Система непересекающихся множеств (СНМ)

Disjoint Set Union (DSU)

Задача

Есть n чисел $0, 1, 2, ..., n-1$. Они разбиты на несколько непересекающихся множеств $A_1 = \{0, 2, 4\}, A_2 = \{1, 6\}, ..., A_k = \{5\}$.

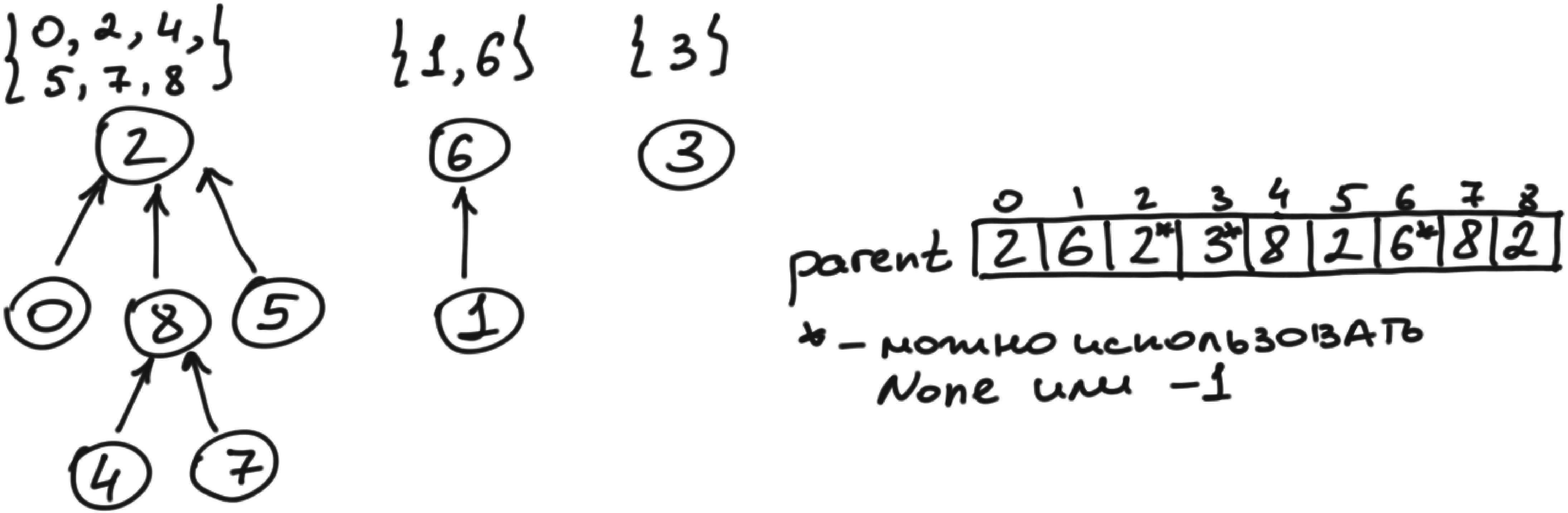
Требуется построить структуру данных, которая поддерживает следующие операции:

- $MakeSet()$ - создает новое одноэлементное множество $\{n\}$
- $FindSet(x)$ - возвращает идентификатор множества, в котором лежит x.
  Важное свойство:
  x *и* y *лежат в одном множестве* $\iff FindSet(x) == FindSet(y)$
- $Union(x, y)$ - объединить множество, в котором лежит x, со множеством, в котором находится y. После этого $FindSet(x) == FindSet(y)$

Лес непересекающихся множеств

Организуем из элементов каждого множества корневое дерево, то есть каждому элементу поставим в соответствие некоторого "представителя".

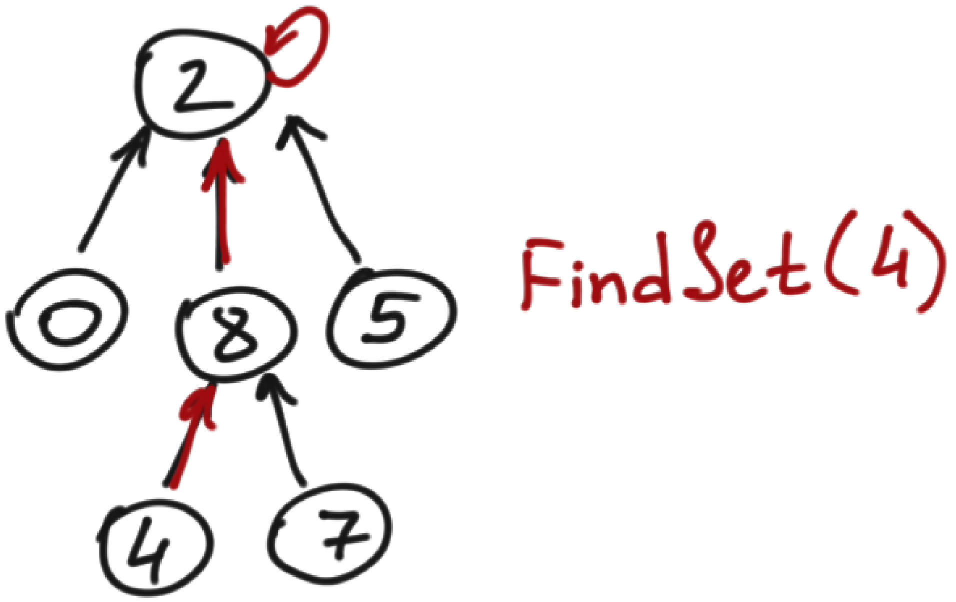
Идентификатором множества будет считаться корень дерева (именно он и возвращается в $FindSet$).



