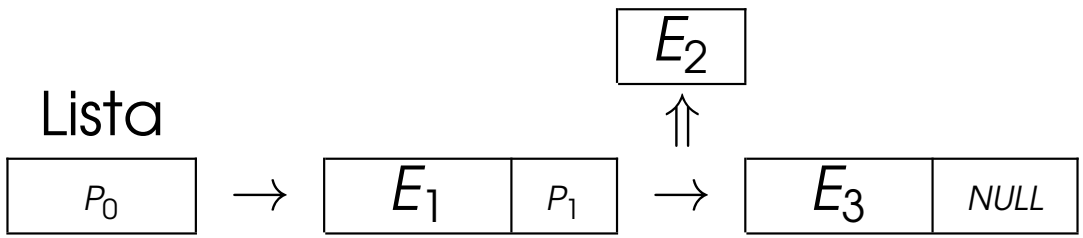
Lista Encadeada

- 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

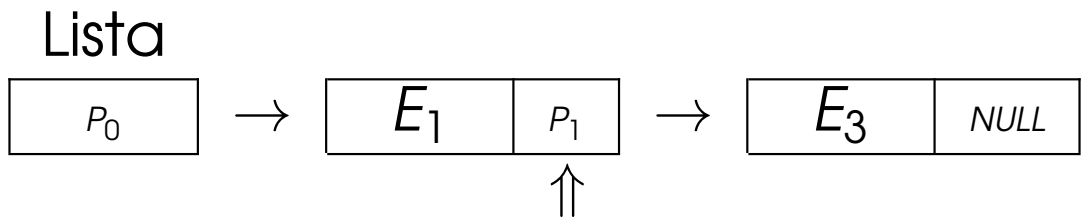


Lista Encadeada

- Análise de complexidade
  - Busca $O(n)$
  - Remoção $O(1)$

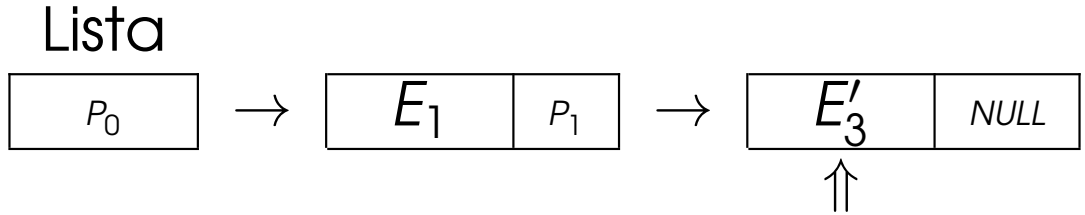
Lista Encadeada

- 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

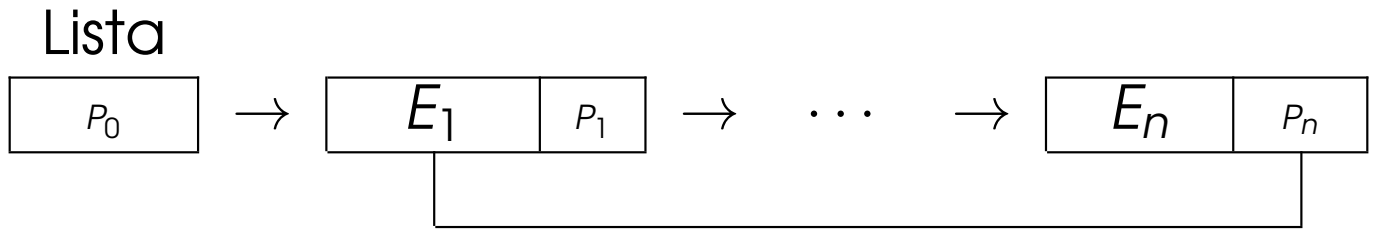


Lista Encadeada

- Análise de complexidade
  - Busca $O(n)$
  - Modificação $O(1)$

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

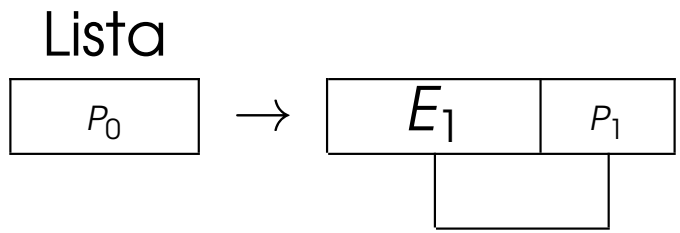


Lista Circular

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

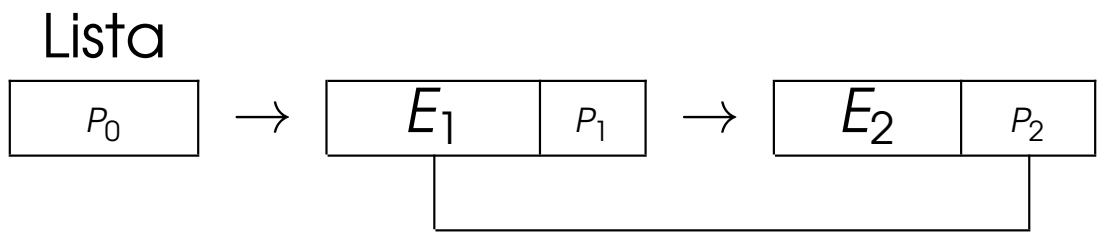
Lista Circular

