

Análise de Algoritmos

**Parte destes slides são adaptações de slides
do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.**

Análise do Union-Find

CLRS cap 21

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma **partição de um conjunto**, e as seguintes operações sobre a partição:

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma **partição de um conjunto**, e as seguintes operações sobre a partição:

- **MAKESET**(x): cria um conjunto unitário com o elemento x ;
- **FINDSET**(x): devolve o identificador do conjunto da partição que contém x ;
- **UNION**(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma **partição de um conjunto**, e as seguintes operações sobre a partição:

- **MAKESET**(x): cria um conjunto unitário com o elemento x ;
- **FINDSET**(x): devolve o identificador do conjunto da partição que contém x ;
- **UNION**(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

O **identificador de um conjunto** é um elemento do conjunto: o **seu representante**.

Coleção de conjuntos disjuntos

Queremos uma ED boa para representar uma **partição de um conjunto**, e as seguintes operações sobre a partição:

- **MAKESET**(x): cria um conjunto unitário com o elemento x ;
- **FINDSET**(x): devolve o identificador do conjunto da partição que contém x ;
- **UNION**(x, y): substitui os conjuntos da partição que contêm x e y pela união deles.

O **identificador de um conjunto** é um elemento do conjunto: o **seu representante**.

Como podemos armazenar cada conjunto da partição?

Implementação 1 do union-find

Make-Set (x)

1 $\text{pai}[x] \leftarrow x$

Implementação 1 do union-find

Make-Set (x)

1 $\text{pai}[x] \leftarrow x$

Find (x)

1 $r \leftarrow x$

2 **enquanto** $\text{pai}[r] \neq r$ **faça**

3 $r \leftarrow \text{pai}[r]$

4 **devolva** r

Implementação 1 do union-find

Make-Set (x)

1 **pai**[x] $\leftarrow x$

Find (x)

1 $r \leftarrow x$

2 **enquanto** **pai**[r] $\neq r$ **faça**

3 $r \leftarrow$ **pai**[r]

4 **devolva** r

Union (x, y)

$\triangleright x$ e y representantes distintos

1 **pai**[y] $\leftarrow x$

Implementação 1 do union-find

Make-Set (x)

1 $\text{pai}[x] \leftarrow x$

Find (x)

1 $r \leftarrow x$

2 **enquanto** $\text{pai}[r] \neq r$ **faça**

3 $r \leftarrow \text{pai}[r]$

4 **devolva** r

Union (x, y) $\triangleright x$ e y representantes distintos

1 $\text{pai}[y] \leftarrow x$

Consumo de tempo: do **Find** pode ser muito ruim... $\Theta(n)$.

Temos que fazer melhor...

Implementação 2

Heurística dos tamanhos

Make-Set (x)

1 $\text{pai}[x] \leftarrow x$

2 $\text{rank}[x] \leftarrow 0$

Implementação 2

Heurística dos tamanhos

Make-Set (x)

1 **pai** $[x] \leftarrow x$

2 **rank** $[x] \leftarrow 0$

Find (x): o mesmo de antes

Implementação 2

Heurística dos tamanhos

Make-Set (x)

1 **pai**[x] $\leftarrow x$

2 **rank**[x] $\leftarrow 0$

Find (x): o mesmo de antes

Union (x, y) $\triangleright x$ e y representantes distintos

1 **se** **rank**[x] \geq **rank**[y]

2 **então** **pai**[y] $\leftarrow x$

3 **se** **rank**[x] = **rank**[y]

4 **então** **rank**[x] \leftarrow **rank**[x] + 1

5 **senão** **pai**[x] $\leftarrow y$

Implementação 2

Heurística dos tamanhos

Make-Set (x)

1 **pai**[x] $\leftarrow x$

2 **rank**[x] $\leftarrow 0$

Find (x): o mesmo de antes

Union (x, y) $\triangleright x$ e y representantes distintos

1 **se** **rank**[x] \geq **rank**[y]

2 **então** **pai**[y] $\leftarrow x$

3 **se** **rank**[x] = **rank**[y]

4 **então** **rank**[x] \leftarrow **rank**[x] + 1

5 **senão** **pai**[x] $\leftarrow y$

Consumo de tempo: melhor... $\Theta(\lg n)$.

