**Task 1**

a)

Test A1 – generate n = 30000 random values from -35500 to 36600

Test A2 – generate n = 40000 random values from -35500 to 36600

Test A3 – generate n = 50000 random values from -35500 to 36600

Test A4 – generate n = 60000 random values from -13800 to 96800

Test A5 – generate n = 70000 random values from -13800 to 96800

Test A6 – generate n = 80000 random values from -13800 to 96800

Test B1 – generate n = 30000 unique, random values from -35500 to 36600

Test B2 – generate n = 40000 unique, random values from -35500 to 36600

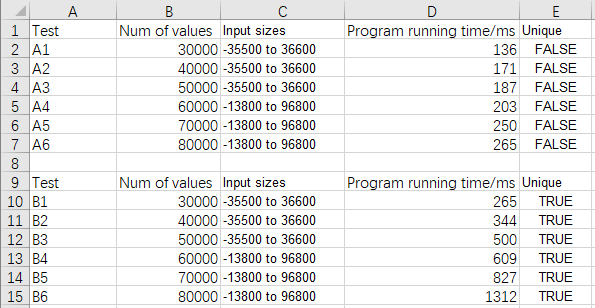
Test B3 – generate n = 50000 unique, random values from -35500 to 36600

Test B4 – generate n = 60000 unique, random values from -13800 to 96800

Test B5 – generate n = 70000 unique, random values from -13800 to 96800

Test B6 – generate n = 80000 unique, random values from -13800 to 96800

b)

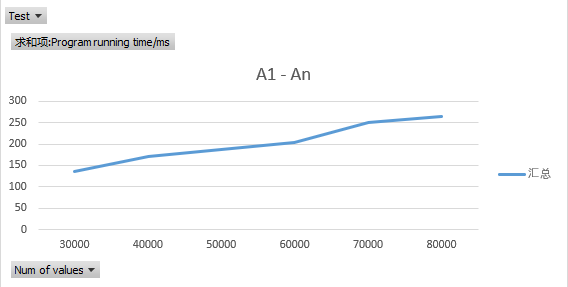


The Java application is in “He Mingli A1 Java\task1” file.

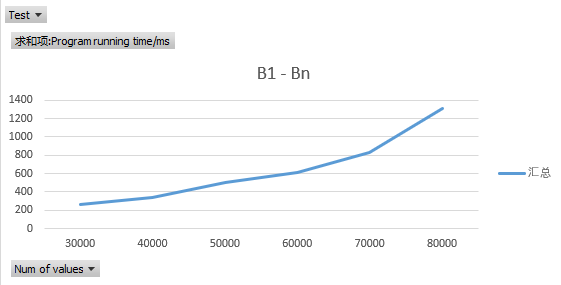
Print results are stored in ‘Test1 results’ file.

c)

Graph #1 – the timing results for Test A1 – An (non-unique random values)



Graph #2 – the timing results for Test B1 – Bn (unique random values)



d)

Following the above results, we can clear to know that the large number of values spend more time to generate numbers. In addition, for A1 to An, these random numbers are only needed to generate once. But unique output needs to traverse the output in the existing list to determine whether it is repeated, so that we need to add loop function, that led to spend high time to algorithm. Furthermore, if n (Num of values) goes bigger, the time cost of unique outputs is much more that A1 – An.

Therefore, A1 – An Big-O characterization:

f(n) = n

B1 – Bn Big-O characterization:

f(n) = n^2

**Task 2**

Test C1 – generate n = 30000 random values from -35500 to 36600, insertion sort

Test C2 – generate n = 40000 random values from -35500 to 36600, insertion sort

Test C3 – generate n = 50000 random values from -35500 to 36600, insertion sort

Test C4 – generate n = 60000 random values from -13800 to 96800, insertion sort

Test C5 – generate n = 70000 random values from -13800 to 96800, insertion sort

Test C6 – generate n = 80000 random values from -13800 to 96800, insertion sort

Test D1 – generate n = 30000 random values from -35500 to 36600, quick sort

Test D2 – generate n = 40000 random values from -35500 to 36600, quick sort

Test D3 – generate n = 50000 random values from -35500 to 36600, quick sort

Test D4 – generate n = 60000 random values from -13800 to 96800, quick sort

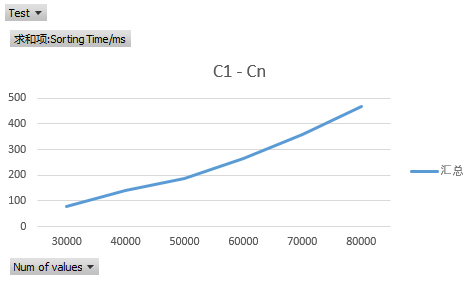
Test D5 – generate n = 70000 random values from -13800 to 96800, quick sort

