******

***Computer Science Department***

***Comp2321***

***Data structure***

***Report for Final Project***

***Sorting and time***

Name :

Wala Qasim Al-Ramoni (1090223 )

Instructure Name : Iyad Jaber

Section : 2 (SMW 10:00 – 10:50 )

Date : wednesday 20-4-2011

# Radix sort

start with radix sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | n=10000 | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | m | s | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| reverse | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 5 |
| random | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 5 |

# As we can see from the table above radix sort is so faster techneqe of sort it just use 0 second for small data and about 5 seconds for huge data ,, so it's too faster and preferred for huge data .. we note that the time approximately independent of the data type (sorted,reverse,random).

The time it take is O(n) which is a best time .

# bubble sort

second with bubble sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 2 | 32 | 4 | 46 | 57 |
| reverse | 0 | 0 | 0 | 0 | 0 | 3 | 0 | 1 | 15 | 0 | 5 | 7 | 13 | 19 | 19 |
| random | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 55 | 0 | 3 | 17 | 4 | 14 | 14 |

# As we can see from the table above bubble sort is so slow techneqe of sort it just use 0 second for small data and about 13 hours for huge data when it's reverse ,, so it's too slower and not preferred for huge data .. we note that the time depend of the data type (sorted,reverse,random). For the sorted and random data type it take less time than when it reversed

**bubble sort have the worst case time of O(n2) Making it impractical for large array , But it's easy to program, easy to debug**

# insertion sort

third with insertion sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| reverse | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 1 | 11 | 0 | 5 | 9 | 4 | 53 | 42 |
| random | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 33 | 0 | 2 | 35 | 2 | 35 | 48 |

**As we can see from the table above insertion sort is faster for small values of n it just use 0 second for small data however it take about 4 hours for huge data ,, .. we note that the time depend of the data type (sorted,reverse,random).**

**Since it's too quick for sorted data type ,, moderate for random and slow for revers**

**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 !**

# heap sort

fourth with heap sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 2 |
| reverse | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| random | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 3 |

**As we can see from the table above heap sort is so so faster techneqe of sort it just use 0 second for small data and about 5 seconds for huge data ,, so it's too faster and preferred for huge data ..**

**It takes time of (NlogN):**

**And it's -Fast**

**Efficient-**

**Complicated, not easy to understand**

# merge sort

fifth with merge sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  |  |  |
| reverse | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |  |  |  |
| random | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |  |  |  |

**As we can see from the table above merge sort is so faster techneqe of sort it just use 0 second for small data and about 1 seconds for huge data ,, so it's too faster and preferred for huge data .. we note that the time approximately independent of the data type (sorted,reverse,random).**

**As for the heap sort the merge sort is It takes time of (NlogN):**

**And it's -Fast**

**Efficient-**

**Complicated, not easy to understand**

**-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 .**

# quick sort

sixth with quick sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| reverse | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| random | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

# As we can see from the table above radix sort is so faster techneqe of sort it just use 0 second for small data and about 1 seconds for huge data ,, so it's too faster and preferred for huge data .. we note that the time approximately independent of the data type (sorted,reverse,random). As for the heap and merge

**It takes time of (NlogN):**

**And it's -Fast**

**Efficient-**

**Complicated, not easy to understand**

# shell sort

finally with shell sort

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data type | n=1000 | | | n=10000 | | | n=50000 | | | n=100000 | | | n=1000000 | | |
|  | h | m | s | h | m | s | h | m | s | h | m | s | h | m | s |
| sorted | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| reverse | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 0 | 42 | 0 | 4 | 53 | 4 | 42 | 12 |
| random | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 22 | 0 | 2 | 49 | 2 | 29 | 52 |

**As we can see from the table above shell sort is so slow techneqe of sort for large data however it's too faster when the data is sorted and preferred for huge data .. we note that the time approximately depend of the data type (sorted,reverse,random).**

**Since for huge data it take more time when the data is reversed ,, moderate time when it's random ..**

**It's take time of O(n2).**

**Preferred for sorted huge data and small data only ..**

Conclusion and comparison of sort types

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

Or **reversed order**: Almost all sorts algorithms take the same time

However bubble sort and shell sort and insertion sort take few seconds for the reverse and random ..

\*if the number of data is **sorted whatever was it's size insertion and shell are preferred**.

\*if the number of data is **sorted whatever was it's size insertion and shell are preferred**.

\*for huge data Quick sort is the fastest and preferred then heap sort and merge sort then radix sort these four sorts are the most sppedy.

\*bubble sort is the most slowest sort for huge and moderate data.