Kolejka priorytetowa

- **Kolejka priorytetowa** – struktura danych (podobna do kolejki, czy też stosu), przetwarzająca elementy o uporządkowanych wartościach.
- Obsługuje dwie operacje:
  - Pobierz i usuń element maksymalny lub minimalny (o najwyższym „priorytecie”)
  - Wstaw element

Kolejka priorytetowa

Opis	Metoda
Konstruktor	`MaxPQ()`
Wstawienie elementu do kolejki priorytetowej	`void insert(Key v)`
Zwróć i usuń element o maksymalnym kluczu z kolejki	`Key delMax()`
Zwróć element o maksymalnym kluczu z kolejki	`Key max()`
Czy kolejka priorytetowa jest pusta?	`boolean isEmpty()`
Zwraca rozmiar kolejki priorytetowej	`int size()`

Kolejka priorytetowa

- Implementacje:
  - Za pomocą tablicy
  - Za pomocą listy powiązanej
  - Za pomocą **kopca** binarnego

Kolejka priorytetowa

- Implementacja za pomocą tablicy nieuporządkowanej i uporządkowanej

insert	P		1	P							P							
insert	Q		2	P	Q						P	Q						
insert	E		3	P	Q	E					E	P	Q					
remove max		Q	2	P	E						E	P						
insert	X		3	P	E	X					E	P	X					
insert	A		4	P	E	X	A				A	E	P	X				
insert	M		5	P	E	X	A	M			A	E	M	P	X			
remove max		X	4	P	E	M	A				A	E	M	P				
insert	P		5	P	E	M	A	P			A	E	M	P	P			
insert	L		6	P	E	M	A	P	L		A	E	L	M	P	P		
insert	E		7	P	E	M	A	P	L	E	A	E	E	L	M	P	P	
remove max		P	6	E	M	A	P	L	E		A	E	E	L	M	P		

[SW-2018]

Kolejka priorytetowa

- Implementacja za pomocą tablicy nieuporządkowanej i uporządkowanej

implementacja	insert()	delMax()	max
Tablica nieuporządkowana	O(1)	O(*N*)	O(*N*)
Tablica uporządkowana	O(*N*)	O(1)	O(1)
	O(log*n*) ?	O(log*n*) ?	O(log*n*) ?

- Czy możliwa implementacja efektywna dla wszystkich operacji?

Kolejka priorytetowa

- Implementacja za pomocą listy powiązanej

-> do ćwiczeń własnych

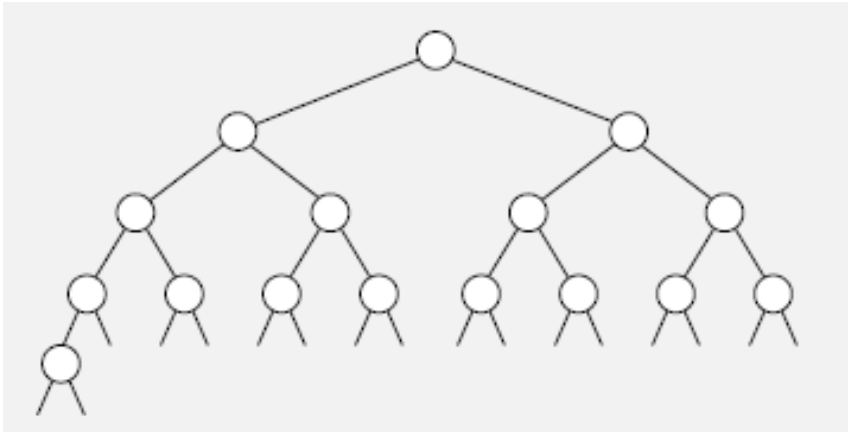
Kopiec binarny

- Implementacja za pomocą **kopca binarnego**

- **Drzewo binarne** – struktura, w której każdy wierzchołek nie ma potomka albo ma dwóch potomków: lewe i prawe drzewo binarne.
- **Kompletne drzewo binarne** – równomiernie zbalansowane z wyjątkiem ostatniego poziomu

- Wysokość kompletnego drzewa binarnego:

  Zaokrąglenie w dół: lg *n*

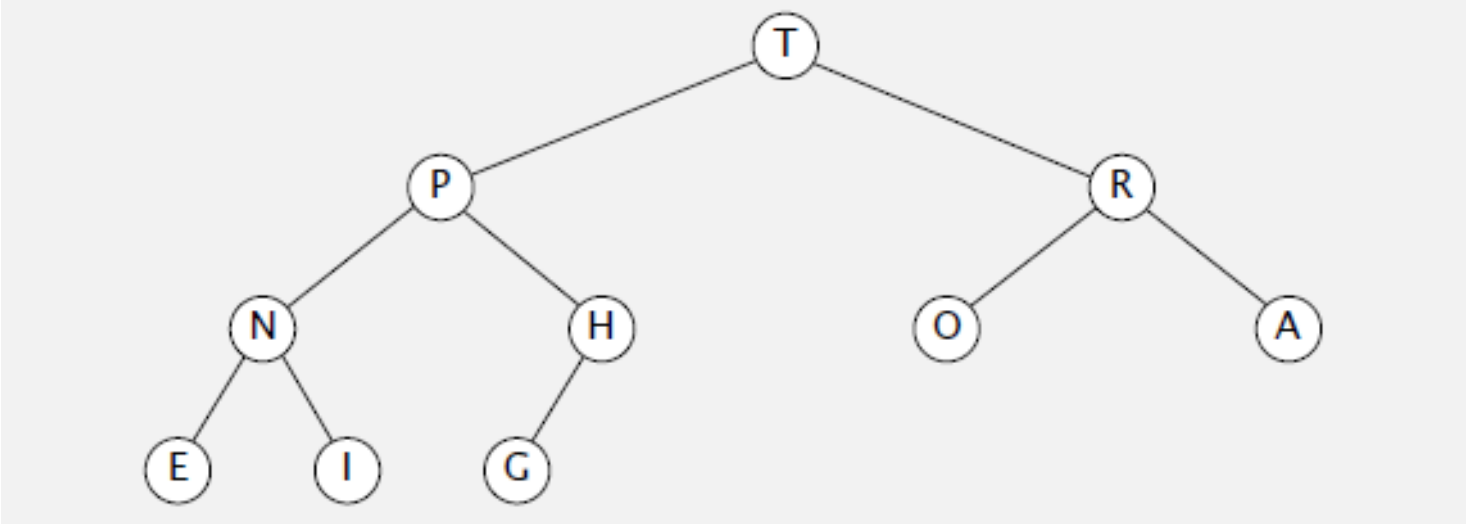


wysokość = 4, 16 wierzchołków

Kopiec binarny

- **Kopiec binarny** – kompletne drzewo binarne, w którym:
  - Wierzchołki zawierają wartości (klucze)
  - Wartości w wierzchołku przodka/rodzica są nie mniejsze niż wartość potomków/dzieci, jeśli istnieją (obserwacja: największa wartość znajduje się w korzeniu drzewa)

[SW-2018]

