วารสารพัฒนบริหารศาสตร์
ปีที่ 33 ฉบับที่ 2 เมษายน-มิถุนายน 2536

Algorithm Visualization: A Revisit to Sorting Algorithms[1]

Somchai Prasitjutrakul[2] and Wittaya Watcharawittayakul[3]

ABSTRACT Algorithm visualization is a research area that explores presentation techniques for visualizing algorithm behavior. Older techniques include single stepping the source code and viewing its variable values. These techniques are widely used in many symbolic debuggers. With higher computational power, newer techniques exploit graphical presentation to display algorithm behavior in action. Complementing the asymptotic analysis, algorithm visualization presents one with abstract behavioral models in terms of animated views. Mapping the behavior to various presentation models helps see behavioral aspects that lead to understanding at a higher abstract level. Understanding the behavior at this level is beneficial for the study of algorithm complexity and design. The goal of this research is to build an algorithm visualization system under Microsoft Windows for such purposes. Under Windows, multiple execution threads can be synchronized and many behavioral views can be displayed simultaneously. This paper presents the experience from the end-user view point in studying sorting algorithms. Although these algorithms have been widely used in similar systems in the last decade, they are revisited as an intermediate step toward other algorithm studies.

1. Introduction

Algorithm visualization is a research area investigating presentation techniques for visualizing algorithm behavior. Interesting aspects relating to algorithm behavior involve the number of data movements, comparisons, and other computational operations. Direct measurement of these aspects usually provides absolute numerical values that do not reflect the inside mechanism. A better approach is to exploit the asymptotic analysis to analyze the algorithm complexity. Although this approach is currently widely used, it needs strong mathematical background and experience. Appreciation of algorithm behavior is usually difficult because of the inability to see through mathematical functions.

Complementing the asymptotic analysis, algorithm visualization presents one with abstract behavioral models in terms of animated views. Each view represents an interesting aspect of the behavior; e.g. data movements. Because interesting aspects can be separated and shown simultaneously, overall understanding of the algorithm behavior can be achieved easily. Furthermore, most systems also provide an interactive session for communicating with the algorithm during execution. With this capability, one can concentrate on aspects as well as directly manipulate the data structures [CoRo93].

This research is to build an algorithm visualization system under the Microsoft Windows. The first step is to investigate techniques for synchronizing multiple execution threads. Second is to explore the abstract behavioral models and to implement them as animated view. Sorting algorithms are selected for study because of their simplicity and known

---

[1] The research is supported by the National Electronics and Computer Technology Center.

[2] Dept. of Computer Engineering, Chulalongkorn University, Bangkok 10210, e-mail : somchaip@chulkn.chula.ac.th

[3] Div. of Computer Science, Natl. Inst. of Dev. Admin., Bangkok, Thailand 10240, e-mail : wittaya@as.nida.ac.th

behavior. Furthermore, using them will enable us to compare with other systems because they are extensively used in the last decade.

This paper describes the usage of our algorithm visualization system from the end-user point of view. Detail of the system as well as usage from the visualizing designer view point can be found in [PrWa94a] and [PrWa94b]. Presentation of the rest of the paper is as follows. Section 2 outlines the overall design of the system. Section 3 explains the interactive session showing how to use the system from the end-user view point. Finally, conclusion follows in section 4.

2. Overall Design

The algorithm visualization system was implemented on the PC platform under Microsoft Window. The system structure is shown in figure 1. There are a single copy of the user interface and algorithm synchronizer and multiple copies of algorithm visualizer. Each copy of the algorithm visualizer consists of an input generator, the algorithm, an output generator, and many views.

The user interface enables the end-users to have overall control of the system. End-users can specify the input pattern, view, and execution speed. Since one may want to observe the behavior of various algorithms at any one time, each algorithm visualizer will support each algorithm execution. These execution threads must synchronize to reflect the amount of work done during the same period. The measured time is in terms of the number of basic operations. A basic operation can be defined by the number of comparisons or assignments or any countable operations.

A view represents a visual abstract model that evolves according to the algorithm execution. There may be many abstract models (views) associated with an algorithm. Each model serves or highlights an interesting aspect of the algorithm.

