

Explicação dos Exercícios de Árvores

Exercícios B, D, E

B - Ancient Berland Roads

— — —

- São dadas N cidades e M rotas bidirecionais
- Com o passar do tempo existem queries que tiram estradas
- E também queries que mudam a população de uma cidade
- Em cada query printar a máxima população de um grupo de cidades

B - Ancient Berland Roads

— — —

- Problema off querie
- Cada mudança de população salvar nas queries a $\text{pop_antiga} - \text{pop_nova}$
- Se tenho uma população 3 na cidade e mudar para 5
- Salvar $3 - 5 = -2$ na querie de mudança de população

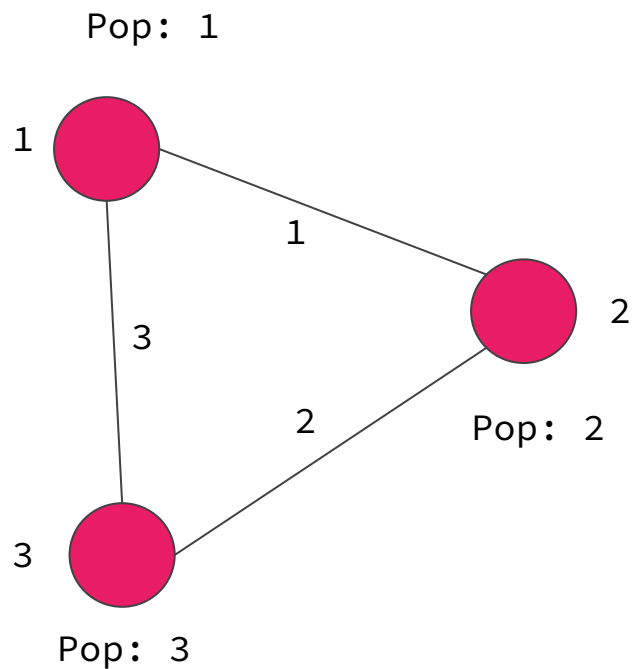
B - Ancient Berland Roads

- Usar um multiset pra escolher a cidade com maior população
- Se aparecer uma querie P tem que retirar a população daquele conjunto de cidades do multiset
- Colocar a população de novo com o valor atualizado
- Em cada querie printar o valor mais alto no multiset.

B - Ancient Berland Roads

— — —

Max = 6



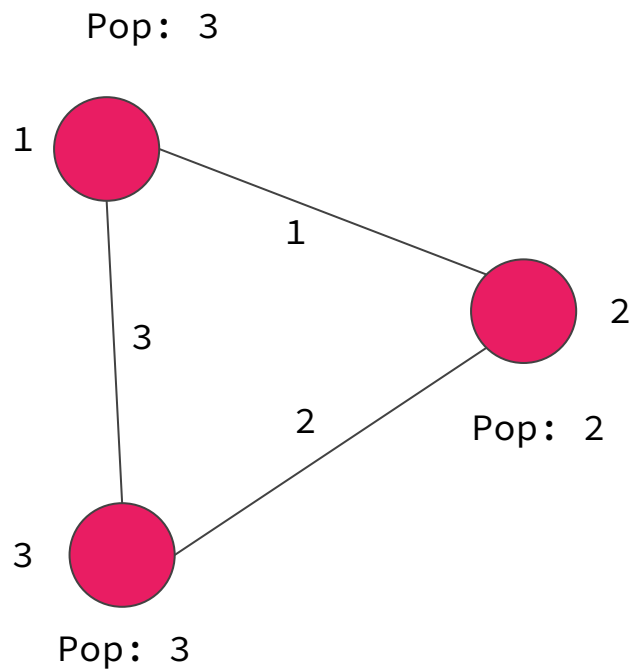
Queries antes:

P 1 3
D 1
P 2 3
D 2
P 3 10
D 3

B - Ancient Berland Roads

— — —

Max = 8



Queries antes:

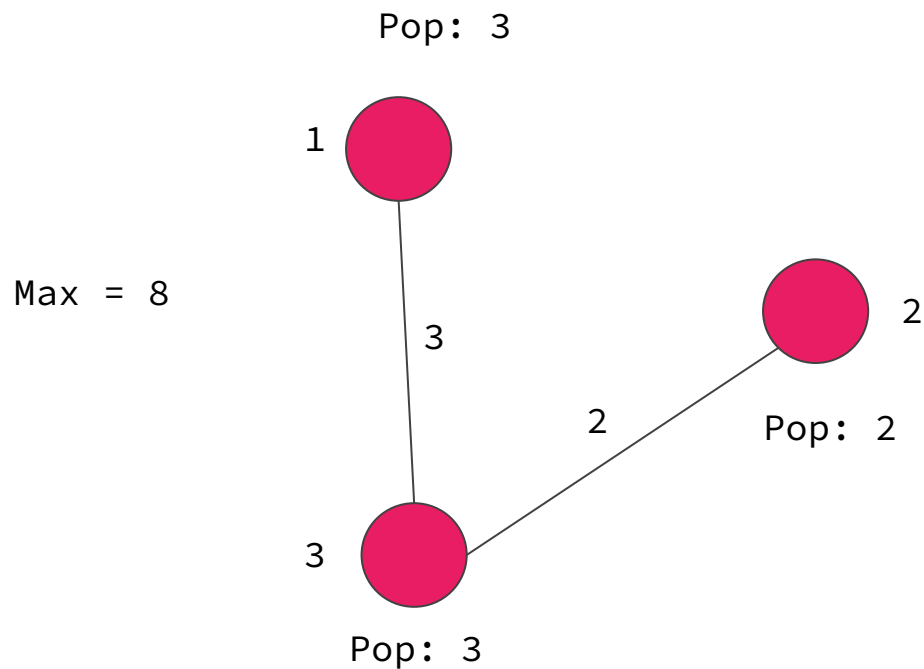
P 1 3

Queries depois:

P 1 -2

B - Ancient Berland Roads

— — —



Queries antes:

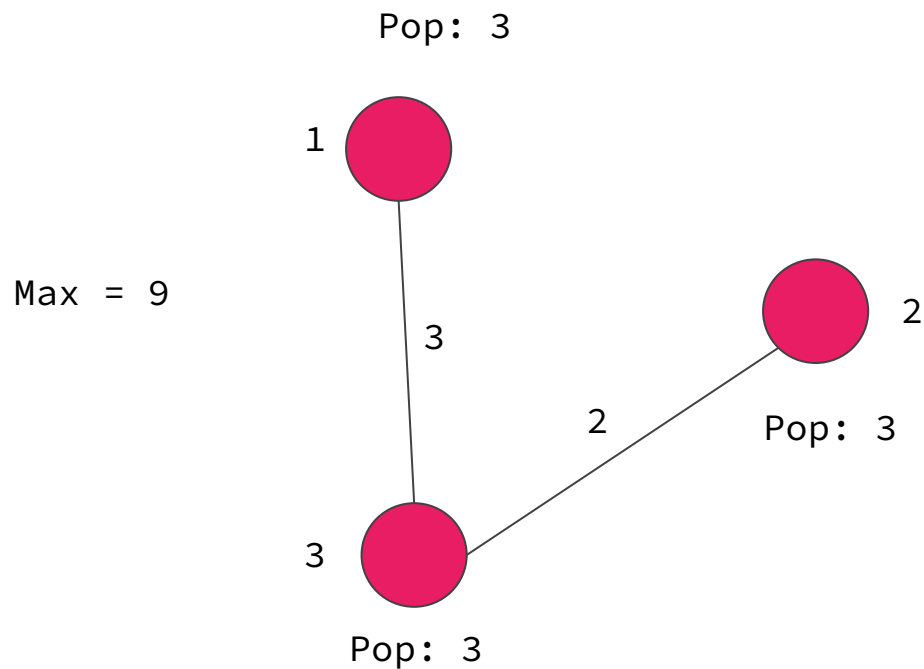
P 1 3
D 1

Queries depois:

P 1 -2
D 1

B - Ancient Berland Roads

— — —



Queries antes:

P 1 3

D 1

P 2 3

Queries depois:

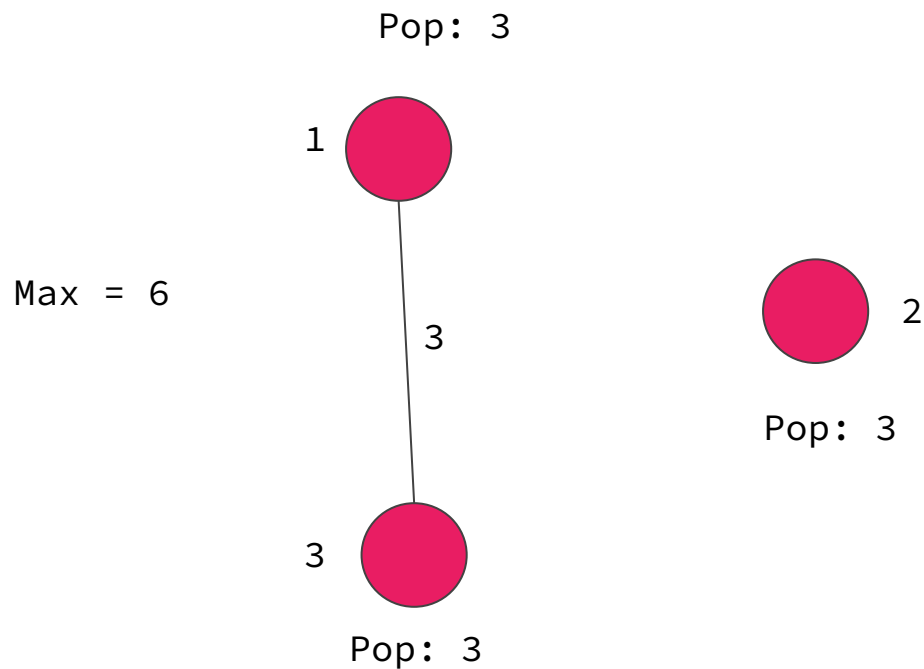
P 1 -2

D 1

P 2 -1

B - Ancient Berland Roads

— — —



Queries antes:

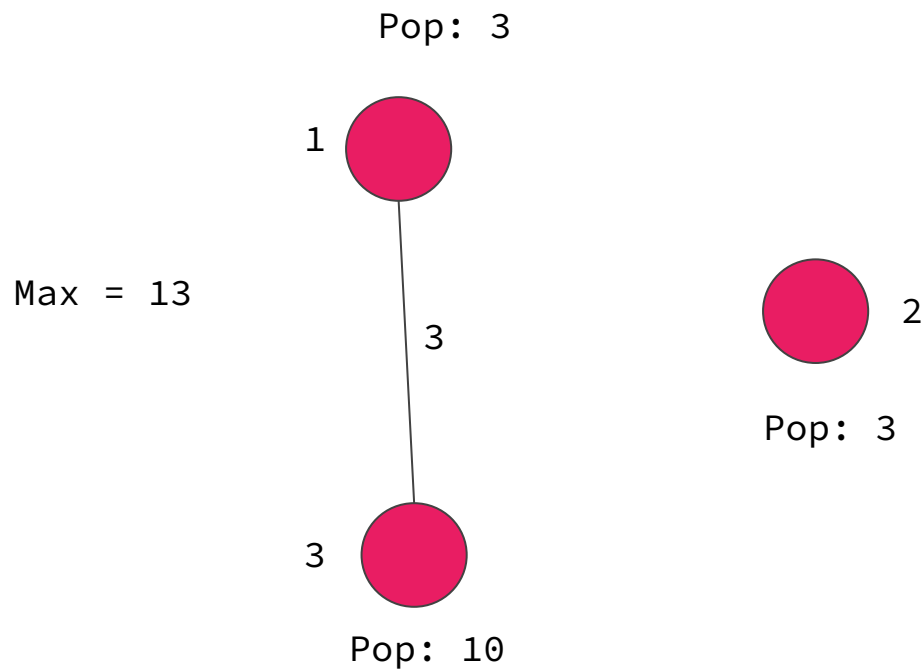
P 1 3
D 1
P 2 3
D 2

Queries depois:

P 1 -2
D 1
P 2 -1
D 2

B - Ancient Berland Roads

— — —



Queries antes:


```
P 1 3
D 1
P 2 3
D 2
P 3 10
```

Queries depois:

```
P 1 -2
D 1
P 2 -1
D 2
P 3 -7
```


B - Ancient Berland Roads

— — —

Pop: 3
1 

Max = 10

 2
Pop: 3

3 
Pop: 10

Queries antes:


P 1 3
D 1
P 2 3
D 2
P 3 10
D 3

Queries depois:


P 1 -2
D 1
P 2 -1
D 2
P 3 -7
D 3


B - Ancient Berland Roads

— — —

Pop: 3
1 

Max = 10

 2
Pop: 3

3 
Pop: 10

Queries depois:

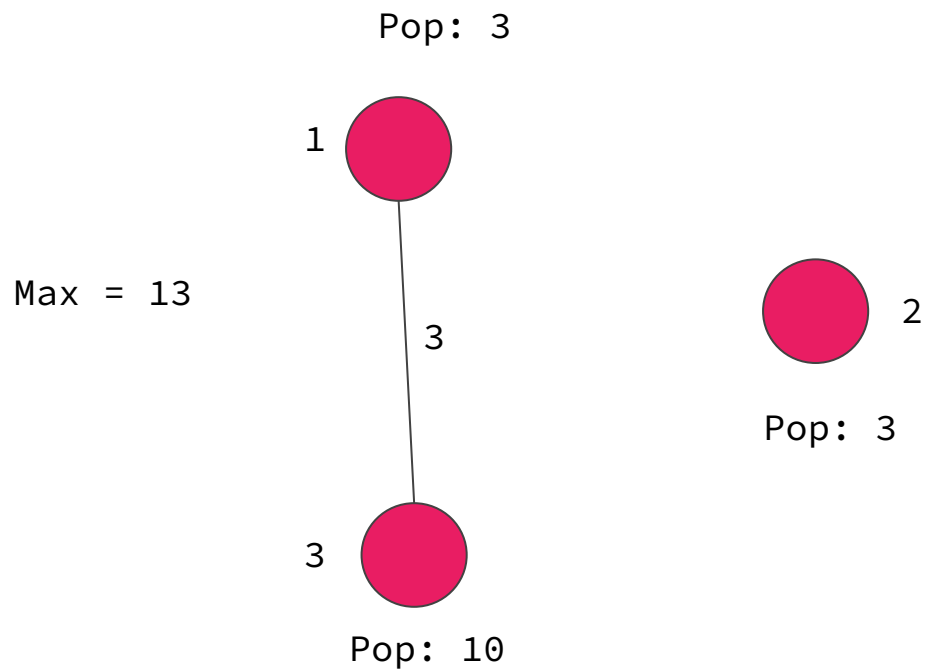
D 3
P 3 -7
D 2
P 2 -1
D 1
P 1 -2

B - Ancient Berland Roads

— — —

Queries depois:

D 3



B - Ancient Berland Roads

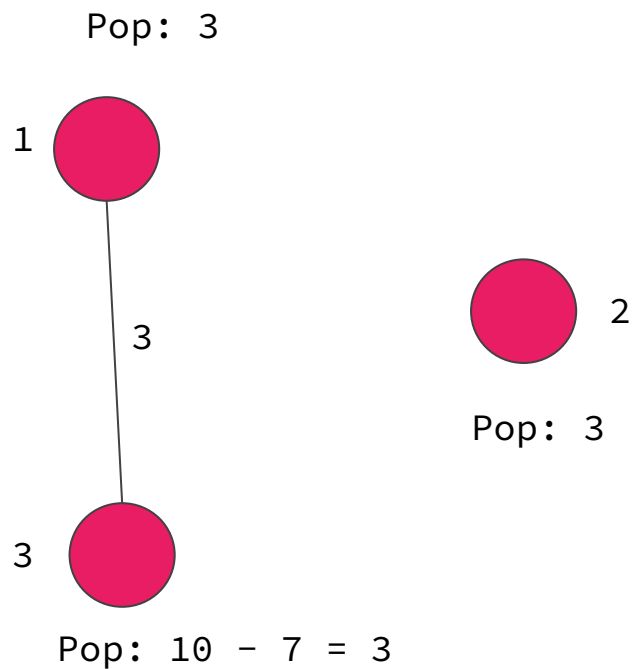
— — —

Queries depois:

D 3

P 3 -7

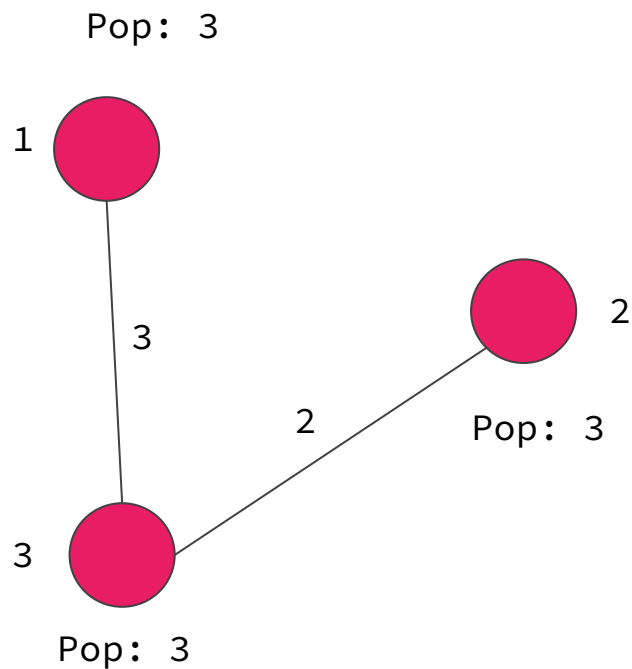
Max = 6



B - Ancient Berland Roads

— — —

Max = 9



Queries depois:

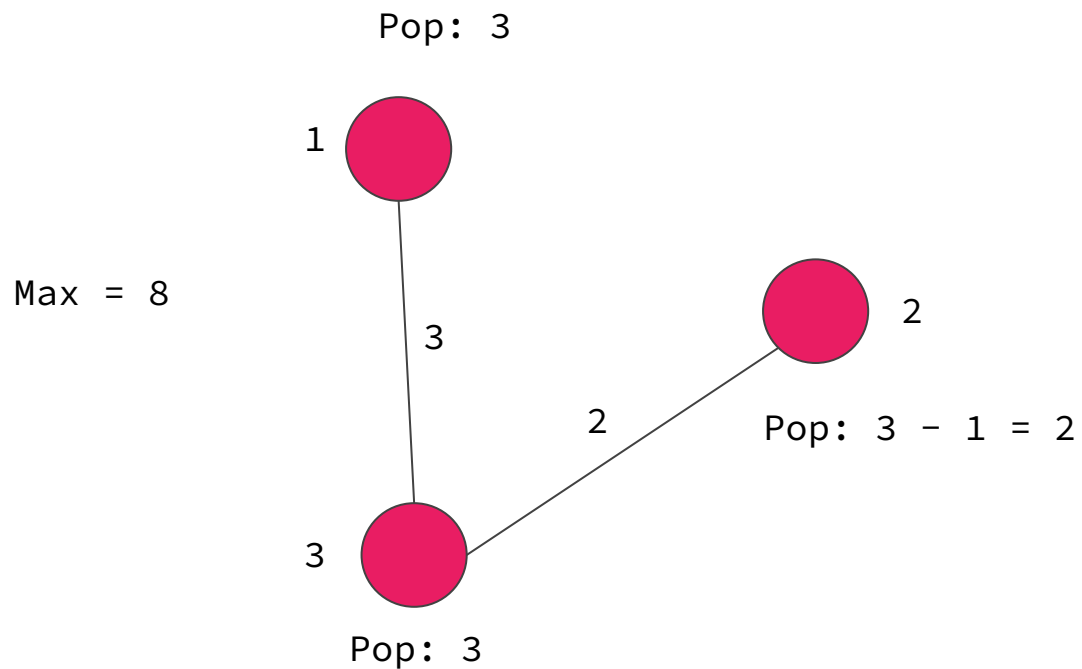
D 3

P 3 -7

D 2

B - Ancient Berland Roads

— — —



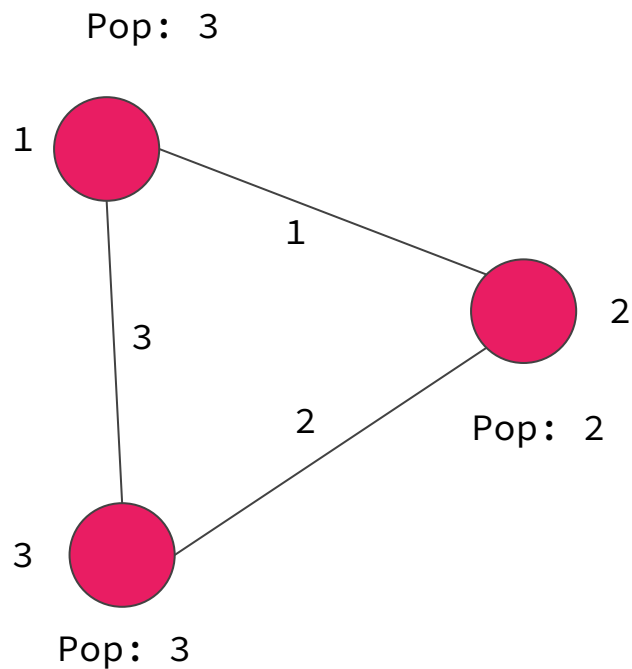
Queries depois:

D 3
P 3 -7
D 2
P 2 -1

B - Ancient Berland Roads

— — —

Max = 8



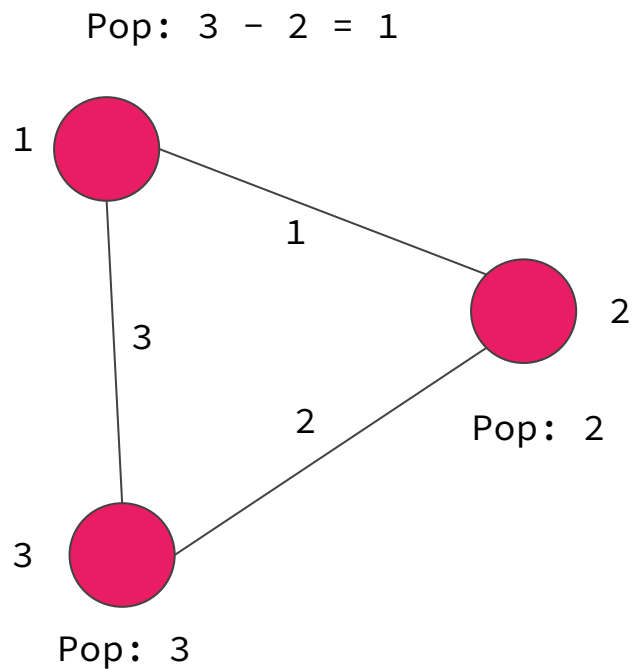
Queries depois:

D 3
P 3 -7
D 2
P 2 -1
D 1

B - Ancient Berland Roads

— — —

Max = 6



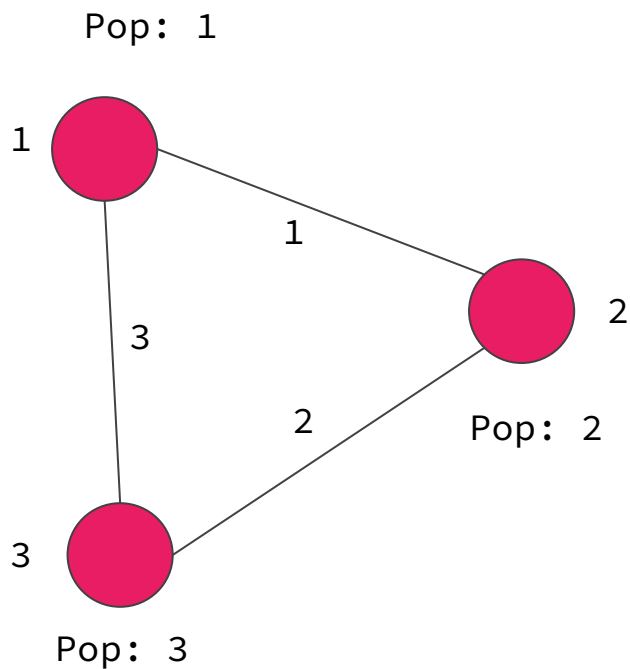
Queries depois:

D 3
P 3 -7
D 2
P 2 -1
D 1
P 1 -2

B - Ancient Berland Roads

— — —

Max = 6

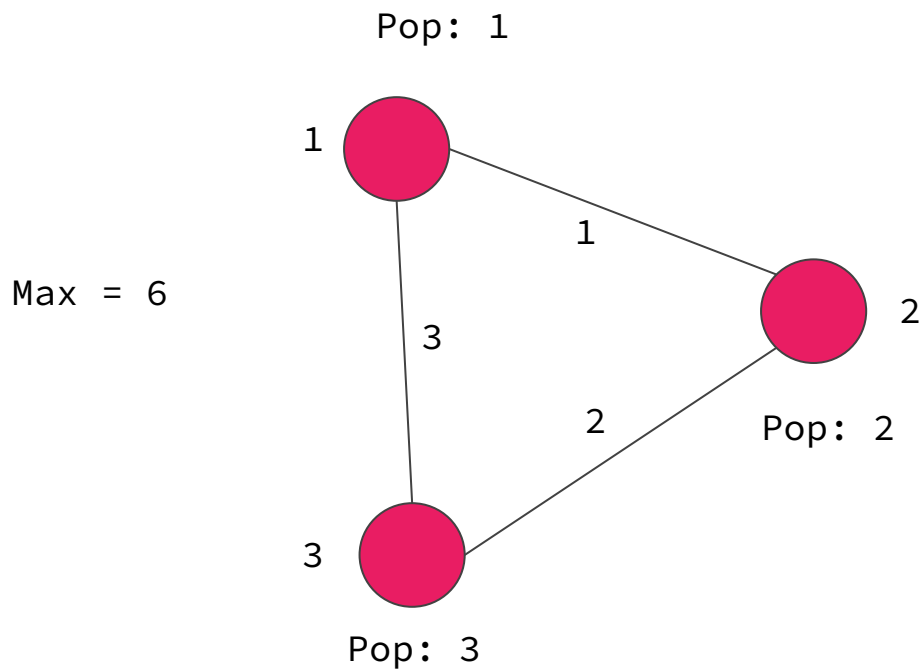


Queries depois:

```
D 3
P 3 -7
D 2
P 2 -1
D 1
P 1 -2
```

Voltou ao estado
inicial :)

B - Ancient Berland Roads



Queries depois:

```
D 3
P 3 -7
D 2
P 2 -1
D 1
P 1 -2
```

Código



QUENTÃO



Voltou
inicia



Accepted! junino

D - Almost Union-Find

— — —

- Implementar uma estrutura de dados que permite as seguintes operações:

COMANDO	ENTRADA	AÇÃO
1	p q	Unir os conjuntos que contêm os elementos p e q . Se p e q fazem parte do mesmo conjunto, ignorar o comando.
2	p q	Mover o elemento p para o conjunto que contém o elemento q . Se p e q fazem parte do mesmo conjunto, ignorar o comando.
3	p	Retorna o número de elementos e a soma dos elementos do conjunto que contém o elemento p .

D - Almost Union-Find

— — —

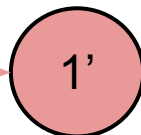
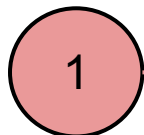
- São dados **N elementos**, com valores de **1 até N** , e **M comandos**;
- Inicialmente, **cada N_i elemento pertence a um subconjunto distinto**;
- **Para cada comando M_i , modificar a estrutura** conforme as **ações** de cada comando na tabela.

D - Almost Union-Find

— — —

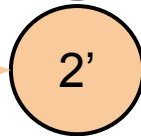
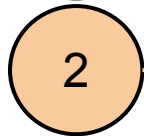
Comandos:

$\text{Pai}[1] = 1'$



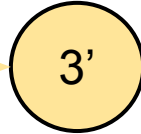
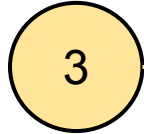
$\text{Tam}[1'] = 1$
 $\text{Soma}[1'] = 1$

$\text{Pai}[2] = 2'$



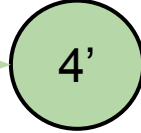
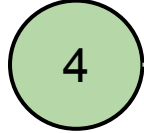
$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Pai}[3] = 3'$



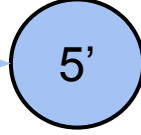
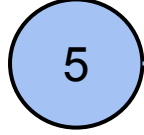
$\text{Tam}[3'] = 1$
 $\text{Soma}[3'] = 3$