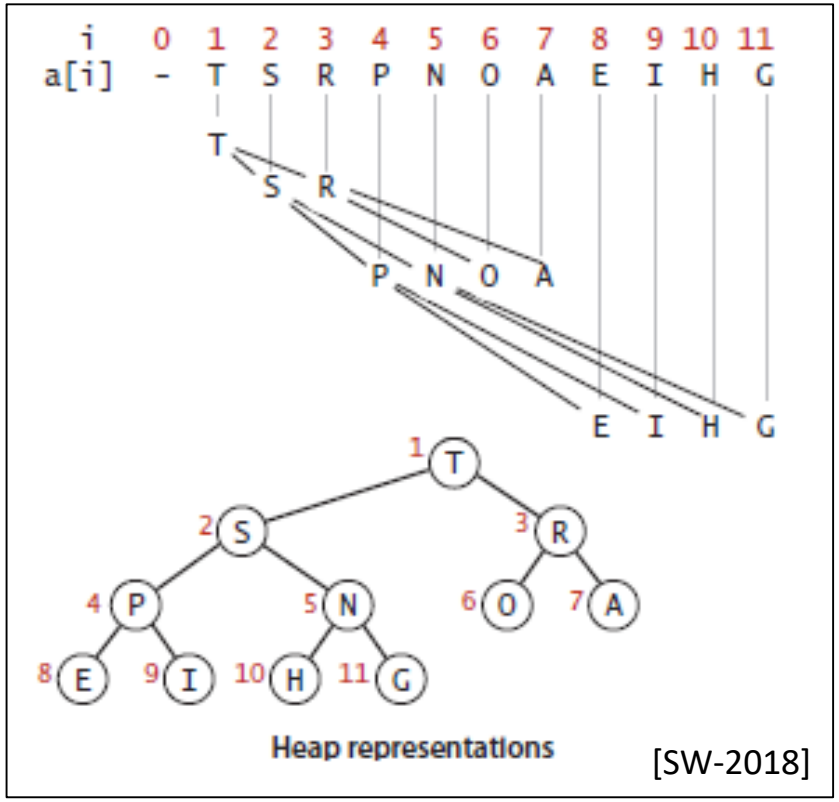


Kopiec binarny

Jak z kolei jego reprezentować?

Tablica

- Indeksy rozpatrujemy od 1
- Zapamiętujemy w kolejności występowania
  od lewej do prawej na każdym poziomie
- Obserwacja:
  - Rodzic wierzchołka *k* jest w *k*/2 (zaokr. w dół),
  - Dzieci wierzchołka *k* są w 2*k* i 2*k*+1



[SW-2018]

Kopiec binarny w postaci tablicy

Fundamentalne operacje

Wstawienie

- Dodaj element na koniec, a następnie zastosuj jego „wynurzanie” (*swim up*) – aby przywrócić strukturę kopca

Pobranie i usunięcie maksimum

- Zamień wierzchołek-korzeń z wierzchołkiem na końcu, a następnie zastosuj jego „zanurzanie” (*sink down*) – aby przywrócić strukturę kopca

Kopiec binarny w postaci tablicy

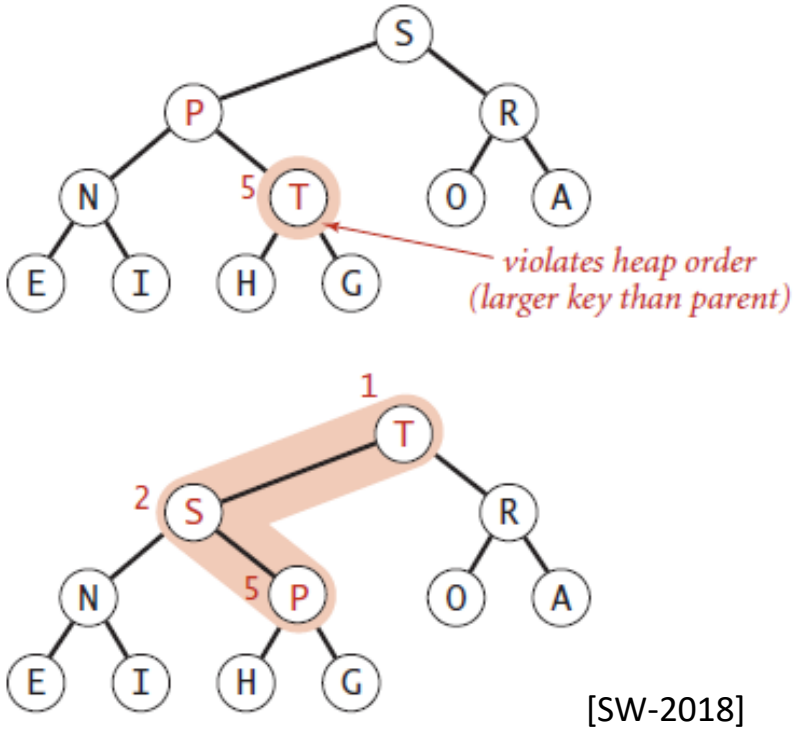
„Wynurzanie” (*swim up*)

Przypadek szczególny:

Wartość w danym wierzchołku jest większa

niż wartość w jego przodku

```
private void swim(int k)
{
   while (k > 1 && less(k/2, k))
   {
      exch(k, k/2);
      k = k/2;
   }
}
```



[SW-2018]

