Київський національний університет технологій та дизайну

КОЛИСКО О.З., КУРИЛОВИЧ М.О.

АВТОМАТИЗАЦІЯ ПРОЦЕСУ ТАБЕЛЮВАННЯ ПРАЦІВНИКІВ ПІДПРИЄМСТВА ЛЕГКОЇ ПРОМИСЛОВОСТІ

KOLISKO O.Z., KURYLOVYCH M.

ENTERPRISE EMPLOYEES WORKING TIME RECORD AUTOMATION

Purpose - Development of a program to help companies monitor their employees by time recording. It will help to reward their workers for their persistent work or to penalize them if they are wasting their time or if they come to work too late. It will help workers to organize their time so they can work properly 40 hours per week and to keep themselves from overworking.

Keywords: Forecast, analysis, monitoring, algorithm.

Вступ

Розв’язання задачі з автоматизації процесу табелювання працівників підприємства легкої промисловості є, насправді, досить важливим завданням для власників підприємств. Вирішення даної задачі дозволить урегулювати та проконтролювати роботу та відвідуваність працівників, а також полегшить керування персоналом.

Постановка завдання

Найбільш важливими алгоритмами для розв’язання поставленої задачі є алгоритми пошуку та сортування. Програма повинна надавати користувачеві прості у використанні засоби для перегляду бази співробітників і пошуку за певними критеріями, а адміністратору - можливість додавання та редагування інформації. Таким чином, головними завданнями є аналіз та дослідження швидкого та надійного алгоритму.

Основна частина

Для розробки доцільно було б вибрати алгоритми, які будуть простими у використанні і функціональними, враховуючи діапазон вхідних даних. В статті Берхарда Коутчана (Dr. Christoph Bernhard Koutschan) [1], команда вчених намагалась скласти список поширених алгоритмів, що найчастіше використовуються в програмуванні. Результатом мала стати добірка з 5 найважливіших алгоритмів, але команді досі важко визначитись. Список містить 32 алгоритми (не відсортовані за популярністю). З тих що можуть нас зацікавити це:

- алгоритм бінарного пошуку (Binary search) ;
- сортування кучею (Heaps (heapsort)) ;
- сортування злиттям (Merge sort).

Щодо бінарного пошуку – немає сумнівів, але було б цікаво розглянути окремі алгоритми сортування щодо реальної швидкості їх роботи на різних даних. В інтернет середовищі [2] до кращих алгоритмів сортування відносять алгоритми:

- Timsort, відносно новий гібридний алгоритм сортування, на даний момент є стандартним сортуванням в Python таGNU Octave, реалізований в OpenJDK 7 та Android JDK 1.5.

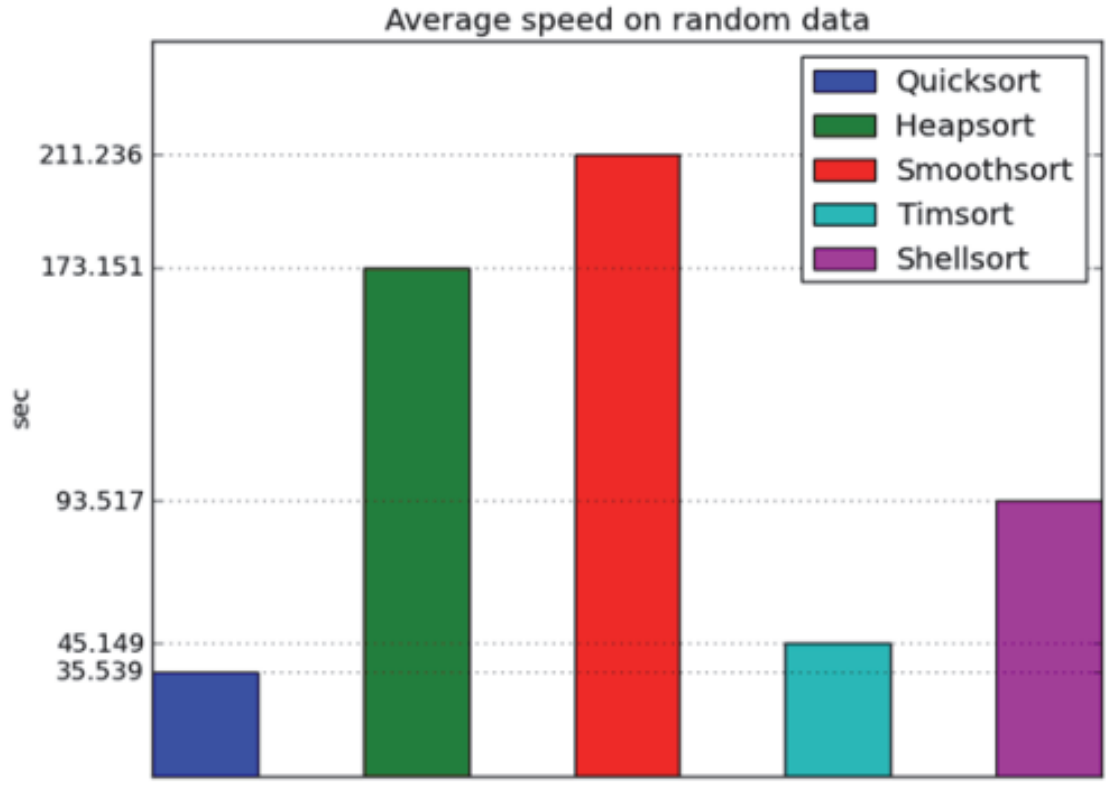
- Smoothsort або Плавне сортування, модифікація «кучі» за допомогою чисел Леонардо[2],

- Shellsort, сортування Шелла;

- також для порівняння взято звичайні рекурсивні реалізації Quicksort (швидке сортування) та Heapsort як «традиційні».

