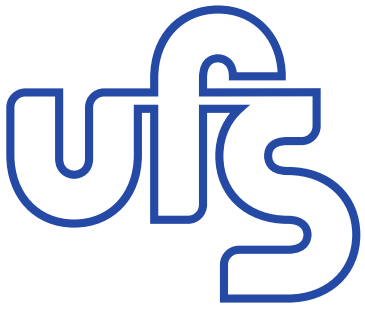
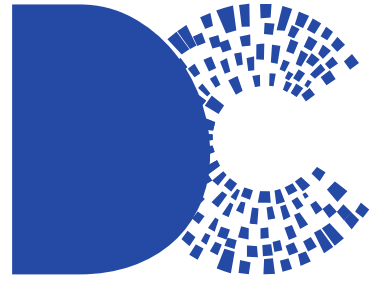




Conjuntos disjuntos

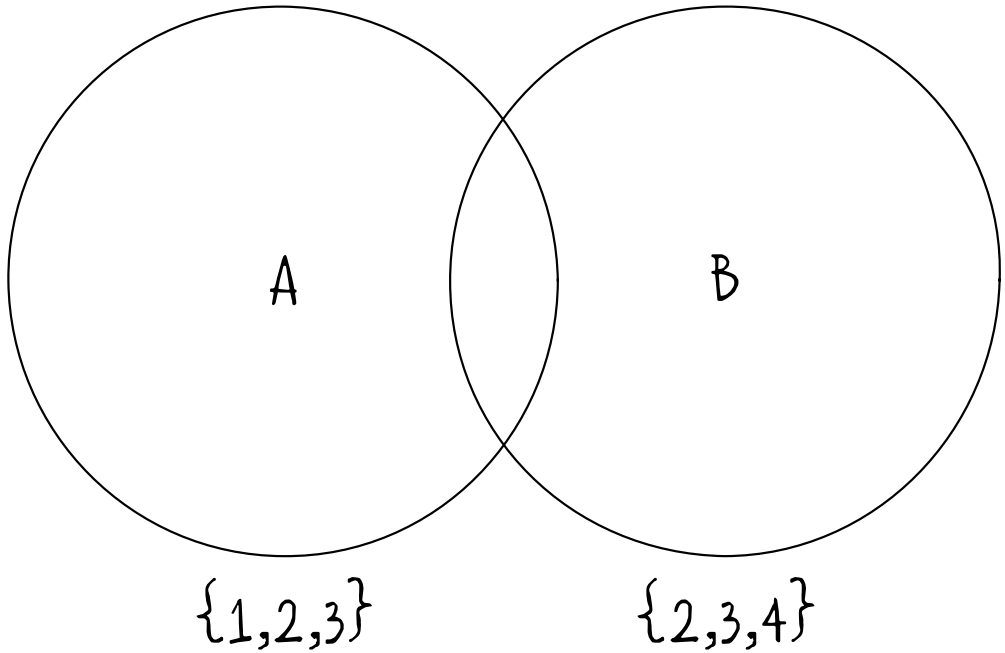
Estruturas de Dados

Bruno Prado

Departamento de Computação / UFS

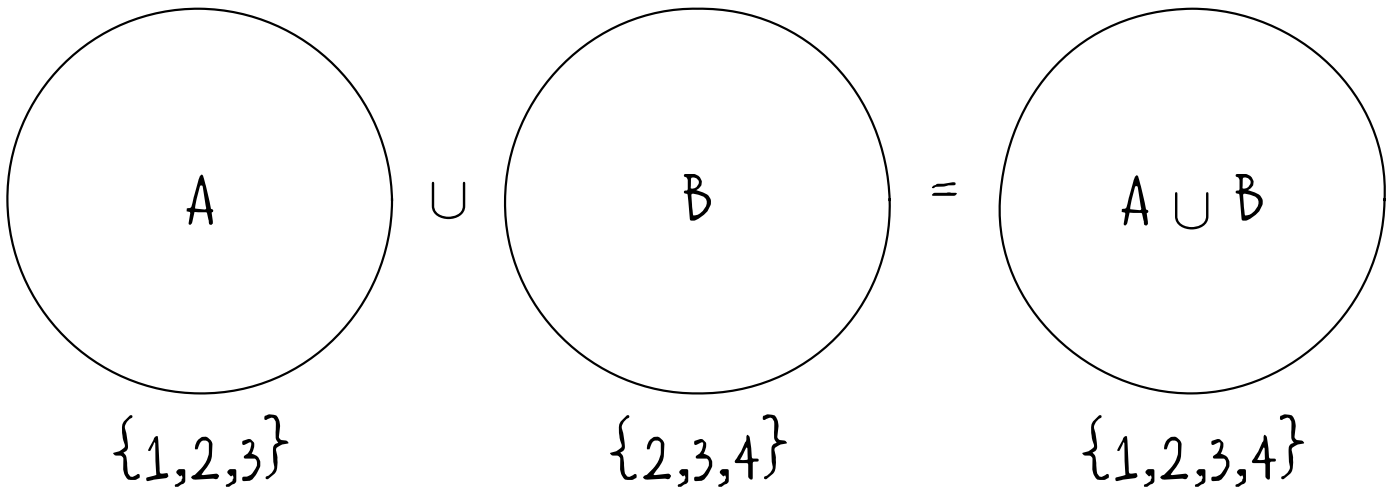
Introdução

- O que é um conjunto?
  - É uma coleção de elementos distintos
  - Podem ser ilustrados através do diagrama de Venn



