Zbiory rozłączne

Zakładamy małe uniwersum: $\{1 \ldots n\}$.
Struktura ma reprezentować rodzinę rozłącznych zbiorów dynamicznych: $\mathcal{S} = \{S_1 \ldots S_k\}$, gdzie każdy zbiór – identyfikowany przez swojego *reprezentanta*.
Operacje:

- `Make-Set(`x`)` tworzy zbiór $\{x\}$
- `Union(`x`,`y`)` łączy zbiory zawierające x i y w ich sumę.
- `Find-Set(`x`)` znajduje reprezentanta zbioru zawierającego x

Oznaczenia: n – liczba operacji `Make-Set`, m – łączna liczba wszystkich operacji.
Stąd: $m \geq n$ oraz łączna liczba `Union` $\leq n - 1$ (po $n - 1$ `Union` powstaje 1 zbiór).

przykład zastosowania

$G = (V[G], E[G])$ – graf nieskierowany.

```
Connected-Components(G)
for each v ∈ V[G] do
    Make-Set(v)
for each (u, v) ∈ E[G] do
    if Find-Set(u)≠Find-Set(v) then
        Union(u,v)
```

Po wykonaniu `Connected-Components(G)` można wykonywać sprawdzenie:

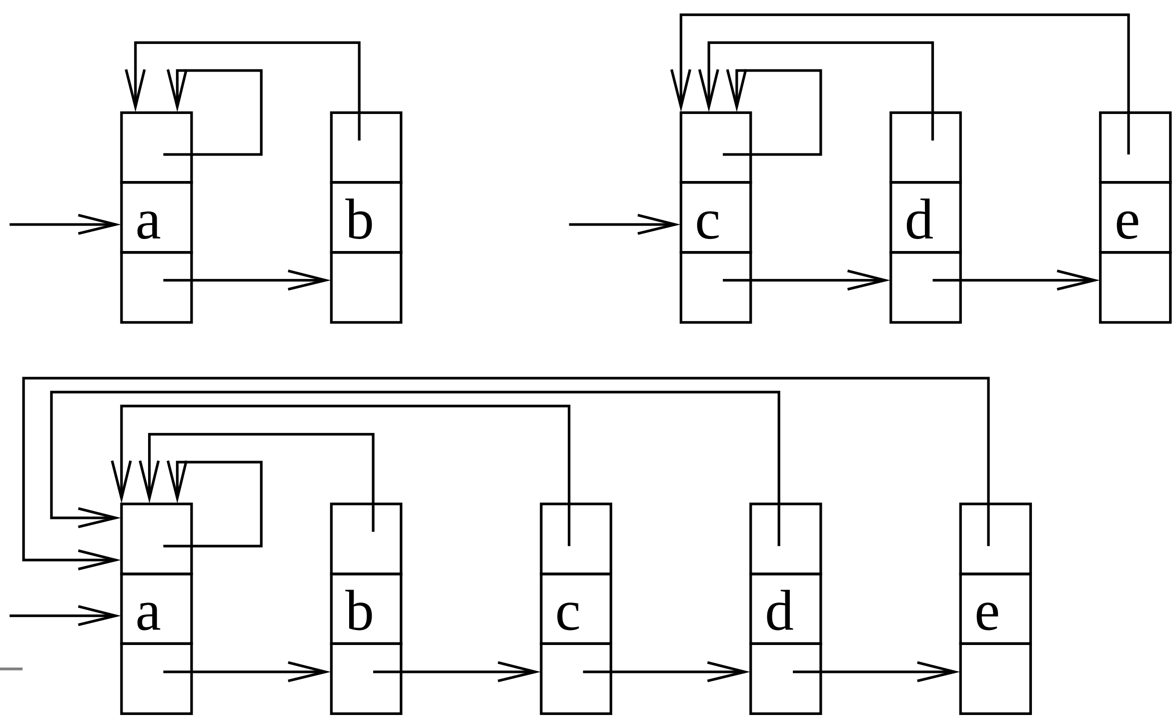
```
Same-Component(u,v)
if Find-Set(u)=Find-Set(v)
    then return TRUE
    else return FALSE
```

