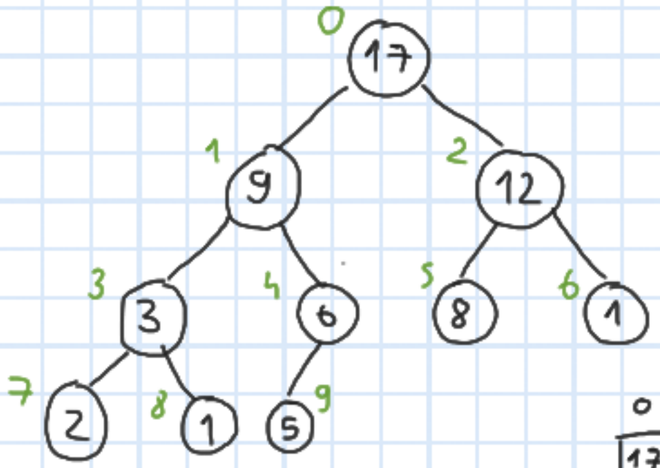
② Sortowanie kopcowe (Heapsort)

Kopiec — drzewo binarne, w którym w każdym węźle A przechowywany wartość ≥ niż wartości w jego dzieciach

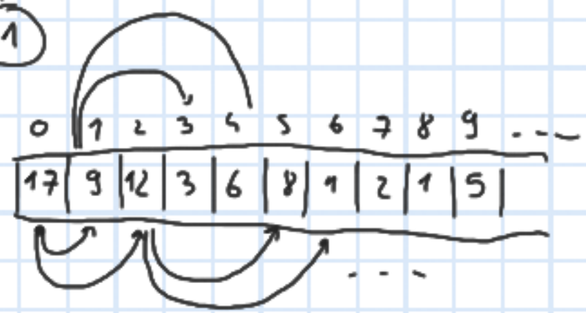
Przykład

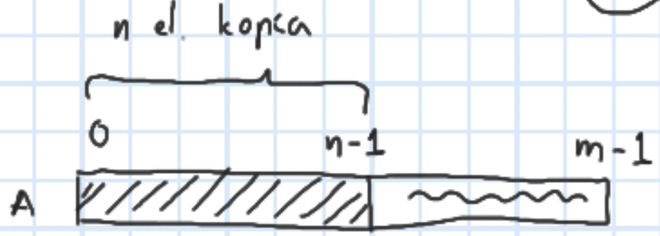


$left(i) = 2i + 1$

$right(i) = 2i + 2$

$parent(i) = \lfloor (i-1)/2 \rfloor$





$len(A) = m$

```
def heapify(A, n, i)
    l = left(i)      , max_ind = i
    r = right(i)
    if l<n and A[l] > A[max_ind]:
        max_ind = l
    if r<n and A[r] > A[max_ind]:
        max_ind = r
    if max_ind ≠ i:
        swap(A[i], A[max_ind])
        heapify(A, n, max_ind)
```

$O(\log n)$

```
def build_heap(A):
    n = len(A)
    for i in range(parent(n-1), -1, -1):
        heapify(A, n, i)
```

$O(n \log n)$

$\Theta(n)$

W tablicy mamy n elementów

$\frac{n}{2}$ tych elementów tworzy kopce o wys. 0

$\frac{n}{4}$ tych el., tworzy kopce o wys 1

$\frac{n}{8}$ tych el. tworzy kopce o wys. 2

$\vdots$

$\left\lceil \frac{n}{2^{h+1}} \right\rceil$ tych el tworzy kopce o wys. h

Koszt build heap wyrażamy jako:

$$\sum_{h=0}^{\lfloor \log n \rfloor} \left\lceil \frac{n}{2^{h+1}} \right\rceil h = O\left(\sum_{h=0}^{\lfloor \log n \rfloor} \frac{n \cdot h}{2^{h+1}} \right)$$

$$= O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h} \right) = O(n)$$

$$f(x) = 1 + x + x^2 + x^3 + \ldots = \frac{1}{1-x}$$

$$f'(x) = 1 + 2x + 3x^2 + \ldots = \frac{1}{(1-x)^2}$$

$$x f'(x) = x + 2x^2 + 3x^3 + \ldots = \frac{x}{(1-x)^2} \xrightarrow{x = \frac{1}{2}} 2$$

```
def heapsort(A):
    n = len(A)
    buildheap(A)
    for i in range(n-1, 0, -1):
        swap(A[i], A[0])
        heapify(A, i, 0)        <-- O(log n)
```

0	1	2	3	4	5	6	7	8	9
1	9	8	3	6	5	1	2	12	17

$\Theta(n \log n)$