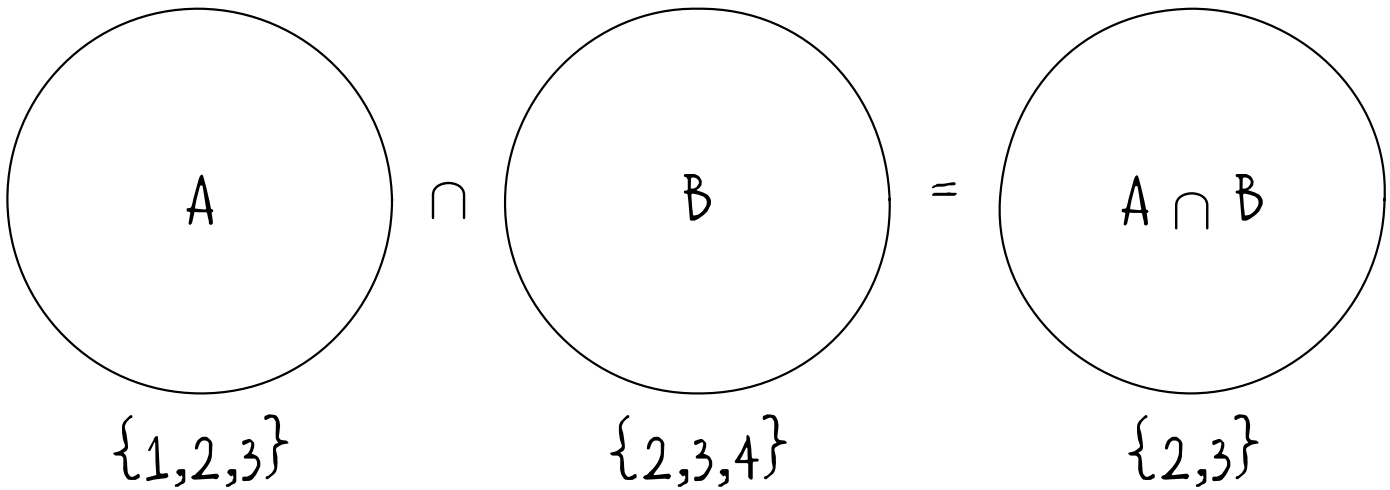
Introdução

- Operação de união de conjuntos
  - Notação $\cup$



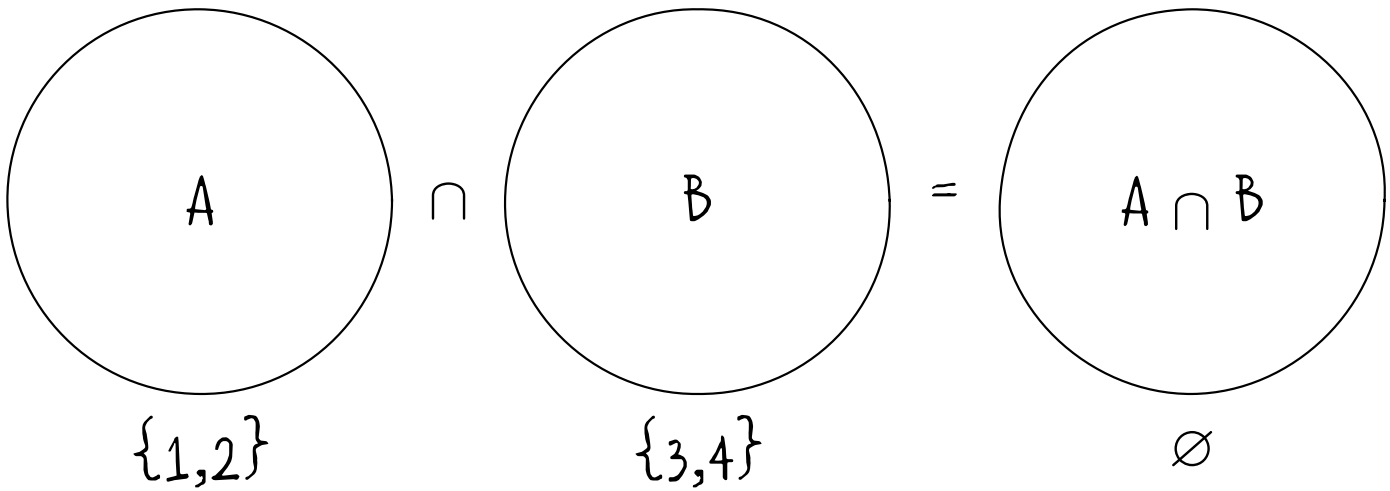
Introdução

- Operação de interseção de conjuntos
  - Notação $\cap$



Introdução

- O que são conjuntos disjuntos?
  - É uma coleção de conjuntos cujos elementos não possuem interseção entre si



Conjuntos disjuntos

- Operações básicas em conjuntos disjuntos
  - Criação (*make-set*)
  - União (*union*)
  - Busca (*find-set*)

Conjuntos disjuntos

- Operações básicas em conjuntos disjuntos
  - Criação (*make-set*)
  - União (*union*)
  - Busca (*find-set*)

Devido a estas operações principais, esta estrutura de dados também é conhecida como conjuntos *union-find*

Conjuntos disjuntos

- Definição da estrutura de conjuntos disjuntos
  - Coleção de conjuntos disjuntos $S = \{S_1, S_2, \ldots, S_n\}$
  - Cada conjunto é identificado por um representante
  - Este representante é algum elemento do conjunto
    - Não importa qual é o elemento
    - Só precisa ser o mesmo em todas as operações
- Estruturas de armazenamento
  - Listas encadeadas
  - Árvores

Conjuntos disjuntos

- Criação de um conjunto disjunto (*make-set*)
  - É feita a criação de um conjunto i com exatamente um elemento $S_i = \{x\}$
  - O elemento x é o representante do conjunto, só existindo neste conjunto disjunto
  - Este novo conjunto criado é adicionado a coleção de conjuntos disjuntos $S = S \cup S_i$

Conjuntos disjuntos

- Criação de um conjunto disjunto (*make-set*)
  - É feita a criação de um conjunto i com exatamente um elemento $S_i = \{x\}$
  - O elemento x é o representante do conjunto, só existindo neste conjunto disjunto
  - Este novo conjunto criado é adicionado a coleção de conjuntos disjuntos $S = S \cup S_i$

Cada elemento é inserido através desta operação, sendo realizadas n operações para criação dos conjuntos disjuntos

Conjuntos disjuntos

- União de dois conjuntos disjuntos (*union*)
  - Assumindo dois conjuntos $S_i = \{x\}$ e $S_j = \{y\}$, é necessário escolher um novo representante de S_i ou S_j
  - É feita a remoção dos conjuntos S_i e S_j dos conjuntos disjuntos $S = S - S_i - S_j$
  - A operação de união cria um novo conjunto $S_k = S_i \cup S_j$ que é incorporado à coleção $S = S \cup S_k$