(Jeśli nowe krawędzie są dodawane “dynamicznie”, to łatwo można poprawiać składowe wykonując `Union`)

listowa reprezentacja zbiorów

Pierwszy element – reprezentant zbioru.
Każdy element listy zawiera pola:

- element zbioru
- wskaźnik na następnika
- wskaźnik na reprezentanta



łączenie zbiorów

`Union(`x`,`y`)`
dołączamy listę z elementem x na koniec listy z elementem y i poprawiamy reprezentantów listy x na reprezentanta y.

Przykład m operacji wymagających czasu $\Theta(m^2)$. Niech $n = \lceil m/2 \rceil + 1$ oraz $q = m - n = \lfloor m/2 \rfloor - 1$, oraz $x_1 \ldots x_n$ – elementy. Wykonujemy ciąg operacji:
`Make-Set(`x_1`)`,`Make-Set(`x_2`)`,...,`Make-Set(`x_n`)`,
`Union(`x_1`,`x_2`)`,`Union(`x_2`,`x_3`)`, ...,`Union(`x_{q-1}`,`x_q`)`
Początkowe n operacji (`Make-Set`) – czas: $\Theta(n)$.
Końcowe $q-1$ operacji (`Union`) – czas: $\sum_{i=1}^{q-1} i = \Theta(q^2)$, bo i-ta z tych operacji dokleja listę długości i na koniec listy długości 1.
Całkowity czas: $\Theta(n+q^2) = \Theta(m^2)$ bo $n = \Theta(m)$ i $q = \Theta(m)$.
Średni (zamortyzowany) czas jednej operacji: $\Theta(m)$.

Heurystyka z wyważaniem

Zakładamy, że reprezentant zawiera informację o długości swej listy i że zawsze doklejamy krótszą listę na koniec dłuższej.

Czas pojedynczej operacji `Union` może nadal być $\Omega(m)$.

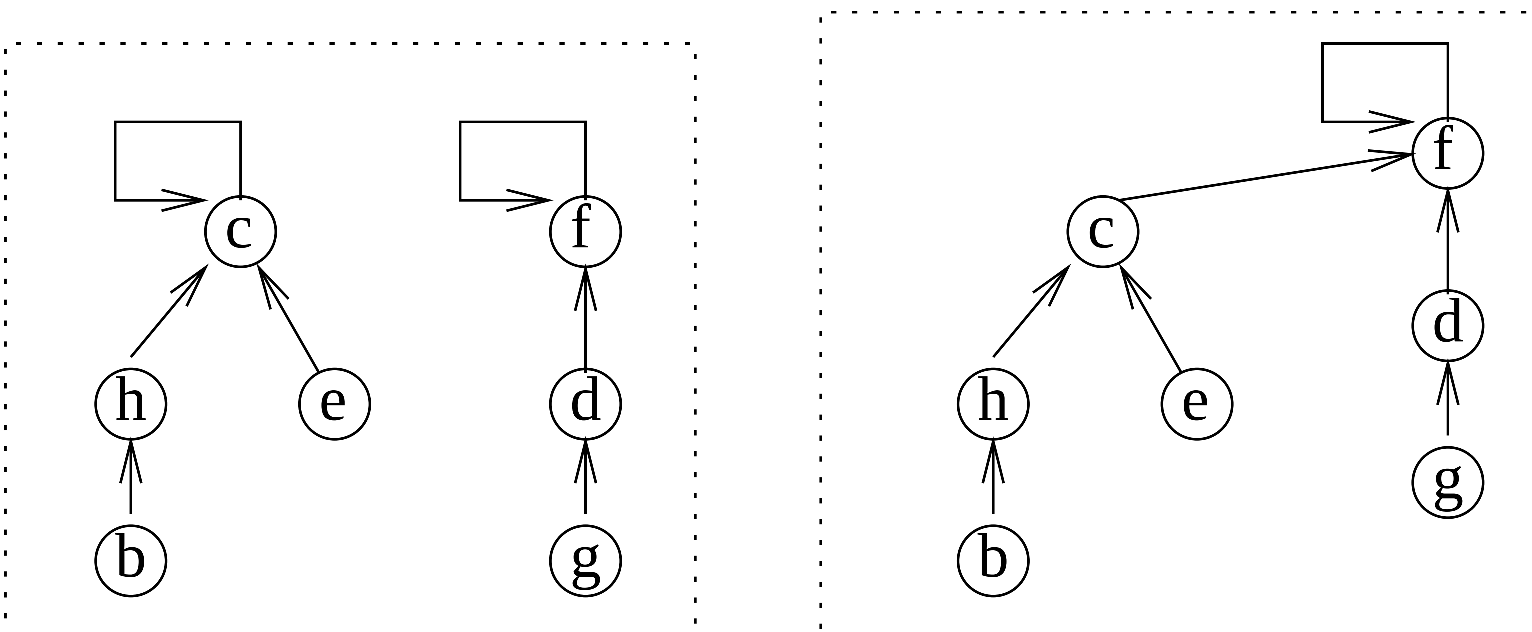
Tw. Jeśli stosujemy heurystykę z wyważaniem, to ciąg m operacji `Make-Set`,`Union`, `Find-Set`, spośród których n operacji to `Make-Set`, zajmuje czas: $O(m + n \lg n)$.

