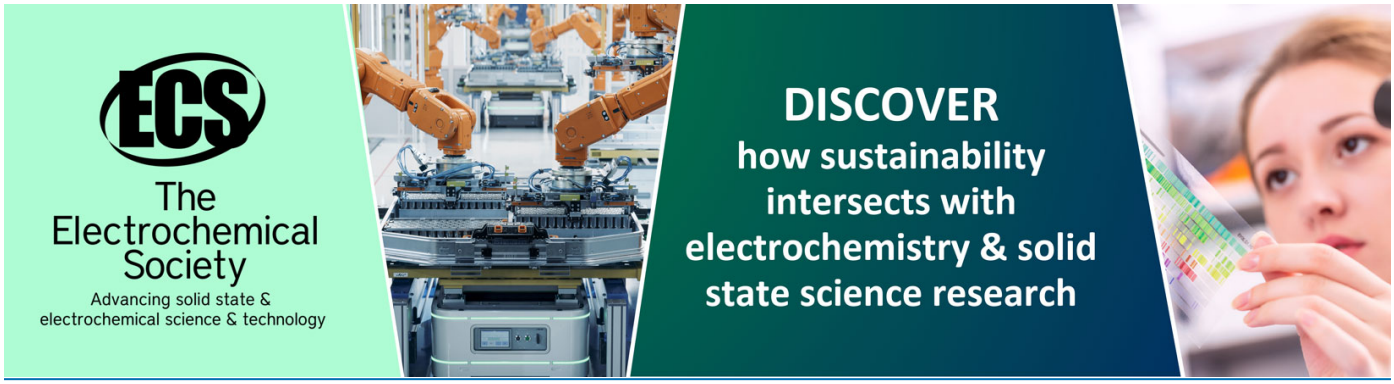
Analysis and Research of Sorting Algorithm in Data Structure Based on C Language



Analysis and Research of Sorting Algorithm in Data Structure Based on C Language

Zhu zhi-gang[1]

City Institute Dalian University Of Technology, Liaoning Dalian, 116600, China

Abstract: In the process of learning the data structure, it is very necessary to master the sorting algorithm, and in the program design, the sorting algorithm is applied frequently. Based on the importance of sorting algorithms, this paper will carefully compare the characteristics of different algorithms, starting with the work efficiency, algorithm implementation, basic ideas, sorting methods and other aspects, and draw conclusions so as to better select sorting algorithms.

1. Introduction

Based on the development of science and technology, people usually use computer technology in the process of comparing different things at this stage, especially in sorting problems, such as comprehensive analysis of product indicators, ranking test results, etc. The sorting problems are increasingly inseparable from the use of data and information processing. For data processing, sorting is a basic function. At the same time, when a computer processes a certain problem, it usually takes at least 35% of the time to sort the data. Regarding this phenomenon, for the purpose of improving the working efficiency of the computer, the more effective method is to study the sorting algorithm and compare to use a more suitable sorting algorithm. The author will combine all aspects of the sorting algorithm to study, and draw a conclusion, which should provide the basis for the actual.

2. Sorting Types and Algorithm Implementation

2.1. Definition and types of sorting

As for sorting, its operating object is a group of elements and data records that do not have a sorting rule. Reorder them according to keywords, and arrange them in descending or increasing sequence during the period. In actual data processing, if the file to be processed contains n records or elements, they are denoted as R1, R2, R3, ..., Rn respectively, and the keywords of these records are k1, k2, k3, ..., kn. The sorting operation shall be performed according to the requirements of keyword conditions, such as ki1≧ki2≧ki3≧…≧kin or ki1≦ki2≦ki3≦…≦kin, re-sorting n records, and the sorting results are Ri1, Ri2, Ri3,..., Rin.

In this process, combined with the algorithm ideas of different methods, it can be divided into four categories. One is insertion sort, including Shell's sort, binary insertion sort and direct insertion sort. The second is select sort, including heap sort and direct select sort. The third is exchange sort, including quick sort and bubble sort. The fourth is merge sort. The fifth is the radix sort [1].

2.2. Insert sort method

With regard to insert sort, the basic idea is to compare keywords and elements to be sorted, and put the elements into the existing sequence column until all elements are inserted. Combined with the

operation method, it includes Shell's sort, binary insertion sort and direct insertion sort.