Dá para fazer melhor ainda!

Implementação 3

Heurística da compressão dos caminhos

Find (x)

1 **if** $\text{pai}[x] \neq x$

2 **então** $\text{pai}[x] \leftarrow \text{Find}(\text{pai}[x])$

3 **devolva** $\text{pai}[x]$

Implementação 3

Heurística da compressão dos caminhos

Find (x)

1 **if** $\text{pai}[x] \neq x$

2 **então** $\text{pai}[x] \leftarrow \text{Find}(\text{pai}[x])$

3 **devolva** $\text{pai}[x]$

Consumo *amortizado* de tempo de cada operação:

$$O(\lg^* n),$$

onde $\lg^* n$ é o número de vezes que temos que aplicar o \lg até atingir um número menor ou igual a 1.

Implementação 3

Heurística da compressão dos caminhos

Find (x)

1 **if** $\text{pai}[x] \neq x$

2 **então** $\text{pai}[x] \leftarrow \text{Find}(\text{pai}[x])$

3 **devolva** $\text{pai}[x]$

Consumo *amortizado* de tempo de cada operação:

$$O(\lg^* n),$$

onde $\lg^* n$ é o número de vezes que temos que aplicar o \lg até atingir um número menor ou igual a 1.

Na verdade, é melhor do que isso,
e há uma análise justa, conforme discutido em aula.

Union-Find

Make-Set (x)

- 1 **pai**[x] $\leftarrow x$
- 2 **rank**[x] $\leftarrow 0$

Find (x)

- 1 **if** **pai**[x] $\neq x$
- 2 **então** **pai**[x] \leftarrow **Find** (**pai**[x])
- 3 **devolva** **pai**[x]

Union (x, y) $\triangleright x$ e y representantes distintos

- 1 **se** **rank**[x] \geq **rank**[y]
- 2 **então** **pai**[y] $\leftarrow x$
- 3 **se** **rank**[x] = **rank**[y]
- 4 **então** **rank**[x] \leftarrow **rank**[x] + 1
- 5 **senão** **pai**[x] $\leftarrow y$

Union-Find

Union (x, y)

```
1   $x' \leftarrow \text{Find}(x)$ 
2   $y' \leftarrow \text{Find}(y)$ 
3  se  $x' \neq y'$ 
4      então Link ( $x', y'$ )
```

Link (x, y) $\triangleright x$ e y representantes distintos

```
1  se  $\text{rank}[x] \geq \text{rank}[y]$ 
2      então  $\text{pai}[y] \leftarrow x$ 
3          se  $\text{rank}[x] = \text{rank}[y]$ 
4              então  $\text{rank}[x] \leftarrow \text{rank}[x] + 1$ 
5  senão  $\text{pai}[x] \leftarrow y$ 
```

Análise

Dada sequência de MAKESET, FINDSET e UNION,
converta-a em uma sequência de MAKESET, FINDSET e LINK.

Análise

Dada sequência de MAKESET, FINDSET e UNION,
converta-a em uma sequência de MAKESET, FINDSET e LINK.

Sequência de m operações MAKESET, FINDSET e LINK
das quais n são MAKESET.

Análise

Dada sequência de MAKESET, FINDSET e UNION, converta-a em uma sequência de MAKESET, FINDSET e LINK.

Sequência de m operações MAKESET, FINDSET e LINK das quais n são MAKESET.

Custo amortizado de cada operação: $O(\lg^* n)$.

Análise

Dada sequência de MAKESET, FINDSET e UNION, converta-a em uma sequência de MAKESET, FINDSET e LINK.

