UDC 004:37

COMPARING SEARCHING AND SORTING ALGORITHMS EFFICIENCY IN IMPLEMENTING COMPUTATIONAL EXPERIMENT IN PROGRAMMING ENVIRONMENT

Sagan R.

Kherson State University

This article considers different aspects which allow defining correctness of choosing sorting algorithms. Also some algorithms, needed for computational experiments for certain class of programs, are compared.

Keywords: *computational experiment, sorting algorithms, results statistical processing unit.*

Introduction

A new version of programming environment "Videointerpreter", in which demonstration of efficiency of algorithms is realized, is created in the Kherson State University. The basic advantage of environment is the organization of independent work.

Environment contains following components:

- parser
- programming code editor
- sorting and searching procedures visual analyzer
- results statistical processing unit

This article will consider the principles of creating of results statistics processing unit. The objects of study are sorting methods. Computational experiment we should understand method of studying objects and processes using mathematic modeling. Experiment requires that after constructing mathematical model its computational research is performing, that allows reproduce behavior investigated object in different conditions or in different modifications.

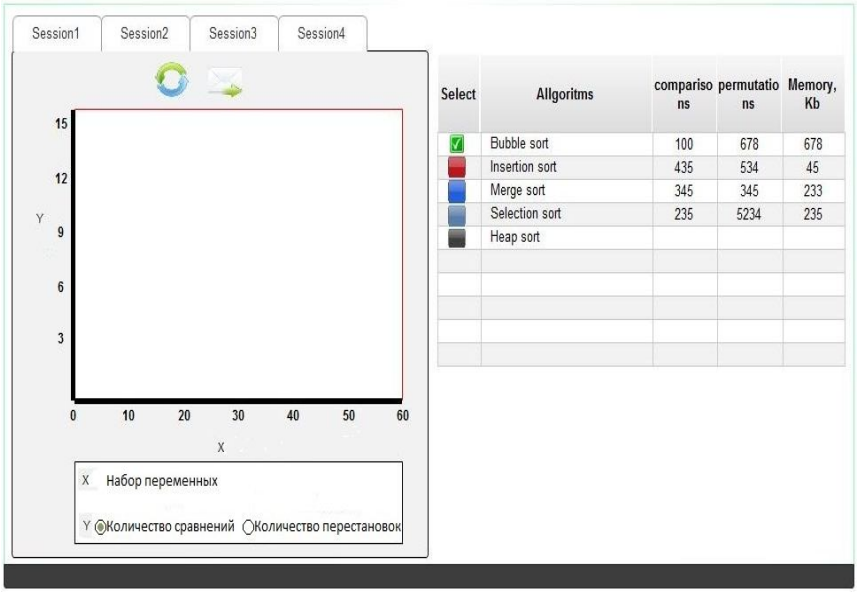
The computer experiment gives us ability to understand singularity of certain algorithms and realize rules of their behavior on such or another data sets. Also this experiment allows make a conclusion about efficiency of sorting methods on these sets with help graphic presentation of dependence on next attributes:

- set of variables;
- quantity of comparing;
- quantity of swaps;
- runtime;
- Picture 1 show the interface of results statistical processing unit.

Today many algorithms of data sorting exist. They contain the operations of comparison and exchange. From their quantity on the n set the complexity of algorithm is determined. In other words complexity is efficiency.

Simple algorithms of sorting, all known as bubble sorting, insertion sorting and selection sort, can have complexity of $O(n^2)$ in the worst case. More effective algorithms, for example, the pyramidal sorting had complexity of $O(n \log n)$.

There are situations, when simple algorithms pass ahead complex ones by efficiency, for example, insertion sorting on small sets can be the best. Such cases can become the objects of researching, and it's one of examples of behavior of sorting types, for the observing of what the system "Videointerpreter" is created.



Picture 1

The criteria of estimating of sorting methods are:

- quantity of operations of key pairs comparing;
- quantity of elements swaps
- economical memory use

With the help of programming environment "Videointerpreter" different problems can be solved, including automatic performing of data sorting. A sorting may be need in different cases, for example for visual displaying data distribution. For such or another data certain sorting methods, improving sorting productivity and speed just for this data type, exist.

It should be mentioned that though complex algorithms need less operations that operations are more complex. Because of that, with relatively small quantity of elements to be sorted, simple methods work enough fast.

Sorting is the process of exchange of objects of the given collection in certain order. The purpose of sorting is to simplify the further elements searching in sorted collection.

Let's consider sorting algorithms they don't use array coping. Convenient measure of such algorithms efficiency is quantity of required comparings in sorting process and quantity of required elements exchange.

Efficient sorting algorithms need nearly $C \equiv N \cdot \log_2(N)$ comparings, where N is elements quantity, and C is quantity of comparings required.

Let's consider some simple sorting methods that need quantity of comparings approximately $C \equiv N^2$.

Sorting methods without array copying can be divided into three main classes:

- selection sort;
- insertion sort;
- bubble sort;

In considered classification different algorithms exist. They differ by complexity, execution speed and operation order. For example incretion sorting, insert sorting, pyramidal sorting, merging sorting, quick sorting etc.

Software development

For programming environment creating Mono toolkit was chosen. It contains C# language compiler and another tools based on .Net for application development acceleration including ASP.Net for Web applications creating and ADO.Net for access to databases.

