Rodziny zbiorów rozłącznych

Pewien zestaw elementów organizujemy w rodziny zbiorów rozłącznych. Każdy zbiór posiada jeden wyróżniony element, który jest *reprezentantem zbioru* zawierającego ten element.

Operacje:

`MakeSet` – tworzy jednoelementowy zbiór

`FindSet(x)` – daje jako wynik reprezentanta zbioru, do którego należy `x`

`Union(x,y)` – łączy zbiory, których reprezentantami są `x` i `y` w jeden zbiór

Przykład: wyznaczanie składowych spójności w grafie nieskierowanym

Dany jest graf `G=(V,E)`.

Zdefiniować funkcję `SameComponents(x,y)`, która dla wierzchołków `x`, `y` odpowie, czy należą one do tej samej składowej spójności w grafie, czyli czy istnieje w grafie ścieżka łącząca `x` i `y`.

Na przykład

```
V = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

```
E = {(1,2) (1,3) (2,3) (2,4) (5,6) (5,7) (8.9)}
```

```
SameComponents(1,4)
```

```
SameComponents(1,6)
```

```
ConnectedComponenets(G)
// G=(V,E) - graf nieskierowany
// V - zbiór wierzchołków
// E - zbiór krawędzi
// organizuje wierzchołki w rodzinę zbiorów odpowiadających
// składowym spójności grafu

  dla każdego wierzchołka v
    Makeset(v)
  dla każdej krawędzi (u,v)
    ru = FindSet(u)
    rv = FindSet(v)
    if ru != rv
      Union(ru,rv)


SameComponents(x,y)
// zwraca "true", jeżeli wierzchołki  x,y należą do tej samej
// składowej spójności, czyli gdy istnieje w grafie ścieżka
// łącząca x i y.
  return FindSet(x) == FindSet(y)
```

`E = {(1,2) (1,3) (2,3) (2,4) (5,6) (5,7) (8.9)}`

Rodzina zbiorów:

$\{\mathbf{1}\}$ $\{\mathbf{2}\}$ $\{\mathbf{3}\}$ $\{\mathbf{4}\}$ $\{\mathbf{5}\}$ $\{\mathbf{6}\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (1,2): `Union(1,2)`

$\{\mathbf{1}, 2\}$ $\{\mathbf{3}\}$ $\{\mathbf{4}\}$ $\{\mathbf{5}\}$ $\{\mathbf{6}\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (1,3): `Union(1,3)`

$\{\mathbf{1}, 2, 3\}$ $\{\mathbf{4}\}$ $\{\mathbf{5}\}$ $\{\mathbf{6}\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (2,3):

$\{\mathbf{1}, 2, 3\}$ $\{\mathbf{4}\}$ $\{\mathbf{5}\}$ $\{\mathbf{6}\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (2,4): `Union(2,4)`

$\{\mathbf{1}, 2, 3, 4\}$ $\{\mathbf{5}\}$ $\{\mathbf{6}\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (5,6): `Union(5,6)`

$\{\mathbf{1}, 2, 3, 4\}$ $\{\mathbf{5}, 6\}$ $\{\mathbf{7}\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (5,7): `Union(5,7)`

$\{\mathbf{1}, 2, 3, 4\}$ $\{\mathbf{5}, 6, 7\}$ $\{\mathbf{8}\}$ $\{\mathbf{9}\}$ $\{\mathbf{10}\}$

krawędź (8,9): `Union(8,9)`

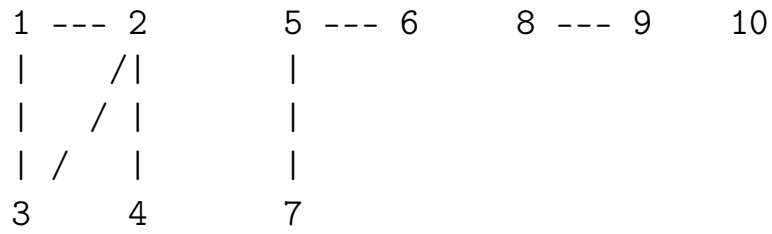
$\{\mathbf{1}, 2, 3, 4\}$ $\{\mathbf{5}, 6, 7\}$ $\{\mathbf{8}, 9\}$ $\{\mathbf{10}\}$

`SameComponents(1,4) :  FindSet(1) == FindSet(4) -> true`

`SameComponents(1,6) :  FindSet(1) == FindSet(6) -> false`

V = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

E = {(1,2) (1,3) (2,3) (2,4) (5,6) (5,7) (8.9)}



$\{\mathbf{1}, 2, 3, 4\}$ $\{\mathbf{5}, 6, 7\}$ $\{\mathbf{8}, 9\}$ $\{\mathbf{10}\}$

SameComponents(1,4) : FindSet(1) == FindSet(4) -> true

SameComponents(1,6) : FindSet(1) == FindSet(6) -> false

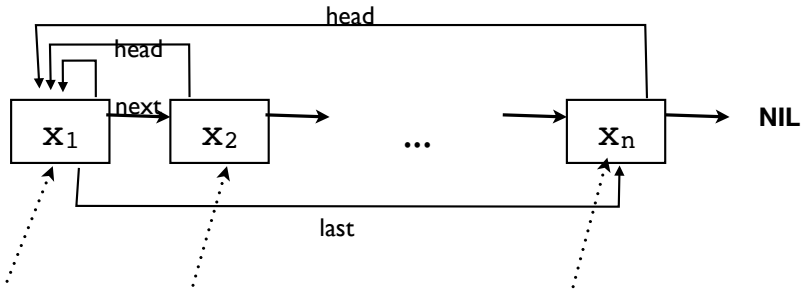
Reprezentacja listowa bez wyważania

Struktura węzła `x` :

- `x.key` - klucz związany z elementem `x` (istotny tylko dla wydruku)
- `x.next` - wskaźnik do następnego
- `x.head` - wskaźnik do reprezentanta zbioru czyli pierwszego elementu listy; pierwszy wskazuje sam na siebie
- `x.last` - wskaźnik do ostatniego elementu listy; określony tylko dla pierwszego elementu listy

Zbiór to lista: $\mathtt{x}_1 \rightarrow \mathtt{x}_2 \rightarrow \ldots \rightarrow \mathtt{x}_n \rightarrow \mathtt{NIL}$

a dokładniej:



Operacje:

```
MakeSet(k)
// utworzenie zbioru reprezentującego jednoelementowy zbiór z kluczem k
   utworz węzeł x
   x.key = k
   x.head = x
   x.last = x
   x.next = NIL
   return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
   return x.head


Union(x,y)
// zamienia zbiory o reprezentantach x,y w jeden zbior będący
// ich sumą rozłączną
   x.last.next = y
   x.last = y.last
   while y != NIL   // aktualizacja pola "head" dla dołączonej listy
       y.head = x
       y = y.next
```

x1 → x2 → x3 → x4 → NIL

y1 → y2 → NIL

Union(x1,y1):

x1 → x2 → x3 → x4 → y1 → y2 → NIL

Reprezentacja listowa z wyważaniem

Struktura węzła `x` :

Tak jak w reprezentacji listowej bez wyważania plus dodatkowo:

`x.length` - długość listy, której pierwszym elementem jest x;
określone tylko dla pierwszego elementu listy

Operacje:

```
MakeSet(k)
// utworzenie zbioru reprezentującego jednoelementowy zbiór z kluczem k
   utworz węzeł x
   x.key = k
   x.head = x
   x.last = x
   x.next = NIL
   x.length = 1
   return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
   return x.head

Union(x,y)
// łączy zbiory o reprezentantach x,y w jeden zbior będący ich sumą
// rozłączną; krótszą listę doczepia na koniec dłuższej
   if y.length > x.length
      zamień x<->y   // lista y jest teraz nie dłuższa niż x
   x.length =  x.length + y.length
   x.last.next = y
   x.last = y.last
   while y != NIL
      y.head = x
      y = y.next
```

Złożoność czasowa

Szacujemy pesymistyczną złożoność czasową ciągu m operacji `MakeSet`, `FindSet`, `Union`, w którym jest n operacji `MakeSet`.

Stwierdzenie 1. Dla reprezentacji listowej bez wyważania powyższa złożoność jest $\Theta(m^2)$ dla $n = \lceil m/2 \rceil + 1$.

Dowód

`MakeSet`(v_1) ... `MakeSet`(v_n)

`Union`(v_2, v_1) $v_2 \to v_1$, 1 aktualizacja pola `head`

`Union`(v_3, v_2) $v_3 \to v_2 \to v_1$, 2 aktualizacje `head`

`Union`(v_4, v_3) $v_4 \to v_3 \to v_2 \to v_1$, 3 aktualizacje `head`

.....

`Union`(v_n, v_{n-1}) $v_n \to v_{n-1} \to \ldots \to v_1$, n-1 aktualizacji `head`

Razem czas:

$$\Theta(n) + \Theta(n-1) + \Theta(1 + 2 + \ldots + (n-1)) = \Theta(n^2) = \Theta(m^2)$$

Szacujemy pesymistyczną złożoność czasową ciągu m operacji `MakeSet`, `FindSet`, `Union`, w którym jest n operacji `MakeSet`.

Stwierdzenie 2. Dla reprezentacji listowej z wyważaniem powyższa złożoność jest $O(m + n \lg n)$.

Dowód

Weźmy dowolny węzeł x i policzmy ile razy mogło być aktualizowane pole `head` dla tego węzła

po pierwszej aktualizacji `head` (w wyniku operacji `Union` , w której brał udział x): x jest on na liście o długości ≥ 2

po drugiej aktualizacji `head` : x jest na liście o długości ≥ 4

po trzeciej aktualizacji `head` : x jest na liście o długości ≥ 8

po k-tej aktualizacji `head` : x jest na liście o długości $\geq 2^k$

Mamy tylko n elementów, więc $2^k \leq n$. Zatem maksymalna ilość aktualizacji pola `head` dla każdego element x jest $lg\, n$. Elementów jet n, a wszystkich operacji m. Stąd $O(m + n \lg n)$