D-d. *Górne oszacowanie liczby uaktualnień wskaźnika na reprezentanta w dowolnym* x: Za każdym razem, gdy reprezentant x był zmieniany – x znajdował się w mniejszym zbiorze. Zatem, po każdym takim uaktualnieniu zbiór zawierający x stawał się ≥ 2 razy większy. (Po k uaktualnieniach – jego rozmiar wynosi: $\geq 2^k$.) Ponieważ każdy zbiór ma rozmiar $\leq n$, więc liczba uaktualnień $\leq \lg n$.

Łączny koszt uaktualniania n obiektów w trakcie wszystkich operacji `Union`: $O(n \lg n)$.
Każda operacja `Make-Set` i `Find-Set` – czas: $O(1)$.
Łączny czas tych operacji: $O(m)$.
Stąd całkowity czas całego ciągu operacji: $O(m + n \lg n)$. $\square$

Lasy zbiorów rozłącznych

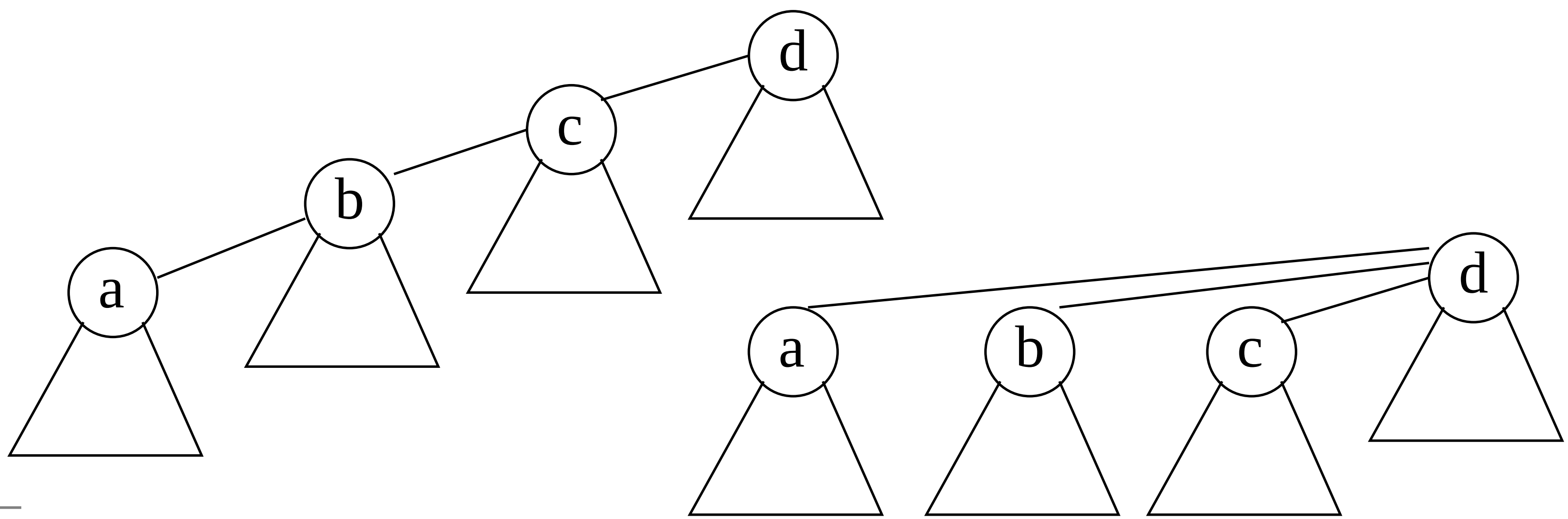
Każde drzewo – zbiór.
Korzeń – reprezentant zbioru.
W węźle x: p$[x]$ – wskaźnik do ojca (jeśli x jest korzeniem, to wskaźnik do x).
`Union(`x`,`y`)` – ustalenie reprezentanta jednego elementu jako ojca reprezentanta drugiego elementu.



Heurystyki poprawiające efektywność

Łączenie wg rangi: Ranga (atrybut *rank*$[x]$) – przybliżenie logarytmu z liczby elementów w poddrzewie x. W `Union` podpinamy korzeń o mniejszej randze pod korzeń o większej randze.

