

## LABORATORIUM NR 2

## SORTOWANIE PRZEZ KOPCOWANIE

**Zadanie AL2.1**

Sprawdź, czy ciąg (23, 17, 14, 6, 13, 10, 1, 5, 7, 12) spełnia własność kopca typu max?

**Zadanie AL2.2**

Zademonstruj kolejne kroki działania procedury budującej kopiec z elementów tablicy  $A = [28, 6, 11, 12, 17, 8, 7, 18, 12, 14, 23]$ . Następnie zilustruj kilka obrotów pętli sortującej przy sortowaniu kopcowym **Heapsort**. Aby wskazywać, które węzły kopca są ze sobą zamieniane, użyj drzewiastej reprezentacji kopca.

**Zadanie AL2.3**

Oszacuj czasy działania algorytmu sortowania przez kopcowanie dla tablicy  $A$  o długości  $n$ , w której (a) wszystkie elementy są takie same, (b) są posortowane malejąco, (c) są posortowane rosnąco. Następnie, w oparciu o rozwiązanie zadania AL2.4 (AL2.5), dokonaj eksperymentalnego pomiaru czasu dla zaproponowanych oszacowań.

**Zadanie AL2.4** (5+1\* pkt.)

- Zaimplementuj omawiany na wykładzie algorytm sortowania przez kopcowanie. (5 pkt)
- Zmodyfikuj procedurę **Heapify** tak, aby używała iteracji zamiast rekursji. (1 pkt)

**Wejście.** Liczby zapisane są w kolejnych wierszach pliku tekstowego.

**Wyjście.** Posortowane liczby z pliku wejściowego zapisane w kolejnych wierszach pliku wyjściowego.

**Zadanie AL2.5** (5+1\* pkt.)

Zaimplementuj algorytm sortowania przez kopcowanie, który sortuje tylko wskazany zakres ciągu  $n$  liczb całkowitych.

**Wejście.**  $n + 2$  liczb zapisanych w kolejnych wierszach pliku tekstowego, z czego dwie pierwsze  $x$  i  $y$  spełniają zależność  $1 \leq x \leq y \leq n$ , a kolejne są elementami tablicy  $A$ .

**Wyjście.** Posortowany ciąg liczb  $A[x], A[x + 1], \dots, A[y]$  zapisany w kolejnych wierszach pliku wyjściowego.