2.2.1. Direct insertion sort

Regarding direct insertion sorting, in actual sorting operations, the keywords of R[i] and R[j] elements (j=1,2, 3, ...) are compared, and once R[i] is less than R[j], R[j] is moved backward; If R[i] is greater than or equal to R[j], the search ends. At the same time, R[i] is placed at position j+1.

2.2.2. Binary insertion sort

Regarding the binary insertion sort, it is an improved version of the direct insertion sort, which uses the binary search method to determine the insertion position, then moves the elements behind the insertion position backward, and finally inserts the elements into the ordered table. In the specific sorting process, if there is a group of data that needs sorting processing, make its sorting interval R[1]-R[n]. In the first sorting, the default R[1] is ordered and R[2]-R[n] is unordered [2]. At the same time, the position of the first element in the ordered data position is set to low and the position of the tail element is set to high.

2.2.3. Shell's sort

As for Shell's sort, it is also called diminishing increment sort. That is, in specific operations, incremental grouping is selected first, and then operations are performed according to the direct insertion principle until all elements are sorted.

The algorithm code is as follows:

```
#define MAXINT 37658
Shellsort（R）
{
  RecType R[n/2+n]；//Define array
  int i，j，d；//Definition element
  RecType x；//Definition element
  for（i=0；i<n/2，i++）
  {
    R[i]=MAXINT；//Assignment
    d=n/2；//Calculate increment
    while（d>=1）//When there are unsorted elements
    {
      for （i=d+n/2；i<n/2+n；i++） //When the increment decreases to 1, the sorting is completed
      {
        x=R[i]；//Assignment
      j=i-d；
      While（x<R[j]）//When R[i] is larger than R[j], insertion processing is performed
      {
        R[j+d]=R[j]；
        j=j-d；
      }
      R[j+d]=x；
      }
    d=d/2；//Calculate incremental values to determine whether to run next time
    }
  }
}
```

2.3. Swap sort method

Regarding swap sort, its basic idea is to compare keywords to be sorted. Once they do not meet the requirements, exchange processing will be carried out. If so, the original order is maintained until all elements are sorted.

2.3.1. Bubble sort

As for bubble sort, it follows the rule of "light above, heavy below". In actual operation, multiple scans are required to determine the order of array R until all elements meet the requirements [3]. The algorithm code is as follows:

```
void bubble_sort (intR[]，int n)
{
    int i，j，lastSwap，tmp；//Definition element
    for (j=n-1；j>0；j=lastSwap) //When there are unsorted elements, bubbling is performed
{
     lastSwap=0；//Each round should be initialized to 0
        for (i=0；i<j；i++)//Starting from the first element, exchange operations are performed
       {
            if (R[i] > R[i+1]) //If the former element is smaller than the latter element
              {
                tmp=R[i]；
                R[i]=R[i+1]；
                R[i+1]=tmp；//The above three are exchange orders
                lastSwap = i；//Last exchange of positions
            }
        }
    }
}
```

2.3.2. Quick sort

For quick sort, it is an improved version of bubble sort, which runs faster. The basic idea followed in its operation is as follows. First, in the unordered array R, any element R[i] is taken as the reference value temp. After that, all elements are divided into two groups by using temp, one group of elements is all smaller than the reference value temp and is on the left; Another group of elements are all greater than the reference value temp, which is on the right. At the same time, the reference value temp is in the middle position and is also the final sorting position [4]. According to the method, the operation is repeated until all elements meet the requirements.

2.4. Select sort method

Regarding the select sort, the basic idea is to select the largest (smallest) of all elements and then put it into the file according to regulations until all elements meet the requirements.

2.4.1. Straight select sort

For direct select sort, when the first run is performed, a largest (smallest) record is selected in the array R, and change the position of R[0] with this element. On the second run, select the largest (smallest) element from the remaining records and change positions with R[1]. Continue to run according to the above rules until all elements are sorted [5].

