***STUDENT NAME : Shaima Ibraheem Yahya.***

***STUDENT ID :1090699.***

 ***. INSTRUCTOR : MR .Sobhi Ghassan Ahmad***

***COURSE : COMP 2321 ..***

***DATE : 19 – 4 -2011 .***

***\* RESULT OF MY PROJECT :***

|  |
| --- |
| ***From 0-1000*** |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ***Shell sort (m sec)*** | ***Quick sort (m sec)***  | ***Merge sort (m sec)*** | ***Heap sort (m sec)*** | ***radix sort (m sec)*** | ***Insertion sort (m sec)*** | ***Bubble sort (m sec)*** | ***Data status/sorting type*** |
|  ***0*** |  ***0 0***  | ***0***  | ***0***  | ***0***  | ***0***  | ***0*** | ***sorted*** |
| ***0*** | ***0***  | ***0***  | ***0***  | ***0***  | ***0***  | ***0*** | ***Reversed order*** |
| ***0*** | ***0***  | ***0*** | ***0***  | ***0*** | ***0***  | ***0*** | ***Random***  |

|  |
| --- |
| ***From 0-10000*** |
| ***Shell sort (m sec)*** | ***Quick sort (m sec)***  | ***Merge sort (m sec)*** | ***Heap sort (m sec)*** | ***radix sort (m sec)*** | ***Insertion sort (m sec)*** | ***Bubble sort (m sec)*** | ***Data status/sorting type*** |
| ***0*** |  ***0 0***  | ***0***  | ***0***  | ***0***  | ***0***  | ***0*** | ***sorted*** |
| ***0*** | ***0***  | ***0***  | ***0***  | ***0***  | ***0***  | ***0*** | ***Reversed order*** |
| ***0*** | ***0***  | ***0*** | ***0***  | ***0*** | ***0***  | ***1*** | ***Random***  |

|  |
| --- |
| ***From 0-100000*** |
| ***Shell sort (m sec)*** | ***Quick sort (m sec)***  | ***Merge sort (m sec)*** | ***Heap sort (m sec)*** | ***radix sort (m sec)*** | ***Insertion sort (m sec)*** | ***Bubble sort (m sec)*** | ***Data status/sorting type*** |
| ***0*** |  ***0***  | ***1***  | ***1***  | ***0***  | ***0***  | ***25*** | ***sorted*** |
| ***0*** | ***0***  | ***0***  | ***0***  | ***0***  | ***34***  | ***56*** | ***Reversed order*** |
| ***0*** | ***0*** | ***1*** | ***0***  | ***0*** | ***16***  | ***49*** | ***Random***  |

|  |
| --- |
| ***From 0-500000*** |
| ***Shell sort (m sec)*** | ***Quick sort (m sec)***  | ***Merge sort (m sec)*** | ***Heap sort (m sec)*** | ***radix sort (m sec)*** | ***Insertion sort (m sec)*** | ***Bubble sort (m sec)*** | ***Data status/sorting type*** |
| ***0*** |  ***0***  | ***\_\_***  | ***1***  | ***0***  | ***0***  | ***667*** | ***sorted*** |
| ***0*** | ***0***  | ***\_\_***  | ***0***  | ***0***  | ***886***  | ***1446*** | ***Reversed order*** |
| ***0*** | ***0***  | ***\_\_*** | ***0***  | ***0*** | ***434***  | ***837*** | ***Random***  |

|  |
| --- |
| ***From 0-1000000*** |
| ***Shell sort (m sec)*** | ***Quick sort (m sec)***  | ***Merge sort (m sec)*** | ***Heap sort (m sec)*** | ***radix sort (m sec)*** | ***Insertion sort (m sec)*** | ***Bubble sort (m sec)*** | ***Data status/sorting type*** |
| ***0*** |  ***0***  | ***\_\_***  | ***1***  | ***0***  | ***0***  | ***2648*** | ***sorted*** |
| ***1*** | ***0***  | ***\_\_***  | ***0***  | ***0***  | ***3591***  | ***4532*** | ***Reversed order*** |
| ***0*** | ***0***  | ***\_\_*** | ***1***  | ***0*** | ***1770***  | ***2120*** | ***Random***  |

***Summary :***

From the data in the table, and after you compare the time for all types of sort, we conclude that the best type of sort:

\*if the number of data is **small**(0-10000) and it's **sorted**:

All the sorts type have the same time .