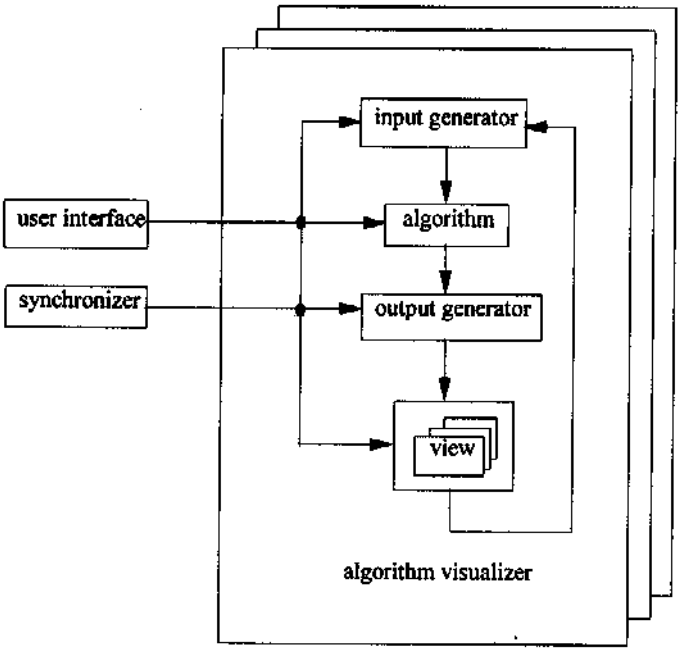


Figure 1. Structure of the environment.

There are two classes of users of the system. First is the visualizing designers. The designer will map the algorithm behavior to visual abstract models. Given an algorithm, the designer will define the basic operation and insert the synchronization points accordingly. Output events are also inserted. The system will automatically transform and transfer these events to the appropriate views. After the design step, code will be implemented and catalogued into the system.

Second is the end-users (usually students) who select the catalogued algorithms and run them. The end-user can play around by specifying various input patterns, and various views for each algorithm. Many algorithms can be run at the same time. During execution, one may slow down, pause, or restart the execution.

3. The Environment

This section describes the environment emphasizing the interaction between the end-user and the visualization system for sorting algorithms. Sorting is chosen because it is easy to understand and we can use it to compare with other visualization systems.

There are two classes of windows in the system. First is the control center window. Users interact with the control center to create the second class of windows, the algorithm windows. Many algorithm windows can be created and each is associated with an algorithm. With multiple algorithm windows, many algorithm executions can be compared.

