UDC 004.942(045)

[1]**Victor M. Sineglazov**, D.E., Prof.
[2]**Sergiy G. Kyevorkov,** Post-graduade student
[3]**Raad Kareem Kadhim,** Post-graduate student

CHOICE OF THE PROJECT DESIGN DATA INTEGRATION MODULE SORTING ALGORITHM OF A COMPUTER-AIDED DESIGN

National Aviation University
[1,3]E-mail: svm@nau.edu.ua
[2]E-mail: kyevorkov@ukr.net

Investigation of computer-aided design data module integration algorithms. The analysis of data sorting algorithms was realized. Algorithms for tasks sorting of project data integration module were suggested.

Проведено дослідження формування метамоделі системи автоматизованого проектування та її компонентів. Розроблено ієрархію метамоделей та моделей системи. Проведено аналіз комплексів системи.

Проведены исследования формирования метамодели системы автоматизированного проектирования и ее компонентов. Разработана иерархия метамоделей и моделей системы. Проведен анализ комплексов системы.

Statement of purpose

At present time, much attention is paid to essential task of improving the effectiveness of computer-aided design systems in the way of more full use of databases (in digital format and symbolic) for solving of problems that are not directly linked with design, for example in the technological preparation of production. For solving such a problem in the work [1] was suggested the approach of CAD development to the problems of production technological preparation. Structure scheme of such CAD is given on fig. 1.

A distinctive feature of the CAD is the presence of project data integration module. The main elements of the module are: database, adapters to CAD systems, interfaces to external systems, end-user interface, administration tools and customization of data, algorithms of sorting, consolidation, retrieval, conversion and data conversion. The design data integration module structure is illustrated fig. 2.

An example of using a design data integration module at the stage of technological preparation of production (TPP) is to launch in the production of a list of objects (parts and assembly unit) in the official notes. During processing of the official note the list of incoming objects is defined. Depending on the number of objects sorting on the marking, on the material, on the group of technological processing is performed. During performance of the object analysis (part or assembly unit) a query to find the information objects (three-dimensional models, electronic circuits or drawings), is carried out in the CAD environment.

During the TPP objects sorting I done according to the following grounds:

– matching the model detail or assembly unit, for example: 77.01.0110.912;

– name of the model parts or assembly units, such as: bracket, stringer, left beam;

– material; for example: B95, 1163, D16T;

– type of the detail technological process; for example: cast, extruded, bent.

Sorting is included into the list of tasks necessary to perform the TTP and access to CAD data. Let's analyze the sorting algorithms in terms of their use in the project data integration module.

