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

```
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)
```

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

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
       y.head = x
       y = y.next
```

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
```

Reprezentacja drzewiasta

Struktura węzła `x` :

`x.key` - klucz związany z elementem `x` (istotny tylko dla wydruku)
`x.p` - wskaźnik do ojca; korzeń wskazuje sam na siebie

Ponadto dla wersji z rangą

`x.rank` - ranga drzewa czyli oszacowanie wysokości, określone tylko dla korzenia

Operacje:

```
MakeSet(k)
// utworzenie zbioru reprezentującego jednoelementowy zbiór z kluczem k
   utworz węzeł x
   x.key = k
   x.p = x
   x.rank = 0 // tylko dla wersji z rangą
   return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
// Wersja bez kompresji
   if x != x.p
      return FindSet(x.p)
   else
      return x

FindSet(x)
// zwraca reprezentanta zbioru do którego należy element (czyli węzeł) x
// Wersja z kompresją
   if x != x.p
       x.p = FindSet(x.p)
   return x.p
```

```
Union(x,y)
// zamienia zbiory o reprezentantach x,y w jeden zbior będący ich sumą rozłączną
// Wersja bez rangi
   y.p=x
```

```
Union(x,y)
// zamienia zbiory o reprezentantach x,y w jeden zbior będący ich sumą rozłączną
// Wersja z rangą
   if x.rank > y.rank
      y.p = x
   else
      x.p = y
      if x.rank==y.rank
         y.rank = y.rank+1
```

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 $n = \lceil m/2 \rceil + 1$.

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

Stwierdzenie 3. Dla reprezentacji drzewiastej bez rangi i bez kompresji ścieżek powyższa złożoność jest $\Theta(m\,n)$.

Stwierdzenie 4. Dla reprezentacji drzewiastej z rangą i bez kompresji ścieżek powyższa złożoność jest $O(m \lg n)$.

Stwierdzenie 5. Dla reprezentacji drzewiastej z rangą i z kompresją ścieżek powyższa złożoność jest $O(m \cdot \alpha(m, n))$, gdzie $\alpha(m, n)$ jest pewną bardzo wolno rosnącą funkcją. Dla wartości m, n spotykanych w praktyce można przyjąć, że $\alpha(m, n) \leq 5$, zatem powyższa złożoność to $O(m)$.