

# Análise e Síntese de Algoritmos

Estruturas de Dados para Conjuntos Disjuntos [CLRS, Cap. 21]

2011/2012

# Contexto

- Revisão [CLRS, Cap.1-13]
  - Fundamentos; notação; exemplos
- Algoritmos em Grafos [CLRS, Cap.21-26]
  - Algoritmos elementares
  - Árvores abrangentes
  - Caminhos mais curtos
  - Fluxos máximos
- Programação Linear [CLRS, Cap.29]
  - Algoritmos e modelação de problemas com restrições lineares
- Técnicas de Síntese de Algoritmos [CLRS, Cap.15-16]
  - Programação dinâmica
  - Algoritmos greedy
- Tópicos Adicionais [CLRS, Cap.32-35]
  - Emparelhamento de Cadeias de Caracteres
  - Complexidade Computacional
  - Algoritmos de Aproximação

# Resumo

- 1 Definições
  - Estrutura de dados para conjuntos disjuntos
  - Operações
- 2 Aplicações
- 3 Estruturas Baseadas em Listas
- 4 Estruturas Baseadas em Árvores



# Estrutura de Dados para Conjuntos Disjuntos

## Estrutura de Dados para Conjuntos Disjuntos

Permite manter uma coleção de **conjuntos dinâmicos disjuntos**

- Cada conjunto caracterizado por **representante**, um elemento do conjunto
- Representante não alterado devido a consultas à estrutura de dados



# Operações sobre Conjuntos Disjuntos

## Operações sobre Conjuntos Disjuntos

Cada elemento da estrutura é representado por objecto  $x$

- **Make-Set( $x$ )**
  - Cria novo conjunto que apenas inclui elemento apontado por  $x$
  - $x$  aponta para único elemento do conjunto, o representante do conjunto
- **Union( $x, y$ )**
  - Realiza a união dos conjuntos que contém  $x$  e  $y$ , respectivamente  $S_x$  e  $S_y$ 
    - Novo conjunto criado:  $S_x \cup S_y$
    - $S_x$  e  $S_y$  eliminados (conjuntos disjuntos)
    - Novo representante será o representante de  $S_x$  ou  $S_y$
- **Find-Set( $x$ )**
  - Retorna apontador para representante do conjunto que contém  $x$

# Exemplo Aplicação

## Problema

Considere que está na equipa de projecto de uma rede de distribuição entre um conjunto de cidades. Foram efectuados estudos que calcularam o custo  $c(u, v)$  associado a cada ligação possível da nova rede. Pretende-se saber qual o menor custo total de uma rede que interligue todas as cidades.

# Exemplo Aplicação

## Problema

Considere que está na equipa de projecto de uma rede de distribuição entre um conjunto de cidades. Foram efectuados estudos que calcularam o custo  $c(u, v)$  associado a cada ligação possível da nova rede. Pretende-se saber qual o menor custo total de uma rede que interligue todas as cidades.

## Solução

- Representar a rede como um grafo não-dirigido e pesado
- Função de pesos é definida pelo custo entre as possíveis ligações
- Rede de menor custo será a Árvore Abrangente de Menor Custo (MST) do grafo

# Exemplo Aplicação

## Algoritmo Kruskal (Cap. 23)

- **Make-Set:** Cria cada conjunto (disjunto) que representa conjunto de vértices
- **Find-Set:** Identifica conjunto a que pertence um dado vértice
- **Union:** Coloca conjuntos de vértices num mesmo conjunto



# Exemplo Aplicação

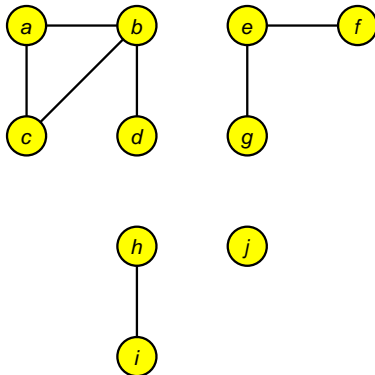
Identificar os componentes ligados de um grafo não dirigido  $G = (V, E)$

## Connected-Components( $G$ )

```
1  for each  $v \in V[G]$ 
2    do Make-Set( $v$ )
3  for each  $(u, v) \in E[G]$ 
4    do if Find-Set( $u$ )  $\neq$  Find-Set( $v$ )
5       then Union( $u, v$ )
```

## Same-Component( $u, v$ )

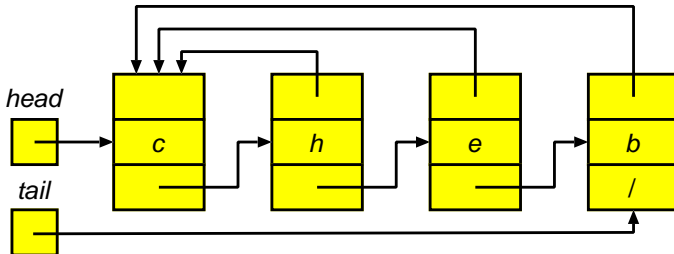
```
1  if Find-Set( $u$ )  $\neq$  Find-Set( $v$ )
2    then return FALSE
3  else return TRUE
```