Sequência de m operações MAKESET, FINDSET e LINK das quais n são MAKESET.

Custo amortizado de cada operação: $O(\lg^* n)$.

Seja $\lg^{(1)} x = \lg x$.

Para $i \geq 2$, seja $\lg^{(i)} x = \lg(\lg^{(i-1)} x)$.

Análise

Dada sequência de MAKESET, FINDSET e UNION, converta-a em uma sequência de MAKESET, FINDSET e LINK.

Sequência de m operações MAKESET, FINDSET e LINK das quais n são MAKESET.

Custo amortizado de cada operação: $O(\lg^* n)$.

Seja $\lg^{(1)} x = \lg x$.

Para $i \geq 2$, seja $\lg^{(i)} x = \lg(\lg^{(i-1)} x)$.

A função $\lg^* n$ é definida da seguinte maneira:

$$\lg^* n = \min\{i : \lg^{(i)} n \leq 1\}.$$

Análise: a função potencial

Para cada nó x da floresta, o **grupo** de x é o conjunto

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}.$$

Análise: a função potencial

Para cada nó x da floresta, o **grupo** de x é o conjunto

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}.$$

Função potencial:

$$\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$$

Análise: a função potencial

Para cada nó x da floresta, o **grupo** de x é o conjunto

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}.$$

Função potencial:

$$\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$$

$\Phi \geq 0$ e o valor inicial $\Phi_0 = 0$.

(não há nenhum conjunto na coleção inicialmente)

Análise: a função potencial

Para cada nó x da floresta, o **grupo** de x é o conjunto

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}.$$

Função potencial:

$$\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$$

$\Phi \geq 0$ e o valor inicial $\Phi_0 = 0$.

(não há nenhum conjunto na coleção inicialmente)

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

Análise: a função potencial

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$$

Função potencial: $\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

Análise: a função potencial

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$$

Função potencial: $\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

O custo amortizado de cada operação é:

● MAKESET(x): $\hat{c} = c + \Phi_d - \Phi_a = 1 + 1 = 2.$

Análise: a função potencial

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$$

Função potencial: $\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

O custo amortizado de cada operação é:

● MAKESET(x): $\hat{c} = c + \Phi_d - \Phi_a = 1 + 1 = 2.$

● LINK(x, y): $\hat{c} = c + \Phi_d - \Phi_a \leq 1.$

Análise: a função potencial

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$$

Função potencial: $\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

O custo amortizado de cada operação é:

● MAKESET(x): $\hat{c} = c + \Phi_d - \Phi_a = 1 + 1 = 2.$

● LINK(x, y): $\hat{c} = c + \Phi_d - \Phi_a \leq 1.$

De fato, $c = 1$ e $\Phi_d - \Phi_a \leq 0$:

pai é alterado para nó que contava no potencial.

se nó deixa de contar, $\Phi_d - \Phi_a = -1$, se continua contando, $\Phi_d - \Phi_a = 0$.

Análise: a função potencial

$$G(x) = \{y : \lg^*(\text{rank}[y]) = \lg^*(\text{rank}[x])\}$$

Função potencial: $\Phi = |\{x : G(x) = G(\text{pai}[x])\}|$

Φ_d : função potencial **depois** da operação

Φ_a : função potencial **antes** da operação

O custo amortizado de cada operação é:

● MAKESET(x): $\hat{c} = c + \Phi_d - \Phi_a = 1 + 1 = 2.$

● LINK(x, y): $\hat{c} = c + \Phi_d - \Phi_a \leq 1.$

De fato, $c = 1$ e $\Phi_d - \Phi_a \leq 0$:

pai é alterado para nó que contava no potencial.

se nó deixa de contar, $\Phi_d - \Phi_a = -1$, se continua contando, $\Phi_d - \Phi_a = 0$.

Incremento de $\text{rank}[y]$ pode fazer o potencial diminuir.