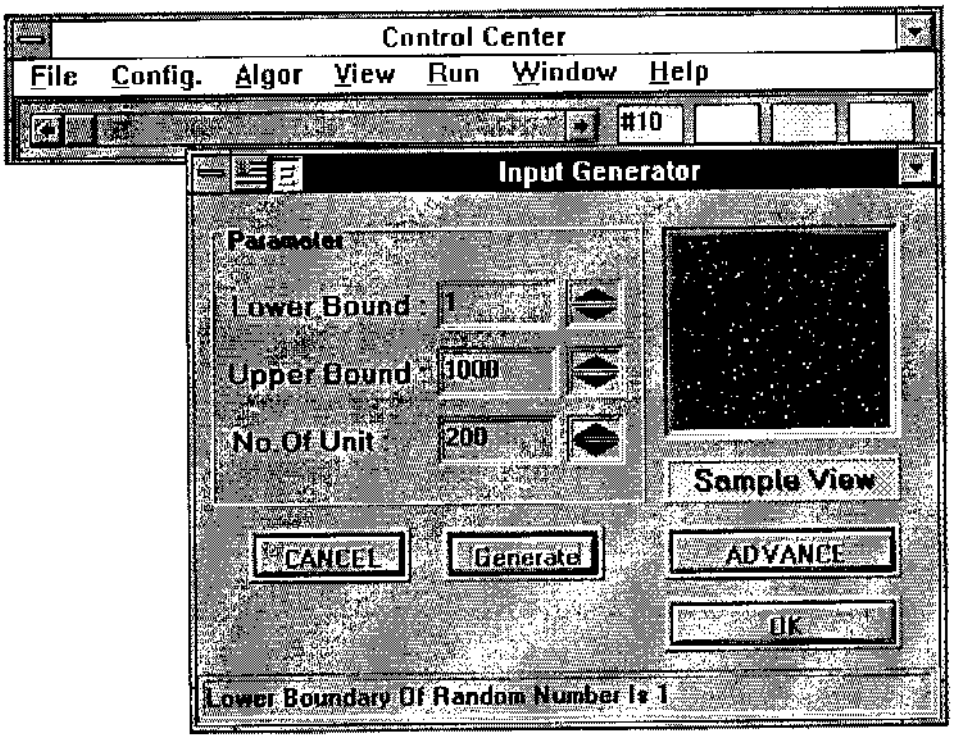
3.1 Control Center

Figure 2. Commands within the Control Center.

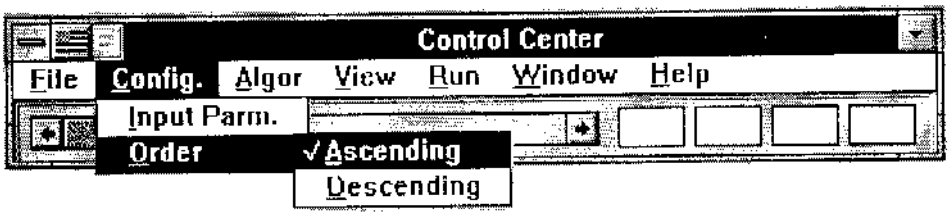
Figure 2 shows the control center window with important commands in the menu bar. The menu bar consists of seven pull-down menus: File, Config., Algor, View, Run, Window, and Help. All menu choices will not be explained. Only those containing relevant commands are described. The scroll bar in the window is used for controlling the executing speed of algorithms.

The Config. menu consists of two commands for generating input. Input Parm. pops up the input generator window as shown in figure 3a. End-users can specify the amount of data and their lower and upper bound. Clicking the Generate button will randomly generate the data and show them in the sub-window. Each point in the sub-window represents a datum. The horizontal axis is the datum number, whereas the vertical axis is the datum value. Because data are generated randomly, all points appear scattering in the sub-window. The Order command in the Config. menu enables one to sort in the ascending or descending order (see figure 3b).

Figure 4 shows the Algor, View, and Run menu. The visualized algorithm can be selected from the Algor menu. In this experiment, sorting algorithms are the only choice. The View menu enables one to create various views for an algorithm. Two kinds of views exist in this experiment. The Points view presents data as points in a two-dimensional space described earlier. The Stick Bars view presents data as vertical sticks along a horizontal axis. The stick length is proportional to the datum value. Finally, with the Run menu ones can start, pause, and stop the execution of all algorithms.

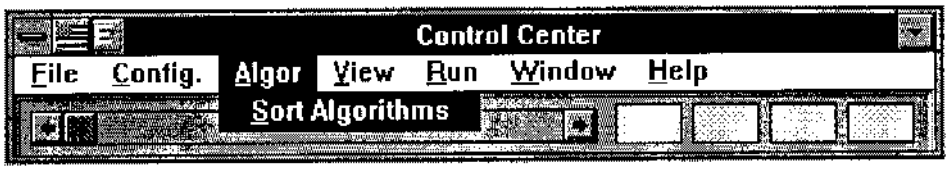


(3a) Input Characteristics.

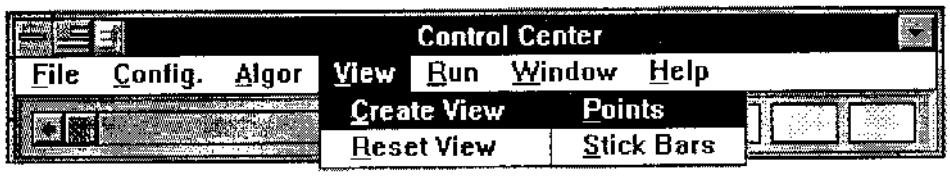


(3b) Input Menu.