# Utilização de Lista Ligada

## Organização

- Elementos de cada conjunto em lista (simplesmente) ligada
- Primeiro elemento é o representante do conjunto
- Todos os elementos incluem apontador para o representante do conjunto



# Utilização de Lista Ligada

## Tempos de Execução

- Union( $x, y$ ):
  - Colocar elementos de  $x$  no fim da lista de  $y$
  - Actualizar ponteiros de elementos de  $x$  para representante
- Operações sobre  $n$  elementos  $x_1, x_2, \dots, x_n$ 
  - $n$  operações Make-Set( $x_i$ )
    - $\Theta(n)$
  - $n - 1$  operações Union( $x_{i-1}, x_i$ ), para  $i = 2, \dots, n$ 
    - Cada operação Union( $x_{i-1}, x_i$ ) actualiza  $i - 1$  elementos
    - Custo das  $n - 1$  operações:  $\sum_{i=1}^{n-1} i = \Theta(n^2)$
  - Número total de operações é  $m = 2n - 1$  logo, em média, cada operação requer tempo  $\Theta(n)$

# Utilização de Lista Ligada

## Heurística União Pesada

- A cada conjunto associar o número de elementos
- Para cada operação Union, juntar lista com menor número de elementos à lista com maior número de elementos
  - Necessário actualizar menor número de ponteiros para representante
- Custo total de  $m$  operações é melhorado

# Utilização de Lista Ligada

## Heurística União Pesada

- A cada conjunto associar o número de elementos
- Para cada operação Union, juntar lista com menor número de elementos à lista com maior número de elementos
  - Necessário actualizar menor número de ponteiros para representante
- Custo total de  $m$  operações é melhorado
- Sequência de  $m$  operações (que incluem  $n$  operações Union) é:  
 $O(m + n \lg n)$
- Prova:
  - Para cada objecto num conjunto com  $n$  elementos, calcular limite superior do número de vezes que ponteiro para representante é actualizado

# Utilização de Lista Ligada

## Tempos de Execução (com Heurística)

- Sempre que o ponteiro para o representante  $x$  é actualizado,  $x$  encontra-se no conjunto com menor número de elementos
  - Da 1ª vez, conjunto resultante com pelo menos 2 elementos
  - Da 2ª vez, conjunto resultante com pelo menos 4 elementos
  - ...
  - Após representante de  $x$  ter sido actualizado  $\lceil \lg k \rceil$  vezes, conjunto resultante tem pelo menos  $k$  elementos
- Maior conjunto tem  $n$  elementos
  - Cada ponteiro actualizado não mais do que  $\lceil \lg n \rceil$  vezes
- Tempo total para actualizar  $n$  elementos é  $O(n \lg n)$
- Make-Set e Find-Set têm tempos de execução  $O(1)$  e há  $O(m)$  destas operações
- Tempo total para  $m$  operações (com  $n$  Union) é  $O(m + n \lg n)$

# Utilização de Árvores

## Organização

- Cada conjunto representado por uma árvore
- Cada elemento aponta apenas para antecessor na árvore
- Representante da árvore é a raiz
- Antecessor da raiz é a própria raiz

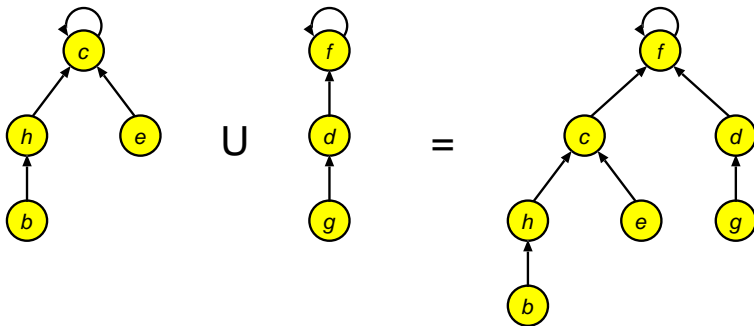
## Operações

- Find-Set: Percorrer antecessores até raiz ser encontrada
- Union: Raiz de uma árvore aponta para raiz da outra árvore

## Complexidade

- Sequência de  $O(m)$  operações é  $O(m n)$
- Pior caso ocorre quando as árvores que são apenas listas dos  $n$  elementos

# Utilização de Árvores





# Utilização de Árvores

## Heurística - União por Categoria

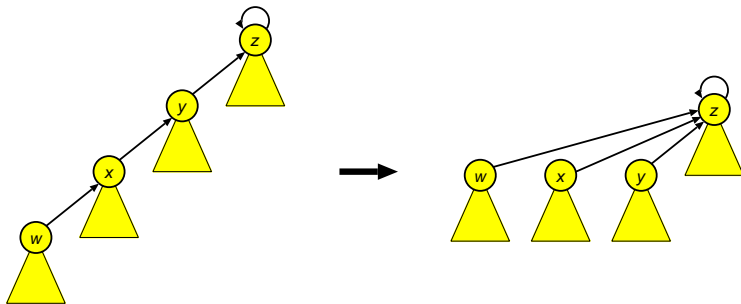
Numa união de dois conjuntos, colocar árvore com menos elementos a apontar para árvore com mais elementos