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



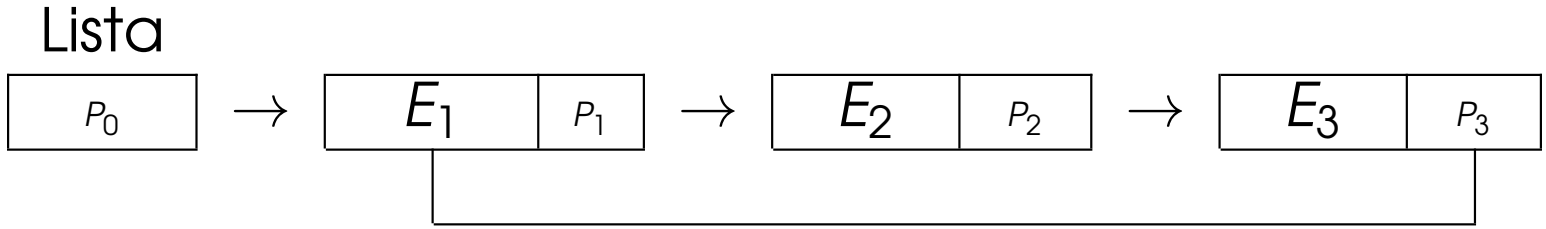
Lista Circular

- 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

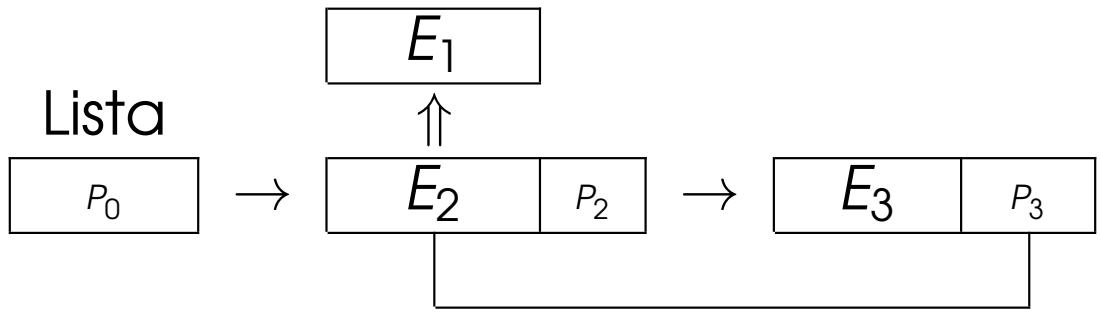


Lista Circular

- Análise de complexidade
  - Busca $O(n)$
  - Inserção $O(1)$

Lista Circular

- Remoção
  - É feita a busca pelo valor do elemento E_1
  - O ponteiro P_3 é aponta para o elemento E_2

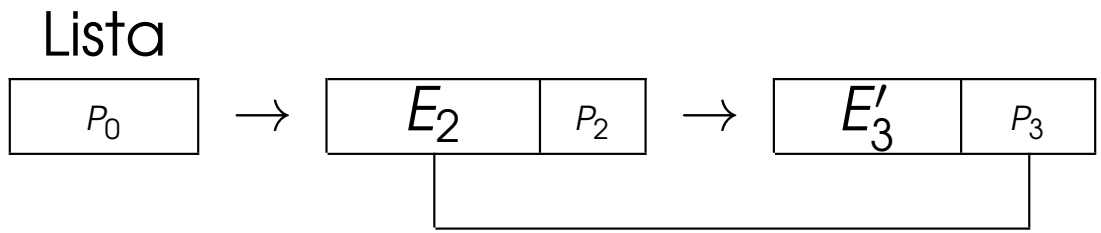


Lista Circular

- Análise de complexidade
  - Busca $O(n)$
  - Remoção $O(1)$

Lista Circular

- Modificação
  - É feita a busca pelo valor do elemento E_3
  - A estrutura do elemento E_3 é acessado através de sua referência e o seu valor é atualizado

