Algorytmy Sortujące

Miłosz Sawicki

22 marca 2019

1 Założenia projektu

Program jest skonstruowany do sortowania listy o długosci odpowiednio 10000, 25000, 50000, 75000, 100000, 150000 losowo wygenerowanych elementów ze zbioru 1-100000 przy użyciu algorytmow sortujacych shellsort, mergesort i quicksort. Wygenerowne losowe i posortowane listy zapisuje do plików, a uzyskane dane czasu i ilosci operacji przy pomocy każdego z algorytmów przedstawia pod koniec na dwóch wykresach.

2 Opis działania programu

Program jest napisany w języku Python3. Z uwagi na ograniczenie ilosci rekurcji w tym jezyku oraz żeby uniknać pojawiających się błędów byłem zmuszony podzielic program na dwa osobne skrypty.

W pierwszym skryptcie (projekt2.miloszsawicki.part1.py) generowane sa wszystkie listy o dlugosci n losowych elementów z przedzialu 1-10000 odpowiednio n =[10000, 25000, 50000, 75000, 100000, 150000], sa one zapisywane do plikow txt o nazwach in.(dlugosc listy).txt. Nastepnie program sortuje kazda z tych list za pomoca algorytmu QuickSort, zlicza liczbe wykonanych operacji przesuniec i przestawien, czas wykonania algorytmu, zapisuje te dane w listach oraz do pliku txt (operacje.q.txt , czas.q.txt). Beda one wykorzystane w drugim skryptcie przy okazji rysowania wykresu przesunięć i przestawien oraz czasu wykonania algorytmu dla kazdego przedstawionego w programie algorytmu sortujacego. Do pliku zapisywana jest również posortowana lista wyjsciowa dla kazdej długosci listy n, a pliki te mają nazwe out.(dlugosc listy).txt.

W drugim skryptcie (projekt2.miloszsawicki.part2.py) program na wstępie zczytuje wszystkie dane z plikow txt wytworzone przez poprzedni skrypt, przy pomocy biblioteki ast konwertuje ciagi znakow na listy, dodaje nieposortowane listy o n dlugosciach do zbioru nieposortowanych list i rozpoczyna sortowanie kazdej z nich za pomoca algorytmów ShellSort i MergeSort. Dla kazdego z nich liczony jest czas wykonania i ilosc operacji. Te dane sa przechowywane w listach, ktore nastepnie zostaja wykorzystane do narysowania wykresow czasu wykonania i ilosci operacji. Dzieki wykorzystywaniu biblioteki matplotlib rysowane są po kolei dwa wykresy przedstawiajace wszystkie dane zebrane w czasie działania dwóch skryptów.

3 Opis działania poszczególnych algorytmow sortujacych

3.1 Sortowanie Shella (ShellSort)

Przy sortowaniu program wspiera sie algorytmem sortujacym Insertion Sort którego Shell Sort jest rozszerzeniem. Złożonosc obliczeniowa algorytmu zalezy od użytych w nim odstepow (step). W programie dlugosc listy wejsciowej dzielona jest przez 2 i tak uzyskiwana jest ilosc podlist ktore pelnia funkcje step. Z kazdą iteracją petli ilosc podlist do posortowania Insertion Sort jest dzielona przez 2 az do momentu kiedy podlista bedzie jedna i bedzie to posortowana lista wyjsciowa. Do algorytmu dołączona jest zmienna globalna licząca liczbę dokonanych przestawień podczas sortowania.

3.2 Sortowanie Merge (MergeSort)

Algorytm stosuje metode "dziel i zwyciężaj". Najpierw dzieli zestaw danych na dwie równe częsci, potem kazda z nich sortuje rekurencyjnie, chyba ze pozostal juz tylko jeden element, wtedy posortowane podciągi łaczy w jeden ciąg posortowany. Do algorytmu dołączona jest zmienna globalna licząca liczbę dokonanych przestawień podczas sortowania.

3.3 Sortowanie Szybkie (QuickSort)

Algorytm QuickSort również działa na zasadzie "dziel i zwyciężaj". Z początku wybierany jest element rozdzielajacy (pivot - w programie jest to pierwszy element listy), w funkcji pomocniczej partition program rodziela elementy listy na te wszyskie ktore sa mniejsze (na lewo) lub wieksze (na prawo) od elementu rodzielajacego - pivot. W nastepnych krokach algorytm wywołuje sie rekurencyjnie dla kazdej powstalej podlisty powstalej z podzielenia elementem rozdzielajacym i powtarza ta prodecure az kolejny fragment uzyskany z podzialu zawiera tylko jeden element. Do algorytmu dołączona jest zmienna globalna licząca liczbę dokonanych przestawień podczas sortowania.

4 Instrukcja dla użytkownika programu

Program składa się z dwóch skryptów napisanych w języku Python. Proszę najpierw uruchomić skrypt pierwszy projekt2.miloszsawicki.part1.py, a następnie skrypt drugi projekt2.miloszsawicki.part2.py

Wszystkie dane zostaną automatycznie spisane, a wykresy czasu i operacji są wbudowane w skrypt drugi.

5 Wnioski

- Program pozwala zrozumieć działanie algorytmów sortujących.
- Z przedstawionych wykresów widac złozonosc operacyjną kazdego z algorytmow. Wynika z niej ze najmniej złożony operacyjnie jest QuickSort, a najwiecej operacji do posortowania listy potrzebuje ShellSort.
- Z wykresu czasu wynika że dla listy złożonej z losowych elementów najwiecej czasu zajmuje działanie algorytmu MergeSort i ta wartosc zdecydowanie rosnie wraz z długoscią listy początkowej.
- Widać rówież, że ilosć wykonywanych operacji dla co raz wiekszej długosci listy wejsciowej najwolniej rosnie przy algorytmie QuickSort, stąd jest on najlepszy dla bardzo dużych zbiorów danych.