Mono's advantage is also in ability use Novel Moonlight for created animated objects. The technology allows use applications separately from browser. Moonlight also was chosen for a new ability to use computer mouse to move objects on a web page.

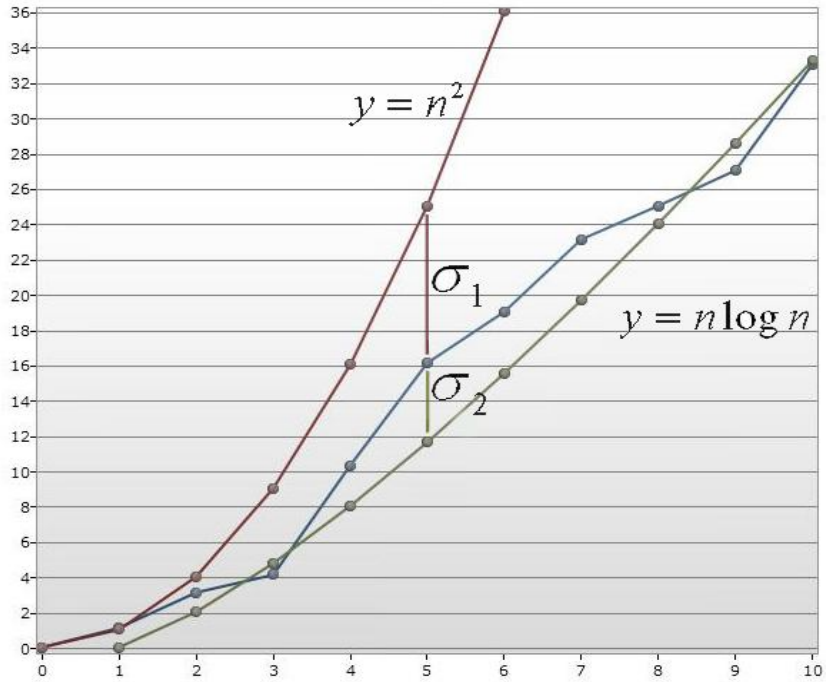
Most part of my work, e.g. computational experiment implementation is executed using namely Moonlight technology.

Logic is described by two classes - Data and SingleSorting. Class SingleSorting is a subclass of class Data and describes statistics of separate sorting: quantities of comparings, exchanges and time spent. Class Data is also place to keep sorting statistics and methods of its processing. It's presented by dictionary whose keys are names of sorting methods and values are dictionaries whose keys are quantities of elements in sorted arrays, and values are instances of SingleSorting class.

Class Data subscribes for events of interpreter:

- SortingBegin gets pointer interpreter, sorting type and quantity of elements , add to itself a new record of type SingleSorting and also subscribes on other events of this interpreter;
- Comparing increment quantity of comparings;
- Swap increment quantity of swaps;
- SortingEnd gets pointer to interpreter , calculates sorting time, unsubscribes on the all events of interpreter.

After getting required data the conclusions are displayed as a chart. On example of picture it's understandable in what way the quicker sorting on a ten elements collection is chosen (Picture 2). A curve built on data got from the interpreter is marked blue. It's required to define a speed with which this sorting was performed. σ_1 and σ_2 - values deviation of the curve from functions $y = n^2$ and $y = n \log n$ correspondingly. On each $n[j]$ deviation from both charts, and $\sum_{j=1}^{n} \sigma_{1_j}$ with $\sum_{j=1}^{n} \sigma_{2_j}$ are compared. The complexity with minimal some of all σ_i is chosen.



Picture 2

Also this class defines methods of self serializing in different formats.

Conclusions for simple sorting methods.

Sorting times is proportional to square of array length. During insertion sorting the time of algorithms execution depends on input data: as the collection to sort is longer so longer the sorting will be performed. Also starting orderliness influences on the sorting time. The best case is sorted array, the worst one is array sorted reversely to needed order. This algorithm is efficient on small data collections and on arrays up to few tens may be the best. Time complexity on the worst variant of input data is $O(n^2)$ on n-element array.

Selection sorting belongs to unstable sorting algorithms. On n-element array execution time in the worst, typical and the best cases is $O(n^2)$.

Exchanging sorting is simple algorithm. It's efficient only for small arrays. The algorithms complexity is $O(n^2)$. Even on the last step, when the array is fully sorted, the algorithm doesn't know that and makes the last full passing through to determine that there were no any elements exchanges.

More precious estimating of simple sorting methods quality show that the fastest is insertion sorting and the slowest is exchanges sorting.

Thought the low speed, simple sorting algorithms should be used for small sized arrays to be sorted.

REFERENCES

1. Лавинский Г. В., Петренко П.А.., Семенов Н.П. Проблемы оценки сложности алгоритмов и вычислений при проектировании управляющих систем // УСиМ – 1997. – № 2. – С. 6-13.
2. Трахтенброт Б. А. Сложность алгоритмов и вычислений. – Новосибирск: НГУ, 1967. – 211 с. 1970.
3. Хартманис Дж., Хопкрофт Дж. Э. Обзор теории сложности вычислений // Кибернетический сборник – 1974. – № 11.