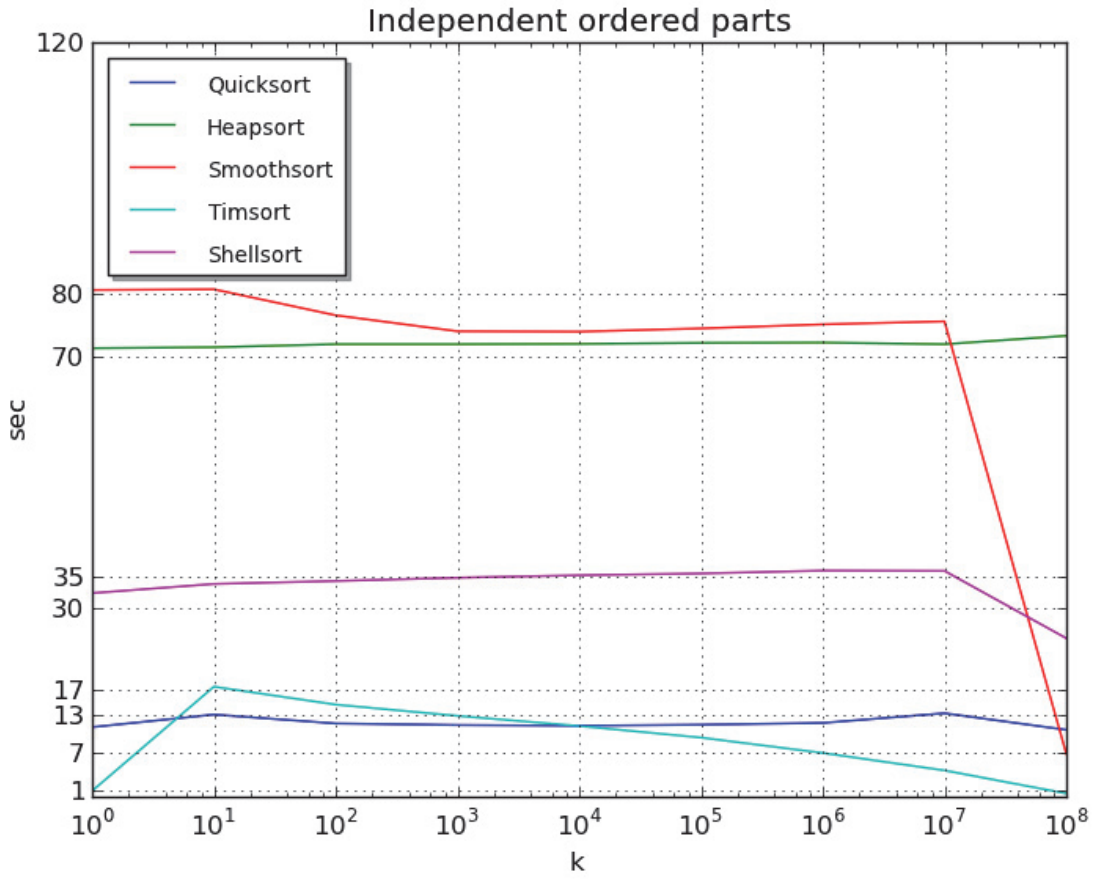
Порівнювався реальний час роботи алгоритмів на різних вхідних даних:
для ніяк не впорядкованих даних це -

1.Quicksort
2.Timsort
3.Shellsort
4.Heapsort
5.Smoothsort;