Kompresja ścieżki: Znajdowanie reprezentanta zbioru zawierającego x – spacer po ścieżce od x do korzenia. Zmieniamy wskaźnik *p* elementów na tej ścieżce aby wskazywał bezpośrednio na korzeń.



Operacje na lesie zbiorów

Make-Set(x)

$p[x] \leftarrow x$

$rank[x] \leftarrow 0$

Operacje na lesie zbiorów

```
Make-Set(x) ▷
p[x] ← x
rank[x] ← 0

Union(x,y) ▷
Link(Find-Set(x),Find-Set(y))
```

Operacje na lesie zbiorów

```
Make-Set(x) ▷
p[x] ← x
rank[x] ← 0

Union(x,y) ▷
Link(Find-Set(x),Find-Set(y))

Link(x,y) ▷ x,y - korzenie
if rank[x] > rank[y]
   then p[y] ← x
   else p[x] ← y
        if rank[x] = rank[y]
           then rank[y] ← rank[y] + 1
```

Operacje na lesie zbiorów

```
Make-Set(x) ▷
p[x] ← x
rank[x] ← 0

Union(x,y) ▷
Link(Find-Set(x),Find-Set(y))

Link(x,y) ▷ x,y - korzenie
if rank[x] > rank[y]
   then p[y] ← x
   else p[x] ← y
        if rank[x] = rank[y]
           then rank[y] ← rank[y] + 1

Find-Set(x) ▷
if x ≠ p[x]
   then p[x] ← Find-Set(p[x])
return p[x]
```

Analiza

Definicja.

$$\lg^{(i)} n = \begin{cases} n, & \text{dla } i = 0 \\ \lg(\lg^{(i-1)} n), & \text{dla } i > 0 \text{ i } \lg^{(i-1)} n > 0 \\ \text{nieokreślone}, & \text{jeśli } i > 0 \text{ i } \lg^{(i-1)} n \leq 0 \\ & \text{lub } \lg^{(i-1)} n \text{ nieokreślone} \end{cases}$$