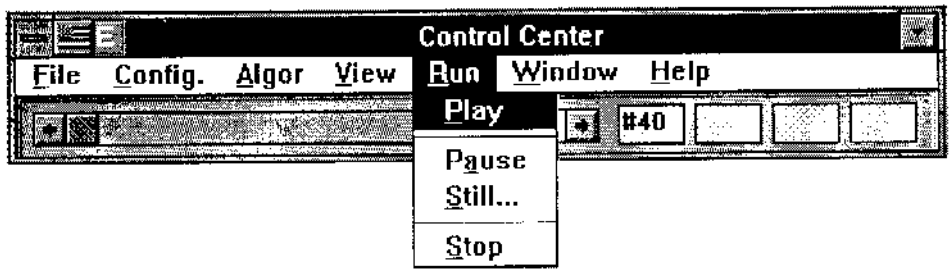
Figure 3. Input Generator.



(4a) Algorithm Menu.



(4b) View Menu.



(4c) Run Menu.

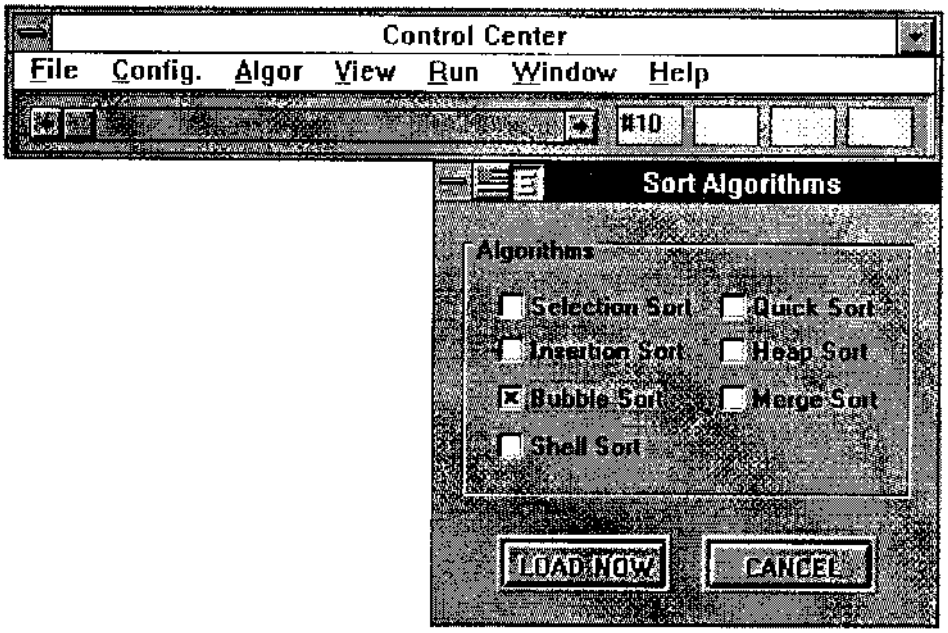
Figure 4. Algor, View, and Run Menu.