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$

$k = \text{comprimento do caminho } P \text{ de } x \text{ a } z$

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$

$k =$ comprimento do caminho P de x a z

Então $c = k$ e $\hat{c} = k + \Phi_d - \Phi_a$.

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$

$k =$ comprimento do caminho P de x a z

Então $c = k$ e $\hat{c} = k + \Phi_d - \Phi_a$.

Quanto vale $\Phi_d - \Phi_a$?

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$

$k =$ comprimento do caminho P de x a z

Então $c = k$ e $\hat{c} = k + \Phi_d - \Phi_a$.

Quanto vale $\Phi_d - \Phi_a$?

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$

$k =$ comprimento do caminho P de x a z

Então $c = k$ e $\hat{c} = k + \Phi_d - \Phi_a$.

Quanto vale $\Phi_d - \Phi_a$?

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

- **vermelha** se $G(\text{pai}[u]) = G(u) = G(r)$,
onde r é a raiz da árvore em que u se encontra.
- **azul** se $G(\text{pai}[u]) \neq G(u)$.
- **amarela** se $G(\text{pai}[u]) = G(u) \neq G(r)$.

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$ $k =$ comprimento do caminho P de x a z

$$\hat{c} = k + \Phi_d - \Phi_a.$$

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

- **vermelha** se $G(\text{pai}[u]) = G(u) = G(r)$,
onde r é a raiz da árvore em que u se encontra.
- **azul** se $G(\text{pai}[u]) \neq G(u)$.
- **amarela** se $G(\text{pai}[u]) = G(u) \neq G(r)$.

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$ $k =$ comprimento do caminho P de x a z

$$\hat{c} = k + \Phi_d - \Phi_a.$$

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

- **vermelha** se $G(\text{pai}[u]) = G(u) = G(r)$,
onde r é a raiz da árvore em que u se encontra.
- **azul** se $G(\text{pai}[u]) \neq G(u)$.
- **amarela** se $G(\text{pai}[u]) = G(u) \neq G(r)$.

Ao final de $\text{FINDSET}(x)$, nós de P têm z como **pai**.

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$ $k =$ comprimento do caminho P de x a z

$$\hat{c} = k + \Phi_d - \Phi_a.$$

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

- **vermelha** se $G(\text{pai}[u]) = G(u) = G(r)$,
onde r é a raiz da árvore em que u se encontra.
- **azul** se $G(\text{pai}[u]) \neq G(u)$.
- **amarela** se $G(\text{pai}[u]) = G(u) \neq G(r)$.

Ao final de $\text{FINDSET}(x)$, nós de P têm z como **pai**.

Φ decresce do número de **arestas amarelas** em P .

Análise do FINDSET(x)

$z = \text{FINDSET}(x)$ $k =$ comprimento do caminho P de x a z

$$\hat{c} = k + \Phi_d - \Phi_a.$$

Partição das arestas da floresta:

para cada vértice u , a aresta entre u e $\text{pai}[u]$ é

- **vermelha** se $G(\text{pai}[u]) = G(u) = G(r)$,
onde r é a raiz da árvore em que u se encontra.
- **azul** se $G(\text{pai}[u]) \neq G(u)$.
- **amarela** se $G(\text{pai}[u]) = G(u) \neq G(r)$.

Ao final de $\text{FINDSET}(x)$, nós de P têm z como **pai**.

Φ decresce do número de **arestas amarelas** em P .

Então $\hat{c} = v + a$, onde v é o número de **arestas vermelhas** e
 a é o número de **arestas azuis** em P .

Análise do $\text{FINDSET}(x)$

rank cresce estritamente ao subirmos nas árvores.

Análise do $\text{FINDSET}(x)$

rank cresce estritamente ao subirmos nas árvores.

Aresta azul: passamos de um grupo para outro.

Análise do FINDSET(x)

rank cresce estritamente ao subirmos nas árvores.

Aresta azul: passamos de um grupo para outro.