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}$$

Przykłady: $\lg^* 0 = \lg^* 1 = 0$, $\lg^* 2 = 1$,

$$n = \lg^* \left. 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \right\} {\color{red} n}$$

Własności rang

Lemat 1. Dla każdego x, *rank*$[x]$ $\leq$*rank*[*p*$[x]$] (przy czym *rank*$[x]$ $<$*rank*[*p*$[x]$], jeśli $x \neq$*p*$[x]$). Wartość *rank*$[x]$ jest początkowo $= 0$ i rośnie do chwili gdy $x \neq$*p*$[x]$; od tego momentu *rank*$[x]$ sie nie zmienia. Wartość *rank*[*p*$[x]$] wzrasta przy każdej zmianie *p*$[x]$.

D-d. Prosta indukcja względem liczby operacji (ćw.). $\square$

Niech *size*(x) – rozmiar poddrzewa x.

Lemat 2. Dla dowolnego korzenia x zachodzi

size$(x) \geq 2^{\textit{rank}[x]}$

D-d. Indukcja wzgl. liczby operacji `Link`. (Operacje `Find-Set` nie zmieniają rang korzeni ani rozmiarów ich drzew.)

Podstawa indukcji: Przed pierwszym wywołaniem `Link` rangi są $= 0$.

...

Własności rang

Krok indukcyjny: Wykonujemy `Link(`x`,`y`)`. Niech *rank* – ranga węzła przed wykonaniem, a *rank'* – po wykonaniu. (Podobnie – *size* i *size'*.)

Przypadek rank$[x]$ $<$*rank*$[y]$: y zostaje korzeniem, oraz:

$\textit{size'}(y) = \textit{size}(x) + \textit{size}(y) \geq 2^{\textit{rank}[x]} + 2^{\textit{rank}[y]} \geq 2^{\textit{rank}[y]}$
$= 2^{\textit{rank'}[y]}$

Rangi ani rozmiary poddrzew pozostałych węzłow się nie zmieniają.

Przypadek rank$[x]$ $>$*rank*$[y]$: analogiczny.

Przypadek rank$[x]$ $=$*rank*$[y]$: y zostaje korzeniem, oraz:

$\textit{size'}(y) = \textit{size}(x) + \textit{size}(y) \geq 2^{\textit{rank}[x]} + 2^{\textit{rank}[y]} = 2 \cdot 2^{\textit{rank}[y]}$
$= 2^{\textit{rank}[y]+1} = 2^{\textit{rank'}[y]}$

$\square$

Własności rang

Lemat 3. Dla $r \geq 0$ istnieje $\leq n/2^r$ węzłów rangi r.

Ustalmy r. Załóżmy, że przypisując rangę r węzłowi x (wiersz 2 w `Make-Set` lub 5 w `Link`), nalepiamy etykietę x na każdy węzeł poddrzewa x.

Jeśli x przestanie być korzeniem, to (z Lematu 1) wszyscy jego nowi przodkowie będą mieli rangę $\geq r+1$.

Stąd: Każdy węzeł otrzymuje etykietę co najwyżej raz.

Każdy węzeł rangi r powoduje nadanie $\geq 2^r$ etykiet (Lemat 2). Gdyby istniało $> n/2^r$ węzłów rangi r, to etykiety otrzymałoby $> 2^r \cdot (n/2^r) = n$ węzłów – sprzeczność.

$\square$

Wn. Ranga żadnego węzła nie przekracza $\lfloor \lg n \rfloor$.

D-d. Dla $r > \lg n$, $n/2^r < 1$. $\square$

Górne oszacowanie

Lemat. Dany ciąg S' złożony z m' operacji `Make-Set`, `Union` i `Find` przekształcamy na ciąg S złożony z m operacji `Make-Set`, `Link` i `Find-Set`, zamieniając każde `Union` na (wywoływane w nim) dwa `Find-Set` i jedno `Link`. Jeśli S wykonuje się w czasie $O(m \lg^* n)$, to S' wykonuje się w czasie $O(m' \lg^* n)$.