Лес непересекающихся множеств: $FindSet$

```
def FindSet(x):
  while x != parent[x]:
    x = parent[x]
  return x
```

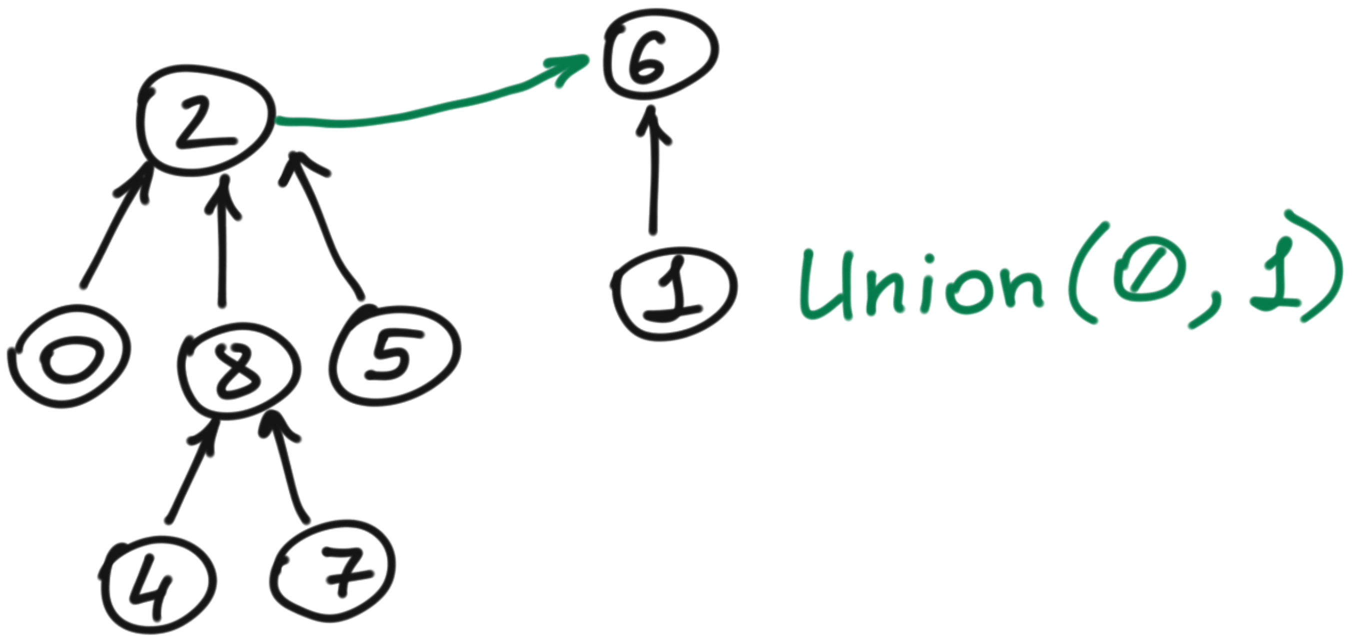
Время работы: $\Theta(h(x))$, где $h(x)$ - глубина вершины.



Лес непересекающихся множеств: $Union$

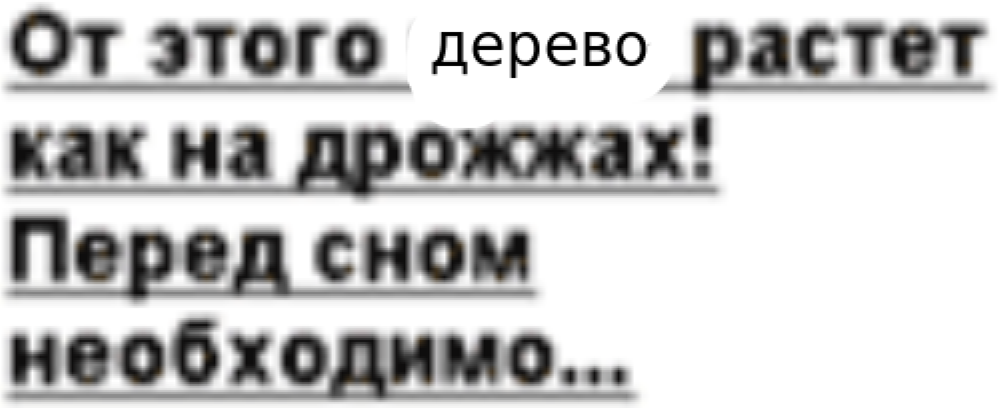
```
def Union(x, y):
  x = FindSet(x)
  y = FindSet(y)
  parent[x] = y
```

