

Name: Patel Darshan Atulbhai

Njit id: dap83

Program 1 (Output & Analysis)

Teacher: David Nassimi

1)Where N=32

- **No of Key Comparison:**

	Merge Sort	Quick Sort	Heap Sort
Best Case	80	159	108
Avg. Case	124	142	115
Worst Case	80	162	108

- **Time in Millisecond:**

	Merge Sort	Quick Sort	Heap Sort
Best Case	0.074543	1.154341	0.082646
Avg. Case	0.086967	0.358132	0.111815
Worst Case	0.085887	1.206737	0.091829

Best Case:

```
Console Tasks Display
<terminated> MergeSort1 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 12:56:16 AM)
Before sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

***MERGE SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

# of Key Comparisons Using Merge Sort = 80
Time Taken by Merge Sort = 0.074543 Milliseconds

***Quick SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

# of Key Comparisons Using Quick Sort = 159
Time Taken by Quick Sort = 1.154341 Milliseconds

***Heap SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
|
# of Key Comparisons Using Heap Sort = 108
Time Taken by Quick Sort = 0.082646 Milliseconds
```

Worst Case:

```
Console Tasks Display
<terminated> MergeSort1 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 12:59:14 AM)
Before sorting number
31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

***MERGE SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

# of Key Comparisons Using Merge Sort = 80
Time Taken by Merge Sort = 0.085887 Milliseconds

***Quick SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

# of Key Comparisons Using Quick Sort = 162
Time Taken by Quick Sort = 1.206737 Milliseconds

***Heap SORT***
After sorting number
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

# of Key Comparisons Using Heap Sort = 108
Time Taken by Quick Sort = 0.091829 Milliseconds
```

AverageCase:

```
Console Tasks Display
<terminated> MergeSort1 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 1:01:25 AM)
Before sorting numer
18 28 3 19 30 21 5 4 24 6 19 26 6 20 31 1 16 13 31 6 10 17 31 30 8 27 15 14 5 16 5 19

***MERGE SORT***
After sorting numer
1 3 4 5 5 5 6 6 6 8 10 13 14 15 16 16 17 18 19 19 19 20 21 24 26 27 28 30 30 31 31 31

# of Key Comparisons Using Merge Sort = 124
Time Taken by Merge Sort = 0.086967 Milliseconds

***Quick SORT***
After sorting numer
1 3 4 5 5 5 6 6 6 8 10 13 14 15 16 16 17 18 19 19 19 20 21 24 26 27 28 30 30 31 31 31

# of Key Comparisons Using Quick Sort = 142
Time Taken by Quick Sort = 0.358132 Milliseconds

***Heap SORT***
After sorting numer
1 3 4 5 5 5 6 6 6 8 10 13 14 15 16 16 17 18 19 19 19 20 21 24 26 27 28 30 30 31 31 31

# of Key Comparisons Using Heap Sort = 115
Time Taken by Quick Sort = 0.111815 Milliseconds
```

2) Large Size Array:

- No of key Comparison:

N	Merge Sort	Quick Sort	Heap Sort
1024	8944	9287	8660
32768	450017	516480	440822
1048576	19644941	22047670	19352696

- Time in Millisecond:

N	Merge Sort	Quick Sort	Heap Sort
1024	14.322037	2.005647	1.863583
32768	2463.406531	21.651047	43.89196
1048576	2025794.820645	557.452629	519.865519

N=1024

```
Console Tasks Display
<terminated> Sort2 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 1:12:08 AM)
Before sorting numer
Where N=1024

***MERGE SORT***
After sorting numer

# of Key Comparisons Using Merge Sort = 8944
Time Taken by Merge Sort = 14.322037 Milliseconds

***Quick SORT***
After sorting numer

# of Key Comparisons Using Quick Sort = 9287
Time Taken by Quick Sort = 2.005647 Milliseconds

***Heap SORT***
After sorting numer

# of Key Comparisons Using Heap Sort = 8660
Time Taken by Quick Sort = 1.863583 Milliseconds
```

N=32768

```
Console Tasks Display
<terminated> Sort2 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 1:16:34 AM)
Before sorting number
Where N=32768

***MERGE SORT***
After sorting number

# of Key Comparisons Using Merge Sort = 450017
Time Taken by Merge Sort = 2463.406531 Milliseconds

***Quick SORT***
After sorting number

|
# of Key Comparisons Using Quick Sort = 516480
Time Taken by Quick Sort = 21.651047 Milliseconds

***Heap SORT***
After sorting number

# of Key Comparisons Using Heap Sort = 440822
Time Taken by Quick Sort = 43.89196 Milliseconds
```

