# Sortowanie przez kopcowanie


WYDAWNICTWO SZKOLNE PWN

# Sortowanie przez kopcowanie (HeapSort)

- Metoda z wykorzystaniem struktury kopca
- Efektywna w wypadku liczb całkowitych
- Zajmuje mało pamięci operacyjnej

WYDAWNICTWO SZKOLNE PWN

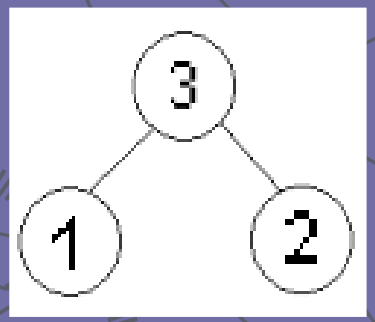
# Definicja kopca

Kopcem nazywa się strukturę danych, którą można rozpatrywać jako drzewo binarne zawarte  w tablicy.

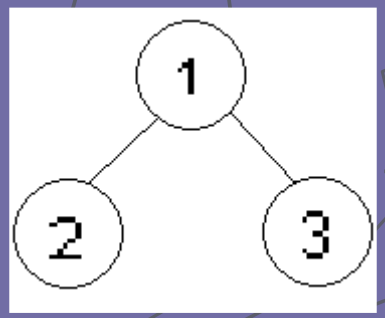
Każdy węzeł drzewa binarnego odpowiada dokładnie jednemu elementowi danych zawartemu w tablicy.

# Warunek kopca

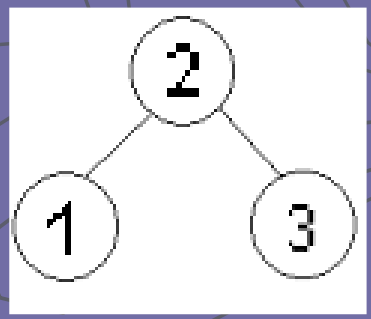
**Wartość przechowywana w dowolnym węźle drzewa binarnego powinna być nie mniejsza (lub nie większa) od wartości przechowywanych w węzłach potomnych.**




Spełnia warunek kopca malejącego



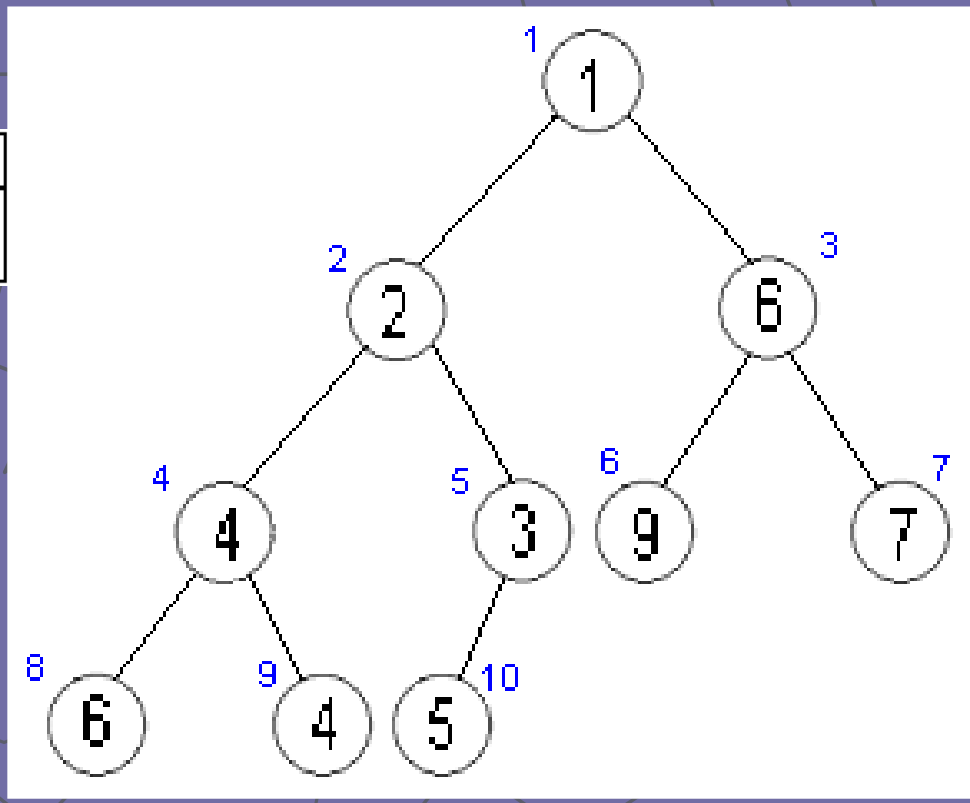

Spełnia warunek kopca rosnącego




Nie spełnia warunku kopca

# Prawidłowo skonstruowany kopiec

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 6 | 4 | 3 | 9 | 7 | 6 | 4 | 5 |



