Celem laboratorium nr 01 jest zapoznanie się z algorytmami dotyczącymi znajdywania wartości minimalnej/maksymalnej w zbiorze, sortowaniem i wyszukiwaniem.

Zapoznajcie się z poniższymi algorytmami i przygotujcie ich implementację. Pamiętajcie, że **macie to zrealizować algorytmicznie** (korzystając z pętli, instrukcji sterujących, etc), a nie „językowo" (użycie sort() czy find()).

1. **Znajdywanie wartości minimalnej/maksymalnej.**

   W przypadku danych liczbowych, znalezienie minimum lub maksimum nie stanowi problemu. Jednakże, gdyby rozważyć zbiór np. ciągów znaków lub jakichś struktur zawierających więcej niż jedną zmienną, pojawia się pewien problem. Należy wpierw ustalić co oznacza A > B w przypadku np. ciągów znaków A i B. Czy chodzi o ich długości? Może o sumę kodów ASCII każdego ze znaków? Czy może o liczbę słów (rozdzielonych białym znakiem) w tekście?

   W ramach zajęć rozważcie zbiór ciągów znaków umieszczonych w dodatkowym pliku (dane.txt) i przygotujcie algorytm, który znajdzie wartość minimalną/maksymalną zgodnie z następującymi regułami (każda reguła to osobny algorytm):
   - liczba znaków w tekście (długość ciągu);
   - suma kodów ASCII znaków w tekście (można użyć funkcji *ord*());
   - alfabetycznie posortowane.

2. **Porównanie algorytmów sortowania.**

   Zaimplementujcie przedstawione na wykładzie algorytmy sortowania:
   - sortowanie bąbelkowe (wersja podstawowa);
   - sortowanie przez wybieranie/selekcję;
   - sortowanie przez wstawianie;
   - sortowanie przez scalanie(*);
   - sortowanie przez zliczanie lub kubełkowe (do wyboru).

   Następnie sprawdźcie czasy działania sortowań dla poszczególnych plików sort1.txt, sort2.txt oraz sort3.txt.

   Poniżej kod pozwalający na analizę czasu działania programu.

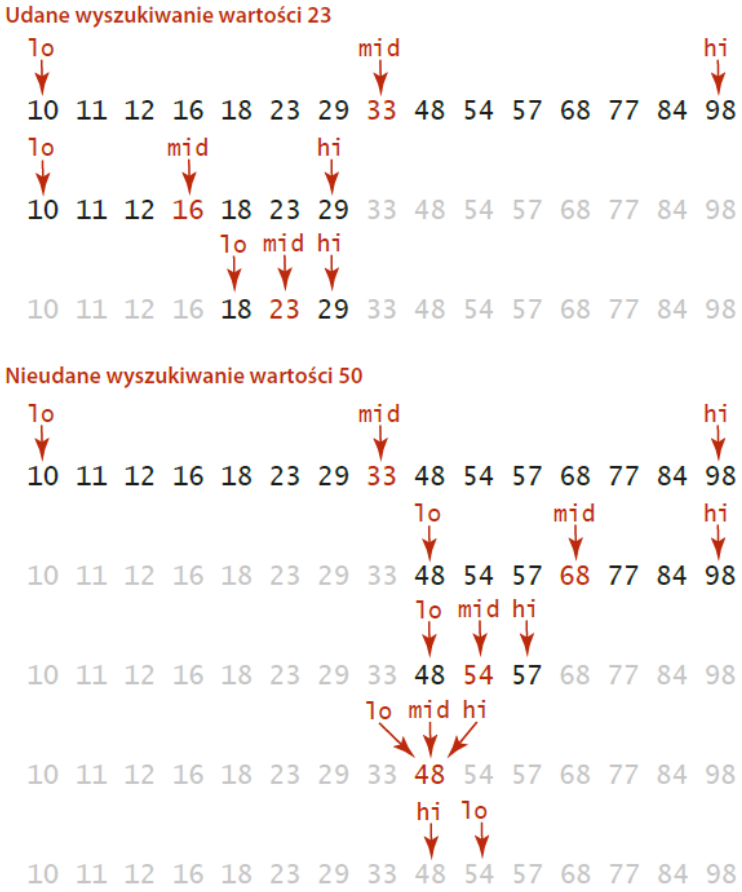
```
import time
start = time.clock()
```

```
funkcja_sortujaca_nrX()
stop = time.clock()
print("--- %s sekund ---" % round(stop – start, 2))
```

3. **Wyszukiwanie binarne (na posortowanym zbiorze).**

Wyszukiwanie binarne wymaga, aby przeszukiwana tablica była już posortowana. Zaletą wyszukiwania binarnego jest to, że przeszukanie n-elementowej tablicy zajmuje w nim tylko O(log n) czasu.

Algorytm jest realizowany metodą "dziel i zwyciężaj". Dzieli on tablicę na mniejsze podtablice do momentu wyszukania pozycji (lub nie w przypadku gdy taki element nie istnieje) elementu szukanego. W każdej kolejnej iteracji wybieramy środek przedziału i sprawdzamy czy jest równy naszej szukanej, jeśli jest większy to szukamy w lewej części przedziału (patrz linia druga na rys. poniżej), jeśli mniejszy to szukamy po prawej. Tak aż do znalezienia wartości lub ograniczenia przedziału do jednej liczby różnej od szukanej wartości (brak rozwiązania, patrz druga część rys. poniżej).



Wyszukiwanie binarne w posortowanej tablicy