N=1048576

```
Console Tasks Display
<terminated> Sort2 [Java Application] C:\Program Files\Java\jre1.8.0_73\bin\javaw.exe (Oct 17, 2017, 1:18:57 AM)
Before sorting number
Where N=1048576

***MERGE SORT***
After sorting number

# of Key Comparisons Using Merge Sort = 19644941
Time Taken by Merge Sort = 2025794.820645 Milliseconds

***Quick SORT***
After sorting number

# of Key Comparisons Using Quick Sort = 22047620
Time Taken by Quick Sort = 557.452629 Milliseconds

***Heap SORT***
After sorting number

# of Key Comparisons Using Heap Sort = 19352696
Time Taken by Quick Sort = 519.865519 Milliseconds
|
```

Analysis: Here we take integer as random number so it is nothing but average case analysis.

Case-1: N = 1024

1) Merge Sort :

According to the Tabulated data, the number of key comparisons = 8944.

And the average case complexity of Merge sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq A \cdot (n \cdot \log n)$$

Here $n=1024$, therefore $n \cdot \log n = 10240$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'A',

So the tabulated data corresponds to the average case analysis of Merge sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 8944 \leq A \cdot (10240)$$

Where $A = 0.873$

2) Quick Sort:

According to the Tabulated data, the number of key comparisons = 9287.

And the average case complexity of Quick sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq B \cdot (n \cdot \log n)$$

Here $n=1024$, therefore $n \cdot \log n = 10240$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'B',

So the tabulated data corresponds to the average case analysis of Quick sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 9287 \leq B \cdot (10240)$$

Where $B = 0.907$

3) Heap Sort:

According to the Tabulated data, the number of key comparisons = 8660.

And the average case complexity of Heap sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq C \cdot (n \cdot \log n)$$

Here $n=1024$, therefore $n \cdot \log n = 10240$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'C',

So the tabulated data corresponds to the average case analysis of Heap sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 8660 \leq C \cdot (10240)$$

Where $C = 0.845$

Case-2: N = 32768

1) Merge Sort:

According to the Tabulated data, the number of key comparisons = 450017.

And the average case complexity of Merge sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq A \cdot (n \cdot \log n)$$

Here $n=32768$, therefore $n \cdot \log n = 491520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'A',

So the tabulated data corresponds to the average case analysis of Merge sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 450017 \leq A \cdot (491520)$$

Where $A = 0.915$

2) Quick Sort:

According to the Tabulated data, the number of key comparisons = 516480.

And the average case complexity of Quick sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq B \cdot (n \cdot \log n)$$

Here $n=32768$, therefore $n*\log n = 491520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'B',

So the tabulated data corresponds to the average case analysis of Quick sort i.e.

$O(n*\log n)$. Hence,

$$T(n)=516480 \leq B*(491520)$$

Where $B = 1.050$

3) Heap Sort:

According to the Tabulated data, the number of key comparisons = 440822.

And the average case complexity of Heap sort is given by,

$$T(n) \leq O(n*\log n) \leq C*(n*\log n)$$

Here $n=32768$, therefore $n*\log n = 491520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'C',

So the tabulated data corresponds to the average case analysis of Heap sort i.e.

$O(n*\log n)$. Hence,

$$T(n)= 440822 \leq C*(491520)$$

Where $C = 0.896$

Case-3: N = 1048576

1) Merge Sort:

According to the Tabulated data,

The number of key comparisons = 19644941.

And the average case complexity of Merge sort is given by,

$$T(n) \leq O(n*\log n) \leq A*(n*\log n)$$

Here $n=1048576$, therefore $n*\log n = 20971520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'A',
So the tabulated data corresponds to the average case analysis of Merge sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 19644941 \leq A \cdot (20971520)$$

Where $A = 0.937$

2) Quick Sort:

According to the Tabulated data,

The number of key comparisons = 22047670.

And the average case complexity of Quick sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq B \cdot (n \cdot \log n)$$

Here $n=1048576$, therefore $n \cdot \log n = 20971520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'B',

So the tabulated data corresponds to the average case analysis of Quick sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 22047670 \leq B \cdot (20971520)$$

Where $B = 1.0513$

3) Heap Sort:

According to the Tabulated data,

The number of key comparisons = 19352696.

And the average case complexity of Heap sort is given by,

$$T(n) \leq O(n \cdot \log n) \leq C \cdot (n \cdot \log n)$$

Here $n=1048576$, therefore $n \cdot \log n = 20971520$.

As the Number of key comparisons are closer to $T(n)$ by some constant factor 'C',
So the tabulated data corresponds to the average case analysis of Heap sort i.e.

$O(n \cdot \log n)$. Hence,

$$T(n) = 19352696 \leq C \cdot (20971520)$$

Where $C = 0.923$