$\text{Pai}[4] = 4'$



$\text{Tam}[4'] = 1$
 $\text{Soma}[4'] = 4$

$\text{Pai}[5] = 5'$



$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

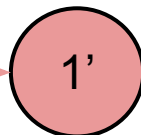
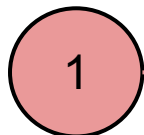
D - Almost Union-Find

— — —

Comandos:

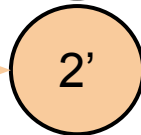
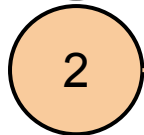
1 1 2

$\text{Pai}[1] = 1'$



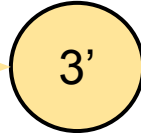
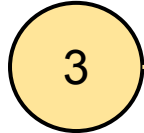
$\text{Tam}[1'] = 1$
 $\text{Soma}[1'] = 1$

$\text{Pai}[2] = 2'$



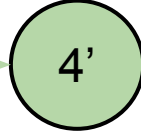
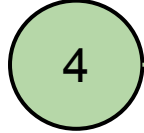
$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Pai}[3] = 3'$



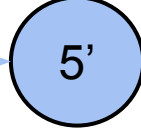
$\text{Tam}[3'] = 1$
 $\text{Soma}[3'] = 3$

$\text{Pai}[4] = 4'$



$\text{Tam}[4'] = 1$
 $\text{Soma}[4'] = 4$

$\text{Pai}[5] = 5'$



$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2

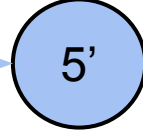
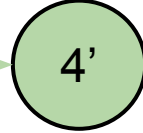
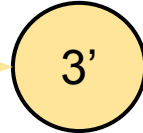
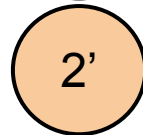
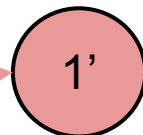
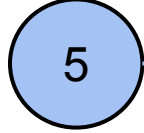
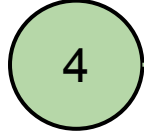
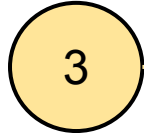
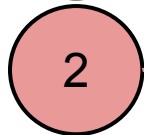
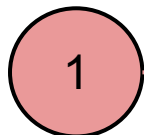
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 3'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 5'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 1$
 $\text{Soma}[3'] = 3$

$\text{Tam}[4'] = 1$
 $\text{Soma}[4'] = 4$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4

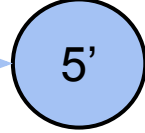
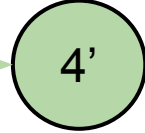
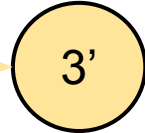
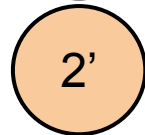
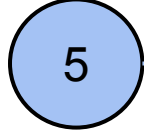
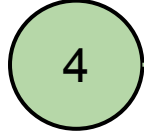
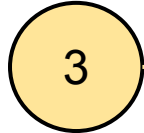
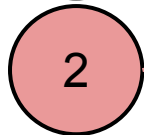
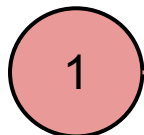
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 3'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 5'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 1$
 $\text{Soma}[3'] = 3$

$\text{Tam}[4'] = 1$
 $\text{Soma}[4'] = 4$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4

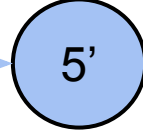
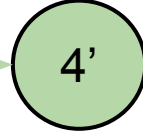
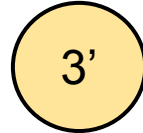
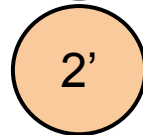
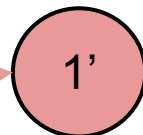
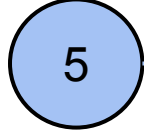
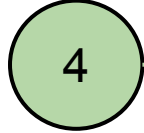
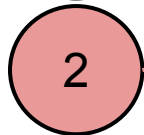
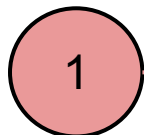
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = ?$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 5'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 1$
 $\text{Soma}[4'] = 4$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4

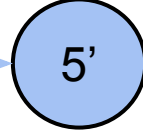
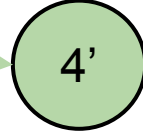
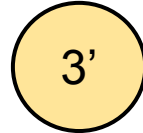
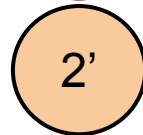
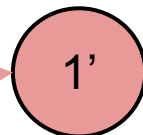
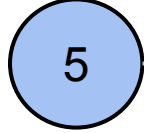
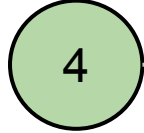
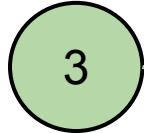
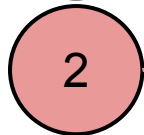
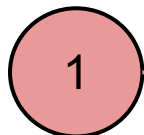
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 5'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 7$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5

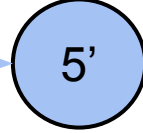
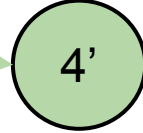
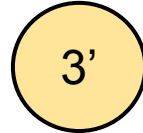
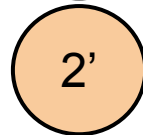
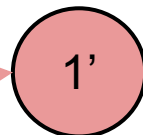
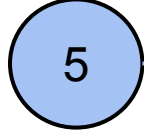
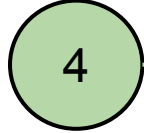
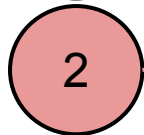
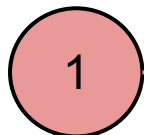
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 5'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 7$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5