Conjuntos disjuntos

- União de dois conjuntos disjuntos (*union*)
  - Assumindo dois conjuntos $S_i = \{x\}$ e $S_j = \{y\}$, é necessário escolher um novo representante de S_i ou S_j
  - É feita a remoção dos conjuntos S_i e S_j dos conjuntos disjuntos $S = S - S_i - S_j$
  - A operação de união cria um novo conjunto $S_k = S_i \cup S_j$ que é incorporado à coleção $S = S \cup S_k$

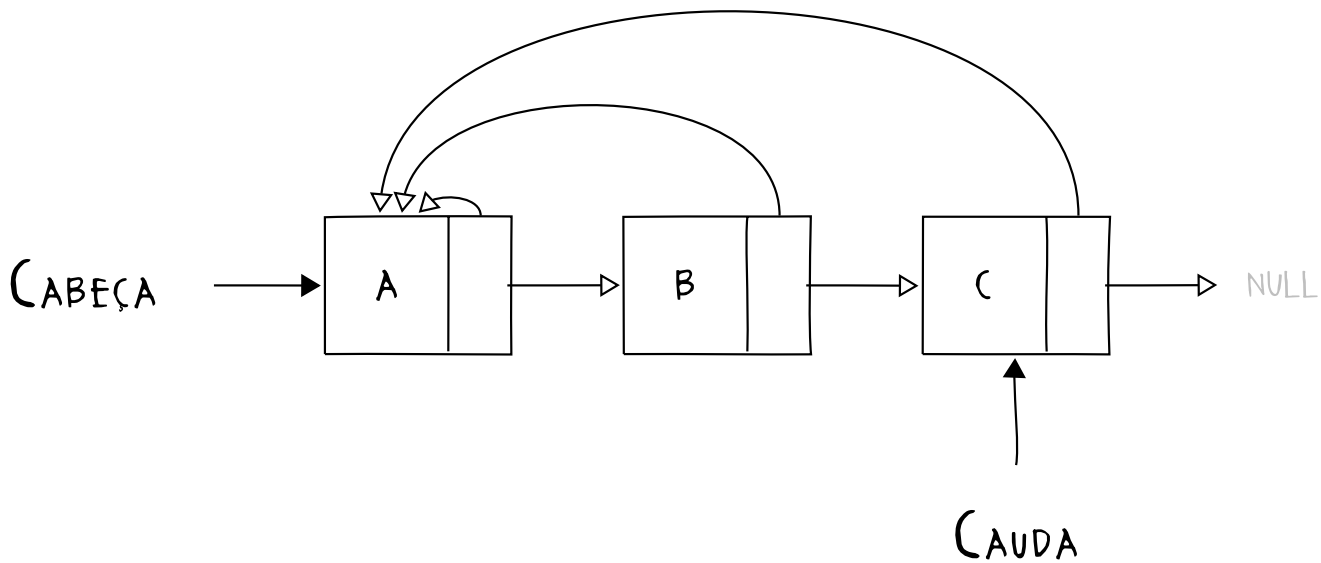
Cada operação de união reduz a quantidade de conjuntos disjuntos por 1, realizando no máximo $n - 1$ uniões até que reste apenas um conjunto

Conjuntos disjuntos

- Busca por um conjunto disjunto (*find-set*)
  - É retornado o conjunto que contém o elemento *x*, através da referência do conjunto
  - Esta referência contém o representante do conjunto que contém o elemento *x*

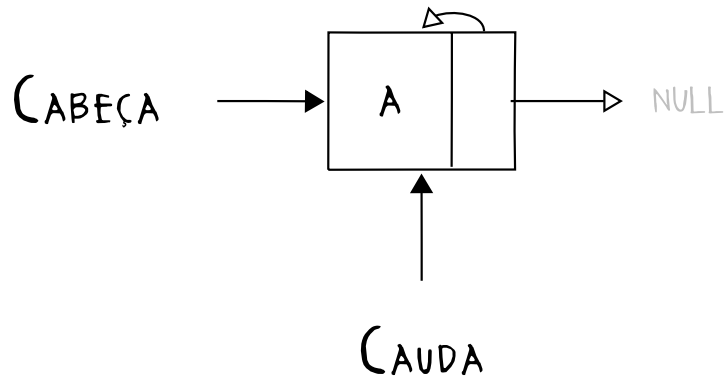
Conjuntos disjuntos

- Representação por lista encadeada
  - Cada conjunto é uma lista encadeada, com ponteiros para cabeça e para cauda da lista
  - Os elementos apontam para o próximo elemento e para cabeça da lista que é o representante