3.2 Algorithm Window

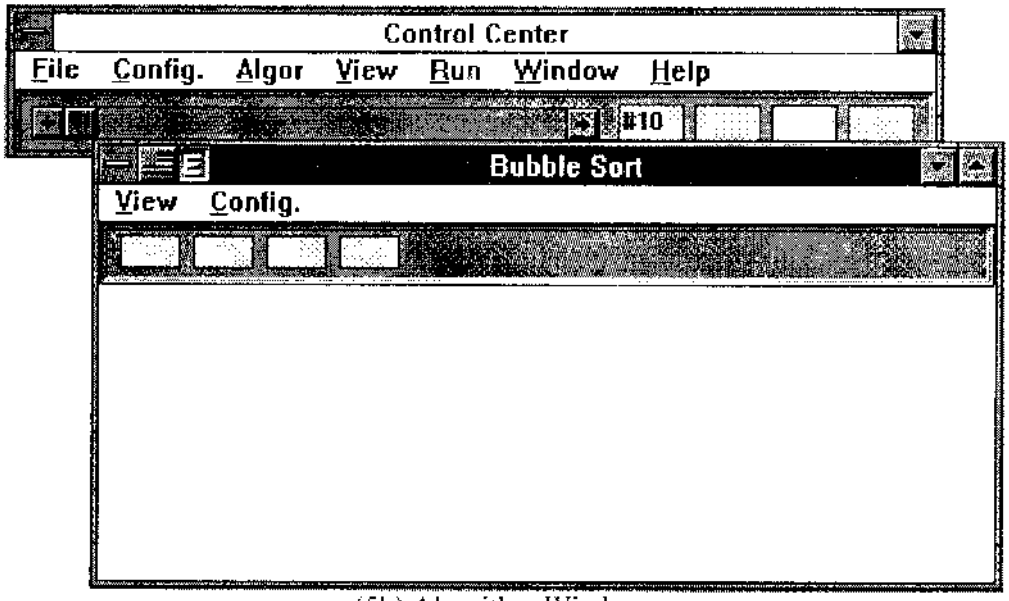
End-users can use the Algor menu to choose one from various sorting algorithms. Figure 5a shows the algorithm selection window. After selecting an algorithm and clicking the LOAD NOW button, an algorithm window will be created. Figure 5b illustrates this effect. The menu bar within the algorithm window consists of menus similar to those of the control center.

After creating an algorithm window, one uses the View menu in the control center to create views. Then input data are generated using the Config. menu. Figure 6a shows the Bubble Sort window with its two views: the Stick Bar view and the Point view. Notice that in this figure because data are randomly generated, data points appear scattering and data sticks appear zigzagging.

To visualize the algorithm, one clicks the start command in the Run menu in the algorithm window and clicks the play command in the Run menu in the control center window. While the algorithm is running, the execution speed can be control form the scroll bar in the control center. Viewing the algorithm execution in action, one should be able to understand how the algorithm gradually works to achieve its goal. Assuming sorting in ascending order, it is not difficult to predict the final view when data are sorted. In the Points view, points should align as a positive-sloped line, whereas in the Stick Bar view, sticks should be arranged as a conic shape along the horizon. Figure 6b and 6c demonstrate these results.



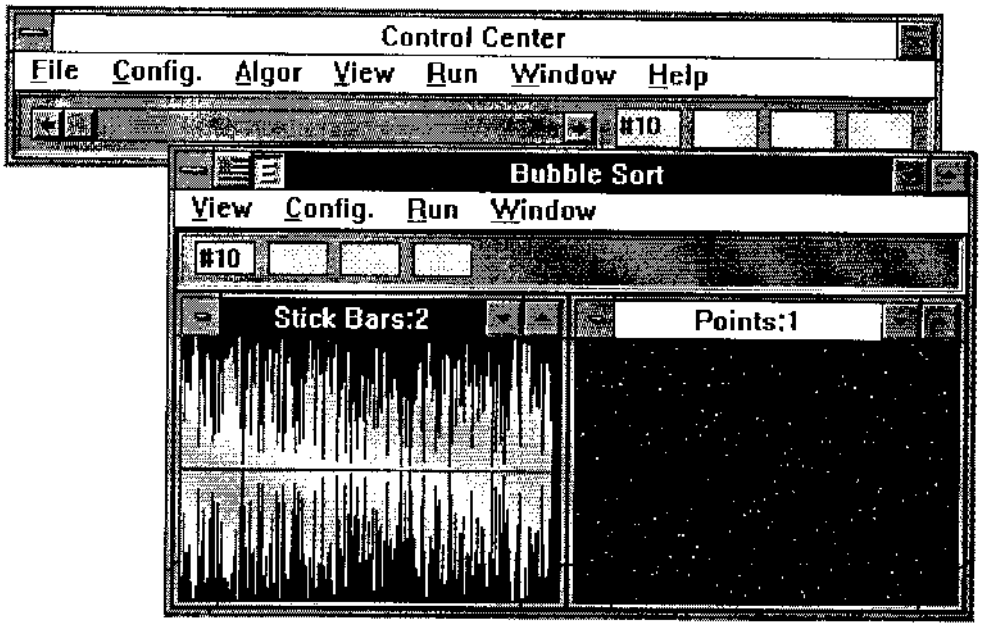
(5a) Selecting an Algorithm.