Test D6 – generate n = 80000 random values from -13800 to 96800, quick sort

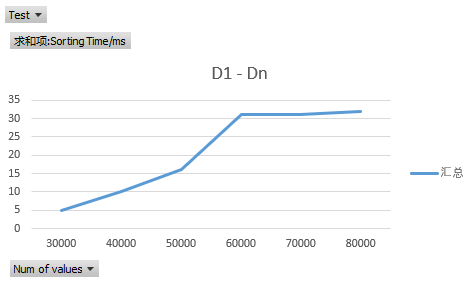
The Java application is in “He Mingli A1 Java\task2” file.

Print results are stored in ‘Test2 results’ file.

Graph #1 – the timing results for Test C1 – Cn (insertion sort algo)

****

Graph #2 – the timing results for Test D1 – Dn (quick sort algo)

****

Following the above results, we find that the quick sort algo spend much lower time than insertion sort algo, because the complexity of quick sorting is lower than insert sorting.

Big-O characterization:

Insertion sort: f(n) = n^2

Quick sort: f(n)=n log(n)

|  |  |  |  |
| --- | --- | --- | --- |
| Sort Algorithm | Sorting Time | | |
| Worst | Average | Best |
| Insertion Sort | O (n(n-1)/2) | O (n^2) | O (n-1) |
| Quick Sort | O (n^2) | O (nlog(n)) | O (n/2) |

If the goal is to rank the sequences of n elements in ascending order, then there are best and worst cases for insert sorting. In the best case, the sequence is already in ascending order. In this case, the comparison operation that needs to be performed needs to be (n-1) times. The worst case is that the sequence is in descending order, then the comparisons that need to be made at this time are n(n-1)/2 times. The assignment of insert sorts is the number of comparison operations plus (n-1) times. On average, the time complexity of inserting the sorting algorithm is O(n^2). Thus, the insertion ordering is not suitable for sorting applications where the amount of data is relatively large. However, if the amount of data that needs to be sorted is small, for example, the magnitude is less than a thousand, then insert sorting is still a good choice.

For quick sort, the worst case occurs when the two intervals generated by each partitioning process contain n-1 elements and 1 element respectively, so that the worst sorting timing is n^2. The best case of quick sort is when the interval generated by each partitioning process is n/2. Even if we can't randomly select a baseline value, fast sorting still only requires nlog(n) time for all possible permutations of its inputs.

**Task 3:**

1. CPU (central processing unit) is a very large-scale integrated circuit, which is the operation core and control unit of a computer. Its function is mainly to interpret computer instructions and process data in computer software. The performance of a computer is largely determined by the performance of the CPU, and the performance of the CPU is mainly reflected in the speed of its running program. Performance indicators affecting the running speed include CPU operating frequency, Cache capacity, instruction system and logical structure. The running speed and usage of CPU have a great influence on our applications, algorithms and so on. Interconnection structure of internal multiple CPUs without considering the burden of communication and synchronization, can greatly improve performance. If communication and synchronization are bottlenecks, we need to change the problem algorithm and data access and processing methods to solve them. At this time, simply increasing the number of processors will not achieve much effect.
2. Random Access Memory (RAM), also known as main memory, is an internal memory that directly exchanges data with the CPU. Ram has the characteristics of random access, which means that when the data in the memory is read or written, the time required is independent of the location of the information or the location it is written. Modern random-access memory is the fastest speed of writing and reading in almost all access devices. When the input size is very small especially below 1M, the performance gap is not obvious, increasing RAM capacity can significantly improve performance if data is very large.
3. Operating system (OS) is a computer program that manages computer hardware and software resources. It is also the core and cornerstone of computer system. Operating systems need to deal with such basic tasks as managing and configuring memory, determining the order of supply and demand of system resources, controlling input and output devices, operating networks and managing file systems. It is not possible to determine whether the operating system will affect performance because no two computers with the same configuration, but different operating systems are tested. In theory, when the configuration is the same, and the factors such as redundant file log files, registry files and programs are not considered, the different operating systems will not affect the performance.