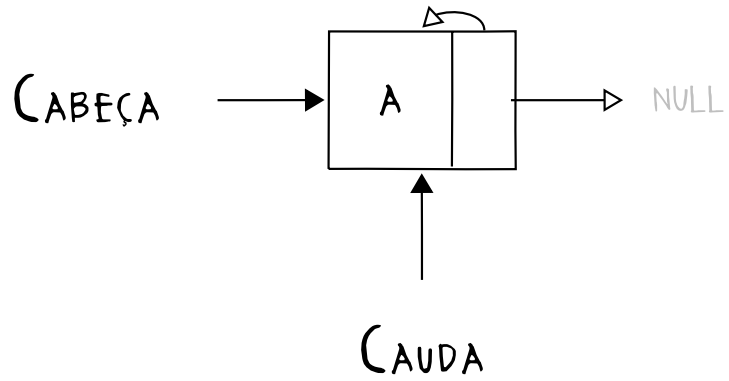
Conjuntos disjuntos

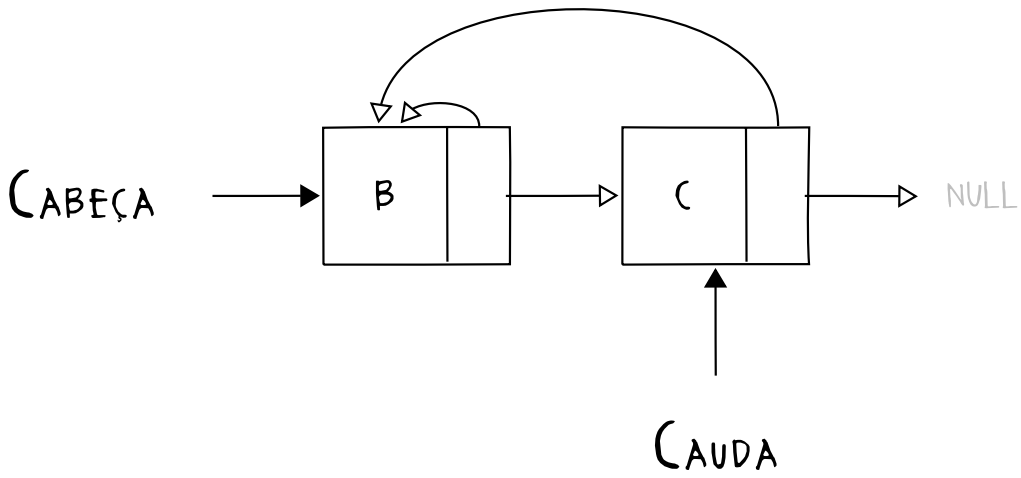
- Representação por lista encadeada
  - Criando um conjunto para o elemento *a*
  - A criação da lista possui custo $O(1)$



Conjuntos disjuntos

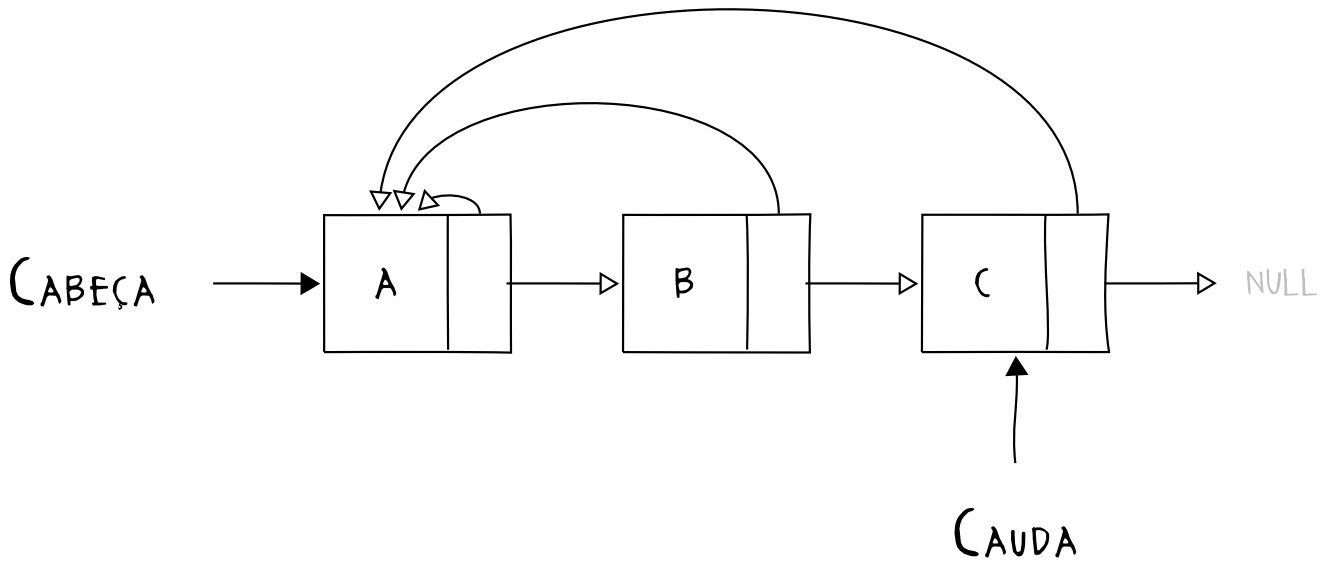
- Representação por lista encadeada
  - Unindo dois conjuntos disjuntos
  - Para realizar a união dos elementos *a*, *b* e *c* é necessário ajustar os ponteiros





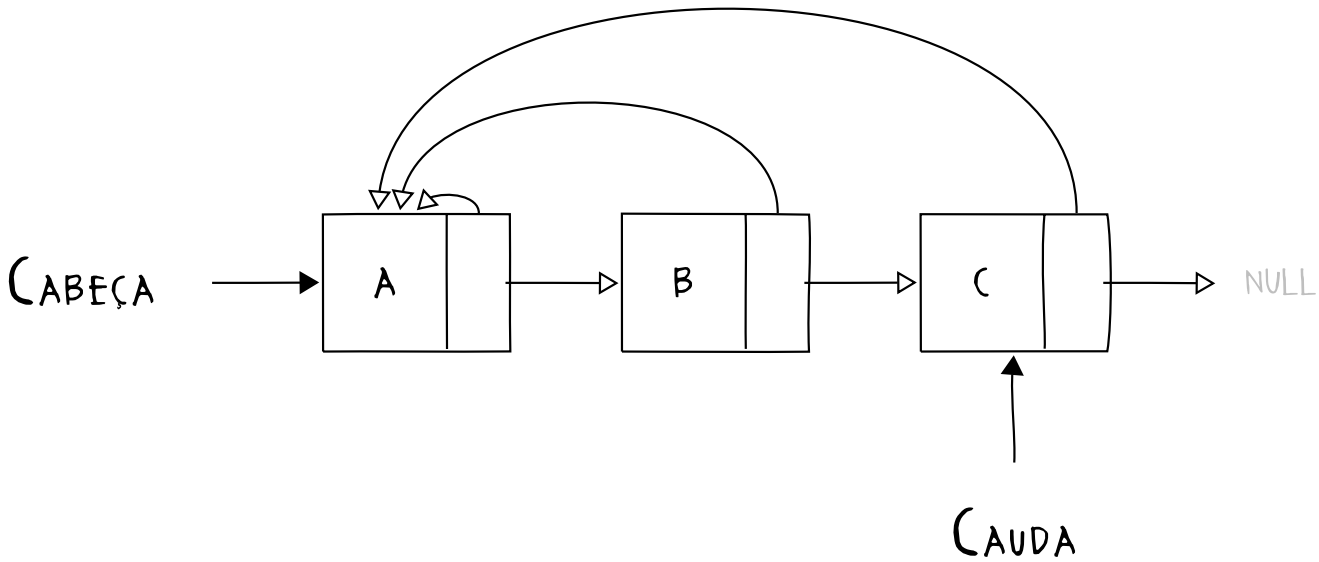
Conjuntos disjuntos

- Representação por lista encadeada
  - Unindo dois conjuntos disjuntos
  - O ajuste dos ponteiros tem custo $O(n)$