Número de **arestas azuis** no caminho de qualquer nó a uma raiz é no máximo o número de grupos, que é $\leq 1 + \lg^* n$.

Análise do $\text{FINDSET}(x)$

rank cresce estritamente ao subirmos nas árvores.

Aresta azul: passamos de um grupo para outro.

Número de **arestas azuis** no caminho de qualquer nó a uma raiz é no máximo o número de grupos, que é $\leq 1 + \lg^* n$.

rank assume valores entre 0 e $\lg n$,
assim os grupos vão de 0 a $\lg^*(\lg n) \leq \lg^* n$.

Análise do FINDSET(x)

rank cresce estritamente ao subirmos nas árvores.

Aresta azul: passamos de um grupo para outro.

Número de **arestas azuis** no caminho de qualquer nó a uma raiz é no máximo o número de grupos, que é $\leq 1 + \lg^* n$.

rank assume valores entre 0 e $\lg n$,
assim os grupos vão de 0 a $\lg^*(\lg n) \leq \lg^* n$.

Ou seja, $a \leq 1 + \lg^* n$
e o custo amortizado do FINDSET fica

$$\hat{c} = v + a \leq v + 1 + \lg^* n.$$

O que falta?

\hat{c}_i : custo amortizado da i -ésima operação

c_i : custo amortizado da i -ésima operação

Queremos mostrar que

$$\sum_{i=1}^m \hat{c}_i = O(m \lg^* n).$$

O que falta?

\hat{c}_i : custo amortizado da i -ésima operação

c_i : custo amortizado da i -ésima operação

Queremos mostrar que

$$\sum_{i=1}^m \hat{c}_i = O(m \lg^* n).$$

Assim, o **custo amortizado** por operação é $O(\lg^* n)$.

O que falta?

\hat{c}_i : custo amortizado da i -ésima operação

c_i : custo amortizado da i -ésima operação

Queremos mostrar que

$$\sum_{i=1}^m \hat{c}_i = O(m \lg^* n).$$

Assim, o **custo amortizado** por operação é $O(\lg^* n)$.

Disso, $\sum_{i=1}^m \hat{c}_i = \sum_{i=1}^m c_i + \Phi_f - \Phi_0$,
onde Φ_f é o potencial final da floresta.

O que falta?

\hat{c}_i : custo amortizado da i -ésima operação

c_i : custo amortizado da i -ésima operação

Queremos mostrar que

$$\sum_{i=1}^m \hat{c}_i = O(m \lg^* n).$$

Assim, o **custo amortizado** por operação é $O(\lg^* n)$.

Disso, $\sum_{i=1}^m \hat{c}_i = \sum_{i=1}^m c_i + \Phi_f - \Phi_0$,
onde Φ_f é o potencial final da floresta.

Como $\Phi_0 = 0$ e $\Phi_f \geq 0$, temos que $\sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i$.

Juntando tudo

Das deduções anteriores,

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde f é o número de operações FINDSET na sequência,

v_i é 0 se a i -ésima operação não é FINDSET e

é o número de **arestas vermelhas** em P se é FINDSET(x).

Juntando tudo

Das deduções anteriores,

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde f é o número de operações FINDSET na sequência,

v_i é 0 se a i -ésima operação não é FINDSET e

é o número de **arestas vermelhas** em P se é FINDSET(x).

Para $\sum_{i=1}^m v_i$: se um u deixa de ser ponta inferior de **aresta vermelha**, não volta a ser ponta inferior de **aresta vermelha**.

Juntando tudo

Das deduções anteriores,

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde f é o número de operações FINDSET na sequência,
 v_i é 0 se a i -ésima operação não é FINDSET e
é o número de **arestas vermelhas** em P se é FINDSET(x).

Para $\sum_{i=1}^m v_i$: se um u deixa de ser ponta inferior de **aresta vermelha**, não volta a ser ponta inferior de **aresta vermelha**.

