# Kopiec Fibonacciego

Autor: Paweł Jelonek

Kopiec Fibonacciego to struktura danych realizująca operacje kolejki priorytetowej, składająca się z kolekcji drzew z porządkiem kopcowym. Kopce te mają lepszy czas zamortyzowany, niż wiele innych implementacji kolejek priorytetowych, w tym kopce binarne i dwumianowe. Michael L. Friedman i Robert E. Tarjan odkryli kopce Fibonacciego w 1984 roku i opublikowali ich opis w czasopiśmie naukowym w 1987 roku. Nazwali je kopcami Fibonacciego w nawiązaniu do liczb Fibonacciego, które są używane do ich analizy.

Dla kopców Fibonacciego operacja znalezienia minimum zajmuje czas stały (*O*(1)) w sensie zamortyzowanym, podobnie jak operacje wstawiania oraz zmniejszania klucza. Usuwanie elementu działa w czasie zamortyzowanym *O*(log *n*), gdzie *n* to rozmiar stosu. Oznacza to, że zaczynając od pustej struktury, dowolny ciąg a operacji wstawiania oraz zmniejszania kluczy i b operacji usunięć zajmie O(a + b log n) czasu (najgorszy przypadek), gdzie n to maksymalny rozmiar kopca. Taki sam ciąg operacji w kopcu dwumianowym miałby złożoność O((a + b) log n ) Możliwa jest także operacja łączenia dwóch kopców Fibonacciego w stałym czasie (zamortyzowanym).

Przykład działania operacji push():

Mamy dwa kroki:

1. Tworzymy nowe drzewo z samym korzeniem.
2. Dodajemy nowe drzewo na koniec listy korzeni(uwzględniając możliwość dodania nowego minimum!)

Złożoność operacji O(1)

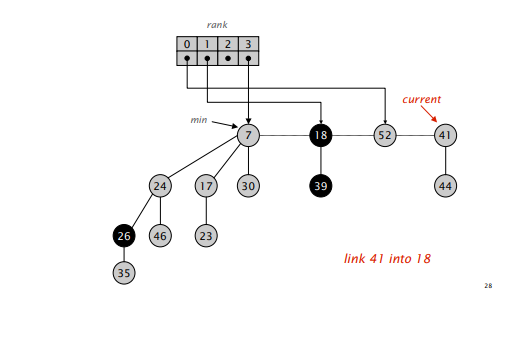
Przykład działania operacji pop() – usuwania minimum:

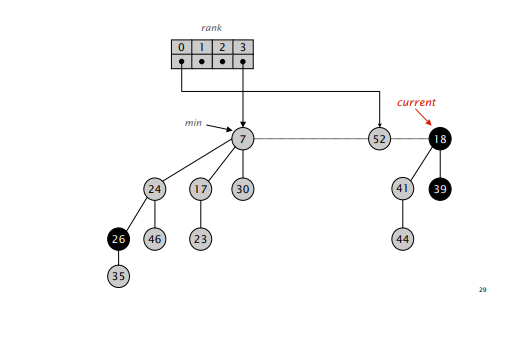
1. Usuwamy obecne minimum, jeśli minimum posiada dzieci to dodać je na koniec listy korzeni, zaktualizuj minimum
2. Konsoliduj\* listę korzeni

Konsolidacja kopca Fibonacciego – jest to operacja polegają na łączeniu korzeni drzew w taki sposób, że drzewo z większym kluczem staje się dzieckiem drzewa z mniejszym kluczem. Łączone są tylko drzewa z taką samą liczbą dzieci. Operacja jest wykonywana tak długo, aż na kopcu nie zostanie ani jedno drzewo z taką samą liczbą dzieci.

Przykład z <https://www.cs.princeton.edu/~wayne/teaching/fibonacci-heap.pdf>

Na poniższym rysunku zamalowaliśmy drzewo z kluczem 18 na czarno, to drzewo ma klucz 18 i range 1, następnie iterując dalej liste drzew natkniemy się na drzewo z kluczem 41 – oznaczone jako current. Ich ranga jest taka sama – ma wartość 1- więc je łączymy.





Złożoność operacji:

- O(log n) by dodać dzieci do listy korzenia

- O(log n) by zaktualizować min

- O(log n) by skonsolidować drzewo

Ogólna złożoność: O(log n)

Złożoność pesymistyczna: O(n\*log n)

Przykład działania operacji decrease\_key():

- Znajdź korzeń z kluczem przeszukując drzewo i zmień na nowy klucz.

- Jeżeli porządek jest zachwiany tj. klucz dziecka < klucz rodzica to:

- Usuń drzewo z zmienionym kluczem i dodaj do listy korzeni

- Sprawdź czy porządek jest zachowany w relacji korzenia z dziećmi

- Jeśli nie to dodaj dzieci na koniec listy korzeni

- Jeżeli rodzic korzenia któremu zmienialiśmy klucz jest oznaczony:

- Usuń rodzica i dodaj do listy korzeni odznaczając go w trakcie

- Jeżeli nie jest oznaczony, oznacz go.

-Sprawdzenie rodzica wykonuj rekursywnie do momentu trafienie na nieoznaczonego rodzica.