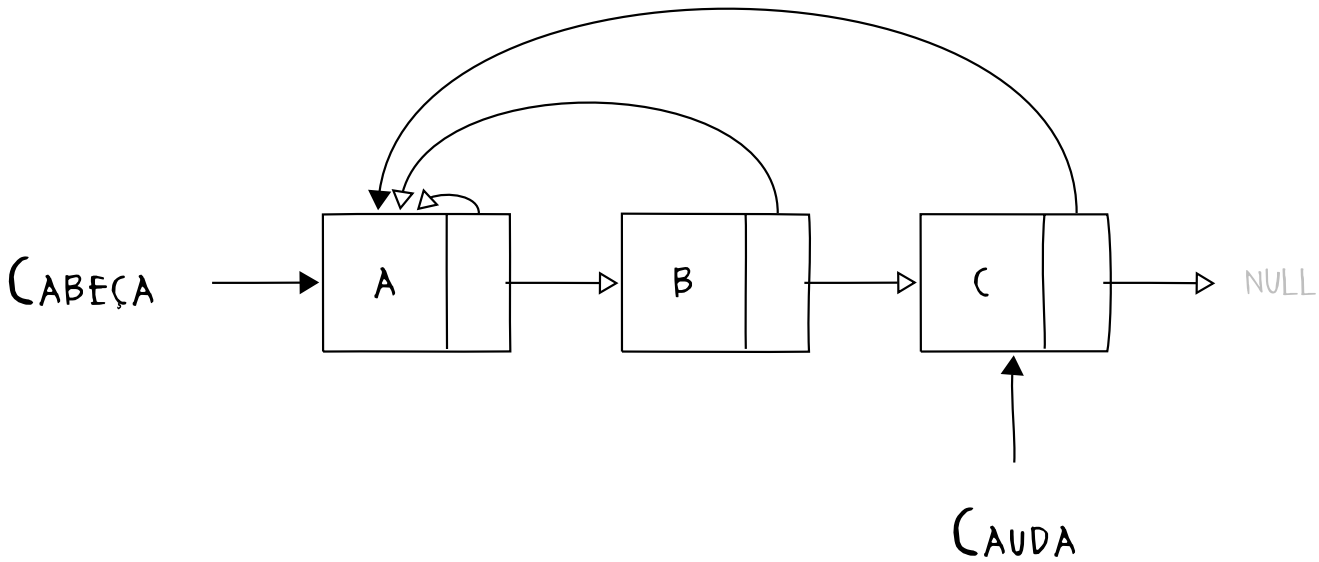
Conjuntos disjuntos

- Representação por lista encadeada
  - Buscando o conjunto do elemento *c*
  - O custo desta operação é *O*(1)



Conjuntos disjuntos

- Representação por lista encadeada
  - Buscando o conjunto do elemento *c*
  - O custo desta operação é *O*(1)



Conjuntos disjuntos

- Representação por lista encadeada
  - Considerando os conjuntos disjuntos $S = \{S_1, S_2, \ldots, S_n\}$
  - Realizando n criações e $n-1$ uniões de conjuntos

Operação	Número de atualizações
Criação(S_1)	1
Criação(S_2)	1
$\vdots$	1
Criação(S_n)	1
União(S_1, S_2)	1
União(S_2, S_3)	2
$\vdots$	$\vdots$
União(S_{n-1}, S_n)	$n-1$

Conjuntos disjuntos

- Representação por lista encadeada
  - Considerando os conjuntos disjuntos $S = \{S_1, S_2, \ldots, S_n\}$
  - Realizando n criações e $n-1$ uniões de conjuntos

Operação	Número de atualizações
Criação(S_1)	1
Criação(S_2)	1
$\vdots$	1
Criação(S_n)	1
União(S_1, S_2)	1
União(S_2, S_3)	2
$\vdots$	$\vdots$
União(S_{n-1}, S_n)	$n-1$

$$O(n + \sum_{i=1}^{n-1} i) = O(n + n^2) = O(n^2)$$

Conjuntos disjuntos

- Representação por lista encadeada
  - Para tornar a estrutura mais eficiente, é aplicada uma heurística de união ponderada
  - É feita a união do conjunto de menor tamanho com o que possui maior tamanho, atualizando no pior caso metade das referências da lista

Conjuntos disjuntos

- Representação por lista encadeada
  - Para tornar a estrutura mais eficiente, é aplicada uma heurística de união ponderada
  - É feita a união do conjunto de menor tamanho com o que possui maior tamanho, atualizando no pior caso metade das referências da lista

A união de n elementos tem custo de $O(n \log_2 n)$

Conjuntos disjuntos

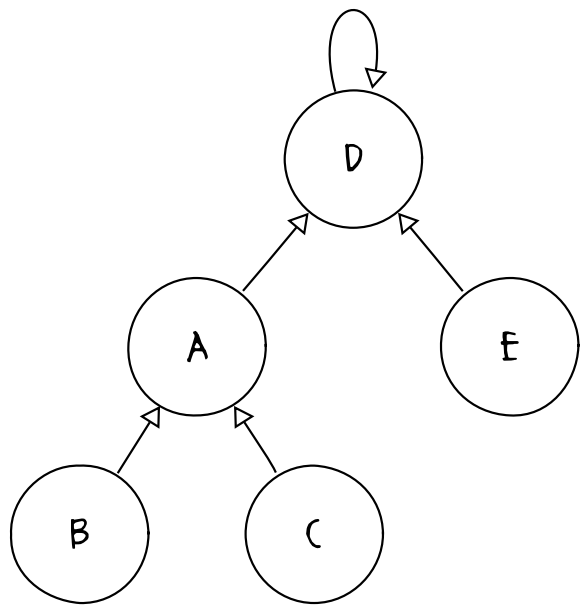
- Análise de complexidade
  - O custo para realização de m operações constantes para criação e busca de conjuntos disjuntos é $O(m)$
  - Espaço: $\Theta(n)$
  - Tempo: $\Omega(n + m)$ e $O(n \log_2 n + m)$