\*\*If the number of data is **small** and it's **reversed order**:

Almost all sorts algorithms take the same time

\*\*\*If the number of data is **small** and it's **random**:

all sorts take the zero time except the bubble sort which take 1 m sec

\*If the data **isn't large** (100000)and it's **sorted**:

Quick sort, shell and radix and insertion were the best and they took zero msec ,, then heap sort, merge sort they took the same time which was 1 msec ,,, and the sort which took the big time was bubble sort

\*\*If the data **isn't large** and it's **reversed order**:

Quick sort , shell sort ,radix sort ,merge sort ,heap sort .. take the same zero time

But there's a noted difference between sorted and reversed order for the insertion sort ,it is take zero msec for sorted and 34 msec for the reversed !!!

The biggest time algorithms was for the bubble sort it took 56 msec !

\*\*\*If the data **isn't large** and it's **random**:

Heap sort, quick sort ,shell sort and radix had the same zero time .. then merge sort took 1 msec then insertion sort(16 msec), and finally bubble sort(49 msec).

\*If the data is **large(500,000)** and it's **sorted**:

Quick sort, shell sort , radix sort and insertion sort took zero msec.. then heap sort which took 1 msec and the worst sort for this type of data was the bubble sort(it's took 667 msec)

Note : there is a problem when we enter large data for merge sort

\*\*If the data is **large** and it's **reversed order**:

 Quick sort, heap sort, radix sort and shell sort had the same zero time .. then insertion sort and finally the bubble sort,.

Note : there is a problem when we enter large data for merge sort

\*\*\*If the data is **large** and it's **random**

we note that this type of data mostly like previous type but the time of bubble here is less than previous .

\*If the data is **huge(1000,000)** and it's **sorted**:

Quick sort, shell sort , radix sort and insertion sort took zero msec.. then heap sort which took 1 msec and the worst sort for this type of data was the bubble sort.

Note : there is a problem when we enter large data for merge sort

\*\*If the data is **huge** and it's **reversed order**:

 Quick sort, heap sort, radix sort had the same zero time .. then the shell sort and then the insertion sort and finally the bubble sort,.

Note : there is a problem when we enter large data for merge sort

\*\*\*If the data is **huge** and it's **random**

we note that this type of data mostly like previous type but the time of bubble here is less than previous .

***SORTS TYPE & PROPERTIES***  :

We used in this project seven type of sort and we can divide them into three group as follow :

 \* sorting algorithms of O(N2) :[insertion sort , bubble sort]

- Easy to program

Simple to understand-

Very slow, especially for large values of **N-**

Almost never used in professional software-

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

\*\* sorting algorithms of (NlogN): [quick sort , heap sort , merge sort ]

 -Fast

Efficient-

Complicated, not easy to understand-

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_-

\* \*\* sorting algorithm of O(N)[radix sort ]: (the best sort because it takes the least time .

***\* CONCLUSIONS AND NOTES :***

**-The merge sort cannot sorted huge data(500000 and 1000000 ), the maximum value that can do was 260000 and the result was 8 msec for sorted and 7 msec for reversed order and 1 msec for random .**

**-- bubble sort and insertion sort both have the worst case time of O(n2) Making them impractical for large array , But they are easy to program, easy to debug.**

**--- insertion sort has good performance when the array is nearly sorted .. the difference is obvious ! .**

**---- its important to note that all algorithms appear to run equally as fast for small value of N.**

**-----For values of N from (1000 – 1000,000) the different between O(N2) and O(NlogN) become dramatically appearnt .**

**---- we noticed that the radix sort and the quick sort had the same time for data that entered but if someone ask me which the best .. I will say that the radix is better because of this**

**For example : sort 1 million 64-bit numbers**

Treat as four-digit radix **216** numbers

Can sort in just **four** passes with radix sort!

Running time: **4**( **1 million + 216** ) ≈ **4 million operations**

**Compare with typical O(*n* log *n*) comparison sort**

Requires approx **l0g *n* = 20** operations per number being sorted

Total running time ≈ **20 million operations**