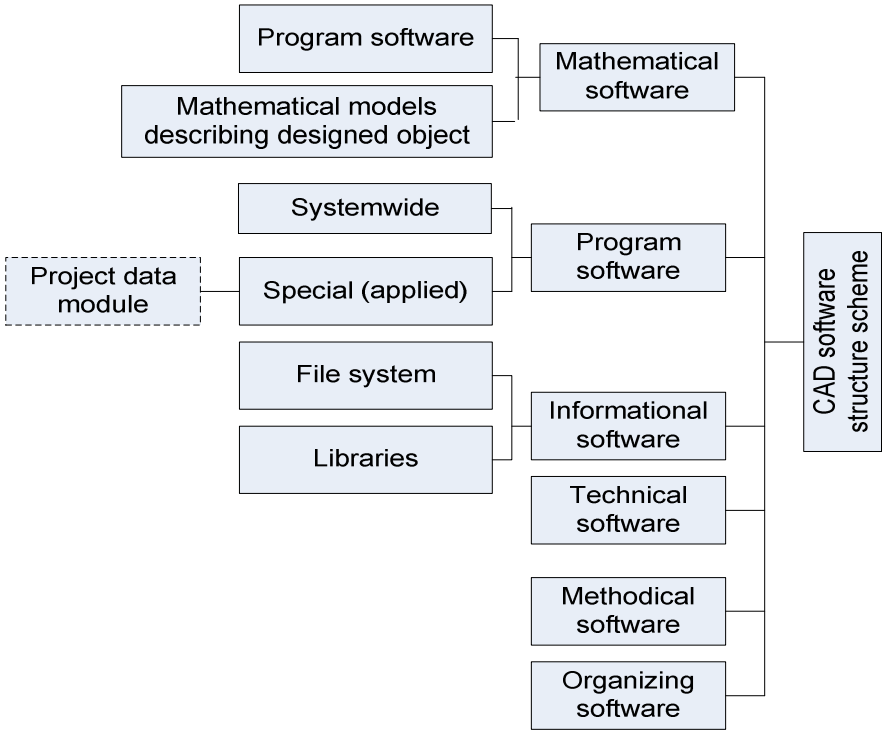

Fig. 1. CAD structure with package data integration module

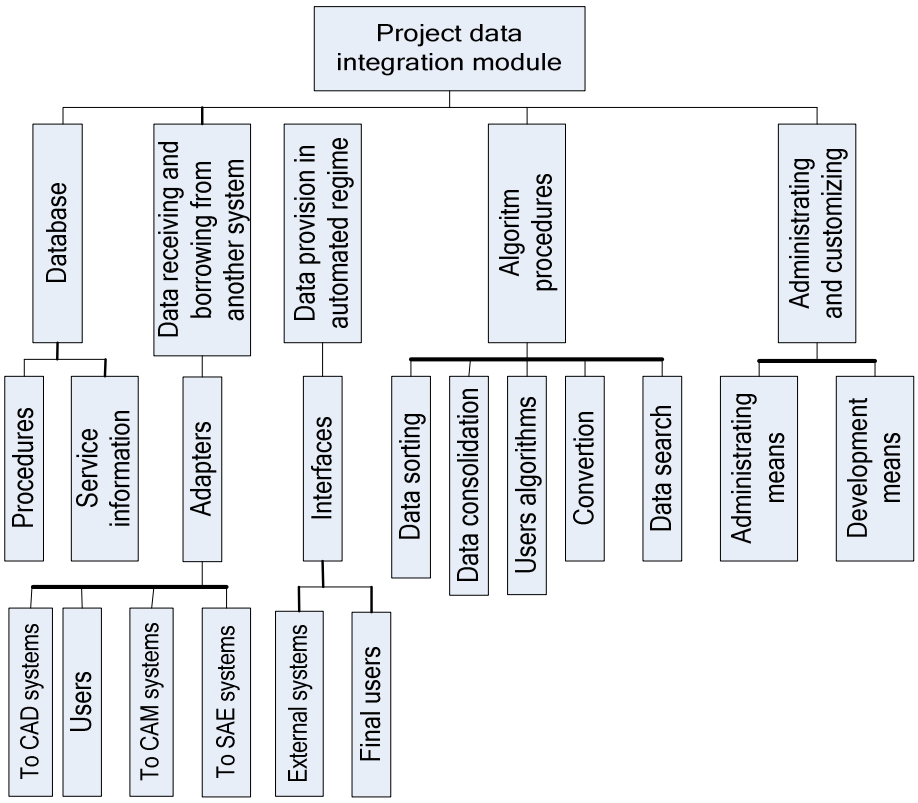

Fig. 2. Design data integration module structure

Problem statement

If there is a sequence a_0, a_1... a_n and comparison function, which on any two elements of sequence takes one of three values: less, than or equal to. The task is to sort the permutation of the members of a sequence in such a way as to satisfy the condition: $a_i <= a_i+1$, for all i from 0 to n. If there are several (*x*, *y*, *z*) fields, then x is called the key, by which the sorting is realized.

For the algorithms analysis the next criteria are used:

– sort time;

– volume of operative memory;

– stability (stable sort does not change the relative position of equal elements).