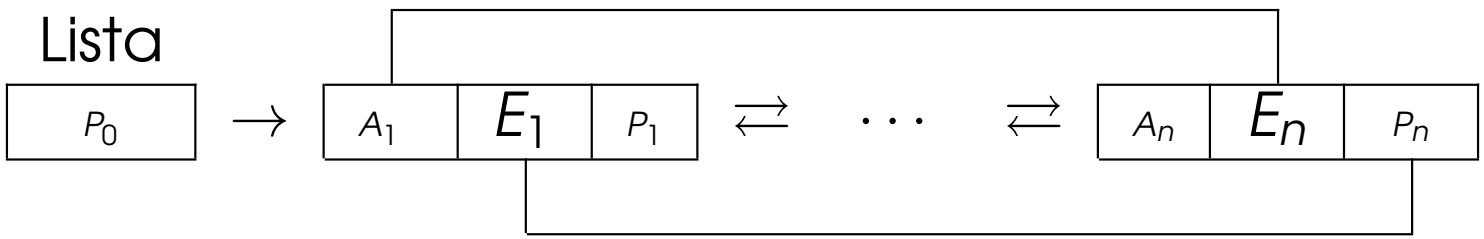


Lista Circular

- Análise de complexidade
  - Busca $O(n)$
  - Modificação $O(1)$

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 das estruturas

```
// Estrutura de elemento
typedef struct elemento {
        // Valor
        int E;
        // Ponteiro para anterior
        elemento* A;
        // Ponteiro para próximo
        elemento* P;
} elemento;
```

Lista Duplamente Encadeada

- Implementação em C
  - Definição das estruturas

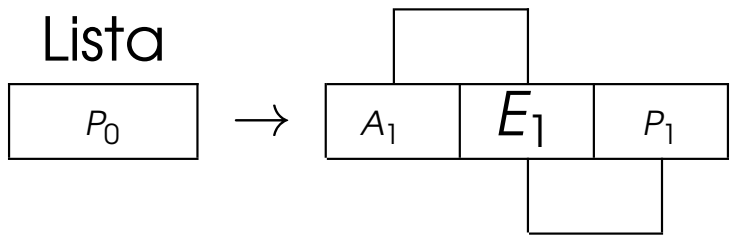
```
// Estrutura de lista
typedef struct lista {
        // Ponteiro
        elemento* P;
} lista;
```