Todos os nós de P , exceto os dois últimos (a raiz e o filho dela no caminho), têm o campo **pai** alterado no FINDSET.

Juntando tudo

Das deduções anteriores,

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde f é o número de operações FINDSET na sequência,
 v_i é 0 se a i -ésima operação não é FINDSET e
é o número de **arestas vermelhas** em P se é FINDSET(x).

Para $\sum_{i=1}^m v_i$: se um u deixa de ser ponta inferior de **aresta vermelha**, não volta a ser ponta inferior de **aresta vermelha**.

Todos os nós de P , exceto os dois últimos (a raiz e o filho dela no caminho), têm o campo **pai** alterado no FINDSET.

Se o **pai** de um nó é alterado, **rank** do seu pai aumenta pois

Juntando tudo

Das deduções anteriores,

$$\sum_{i=1}^m \hat{c}_i \leq 2(m - f) + f(1 + \lg^* n) + \sum_{i=1}^m v_i,$$

onde f é o número de operações FINDSET na sequência,
 v_i é 0 se a i -ésima operação não é FINDSET e
é o número de **arestas vermelhas** em P se é FINDSET(x).

Para $\sum_{i=1}^m v_i$: se um u deixa de ser ponta inferior de **aresta vermelha**, não volta a ser ponta inferior de **aresta vermelha**.

Todos os nós de P , exceto os dois últimos (a raiz e o filho dela no caminho), têm o campo **pai** alterado no FINDSET.

Se o **pai** de um nó é alterado, **rank** do seu pai aumenta pois **rank** cresce estritamente ao subirmos nas árvores.

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Seja $t(s) = \min\{k \in \mathbb{Z} : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$).

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Seja $t(s) = \min\{k \in \mathbb{Z} : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$).

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g + 1) - t(g) \leq t(g + 1)$ vezes.

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Seja $t(s) = \min\{k \in \mathbb{Z} : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$).

Se $g = \lg^*(\mathbf{rank}[y])$, então $\leq t(g + 1) - t(g) \leq t(g + 1)$ vezes.

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\mathbf{rank}[y]) = g\}| t(g + 1).$$

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Seja $t(s) = \min\{k \in \mathbb{Z} : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$).

Se $g = \lg^*(\mathbf{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes.

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\mathbf{rank}[y]) = g\}| t(g+1).$$

O $2f$ é pelos dois nós (a raiz e o filho dela no caminho).

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

Num FINDSET, o campo **pai** sofre alteração para quantos y ?

Seja $t(s) = \min\{k \in \mathbb{Z} : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$).

Se $g = \lg^*(\mathbf{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes.

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\mathbf{rank}[y]) = g\}| t(g+1).$$

O $2f$ é pelos dois nós (a raiz e o filho dela no caminho).

O número de nós y tais que $\mathbf{rank}[y] = r$ é no máximo $n/2^r$.

(Subárvore com raiz de rank r tem pelo menos 2^r nós.)

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

$t(s) = \min\{k \in Z : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$)

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes e

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

O número de nós y tais que $\text{rank}[y] = r$ é no máximo $n/2^r$.

Além disso, $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$.

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

$t(s) = \min\{k \in Z : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$)

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes e

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

O número de nós y tais que $\text{rank}[y] = r$ é no máximo $n/2^r$.

Além disso, $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$.

$$|\{y : \lg^*(\text{rank}[y]) = g\}| \leq \sum_{r=t(g)}^{\infty} |\{y : \text{rank}[y] = r\}|$$

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

$t(s) = \min\{k \in Z : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$)

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes e

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

O número de nós y tais que $\text{rank}[y] = r$ é no máximo $n/2^r$.

Além disso, $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$.

$$\begin{aligned} |\{y : \lg^*(\text{rank}[y]) = g\}| &\leq \sum_{r=t(g)}^{\infty} |\{y : \text{rank}[y] = r\}| \\ &\leq \sum_{r=t(g)}^{\infty} \frac{n}{2^r} \end{aligned}$$

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

$t(s) = \min\{k \in Z : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$)

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes e

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

O número de nós y tais que $\text{rank}[y] = r$ é no máximo $n/2^r$.