- Utilizar estimativa do número de elementos em cada sub-árvore
- categoria (*rank*): aproxima logaritmo do tamanho da sub-árvore e é um limite superior na altura da sub-árvore
- Numa união, raiz da árvore com menor *rank* aponta para raiz da árvore com maior *rank*

# Utilização de Árvores

## Heurística - Compressão de Caminhos

Em cada operação Find-Set coloca cada nó visitado a apontar directamente para a raiz da árvore (representante do conjunto)



# Utilização de Árvores

## Make-Set( $x$ )

```
1  $p[x] \leftarrow x$   
2  $rank[x] \leftarrow 0$ 
```

## Find-Set( $x$ )

```
1 if  $x \neq p[x]$   
2   then  $p[x] \leftarrow \text{Find-Set}(p[x])$   
3 return  $p[x]$ 
```

## Union( $x, y$ )

```
1  $\text{Link}(\text{Find-Set}(x), \text{Find-Set}(y))$ 
```

## Link( $x, y$ )

```
1 if  $rank[x] > rank[y]$   
2   then  $p[y] \leftarrow x$   
3   else  $p[x] \leftarrow y$   
4       if  $rank[x] = rank[y]$   
5         then  $rank[y] \leftarrow rank[y] + 1$ 
```

# Utilização de Árvores

## Complexidade

Execução de  $m$  operações sobre  $n$  elementos:  $O(m \alpha(n))$

Onde  $\alpha(n) = \min\{k : A_k(1) \geq n\}$

$$A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$\begin{cases} A_{k-1}^{(0)}(j) & = j \\ A_{k-1}^{(i)}(j) & = A_{k-1}(A_{k-1}^{(i-1)}(j)) \end{cases}$$

$$A_1(j) = 2 * j + 1$$

$$A_2(j) = 2^{j+1}(j+1) - 1$$

# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\}$$

$$A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\}$$

$$A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

$$A_1(3) = A_0^{(4)}(1) = A_0(A_0(A_0(A_0(3)))) = 7$$

# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\}$$

$$A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

$$A_1(3) = A_0^{(4)}(1) = A_0(A_0(A_0(A_0(3)))) = 7$$

$$A_2(1) = A_1^{(2)}(1) = A_1(A_1(1)) = A_1(3) = 7$$

# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\}$$

$$A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

$$A_1(3) = A_0^{(4)}(1) = A_0(A_0(A_0(A_0(3)))) = 7$$

$$A_2(1) = A_1^{(2)}(1) = A_1(A_1(1)) = A_1(3) = 7$$

$$A_3(1) = A_2^{(2)}(1) = A_2(A_2(1)) = A_2(7) = 2^8 * 8 - 1 = 2047$$



# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\} \qquad A_k(j) = \begin{cases} j+1 & \text{se } k = 0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

$$A_1(3) = A_0^{(4)}(1) = A_0(A_0(A_0(A_0(3)))) = 7$$

$$A_2(1) = A_1^{(2)}(1) = A_1(A_1(1)) = A_1(3) = 7$$

$$A_3(1) = A_2^{(2)}(1) = A_2(A_2(1)) = A_2(7) = 2^8 * 8 - 1 = 2047$$

$$A_4(1) = A_3^{(2)}(1) = A_3(A_3(1)) = A_3(2047) = A_2^{(2048)}(2047) \gg A_2(2047) \gg 10^{80}$$

# Utilização de Árvores

$$\alpha(n) = \min\{k : A_k(1) \geq n\} \qquad A_k(j) = \begin{cases} j+1 & \text{se } k=0 \\ A_{k-1}^{(j+1)}(j) & \text{se } k \geq 1 \end{cases}$$

$$A_1(1) = A_0^{(2)}(1) = A_0(A_0(1)) = A_0(2) = 3$$

$$A_1(3) = A_0^{(4)}(1) = A_0(A_0(A_0(A_0(3)))) = 7$$

$$A_2(1) = A_1^{(2)}(1) = A_1(A_1(1)) = A_1(3) = 7$$

$$A_3(1) = A_2^{(2)}(1) = A_2(A_2(1)) = A_2(7) = 2^8 * 8 - 1 = 2047$$

$$A_4(1) = A_3^{(2)}(1) = A_3(A_3(1)) = A_3(2047) = A_2^{(2048)}(2047) \gg A_2(2047) \gg 10^{80}$$

$$\alpha(n) = \begin{cases} 0 & \text{se } 0 \leq n \leq 2 \\ 1 & \text{se } n = 3 \\ 2 & \text{se } 4 \leq n \leq 7 \\ 3 & \text{se } 8 \leq n \leq 2047 \\ 4 & \text{se } 2048 \leq n \leq A_4(1) \end{cases}$$