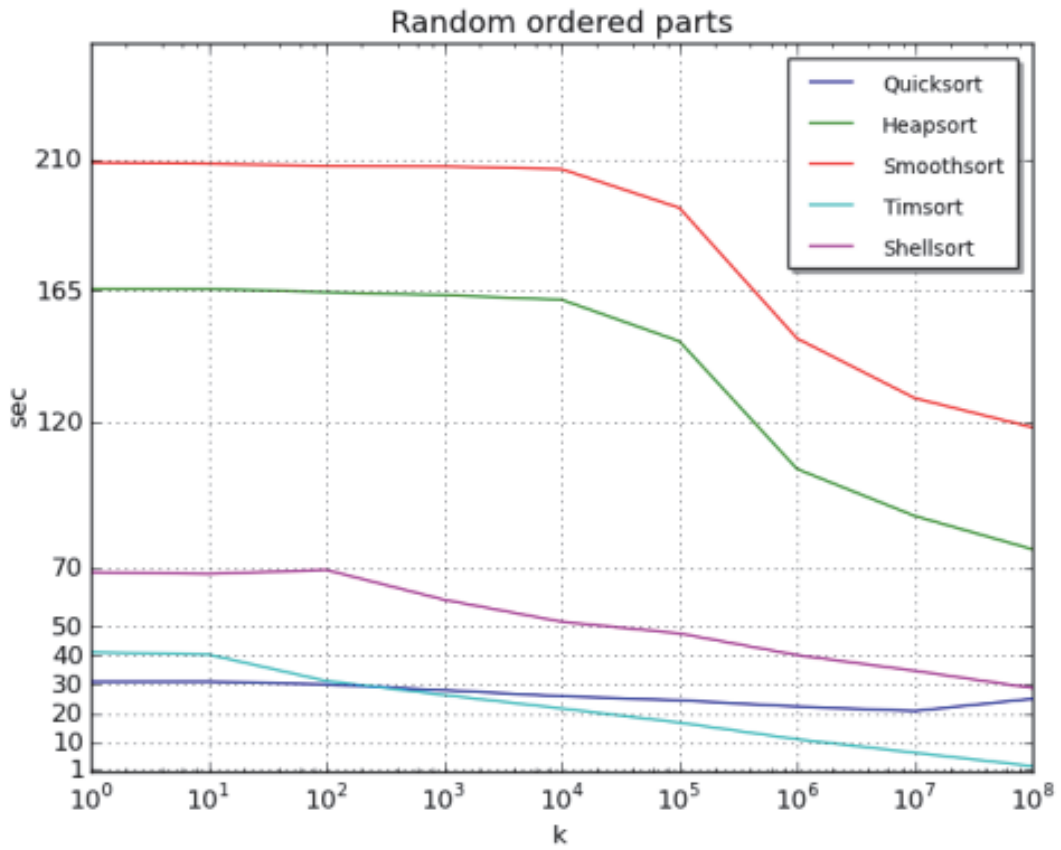
при наявності відсортованих груп однакового розміру -

1.Timsort
2.Quicksort
3.Shellsort
4.Heapsort
5.Smoothsort;



в разі наявності відсортованих підгруп різних розмірів

1.Quicksort
2.Timsort
3.Shellsort
4.Heapsort
5.Smoothsort;



Бачимо наступні закономірності:

- Quicksort і Heapsort люблять, коли дані частково відсортовані . На асимптотичну складність це не впливає, але швидкість роботи все одно зростає в рази
- Timsort'у все одно, в якому порядку відсортовані дані, в усіх випадках він працює дуже швидко
- Smoothsort має своє розуміння часткової впорядкованості даних. Реальний приріст продуктивності в цьому тесті помітний лише при повністю відсортованих даних. Навіть у тому випадку, коли масив вдає із себе невелику кількість повністю відсортованих шматків, для того, щоб переставити їх місцями, алгоритму потрібно провести величезну кількість операцій з купами Леонардо, які досить «важкі»
- Shellsort, не дивлячись на велику складність часто працює за менший час, чим Smoothsort. Це пов'язано з відносною простотою алгоритму. Але при повністю відсортованих даних, Shellsort все одно вимушений зробити всі проходи, нехай навіть і не переставляючи жодного елементу.

Висновки

Як бачимо абсолютним лідером став Timsort який дає помітне прискорення при збільшенні ступеня впорядкованості даних і працює на рівні Quicksort в звичайних случах.

Про Smoothsort в мережі дещо мало інформації (в порівнянні з рештою алгоритмів) бо через складність ідеї та спірної продуктивності він швидше є алгоритмічним експериментом, ніж застосовним методом.

У Shellsort асимптотична складність дещо більша, ніж у решти алгоритмів, але є і дві великі переваги: простота ідеї і реалізації, що дозволяє майже завжди обходити Smoothsort. Тут працюючий код займає всього пару десятків рядків, тоді як хороші реалізації Timsort і Smoothsort вимагає більше сотні рядків.

Тому якщо потрібний дуже хороший алгоритм сортування для вбудовування в бібліотеку або у великий проект то краще за все підійде Timsort. У разі ж невеликих завдань, де немає часу і необхідності писати складні алгоритми, а результат потрібний швидко — те цілком можна обійтися і Quicksort'ом — оскільки він дає кращу з простих в написанні алгоритмів продуктивність.

Література

1. THE MOST IMPORTANT ALGORITHMS., Dr. Christoph Bernhard Koutschan, Johann Radon Institute for Computational and Applied Mathematics (RICAM), Linz, Austria, 2015, http://www.risc.jku.at/people/ckoutsch/stuff/e_algorithms.html;
2. Популярные алгоритмы сортировки., Константин Абакумов, Интернет-ресурс https://habrahabr.ru/post/133996/
3. Вірт Н. (1985), Алгоритми та структури даних, розділ 1.9.2.
4. Дональд Кнут Искусство программирования — 3-е изд. — М. : «Вильямс», 2007. — Т. том 3. Сортировка и поиск. — 824 с.