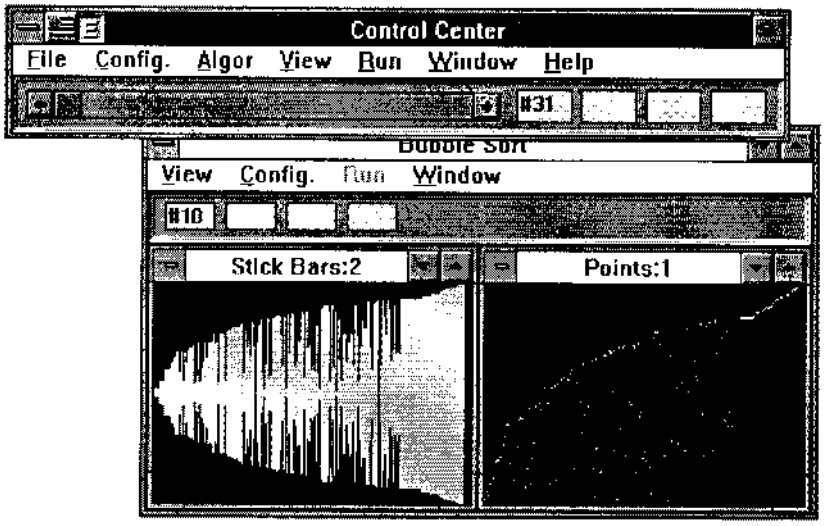
(5b) Algorithm Window.

Figure 5. Visualizing an Algorithm.

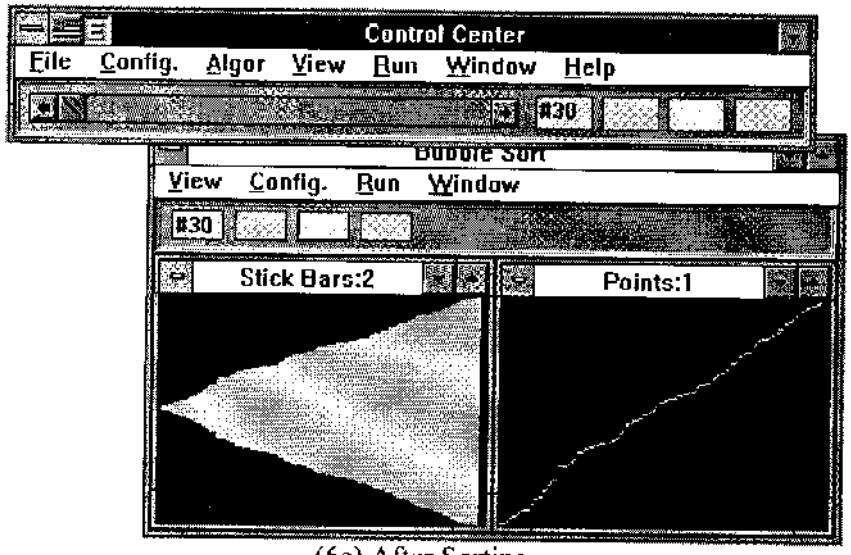
For sorting algorithms, performance (complexity) depends on the number of data movements and comparisons. Asymptotic analysis of various algorithms are presented in most data structure text books. It is hoped that visualizing these two characteristics will help one appreciate the analysis.



(6a) Before Sorting.



(6b) Sorting in Action.



(6c) After Sorting.

Figure 6. Bubble-sort Algorithm Window with two Views.

To compare any two algorithms, two algorithm windows are opened from the control center. Views and data generation are created similarly to the one-algorithm-window case. After starting each algorithm within its window from the Run menu, execution will not commence. Only when clicking the Play command from the Run menu in the control center that all algorithms commence their running simultaneously. Figure 7 demonstrates the visualization of two algorithms: insertion sort and quicksort. In this figure, notice that quicksort should finish much earlier than insertion sort. Hence, the effect of $O(n \log n)$ and $O(n^2)$ can be appreciated.