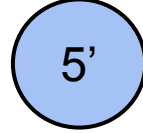
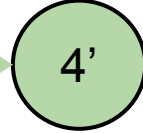
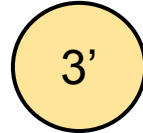
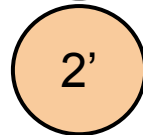
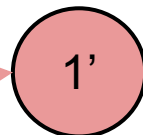
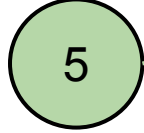
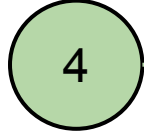
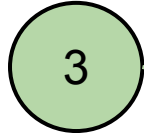
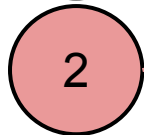
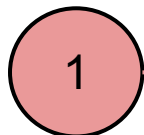
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 3$
 $\text{Soma}[4'] = 12$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4

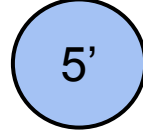
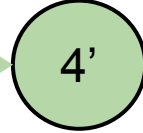
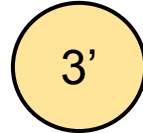
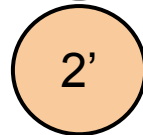
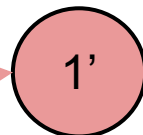
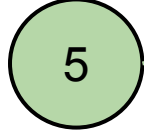
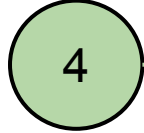
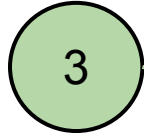
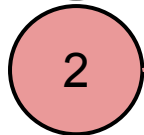
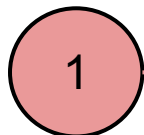
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 3$
 $\text{Soma}[4'] = 12$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4

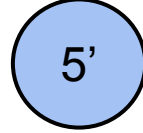
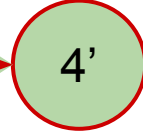
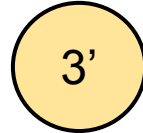
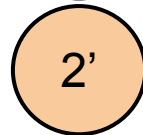
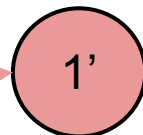
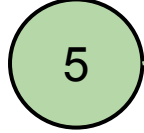
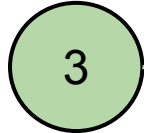
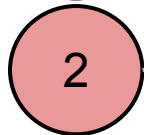
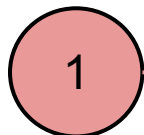
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 3$
 $\text{Soma}[4'] = 12$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1

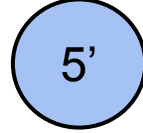
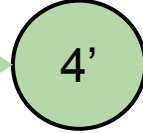
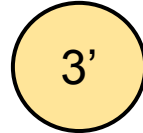
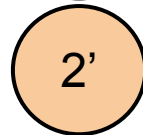
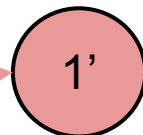
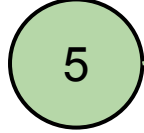
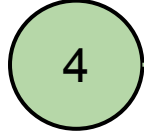
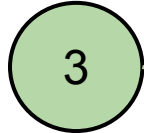
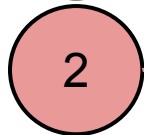
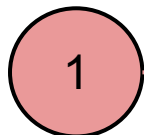
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 4'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 3$
 $\text{Soma}[4'] = 12$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1

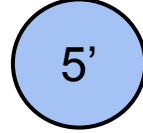
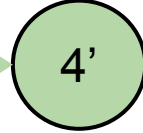
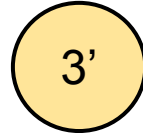
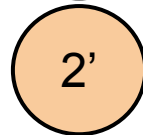
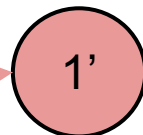
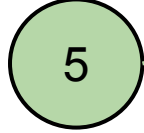
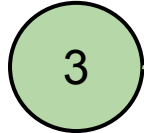
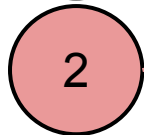
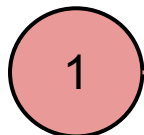
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = ?$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 2$
 $\text{Soma}[1'] = 3$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1

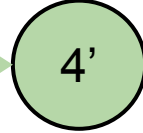
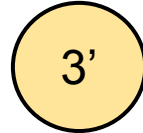
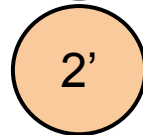
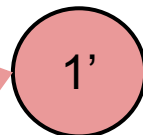
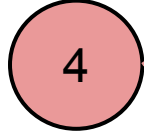
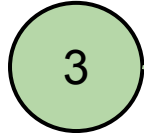
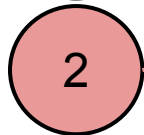
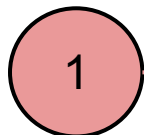
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 1'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 3$
 $\text{Soma}[1'] = 7$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1
3 4

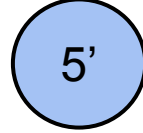
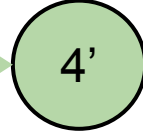
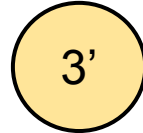
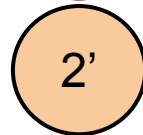
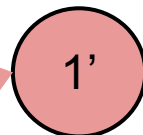
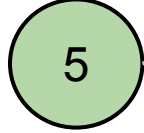
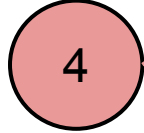
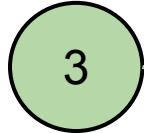
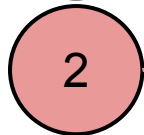
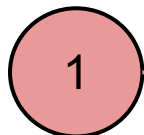
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 1'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 3$
 $\text{Soma}[1'] = 7$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1
3 4

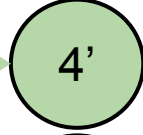
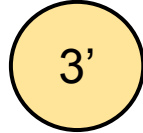
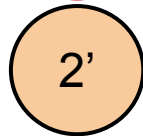
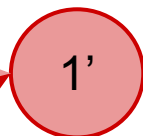
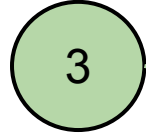
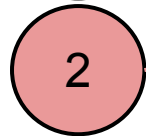
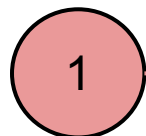
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 1'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 3$
 $\text{Soma}[1'] = 7$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1
3 4
3 5