D-d. Czas wykonania S i S' – taki sam. Każda operacja `Union` – zastąpiona prez 3 operacje. Stąd: $m' \leq m \leq 3m'$. Ponieważ $m = O(m')$, z oszacowania $O(m \lg^* n)$ na czas wykonania S wynika oszacowanie $O(m' \lg^* n)$ na czas wykonania S'. $\square$

Górne oszacowanie

Tw. Ciąg m operacji `Make-Set`, `Link` i `Find-Set`, spośród których n to operacje `Make-Set`, w reprezentacji lasu zbiorów, z łączeniem wg rangi i kompresją ścieżki, wykonuje się w czasie $O(m \lg^* n)$.

D-d. Metoda kosztu sumarycznego. Liczymy sumę *opłat* równą faktycznemu kosztowi ciągu operacji. Opłata za `Make-Set` lub `Link`: $O(1)$.

Opłata za `Find-Set`: Dzielimy zbiór rang węzłów na bloki: ranga r – w bloku $\lg^* r$, dla $0 \leq r \leq \lfloor \lg n \rfloor$ ($\lfloor \lg n \rfloor$ – największa możliwa ranga (**Wn.**)). Najwyższym numerem bloku jest $\lg^*(\lg n) = \lg^* n - 1$.

...

granice bloków

Dla $j \geq -1$, niech:

$$B(j) = \begin{cases} -1 & \text{dla } j = -1 \\ 1 & \text{dla } j = 0 \\ 2 & \text{dla } j = 1 \\ \left. 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}} \right\} j & \text{dla } j \geq 2 \end{cases}$$

Wtedy, dla $j = 0$,. . . ,$\lg^* n - 1$, j-ty blok to:

$$\{B(j-1)+1, \ldots, B(j)\}$$

Opłaty za `Find-Set`

Dwa rodzaje opłat za `Find-Set`: *opłaty za bloki*, *opłaty za ścieżki*.
Załóżmy, że wywołujemy `Find-Set(`x_0`)` oraz $x_0, \ldots, x_l$ – ścieżka z x_0 do korzenia. (T.j., dla $i = 1, \ldots, l$, $p[x_{i-1}] = x_i$ oraz $p[x_l] = x_l$.)
Dla $j = 0, \ldots, \lg^* n - 1$, przypisujemy opłatę 1 zł za blok *ostatniemu* węzłowi na ścieżce o randze należącej do bloku j. (Z Lematu 1: węzły o rangach z jednego bloku – kolejno obok siebie na ścieżce.) Zawsze przypisujemy 1 zł za blok synowi korzenia (t.j. x_{l-1}). (Równoważna zasada: x_i dostaje 1 zł za blok jeśli $p[x_i] = x_l$ lub $\lg^*$*rank*$[x_i] < \lg^*$*rank*[*p*$[x_i]$].)
Pozostałym węzłom przypisujemy opłatę 1 zł za ścieżkę.

Opłaty za `Find-Set`

Uwagi:

- Suma wszystkich opłat za wszystkie `Find-Set` jest równa łącznej liczbie odwiedzonych wierzchołków w operacjach `Find-Set`
- Łączna suma opłat za bloki: $m \cdot (\lg^* n + 1)$ (Bloki numerowane od 0 do $\lg^* n - 1$ ($\lg^* n$ numerów), więc w każdym `Find-Set`: $\leq \lg^* n + 1$ opłat za bloki.)
- Jeśli x nie jest korzeniem ani synem korzenia i x otrzymał opłatę za blok, to x już nigdy nie otrzyma opłaty za ścieżkę. (ranga x już się nie zmieni, a przy każdej zmianie p$[x]$ ranga p$[x]$ rośnie (Lemat 1), więc ranga p$[x]$ zawsze już będzie w innym bloku.)

