Disjoint-set data strucutre / DST / Union-Find

Определение

DST – струкрура данных, позволяющая вести учет некого конечного множества елементов, которые распределены по попарно непересекающимся подмножествам.

Для работы с таким множеством елементов нам необходимы 3 операции:

- `MakeSet(x)`: создание нового множества из единственного елемента
- `Union(x, y)`: объеденение множеств к которым пренадлежат x и y
- `Find(x)`: определить для $x \in S$ подмножество S и вернуть его представителя

История

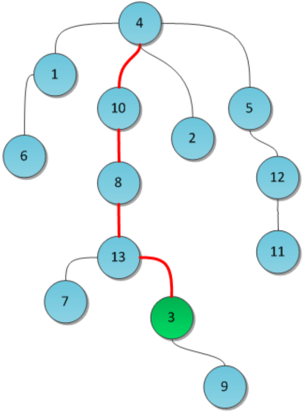
Впервые леса с непересекающимися множествами были описаны Бернардом А. Галлером и Майклом Дж. Фишером в 1964 году. В 1973 их сложность была ограничена $O(\log^*(n))$

$$\log^*(n) = \begin{cases} 0, & \text{if } n \leq 1 \\ 1 + \log^*(\log(n)), & \text{if } n > 1 \end{cases}$$

В 1979 Роберт Тарьян доказал верхнюю границу в $O(\alpha(n))$, где $\alpha(n)$ – обратная ф-ция Аккермана, и в 1979 доказал что это и нижний предел для ограниченого числа случаев.
В 1989 году Фредман и Сакс доказали $\Omega(\alpha(n))$ для любой операции над DSU.

Реализация (структура данных)

Данные DSU как правило хранятся в древовидной структуре, которая состоит из множества елементов(ячеек), в каждой из которых хранится указатель на "родителя и, опционально, идентификатор елемента. Для достижения сложности операций в $\Omega(\alpha(n))$ в ячейках также необходимо хранить дополнительное значение: "ранг"или "размер".



Реализация (операции)

```
MakeSet(x) =
  if x is not already present:
    add x to the disjoint-set tree
    x.parent := x
    x.rank := 0
    x.size := 1
```

$\Rightarrow O(1)$

Реализация (операции)

Path compression:

```
Find(x) =
  if x.parent != x
    x.parent := Find(x.parent)
  return x.parent
```

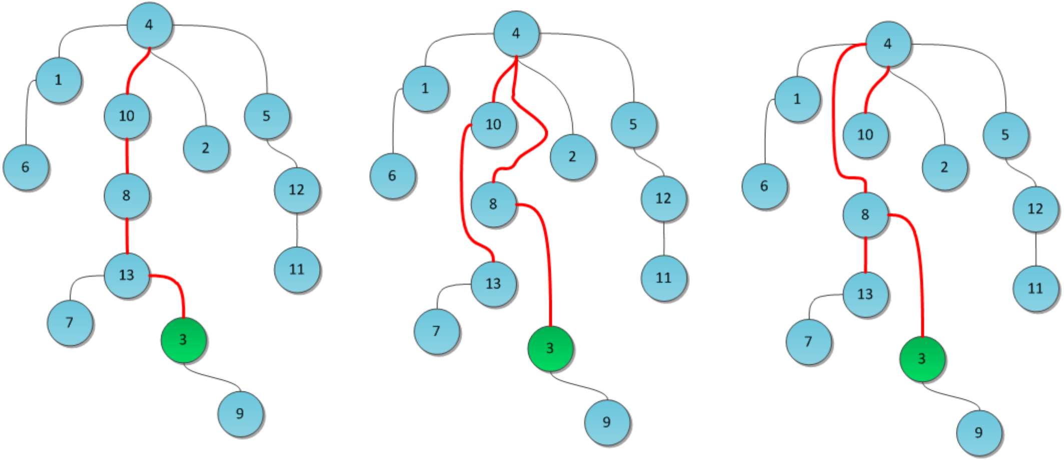
Path halving:

```
Find(x) =
  while x.parent != x
    x.parent := x.parent.parent
    x := x.parent
  return x
```

Path splitting:

```
Find(x) =
  while x.parent != x
    x, x.parent := x.parent,
x.parent.parent
  return x
```

$\Rightarrow O(\alpha(n))$



Initial $\rightarrow$ Path splitting $\rightarrow$ Path halving

Реализация (операции)

Union by rank:

```
Union(x, y) =
  xRoot = Find(x)
  yRoot = Find(y)
 // x and y are already in the same set
  if xRoot == yRoot
    return
 // x and y are not in same set, so we merge them
 if xRoot.rank < yRoot.rank
    // swap xRoot and yRoot
    xRoot, yRoot = yRoot, xRoot
 // merge yRoot into xRoot
 yRoot.parent = xRoot
 if xRoot.rank == yRoot.rank:
    xRoot.rank = xRoot.rank + 1
```

$\Rightarrow O(\alpha(n))$

Реализация (операции)

Union by size:

```
Union(x, y) =
  xRoot := Find(x)
  yRoot := Find(y)
  // x and y are already in the same set
  if xRoot == yRoot
    return
  // x and y are not in same set, so we merge them
  if xRoot.size < yRoot.size
    xRoot, yRoot := yRoot, xRoot // swap xRoot and yRoot
  // merge yRoot into xRoot
  yRoot.parent := xRoot
  xRoot.size := xRoot.size + yRoot.size
```

$$\Rightarrow O(\alpha(n))$$

and Kruskal's algorithm