Exemplo

- Realize a criação de conjuntos para os elementos 1, 2, 3, 4, 5, 6, ilustrando as estruturas de lista a medida que as operações abaixo são executadas
  - União(1, 2)
  - União(3, 4)
  - União(5, 6)
  - União(1, 6)
  - Busca(5)
  - União(3, 5)

Conjuntos disjuntos

- Representação por árvore
  - Cada conjunto é representado por uma árvore enraizada criando uma floresta de conjuntos disjuntos
  - Na implementação das referências o nó possui uma referência para o elemento pai, referenciando a si mesmo quando for raiz da árvore



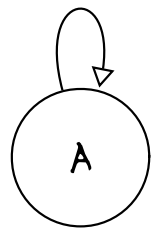
Conjuntos disjuntos

- Implementação em C
  - Estrutura do nó da árvore

```c
// Padrão de tipos por tamanho
#include <stdint.h>
// Estrutura de nó
typedef struct no {
    // Altura do nó
    uint32_t H;
    // Ponteiro para pai
    struct no* P;
} no;
```

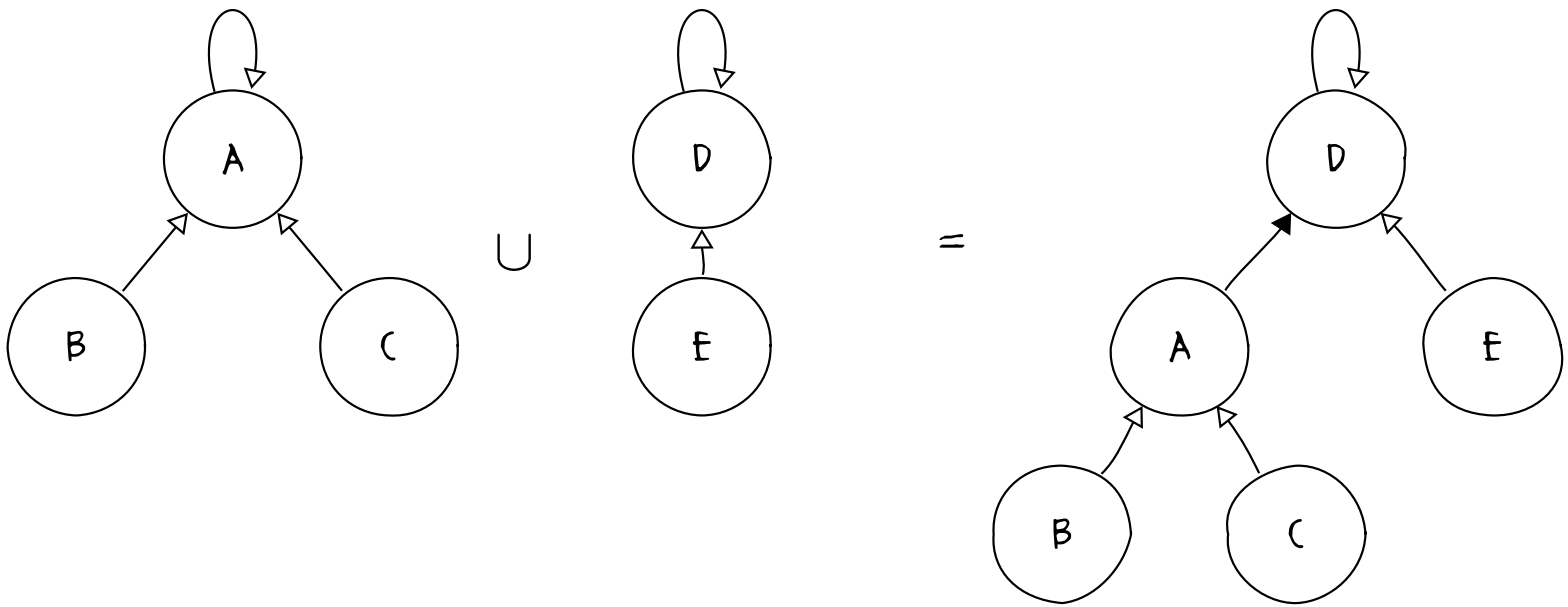
Conjuntos disjuntos

- Representação por árvore
  - Criando um conjunto para o elemento *a*
  - A criação da árvore possui custo $O(1)$



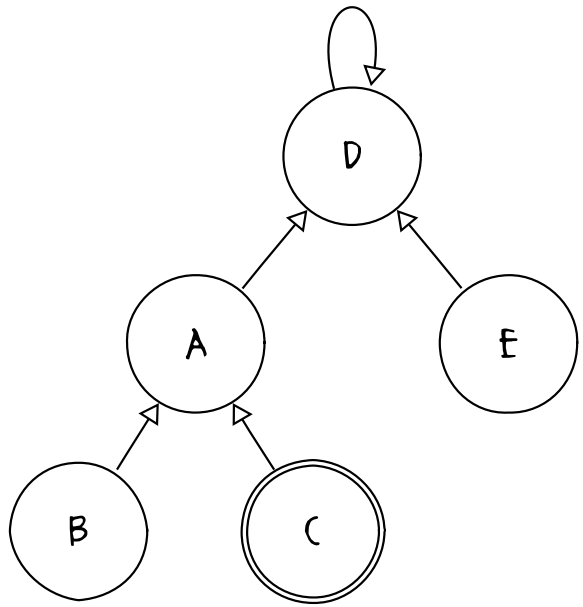
Conjuntos disjuntos

- Representação por árvore
  - Unindo dois conjuntos disjuntos
  - O ajuste dos ponteiros tem custo $O(1)$



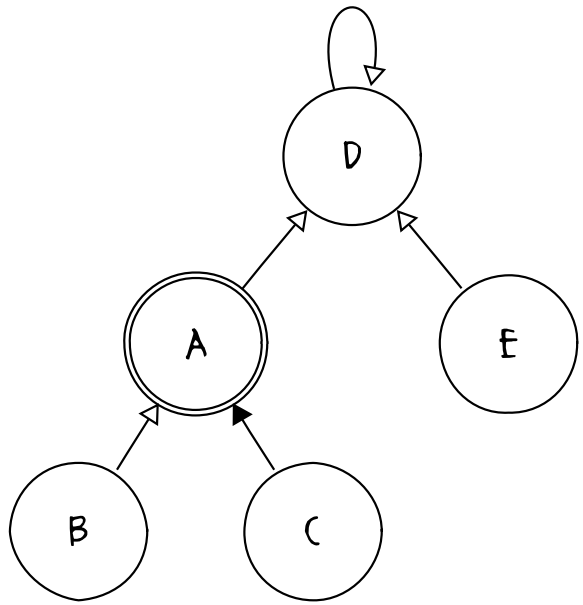
Conjuntos disjuntos

- Representação por árvore
  - Buscando o conjunto do elemento c
  - O custo desta operação é $O(h)$