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

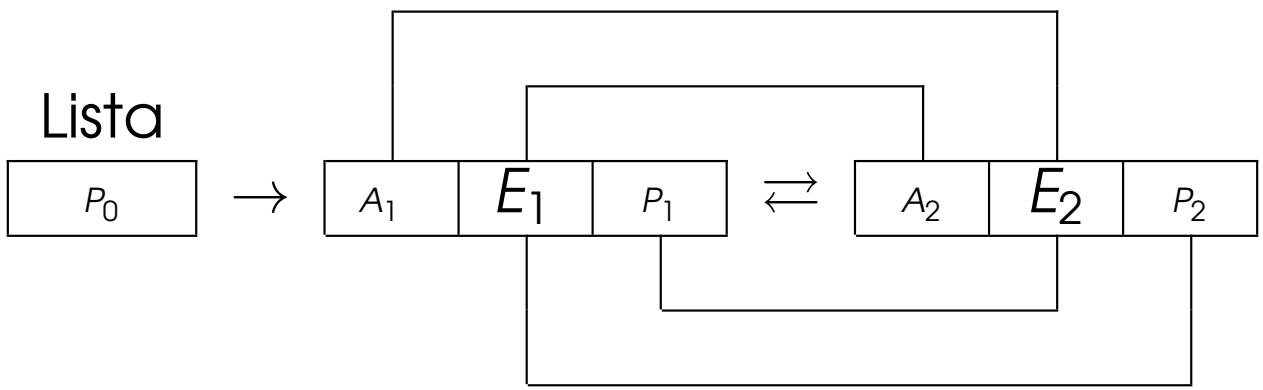
Lista Duplamente Encadeada

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



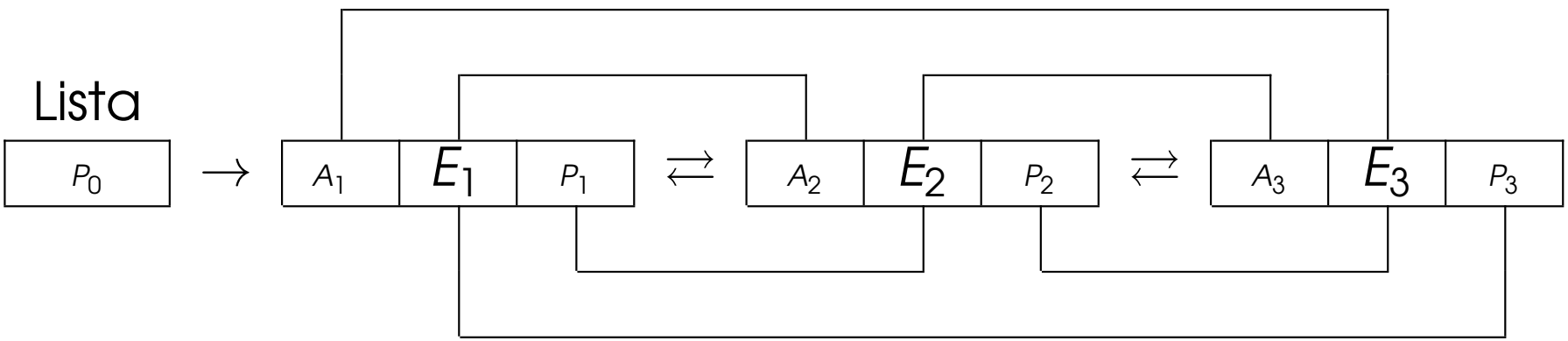
Lista Duplamente Encadeada

- 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

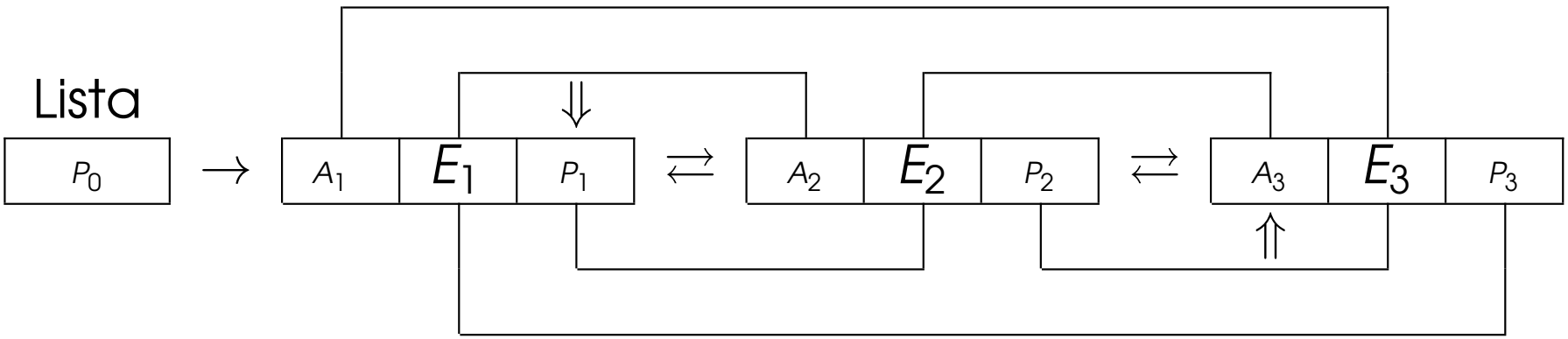


Lista Duplamente Encadeada

- Análise de complexidade
  - Busca $O(1)$
  - Inserção $O(1)$

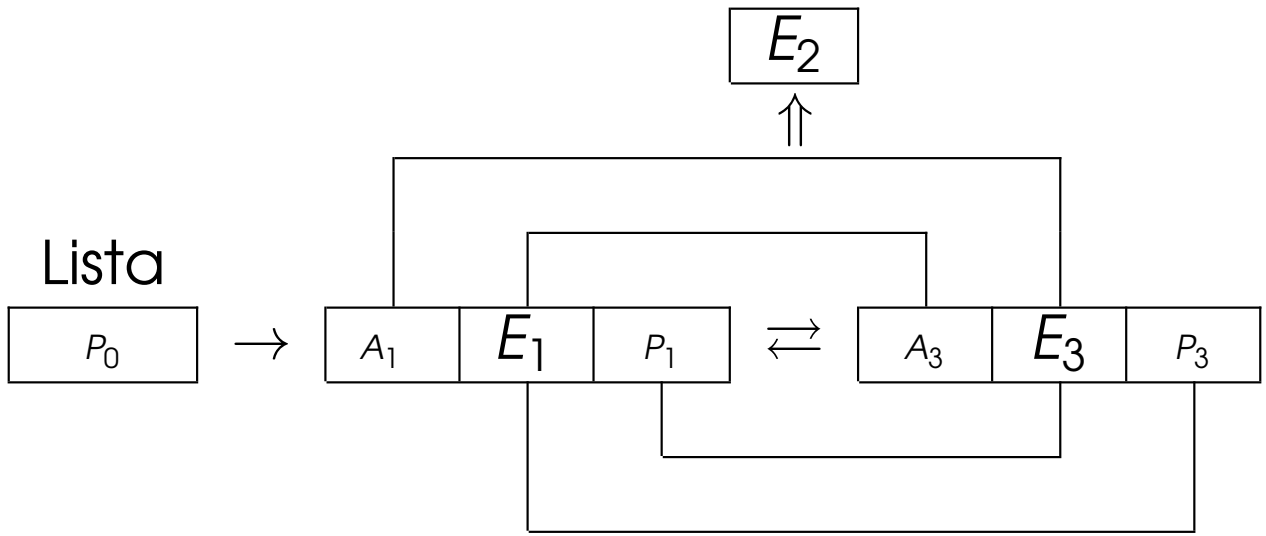
Lista Duplamente Encadeada

- 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

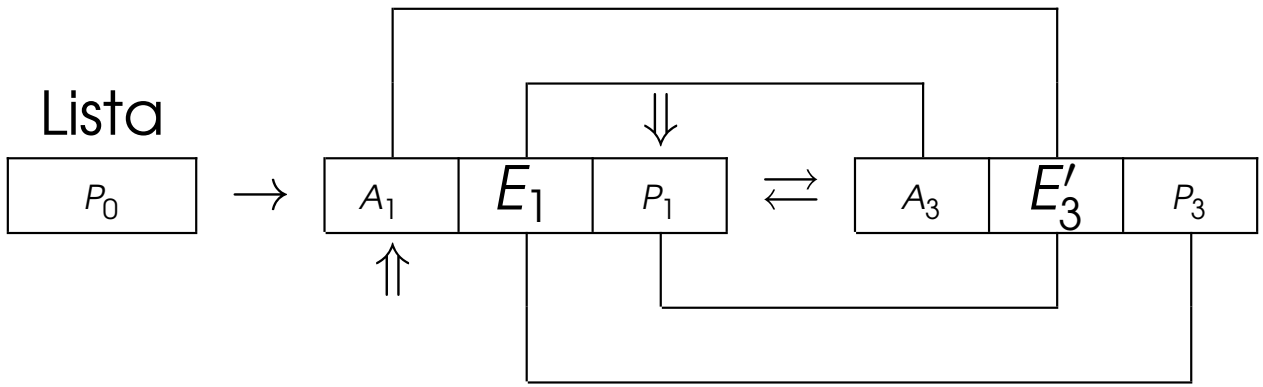


Lista Duplamente Encadeada

- Análise de complexidade
  - Busca $O(n)$
  - Remoção $O(1)$

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

- Análise de complexidade
  - Busca $O(n)$
  - Modificação $O(1)$

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*

```
ADD Jose da Silva
SHOW Jose da Silva
ADD Jose da Silva
ADD Joao dos Santos
ADD Maria da Penha
REMOVE Joao dos Santos
REMOVE Maria da Silva
ADD Alan Turing
SHOW Maria da Penha
SHOW Bruno Prado
```

Exercício

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

```
[Jose da Silva] ADD-OK
[Jose da Silva]<-[Jose da Silva]->[Jose da Silva]
[Jose da Silva] ADD-ERROR
[Joao dos Santos] ADD-OK
[Maria da Penha] ADD-OK
[Joao dos Santos] REMOVE-OK
[Maria da Silva] REMOVE-ERROR
[Alan Turing] ADD-OK
[Jose da Silva]<-[Maria da Penha]->[Alan Turing]
[Bruno Prado] SHOW-ERROR
```