Além disso, $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$.

$$\begin{aligned} |\{y : \lg^*(\text{rank}[y]) = g\}| &\leq \sum_{r=t(g)}^{\infty} |\{y : \text{rank}[y] = r\}| \\ &\leq \sum_{r=t(g)}^{\infty} \frac{n}{2^r} = \frac{n}{2^{t(g)}} \sum_{r=0}^{\infty} \frac{1}{2^r} \end{aligned}$$

Juntando tudo

y : ponta inferior de uma **aresta vermelha**

$t(s) = \min\{k \in Z : \lg^* k \geq s\}$ (tipo de inversa do $\lg^* n$)

Se $g = \lg^*(\text{rank}[y])$, então $\leq t(g+1) - t(g) \leq t(g+1)$ vezes e

$$\sum_{i=1}^m v_i \leq 2f + \sum_{g=1}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1).$$

O número de nós y tais que $\text{rank}[y] = r$ é no máximo $n/2^r$.

Além disso, $t(g) = \min\{k \in Z : \lg^* k \geq g\} \leq \text{rank}[y]$.

$$\begin{aligned} |\{y : \lg^*(\text{rank}[y]) = g\}| &\leq \sum_{r=t(g)}^{\infty} |\{y : \text{rank}[y] = r\}| \\ &\leq \sum_{r=t(g)}^{\infty} \frac{n}{2^r} = \frac{n}{2^{t(g)}} \sum_{r=0}^{\infty} \frac{1}{2^r} \\ &= \frac{2n}{2^{t(g)}}. \end{aligned}$$

Conclusão

Portanto concluímos que

$$\begin{aligned} \sum_{i=1}^m v_i &\leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1) \\ &\leq 2f + 2n \sum_{g=0}^{\lg^* n} \frac{t(g+1)}{2^{t(g)}}. \end{aligned}$$

Conclusão

Portanto concluímos que

$$\begin{aligned}\sum_{i=1}^m v_i &\leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1) \\ &\leq 2f + 2n \sum_{g=0}^{\lg^* n} \frac{t(g+1)}{2^{t(g)}}.\end{aligned}$$

Mas, $t(g+1) \leq 2^{t(g)}$, ou seja, $\sum_{i=1}^m v_i \leq 2f + 2n(1 + \lg^* n)$.

Conclusão

Portanto concluímos que

$$\begin{aligned}\sum_{i=1}^m v_i &\leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1) \\ &\leq 2f + 2n \sum_{g=0}^{\lg^* n} \frac{t(g+1)}{2^{t(g)}}.\end{aligned}$$

Mas, $t(g+1) \leq 2^{t(g)}$, ou seja, $\sum_{i=1}^m v_i \leq 2f + 2n(1 + \lg^* n)$.

Então

$$\sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i \leq 2m + 2n + f + 2m \lg^* n = O(m \lg^* n).$$

Conclusão

Portanto concluímos que

$$\begin{aligned}\sum_{i=1}^m v_i &\leq 2f + \sum_{g=0}^{\lg^* n} |\{y : \lg^*(\text{rank}[y]) = g\}| t(g+1) \\ &\leq 2f + 2n \sum_{g=0}^{\lg^* n} \frac{t(g+1)}{2^{t(g)}}.\end{aligned}$$

Mas, $t(g+1) \leq 2^{t(g)}$, ou seja, $\sum_{i=1}^m v_i \leq 2f + 2n(1 + \lg^* n)$.

Então

$$\sum_{i=1}^m c_i \leq \sum_{i=1}^m \hat{c}_i \leq 2m + 2n + f + 2m \lg^* n = O(m \lg^* n).$$

Logo o custo amortizado por operação é $O(\lg^* n)$.