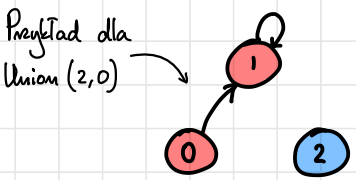
Union Find

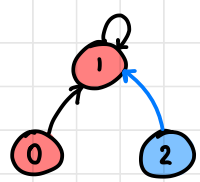
Union Find

- Union – łączymy elementy w zbiory
- Find – Szukamy do jakiego zbioru należy element

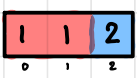
Union (a, b)

Przykład dla Union (2,0)





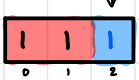
W tablicy mamy 2 grupy - czerwoną i niebieską.



Chcemy scalić niebieską z czerwoną

Przypisz właściciela zbioru a jako dziecko właściciela zbioru b

Ustawiamy na wskaźnik korzenia nowego drzewa

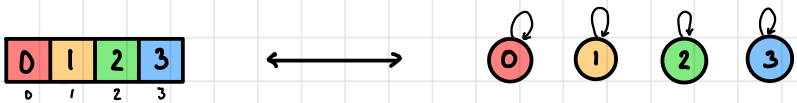


Find (kompresja ścieżek *)

1. Znajdź właściciela zbioru dla danego elementu x
* 2. Dla każdego elementu po kolei wychodząc z rekurencji - przypisz rodzica elementu jako właściciel zbioru

Charakterystyka:

- Początkowy stan to tablica wypełniona wartościami indeksów



- Gdy łączymy dwa zbiory, to przyłączamy mniejszy do większego to nam daje złożoność $O(\log n)$ dla operacji Find oraz Union

Najgorszy przypadek:

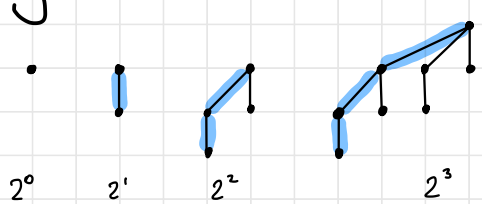
- Kompresja ścieżek zapewnia nam $O(\log^* n)$ czasu zamortyzowanego dla operacji Find

$\log^* n$ — $\log(\log(\log(\dots\log(n)\dots)))$

Dowód czasu zamortyzowanego

Weźmy sekwencję m unionów i findów.

drzewo o wysokości r ma max 2^r wierzchołków



Ponad to wiemy, że wierzchołków o wysokości r jest $\frac{n}{2^k}$

Zdefiniujmy logarytm iteracyjny

$$\log^* n = \log(\log(\ldots(\log n)\ldots))$$

jako odwrotność funkcji:

$$F(k) = \begin{cases} 1 & k = 0 \\ 2^{(k-1)} & \text{wpp.} \end{cases}$$

np. $F(2) = 2^{2^1} = 4$ $\quad F(4) = 2^{2^{2^1}} = 2^8 = 256$

wtedy:

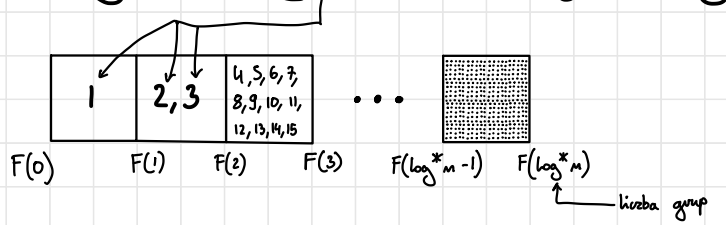
$\log^* 2 = 1$

$\log^* 3 = 2$

$\log^* 4 = 2$

$\log^* 5 = 3$

Podzielmy sobie wysokości n (rzędy) na grupy:



Wykonamy teraz księgową analizę zamortyzowanej złożoności czasowej:

Obciążać będziemy funkcję Find(x) w 3 przypadkach:

- x jest korzeniem
- x jest synem korzenia

← Bo wtedy niczego nie optymalizujemy

- x i jego ojciec są w różnych grupach

w reszcie przypadków obciążamy wierzchołek.

Ile nas kosztuje funkcja find gdy ją obciążamy?

$$koszt(find) = 1 + 1 + \log^* n = O(\log^* n)$$

Ile nas kosztuje obciążenie każdego wierzchołka?

- w każdej grupie mamy $(F(g) - (F(g-1)+1))$ rzędów.
- w każdym rzędzie jest $\frac{n}{2^k}$ wierzchołków.
- koszt każdego wierzchołka w danej grupie będzie wynosić conajwyżej $F(g)$

wszystkie rzędy → ↓ wszystkie wierzchołki

$$\text{koszt}(\forall V \text{ dla grupy } g) = \sum_{r=F(g-1)+1}^{F(g)} \frac{n}{2^r} F(g) = \frac{n}{2^{F(g-1)}} F(g) \sum_{r=0}^{F(g)-F(g-1)-1} \frac{1}{2^{r+1}} \leq$$

↑ wszystkich wierzchołków

$$\frac{2n}{2^{F(g-1)}} F(g) = \frac{n}{F(g)} F(g) = n$$

Wiemy, że grup mamy $\log^* n$, zatem

$$\text{koszt}(\forall V) = n \log^* n$$

zatem dla pojedynczego wierzchołka $\log^* n$.