Kopiec binarny w postaci tablicy

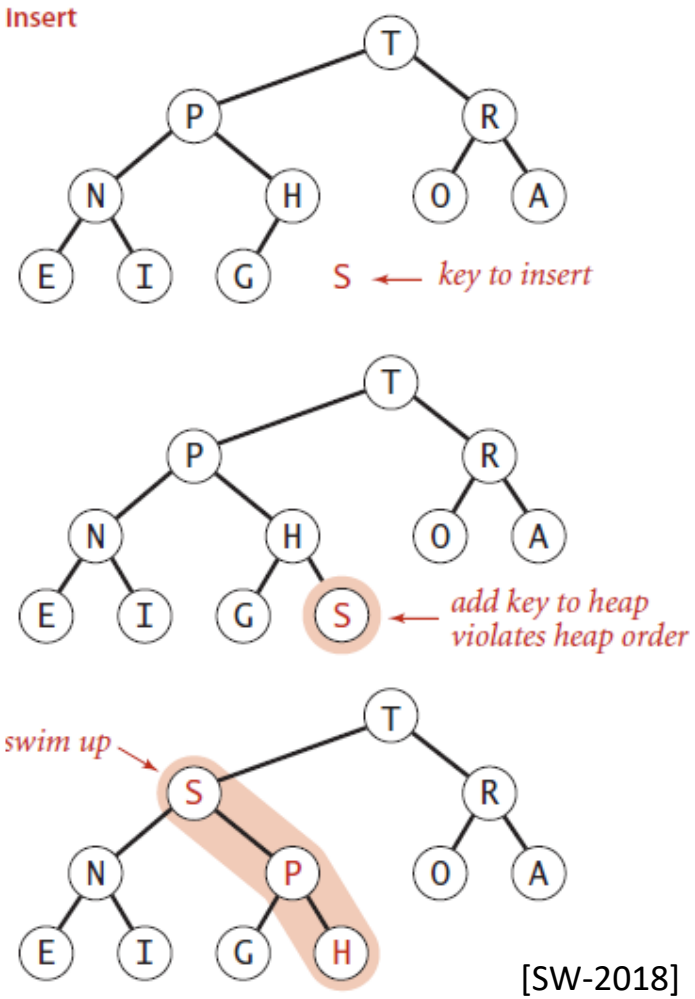
Wstawienie

- Dodaj element na koniec, a następnie zastosuj jego „wynurzanie” (*swim up*)

Koszt:

Co najwyżej 1+log *n* porównań

```
public void insert(Key x)
{
   pq[++n] = x;
   swim(n);
}
```



[SW-2018]

Kopiec binarny w postaci tablicy

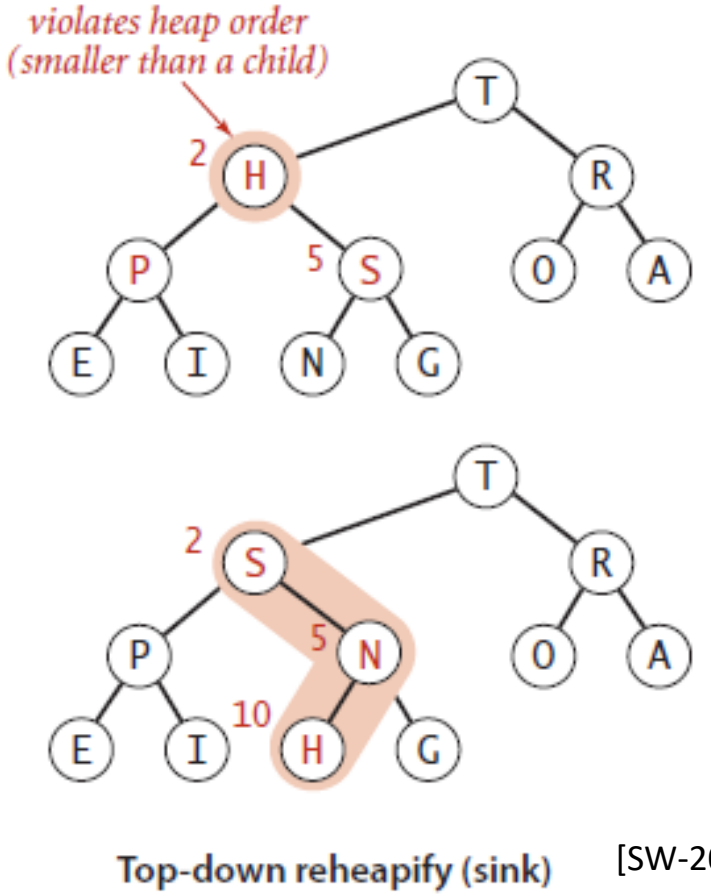
„Zanurzanie” (*sink down*)