The most widespread sorting algorithms are:

– bubble sort;

– insert sort;

– shaker sort;

– merge sort;

– fast sort;

– shell sort.

Let's make a comparison [1, p. 20–80] of these algorithms by suggested above criteria.

Algorithm 1. Bubble sort

Implementing of this method doesn't require any additional memory [2, p. 120–132; 3]. The method is as follows: a pair is taken of the adjacent elements, and if the element with lower index is higher than the element with a larger index, then we change their places. These actions are continued until there are such pairs. When there are no such pairs, then the data will be sorted. To search for such pairs of data is over looked from the beginning to the end. From this it follows that during this overview a maximum is found, which is placed at the end of the array, and so the next time enough to watch a smaller number of elements. As every time at its own place gets at least one element, then do not need more than N passages where N – number of elements.

Algorithm 2. Insert sort

A new array is created, into which elements are consistently inserted from the original array so that the new array was ordered.

Insertis performed in the following way: at the end of the new array is allocated a free cell, then analyzed element, standing in front of an empty cell (if a blank cell is not on the first place), and if this is more an element is inserted, then moves the item into a free cell (at its own place a free cell is formed), and the next element is compared. So we go to the case where the element in front of an empty cell is less than inserted, or an empty cell is at the beginning of the array. Put the plug-in to an empty cell.

So, in turn, insert all the elements of the original array. It is obvious that if, the array has been ordered before the element then after inserting before the inserted element are located all elements smaller than him, and after it - larger. Since the order of elements in the new array does not change, then the array will be formed into an ordered one after each insertion. It means that, after the last insertion, we obtain an ordered array of the source. This sorting can be implemented without additional array B, if you sort the array A directly in the readout-ing, ie, to insert a new element in the array A.

Algorithm 3. Shaker sort

When the data is sorted not in operative memory and hard disk space, especially if the key is associated with a large amount of additional information, the number of displacements of elements substantially affect the operating time. This algorithm reduces the number of such displacements, and acts as follows: by means of one pass through all the elements the minimum and maximum one is selected. Then the minimal element is placed at the beginning of the array, and the maximum, respectively, at the end. Then, the algorithm is executed for the remaining data. Thus, for each pass, two elements are put into their places and, hence, need N / 2 passes, where N - number.

Shaker method is more benefit because of sorting data on external memory devices, and it requires a half less permutations then the algorithm №1 and №2.

Algorithm 4. Merge sort

This sorting uses the following subtasks: there are two sorted arrays, you need to do (unite) one but sorted. Sorting algorithm works on the principle: divide the array into two parts, sort each of them, and then merge the two parts into one sorted.

To estimate the time necessary for this algorithm, we form a recurrence.

Let T (n) - sorting time an array of length n, then for mergesort rightly

T (n) = 2T (n / 2) + O (n) (O (n)

this time, it is necessary to have drained two arrays). Write down this ratio:

It remains to estimate k. We know that

2k = n,

and hence

k = log2n.

The equation takes the form

T (n) = nT (1) + log2nO (n).

Since T (1) - constant, then

T (n) = O (n) + log2nO (n) = O (nlog2n).

That is, the time estimate of the merge sort is less than at the first three algorithms.

Algorithm 5. Fast sort

Like merge sort [4], the array is divided into two parts, with the condition that all elements of the first part is less than any element of the second. Then each part is graded separately. Partitioning is achieved on the part of the ordering with respect to some element of the array, i.e., in the first part of all numbers less than or equal to this element, and second, respectively, greater than or equal. Two indexes are held on the array from different directions and looking for items that were not in their group. Finding such items, exchange them. The one element on which the indices will intersect, determines the group devision.

Time O (nlog2n) is the minimum for sorting, which use only a paired comparison of elements and not use the structure of the elements.

Algorithm 6. Shell sort

For each element it is necessary to find how many less than a certain number items are there, and place this number at the appropriate place [1, p. 60].