2.4.2. Heap sort

As the stability of straight select sort is not strong in the operation process, relevant personnel improved it, forming a heap sort, which belongs to tree selection sort. In the specific sorting, firstly,

according to the requirement of complete binary tree, the relation is established for array R elements [6]. After that, with the help of children and parents nodes of complete binary tree, the minimum keyword is selected, thus achieving the sorting purpose. The code of this algorithm is as follows:

```
Sift（R，i，k） //The screening method adjusts records R[i] to R[m]
  {
    RecType R[]；//Define array
    int i，k；
    int j=2*t；
    RecType temp=R[i]；//Assignment
    while（j<=k）
  {
    if （（R[j]<R[j+1]）&&（j<=k））// Adjust the sequence from the end and narrow the range
    j++；
    if （temp<R[j]）
    {
      R[i]=R[j]；
      i=j；
      j=2*i；
    }
    else break；//Exit loop
  }
    R[i]=temp；//Assignment
  }
```

2.5. Merge sort method

Regarding merge sort, it is to merge multiple sequential files and observe the requirements during the process. Among them, the most frequently used is the two-way merge sort. In the sorting process, firstly, all elements are regarded as ordered subfiles [7]; After that, all the files are merged, and the principle of merging two files is followed during the process. Then, sort the two elements in the file. Follow the above rules for processing until n element files are formed.

```
MergeSort（R）
  {
    RecType R[]；//Define array
    int length=1；//Define initial values
    while（length<n）//If there are elements that are not sorted
    {
      MengePass（R，R1，length）；
      Length=2* length；/Calculate the number of groups/
      MengePass（R1，R，length）；//Reference function
      Length=2* length；
    }
  }
```

2.6. Radix sort method

As for radix sort, it is an extension of bucket sort. In the process of operation, integers are divided into individual numbers according to the number of bits, and then each number is compared in sequence starting from the lowest bit.

3. Performance Analysis of Sorting Algorithm

Comprehensive comparison of the performance of different algorithms, the specific situation is shown in Table 1.

Table 1 Comprehensive Performance Comparison of Different Sorting Methods

Sort type	Average time	Worst case	Storage space	Stability
Simple sort	O（n^2）	O（n^2）	O（1）	Stable
Quick sort	O（$n\log_2 n$）	O（n^2）	O（$n\log_2 n$）	Unstable
Select sort	O（n^2）	O（n^2）	O（1）	Unstable
Heap sort	O（$n\log_2 n$）	O（$n\log_2 n$）	O（1）	Unstable
Merge sort	O（$n\log_2 n$）	O（$n\log_2 n$）	O（n）	Stable
Radix sort	O（d（n+r））	O（d（n+r））	O（r）	Stable

According to Table 1, the following conclusions can be drawn: First, bubble sort and insertion sort are simple sort. If the number of elements n is small or the elements are basically ordered, direct insertion sort is the simplest [8]. Secondly, considering the time performance of different sorting methods, merge sorti, quick sort and heap sort are better. However, if it is in the worst case, the quick sort is poor. If the performance of storage space is compared, merge sort needs more space. Finally, it is relatively stable and simple sort is the best.

4. Conclusion

To sum up, in actual operation, it is necessary to select an appropriate algorithm. During this period, it is necessary to comprehensively analyze the stability, occupied space, time performance, etc. and consider the actual situation. For example, when the number of elements is small, quick sorting, heap sorting and merge sorting are more practical. At the same time, none of the sorting methods is optimal, and the required conditions are favorable, so multiple methods can be comprehensively applied.

References：

[1] Lu Yang, Diao Mingguang. Teaching Exploration of Classical Algorithms in "Data Structure" Course[J]. Education Forum, 2017(37):136-137.
[2] Liang Jianhua. Analysis and Research of Sorting Algorithm in Data Structure Based on C Language[J]. Journal of Luohe Vocational Technology College, 2019(3):2-3.
[3] Li Feng. Research on Uncertain Frequent Pattern Mining Algorithm Based on Minimal Data Structure[J]. Journal of Hunan University of Arts and Science(Natural Science Edition), 2019(2):14-14.
[4] Deng Dingsheng. Discrete Hybrid Research Based on C Programming and Data Structure[J]. Computer Knowledge and Technology, 2019(14):14-19.
[5] Ye Guo. Digital Elevation Model and Data Structure Construction Based on Hybrid Algorithm[J]. Electronic Technology & Software Engineering, 2019(12):4-4.
[6] Shi Tingting, Liu Weihua, He Chaobo. Research on the Practical Effect of Applying WeChat Public Platform in the Curriculum Reform of "Data Structure and Algorithm"[J]. Information Recording Materials, 2019(7):19-19.
[7] Gong Shengdong. Research on Practical Teaching Method of "Data Structure" Course[J]. Digital World, 2017(9):194-194.
[8] Zhou Yihua, Ji Wen, Yang Yuguang. Database ciphertext retrieval scheme based on f-mOPE[J]. Journal of Electronics & Information Technology, 2019, 41(8):14-14.