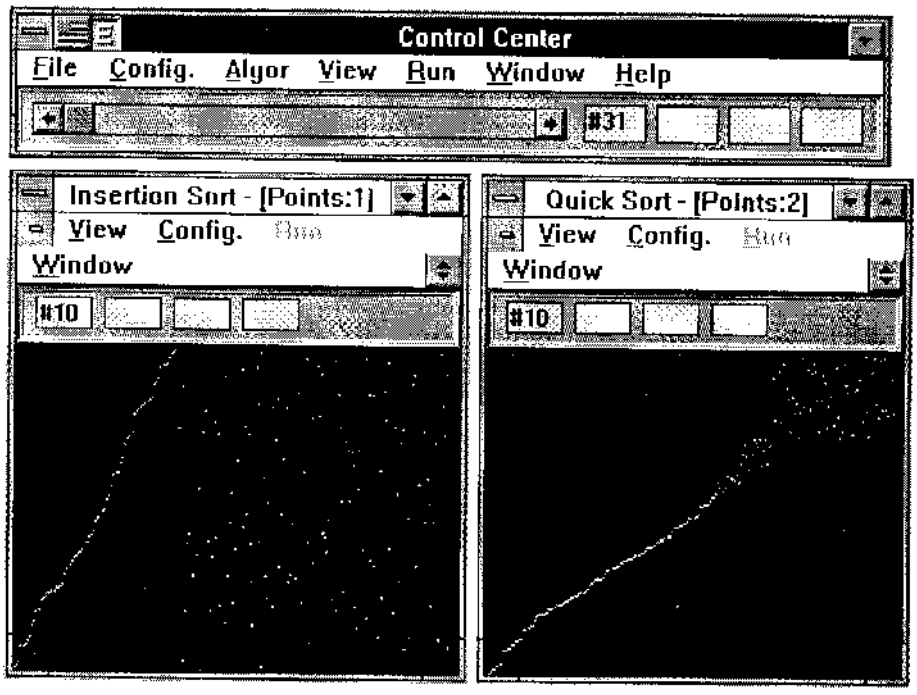


Figure 7. Insertion Sort versus Quicksort.

The above examples provide the general idea that can be used to explore any sorting algorithm. With multiple algorithm windows, many algorithm behaviors can be investigated at the same time. One can also study the worst case and the best case. Furthermore, the input generator can be extended to generate various data patterns. Visualizing the algorithm behavior driven by these patterns can be interesting.

4. Conclusion

This paper describes an algorithm visualization system for sorting algorithms. The system is run under a window environment. End-user interaction is centered in the control center window. From the control center, many algorithm windows can be created. The control center will control and synchronize the execution speed and operations of all algorithms. The control center is also equipped with the input generator that can create various input patterns. Within an algorithm window, many views can be opened. A view presents an abstract behavioral model that represents the algorithm in execution. A good abstract behavioral model should help one understand how the algorithm gradually works to achieve its goal.

Although sorting algorithms have been used extensively in many algorithm visualization systems, other algorithms can follow the same trait. The research is currently exploring the abstract behavioral models for various graph algorithms. A good algorithm visualization system should be a powerful tool for studying algorithm analysis and design.

References

[CoRo93] K.C. Cox, and G-C Roman, A *Taxonomy of Program Visualization Systems*, Computer, Vol. 26, No. 12, Dec. 1993.

[PrWa94a] S. Prasitjutrakul, and W. Watcharawittayakul, *Design and Implementation of an algorithm visualization system*, Progress Report (in Thai), National Electronics and Computer Technology Center, Bangkok 10400, Thailand.

[PrWa94b] S. Prasitjutrakul, and W. Watcharawittayakul, *An Algorithm Visualization System under Microsoft Windows Operating Environment*, Proceedings, National Computer Symposium 1994, Bangkok, Thailand.