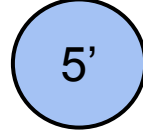
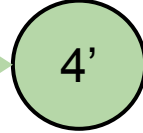
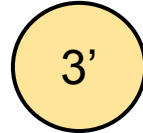
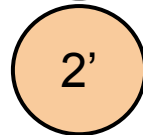
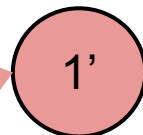
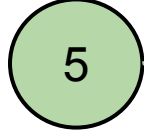
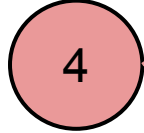
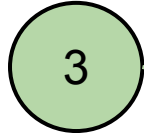
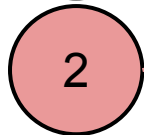
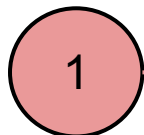
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 1'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 3$
 $\text{Soma}[1'] = 7$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

Comandos:

1 1 2
2 3 4
1 3 5
3 4
2 4 1
3 4
3 5

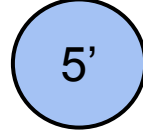
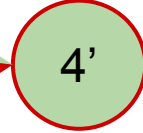
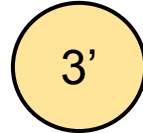
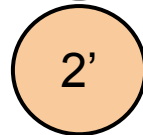
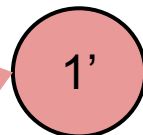
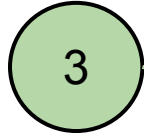
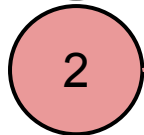
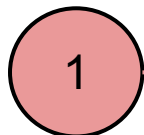
$\text{Pai}[1] = 1'$

$\text{Pai}[2] = 1'$

$\text{Pai}[3] = 4'$

$\text{Pai}[4] = 1'$

$\text{Pai}[5] = 4'$



$\text{Tam}[1'] = 3$
 $\text{Soma}[1'] = 7$

$\text{Tam}[2'] = 1$
 $\text{Soma}[2'] = 2$

$\text{Tam}[3'] = 0$
 $\text{Soma}[3'] = 0$

$\text{Tam}[4'] = 2$
 $\text{Soma}[4'] = 8$

$\text{Tam}[5'] = 1$
 $\text{Soma}[5'] = 5$

D - Almost Union-Find

— — —

```
int main() {
    int n, m;

    while (cin >> n >> m) {
        init();

        while (m--) {
            int c, p, q;
            cin >> c >> p;

            if (c != 3) {
                cin >> q;
                c == 1 ? unite(p, q) : move(p, q);
            } else {
                p = find(p);
                cout << len[p] << " " << sum[p] << "\n";
            }
        }
    }

    return 0;
}
```


D - Almost Union-Find

— — —

```
int main() {
    int n, m;

    while (cin >> n >> m) {
        init();

        while (m--) {
            int c, p, q;
            cin >> c >> p;

            if (c != 3) {
                cin >> q;
                c == 1 ? unite(p, q) : move(p, q);
            } else {
                p = find(p);
                cout << len[p] << " " << sum[p] << "\n";
            }
        }
    }

    return 0;
}
```

```
void init() {
    parent = len = sum = vi(2 * n + 1);

    for (int u = 1; u <= n; u++) {
        parent[u] = parent[u + n] = u + n;
        len[u + n] = 1;
        sum[u + n] = u;
    }
}

int find(int u) {
    if (u == parent[u])
        return u;

    return parent[u] = find(parent[u]);
}
```

D - Almost Union-Find

— — —

```
int main() {
    int n, m;

    while (cin >> n >> m) {
        init();

        while (m--) {
            int c, p, q;
            cin >> c >> p;

            if (c != 3) {
                cin >> q;
                c == 1 ? unite(p, q) : move(p, q);
            } else {
                p = find(p);
                cout << len[p] << " " << sum[p] << "\n";
            }
        }

        return 0;
    }
}
```

```
void unite(int u, int v) {
    u = find(u);
    v = find(v);

    if (u != v) {
        if (len[u] < len[v])
            swap(u, v);

        parent[v] = u;
        len[u] += len[v];
        sum[u] += sum[v];
    }
}

void move(int u, int v) {
    int p;
    p = find(u);
    v = find(v);

    if (p != v) {
        parent[u] = v;
        len[p]--;
        sum[p] -= u;
        len[v]++;
        sum[v] += u;
    }
}
```

D - Almost Union-Find

```
int main() {
    int n, m;

    while (cin >> n >> m) {
        init();

        while (m--) {
            int c, p, q;
            cin >> c >> p;

            if (c != 3) {
                cin >> q;
                c == 1 ? unite(p, q) : move(p, q);
            } else {
                p = find(p);
                cout << len[p] << " " << sum[p] << "\n";
            }
        }

        return 0;
    }
}
```

```
void unite(int u, int v) {
    u = find(u);
    v = find(v);

    if (u != v) {
        if (len[u] < len[v])
            swap(u, v);

        parent[v] = u;
        len[u] += len[v];
        sum[u] += sum[v];
    }
}
```

```
void move(int u, int v) {
    int p;
    p = find(u);
    v = find(v);

    if (p != v) {
        parent[u] = v;
        len[p]--;
        sum[p] -= u;
        len[v]++;
        sum[v] += u;
    }
}
```





Programação Competitiva Unesp Bauru
apresenta



Arissa “Tokidebug” Yoshida

Professora, programadora, musicista, futeboleira, 42

POJ 1988 – CUBE STACKING

“Como resolver esse raio de exercício sem poder utilizar a biblioteca `<bits/stdc++.h>`!”

27/Junho ◦ **Local:** aqui ◦ **Horário:** agora

Valor: R\$ 100.000.000,00
(Ingressos a venda com o **Paiola**)