For a linear array, we pass on to the calculation of each possible value, and how many items have the same value. Then add to each of the numbers the sum of all previous. Obtaining in such a way the number of elements whose values are not more of the given value. For the linear passage formed from the original array new sorted. Two same elements are tracked not to be recorded in one place.

The method does not use nested loops and, taking into account the small range of values, hits work duration is O (n).

Another important feature of algorithms is its sphere of application. There are two main positions:

– internal sorting work with data in memory with random access;

– external sorting organizes information located on external memory devices.

This imposes some additional restrictions on the algorithm:

– access to the memory device has been ongoing the consistent way: each instant period of time one can read or write only the element following by the current one t

– the amount of data does not permit them to stay in RAM

In addition, access to data on the memory device is much slower than the operation of RAM.

This class of algorithms is divided into two main sub-classes:

Internal sorting operates with arrays, entirely fist in memory with random access to any cell. All data usually graded at the same place, without its Additional Cost.

External sorting operates with storage devices of a large volume, but access is not arbitrary, but consistent (sort of files), i.e. At the moment we 'see' only one element, and the cost of skipping over the memory of unfairly high. This leads to special methods of sorting, typically using additional storage space.

The results of given algorithms comparison using time criteria are shown on fig. 3 [5].

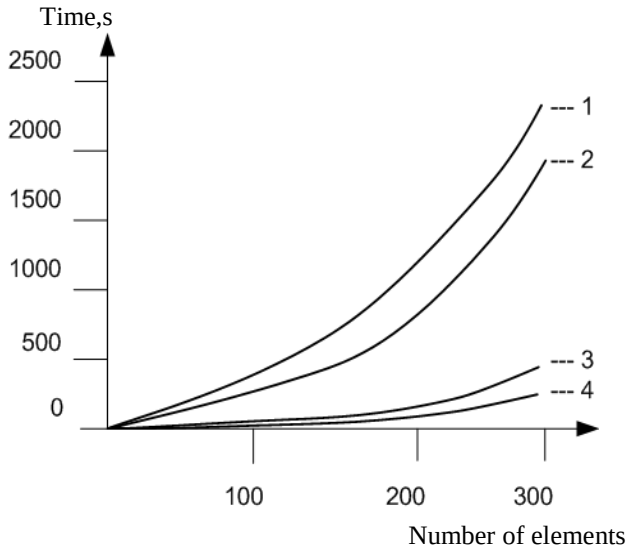

Fig. 3. Comparison of sorting time duration depending on the number of elements:
1 – bubble sort;
2 – shaker sort;
3 – insert sort;
4 – сортировка Шелла

Given analysis shows that the most optimal in time is Shell's sorting.

Conclusion

As a result of conducted research the analysis of sorting algorithms was performed. For use of project data in integration module it is better to use Shell's algorithm.

References

1. *Кнут Д.* Искусство программирования. Т. 3. Сортировка и поиск / Д. Кнут. = The Art of Computer Programming. Vol. 3. Sorting and Searching. – 2-е изд. – М.: Вильямс, 2007. – 824 с.

2. *Алгоритмы*: построение и анализ: 2-е изд. / Томас Х. Кормен, Чарльз И. Лейзерсон, Рональд Л. Ривест, Клиффорд Штайн. — М.: Вильямс, 2007. – 1296 с.

3. *Ткачук В.* Алгоритмы сортировки. Ч. 1 / В. Ткачук. – Режим доступа: http://docs.com.ru/algoritm_1.php.

4. *Ткачук В.* Алгоритмы сортировки. Ч. 2 / В. Ткачук. – Режим доступа: http://docs.com.ru/algoritm_2.php.

5. *Кантор И.* Алгоритмы сортировки / И. Кантор. – Режим доступа: http://algolist.manual.ru.

The editors received the article on 26 May 2010.