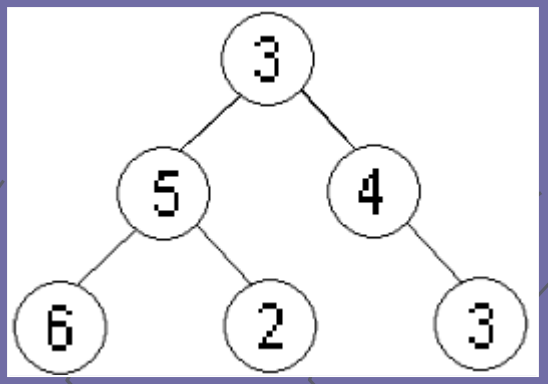
- Index rodzica = index syna div 2
- Index lewego syna = index ojca * 2
- Index prawego syna = index ojca *2 + 1

# Procedure Heapify(x: integer);

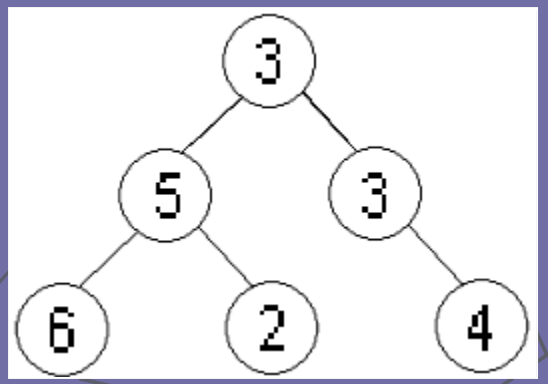
```
var pmin, l, r : integer;
begin
 l:= x*2;
 p:=x*2+1;
 if (l<=rozmiar) and (heap[l]<heap[x]) then pmin:=l
 else pmin:=x;
 if (p<=rozmiar) and (heap[p]<heap[min]) then
pmin:=p;
 if pmin<>x then
          begin
           zamien(x,pmin);
           Heapify(pmin);
          end;
end;
```

WYDAWNICTWO SZKOLNE PWN

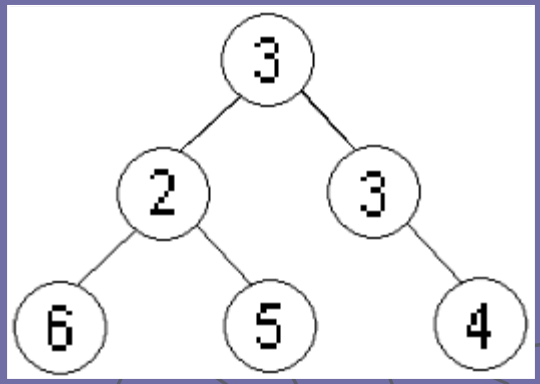
# Przebieg kopcowania



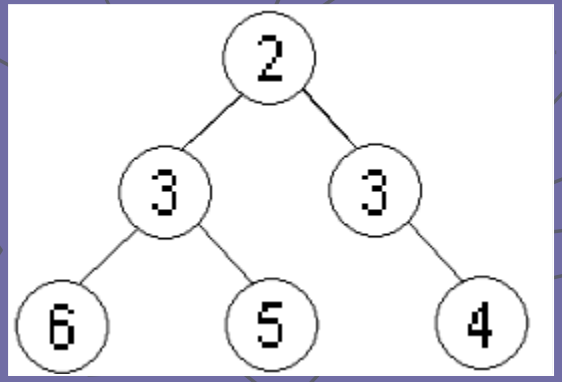

| 3 | 5 | 4 | 6 | 2 | 3 |
|---|---|---|---|---|---|




| 3 | 5 | 3 | 6 | 2 | 4 |
|---|---|---|---|---|---|




| 3 | 2 | 3 | 6 | 5 | 4 |
|---|---|---|---|---|---|




| 2 | 3 | 3 | 6 | 5 | 4 |
|---|---|---|---|---|---|

# Procedure Build_Heap (rozmiar:integer);

```
var x:integer;

begin

 For x:=(rozmiar div 2) downto 1 do
Heapify(x);

End;
```

# Idea sortowania

- Zaczynasz od zbudowania kopca – deklarujesz tablicę i za pomocą procedury Build_Heap doprowadzasz do spełnienia w obrębie całej tablicy warunków wymaganych dla kopca

# Kolejno, w pętli wykonujesz następujące czynności, aż do momentu, gdy rozmiar kopca będzie równy 2:

- zamieniasz wartość, która znalazła się na wierzchołku z ostatnim – rozpatrywanym elementem kopca,
- zmniejszasz wartość zmiennej przechowującej informację o rozmiarze kopca (chodzi o to, by powtórnie nie rozpatrywać elementu ustawionego na końcu),
- przywracasz procedurą Heapify(1) konstrukcję kopca dla zmniejszonej liczby elementów.