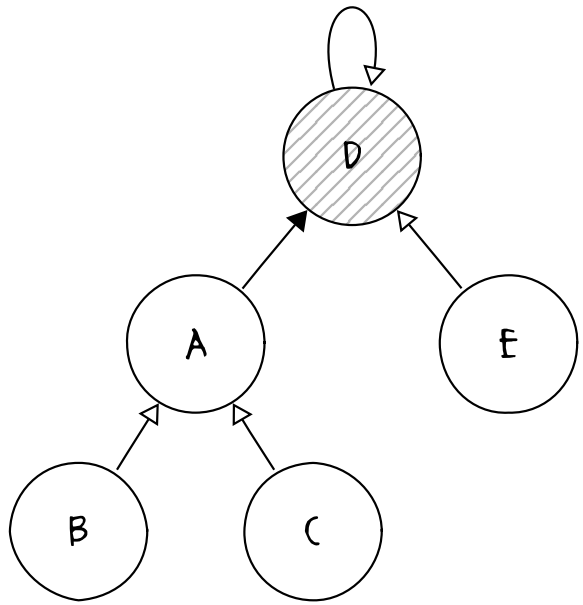
Conjuntos disjuntos

- Representação por árvore
  - Buscando o conjunto do elemento c
  - O custo desta operação é $O(h)$



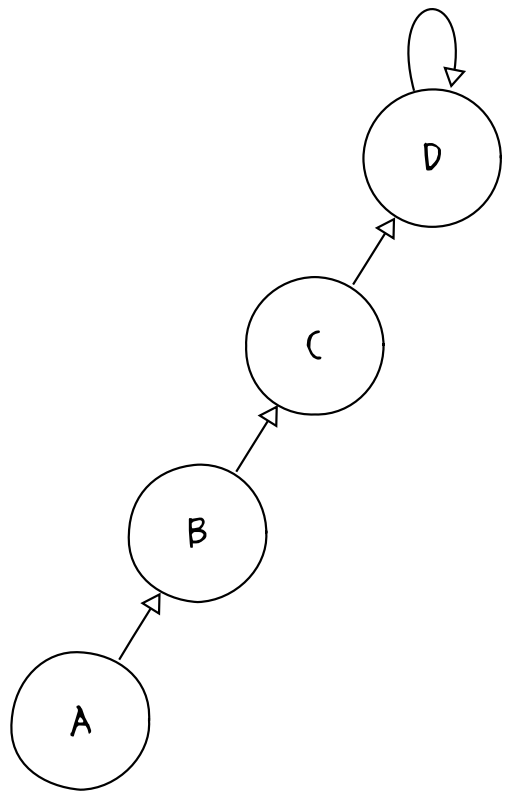
Conjuntos disjuntos

- Representação por árvore
  - Buscando o conjunto do elemento c
  - O custo desta operação é $O(h)$



Conjuntos disjuntos

- Representação por árvore
  - Considerando os conjuntos disjuntos $S = \{S_1, S_2, \ldots, S_n\}$
  - Realizando n criações e $n-1$ uniões de conjuntos
  - No pior caso, a árvore é uma lista encadeada com $O(n^2)$