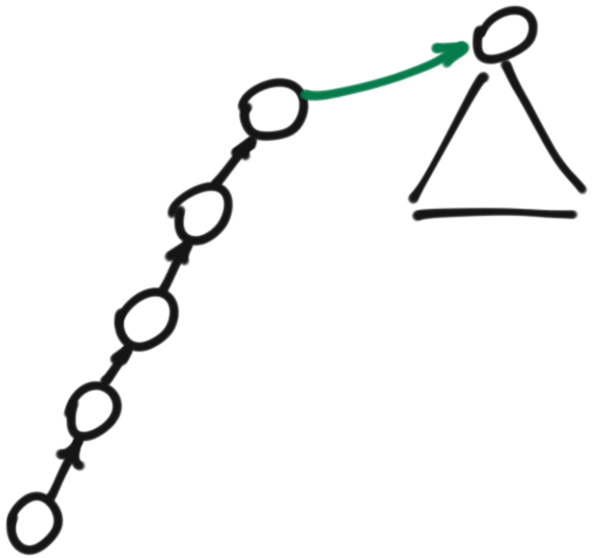
Время работы: $O(h(x) + h(y))$



Лес непересекающихся множеств: проблема

Возможен линейный рост глубины деревьев. То есть $O(h)$ вырождается в $O(n)$.





Улучшение 1: ранговая эвристика

Заметим, что при подвешивании менее глубокого дерева к более глубокому, глубина последнего не увеличивается.

При объединении деревьев одинаковой глубины, глубина растет, но только на 1.

```
# rank - глубина дерева
def Union(x, y):
  x = FindSet(x)
  y = FindSet(y)
  if rank[x] < rank[y]:
    parent[x] = y
  elif rank[y] < rank[x]:
    parent[y] = x
  else:  # rank[x] == rank[y]
    parent[x] = y
    rank[y] += 1
```

Улучшение 1: ранговая эвристика

Утверждение 1.

При использовании ранговой эвристики размер дерева с корнем x равен $size(x) \geq 2^{rank[x]}$

Улучшение 1: ранговая эвристика

Утверждение 1.

При использовании ранговой эвристики размер дерева с корнем x равен $size(x) \geq 2^{rank[x]}$

Доказательство.

В начальный момент это верно: $size(x) = 1 \geq 2^{rank[x]} = 2^0 = 1$.

Перед объединением деревьев x и y: $size(x) \geq 2^{rank[x]}$ и $size(y) \geq 2^{rank[y]}$.

- Если $rank[x] > rank[y]$, то $size'(x) = size(x) + size(y) \geq 2^{rank[x]} + 2^{rank[y]} \geq 2^{rank[x]} = 2^{rank'[x]}$
- Если $rank[x] = rank[y]$, то $size'(x) = size(x) + size(y) \geq 2^{rank[x]} + 2^{rank[y]} = 2^{rank[x]+1} = 2^{rank'[x]}$

■

Следствие: $h(x) \leq rank[x] = O(\log size(x))$

Улучшение 2: эвристика сжатия путей

При рекурсивном подъеме в $FindSet$ для каждой вершины узнаем самого главного представителя (корень). Но тогда можно подвесить эти вершины непосредственно к корню.

```
def FindSet(x):
  if x == parent[x]:
    return x
  return parent[x] = FindSet(parent[x])
```

Улучшение 2: эвристика сжатия путей

Утверждение 2 (б/д).

При использовании ранговой эвристики совместно с эвристикой сжатия путей амортизационная сложность $FindSet$ есть $O(\alpha(n))$, где α - обратная функция Аккермана.

$\alpha(1) = 0$, $\alpha(3) = 1$, $\alpha(7) = 2$, $\alpha(61) = 3$

Улучшение 2: эвристика сжатия путей

Утверждение 2 (б/д).

При использовании ранговой эвристики совместно с эвристикой сжатия путей амортизационная стоимость $FindSet$ есть $O(\alpha(n))$, где α - обратная функция Аккермана.

$$\alpha(1) = 0, \alpha(3) = 1, \alpha(7) = 2, \alpha(61) = 3$$

$$\alpha(2^{2^{2^{65536}}} - 3) = 4$$