Opłaty za ścieżkę

Jeśli x_i otrzymuje opłatę za ścieżkę, to przed kompresją *p*$[x_i] \neq x_l$, czyli zmienia się *p*$[x_i]$ (i wzrasta *rank*[*p*$[x_i]$]).
Załóżmy, że ranga x_i w bloku j. Ile razy może wzrosnąć *rank*[*p*$[x_i]$] pozostając w tym samym bloku co *rank*$[x_i]$?
Najgorszy przypadek: *rank*$[x_i] = B(j-1)+1$, oraz *rank*[*p*$[x_i]$] przyjmuje kolejno wszystkie z $B(j) - B(j-1) - 1$ pozostałych wartości w bloku j: $B(j-1)+2, \ldots, B(j)$. (T.j. x_i dostanie $\leq (B(j) - B(j-1) - 1)$ zł za ścieżkę.)
Niech $N(j)$ liczba węzłów o rangach w bloku j. Z Lematu 3 mamy:

$$N(j) \leq \sum_{r=B(j-1)+1}^{B(j)} \frac{n}{2^r}$$

Opłaty za ścieżkę

Dla $j = 0$: $N(0) = n/2^0 + n/2^1 = 3n/2 = 3n/(2B(0))$
Dla $j \geq 1$:

$$\begin{aligned} N(j) &\leq \tfrac{n}{2^{B(j-1)+1}} \sum_{r=0}^{B(j)-(B(j-1)+1)} \tfrac{1}{2^r} \\ &< \tfrac{n}{2^{B(j-1)+1}} \sum_{r=0}^{\infty} \tfrac{1}{2^r} \\ &= \tfrac{n}{2^{B(j-1)}} \\ &= \tfrac{n}{B(j)} \end{aligned}$$

Zatem $N(j) \leq 3n/(2B(j))$ dla wszystkich $j \geq 0$.

Opłaty za ścieżkę

Niech $P(n)$ – całkowita opłata za ścieżki w ciągu operacji `Find-Set`:

$$\begin{aligned} P(n) &\leq \sum_{j=0}^{\lg^* n-1} \frac{3n}{2B(j)}(B(j) - B(j-1) - 1) \\ &\leq \sum_{j=0}^{\lg^* n-1} \frac{3n}{2B(j)}B(j) \\ &= \tfrac{3}{2} n \lg^* n \end{aligned}$$

Zatem łączna opłata przypisywana węzłom podczas operacji `Find-Set` wynosi $O(m(\lg^* n + 1) + n \lg^* n) = O(m \lg^* n)$, gdyż $m \geq n$.
Uwzględniając jeszcze $O(n)$ operacji `Make-Set` i `Link`, wnoszących jednostkowe opłaty, mamy oszacowanie kosztu ciągu wszystkich operacji: $O(m \lg^* n)$. $\square$

Uwagi

- $\lg^*$63536=4, $\lg^* 2^{63536} = 5$, zatem we wszystkich *praktycznych* zastosowaniach mamy $\lg^* n \leq 5$.
- Niech $A(i,j)$ funkcja Ackermana określona dla $i,j \geq 0$:

$$
\begin{array}{ll}
A(1,j) = 2^j & \text{dla } j \geq 1 \\
A(i,1) = A(i-1,2) & \text{dla } i \geq 2 \\
A(i,j) = A(i-1, A(i,j-1)) & \text{dla } i,j \geq 2
\end{array}
$$

Odwrotność f-cji Ackermana definiujemy wzorem:

$$\alpha(m,n) = \min\{i \geq 1 : A(i, \lfloor m/n \rfloor) > \lg n\}$$

Istnieje (trudniejsze) oszacownie $O(m\alpha(m,n))$ na koszt ciągu m operacji `Make-Set`, `Union`, `Find-Set`.