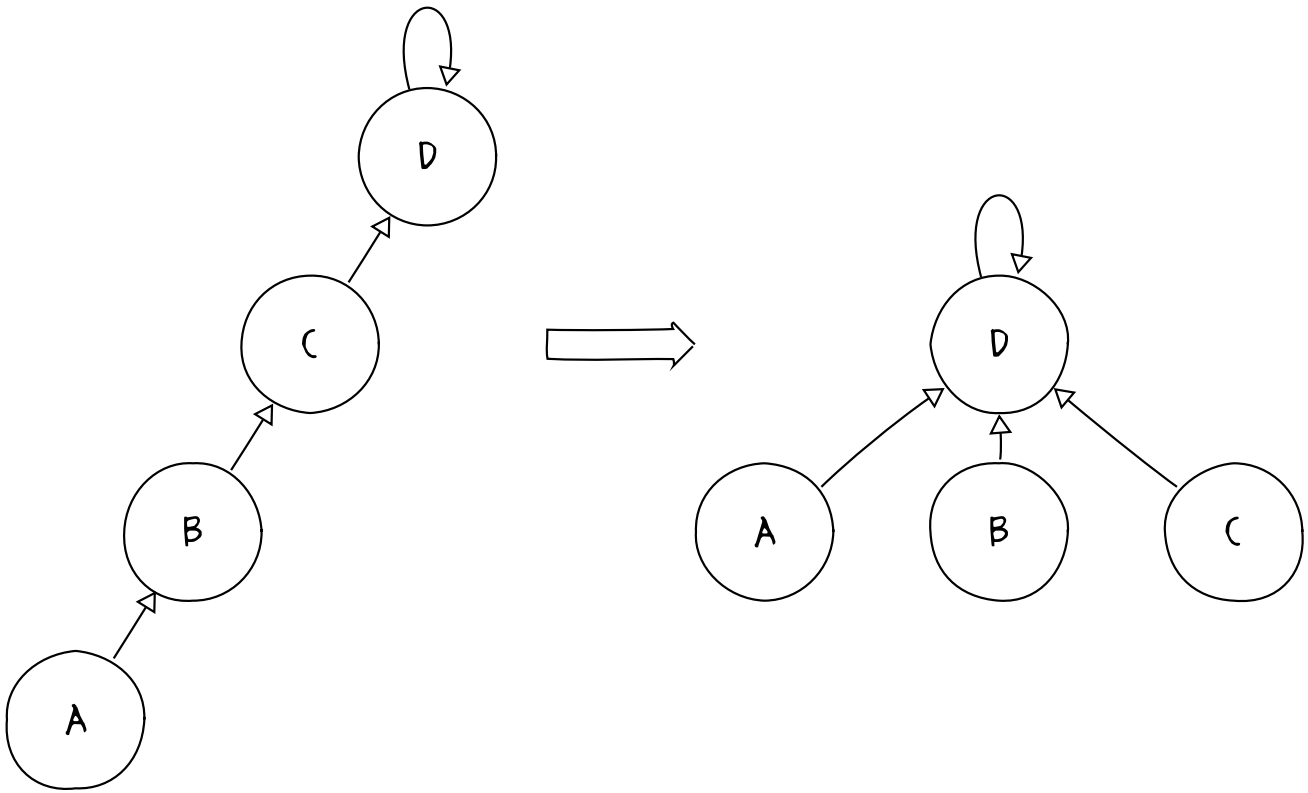


Conjuntos disjuntos

- Representação por árvore
  - Para melhorar a eficiência da estrutura é aplicada a heurística de união por classificação
  - Nesta heurística de união, cada árvore é classificada pelo número de nós que possui e a árvore com menor número de nós fará referência para a árvore com maior número de nós
  - Considerando m operações de criação e de união de conjuntos, esta heurística tem custo $O(m \log_2 n)$

Conjuntos disjuntos

- Representação por árvore
  - A heurística de compressão de caminhos permite tornar a estrutura ainda mais eficiente, fazendo que os nós referenciem diretamente a raiz da árvore
  - Não é alterada a quantidade de nós de cada árvore



Conjuntos disjuntos

- Implementação em C
  - Na criação de um conjunto, o nó referencia a si mesmo e possui altura nula

```
1 // Procedimento para criação de conjunto
2 void criar_conjunto(no* x) {
3     x->P = x;
4     x->H = 0;
5 }
```

Conjuntos disjuntos

- Implementação em C
  - Na busca pelo conjunto é feita a aplicação da heurística de compressão de caminhos

```
1 // Função para encontrar conjunto
2 no* encontrar_conjunto(no* x) {
3     if(x != x->P) {
4         x->P = encontrar_conjunto(x->P);
5     }
6     return x->P;
7 }
```

Conjuntos disjuntos

- Implementação em C
  - A união de dois conjuntos é feita pela aplicação da heurística de união por classificação

```
1  // Procedimento para união de conjuntos
2  void unir_conjuntos(no* x, no* y) {
3      no* rx = encontrar_conjunto(x);
4      no* ry = encontrar_conjunto(y);
5      if(rx->H > ry->H) {
6          ry->P = rx;
7      }
8      else {
9          rx->P = ry;
10         if(rx->H == ry->H) {
11             ry->H++;
12         }
13     }
14 }
```

Conjuntos disjuntos

- Análise de complexidade
  - Aplicando as heurísticas compressão de caminhos e de união por classificação, o custo para realização de m operações em conjuntos disjuntos é $O(m\alpha(n))$, onde $\alpha(n)$ é uma função de crescimento muito lento, tal que para valores práticos de n, o valor desta função é sempre inferior a 5
  - Espaço: $\Theta(n)$
  - Tempo: $\Theta(m)$

Exemplo

- Realize a criação de conjuntos para os elementos 1, 2, 3, 4, 5, 6, ilustrando as estruturas de árvore a medida que as operações abaixo são executadas
  - União(1, 2)
  - União(3, 4)
  - União(5, 6)
  - União(1, 6)
  - Busca(5)
  - União(3, 5)

Conjuntos disjuntos

- Aplicações
  - Árvore de extensão mínima (Kruskal)
  - Particionamento de conjuntos
  - Verificar conectividade dos nós de uma rede
  - Geração de labirintos
  - ...

Exercício

- A empresa de tecnologia Poxim Tech está desenvolvendo um sistema para simulação de propagação de doenças transmissíveis pelo ar
  - A região é definida por uma altura e uma largura, com cada coordenada sendo ocupada por uma única pessoa e com o paciente zero destacado
  - A estimativa de propagação da doença é calculada pela função $f(x, y) = (g(x), g(y))$, onde $g(z) = z + (-1 + myrand() \bmod 3)$, sendo calculada primeiro para x e depois para y dentro dos limites da região definida, com recálculo da coordenada caso já possua uma pessoa infectada na posição gerada

```
1 // Função myrand
2 uint32_t myrand() {
3     static uint32_t next = 1;
4     next = next * 1103515245 + 12345;
5     return next;
6 }
```

Exercício

- Formato de arquivo de entrada
  - [$Regiões(n)$]
  - [$Altura_1$] [$Largura_1$] [x_1] [y_1]
  - ⋮
  - [$Altura_n$] [$Largura_n$] [x_n] [y_n]

```
1 3
2 2␣2␣0␣1
3 5␣5␣2␣3
4 8␣8␣4␣5
```

Exercício

- Formato de arquivo de saída
  - Propagação de doenças entre as pessoas

```
1 1:(0,1);(1,1);(0,0);(1,0)
2 2:(2,3);(3,3);(3,4);(4,4);(4,3);(4,2);(3,1);(3,2)
      ;(2,4);(1,3);(0,3);(1,4);(0,4);(2,1);(2,2);(1,2)
      ;(1,1);(0,0);(0,1);(4,1);(2,0);(3,0);(0,2);(4,0)
      ;(1,0)
3 3:(4,5);(5,5);(6,6);(7,7);(7,6);(6,7);(5,7);(6,5)
      ;(5,6);(7,5);(7,4);(6,3);(7,2);(7,1);(6,0);(5,1)
      ;(6,2);(5,0);(6,1);(7,0);(5,2);(4,2);(4,1);(3,1)
      ;(3,0);(2,0);(2,1);(5,3);(5,4);(6,4);(7,3);(3,3)
      ;(4,3);(4,4);(3,2);(2,2);(3,4);(2,4);(2,5);(3,5)
      ;(3,6);(2,6);(1,7);(2,7);(1,6);(2,3);(0,6);(4,7)
      ;(1,5);(1,4);(0,4);(0,3);(1,3);(0,7);(0,2);(1,2)
      ;(1,1);(4,0);(0,5);(4,6);(3,7);(0,1);(0,0);(1,0)
```