Przypadek szczególny:

Wartość w danym wierzchołku jest mniejsza

niż wartość w jego potomku (jednym lub obu)

```
private void sink(int k)
{
   while (2*k <= n)
   {
      int j = 2*k;
      if (j < n && less(j, j+1)) j++;
      if (!less(k, j)) break;
      exch(k, j);
      k = j;
   }
}
```



Top-down reheapify (sink)

[SW-2018]

Kopiec binarny w postaci tablicy

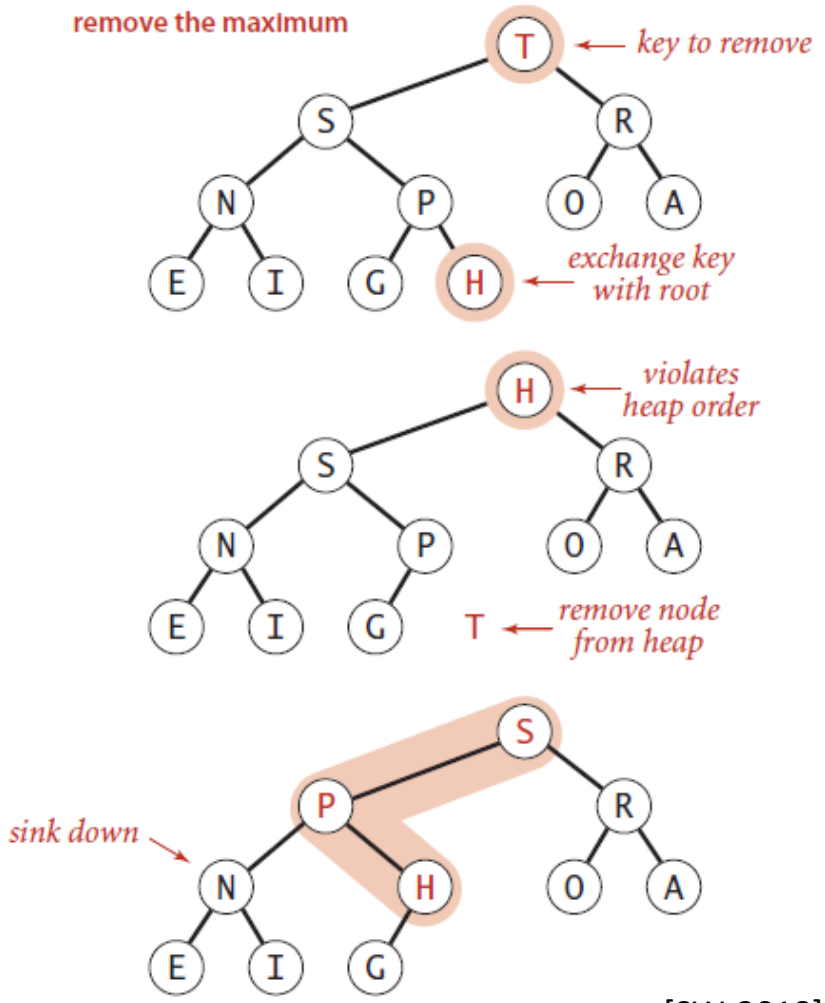
Pobranie i usunięcie maksimum

- Zamień wierzchołek-korzeń z wierzchołkiem na końcu, a następnie zastosuj jego „zanurzanie” (*sink down*)

Koszt:

Co najwyżej 2 log *n* porównań

```
public Key delMax()
{
   Key max = pq[1];
   exch(1, n--);
   sink(1);
   pq[n+1] = null;
   return max;
}
```



[SW-2018]

Kolejka priorytetowa

Implementacja za pomocą tablicy nieuporządkowanej i uporządkowanej + kopiec

implementacja	insert()	delMax()	max
Tablica nieuporządkowana	O(1)	O(*N*)	O(*N*)
Tablica uporządkowana	O(*N*)	O(1)	O(1)
Kopiec	**O(log*n*)**	**O(log*n*)**	**O(1)**

Czy możliwa implementacja efektywna dla wszystkich operacji?

Odp. TAK