

ASSIGNMENT 1

02/02/23

NAME : SHRESTH SONKAR
REGNO : 20214272
GROUP : CS4D
TOPIC : ANALYSIS OF
ALGORITHM LAB
CODE : CS-14202

```
// Analysis of Merge Sort over 100K entries
```

```
#include <stdio.h>
```

```
#include <time.h>
```

```
clock_t begin;
```

```
clock_t end;
```

```
void swap(int *x, int *y) {
```

```
    int temp = *x;
```

```
    *x = *y;
```

```
    *y = temp;
```

```
}
```

```
void merge(int A[],int l,int mid,int r)
```

```
{
```

```
    int i=l,j=mid+1,k=1;
```

```
    int B[1000000];
```

```
    while(i<=mid && j<=r)
```

```
    {
```

```
        if(A[i]<A[j])
```

```
            B[k++]=A[i++];
```

```
        else
```

```
            B[k++]=A[j++];
```

```
    }
```

```
    for(;i<=mid;i++)
```

```
        B[k++]=A[i];
```

```
    for(;j<=r;j++)
```

```
        B[k++]=A[j];
```

```
    for(i=l;i<=r;i++)
```

```
        A[i]=B[i];
```

```
}
```

```
void mSort(int A[],int l,int r)
```

```
{
```

```
    int mid;
```

```
    if(l<r)
```

```
    {
```

```
        mid=(l+r)/2;
```

```
        mSort(A,l,mid);
```

```
        mSort(A,mid+1,r);
```

```
        merge(A,l,mid,r);
```

```
    }
```

```
}
```

```

void writeTable(int size, double time, char *filename)
{
    int i;
    FILE *fp = fopen(filename, "a+");

    if (fp == NULL) printf("FILE CANNOT BE OPENED\n");
    else {
        fprintf(fp, "%d %lf", size, time);
        fprintf(fp, "\n");
    }
    fclose(fp);
}

void readData(int arr[], char *filename) {
    FILE *fp = fopen(filename, "r+");
    char x[16];
    int i, k = 0;

    if (fp == NULL) printf("FILE CANNOT BE OPENED\n");
    else {
        while (fgets(x, 16, fp) != NULL) {
            int num = 0;
            fscanf(fp, "%d", &num);
            if (num == 0) break;
            arr[k++] = num;
        }
    }
    fclose(fp);
}

int main(int argc, char **argv) {
    int arr[100000];
    int size[] = {1000, 5000, 8000, 10000, 20000,
30000, 40000, 50000, 65000, 80000, 90000, 100000};
    int i = 0;
    for (i = 0; i <= 11; i++) {
        readData(arr, argv[1]);
        begin = clock();
        mSort(arr, 0, size[i]);
        end = clock();
        writeTable(size[i], (end - begin) / 100,
argv[2]);
    }
    return 0;
}

```

```
// Analysis of Heap Sort over 100K entries
```

```
#include <stdio.h>
#include <time.h>
```

```
clock_t begin;
clock_t end;
```

```
void insertMaxHeap(int arr[], int n) {
    int t, i = n;
    t = arr[n];

    while (i > 1 && t > arr[i / 2]) {
        arr[i] = arr[i / 2];
        i /= 2;
    }
    arr[i] = t;
}
```

```
int removeMaxHeap(int arr[], int n) {
    int i, j, x, t, val;
    val = arr[1];
    x = arr[n];
    arr[1] = arr[n];
    arr[n] = val;
    i = 1;
    j = i * 2;

    while (j <= n - 1) {
        if (j < n - 1 && arr[j + 1] > arr[j])
            j++;
        if (arr[i] < arr[j]) {
            t = arr[i];
            arr[i] = arr[j];
            arr[j] = t;
            i = j;
            j *= 2;
        } else break;
    }
    return val;
}
```

```
void createMaxHeap(int arr[], int n) {
    int i;
    for (i = 2; i < n; i++)
```

```

        insertMaxHeap(arr, i);
    }

void hSort(int arr[], int n) {
    int i;
    for (i = 2; i < n; i++)
        insertMaxHeap(arr, i);
    for (i = n - 1; i > 1; i--)
        removeMaxHeap(arr, i);
}

void writeTable(int size, double time, char *filename)
{
    int i;
    FILE *fp = fopen(filename, "a+");

    if (fp == NULL) printf("FILE CANNOT BE OPENED\n");
    else {
        fprintf(fp, "%d %lf", size, time);
        fprintf(fp, "\n");
    }
    fclose(fp);
}

void readData(int arr[], char *filename) {
    FILE *fp = fopen(filename, "r+");
    char x[16];
    int i, k = 0;

    if (fp == NULL) printf("FILE CANNOT BE OPENED\n");
    else {
        while (fgets(x, 16, fp) != NULL) {
            int num = 0;
            fscanf(fp, "%d", &num);
            if (num == 0) break;
            arr[k++] = num;
        }
    }
    fclose(fp);
}

int main(int argc, char **argv) {
    int arr[100000];
    int size[] = {1000, 5000, 8000, 10000, 20000,
30000, 40000, 50000, 65000, 80000, 90000, 100000};

```

```

int i;
for (i = 0; i <= 11; i++) {
    readData(arr, argv[1]);
    begin = clock();
    hSort(arr, size[i] - 1);
    end = clock();
    writeTable(size[i], (end - begin) / 100,
argv[2]);
}
return 0;
}

```



A terminal window with a dark blue background and white text. The title bar shows three colored circles (red, yellow, green) and the text ".../sem4/algo/02-02-23". The terminal displays a series of commands and their outputs, showing the compilation of hSort.c and mSort.c using clang, followed by running the executables (hs and ms) with various input files (avg.txt, best.txt, wrst.txt) and output files (hTableAVG.txt, hTableBST.txt, hTableWST.txt, mTableAVG.txt, mTableBST.txt, mTableWST.txt).

```

~/sem4/algo/02-02-23
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ clang hSort.c -o hs
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./hs avg.txt hTableAVG.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./hs best.txt hTableBST.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./hs wrst.txt hTableWST.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ clang mSort.c -o ms
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./ms avg.txt mTableAVG.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./ms best.txt mTableBST.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $ ./ms wrst.txt mTableWST.txt
→ ~/desktop/cse/assgn/sem4/algo/02-02-23 $

```

```
#pltAVG.p
```

```
set autoscale          # scale axes automatically
unset log              # remove any log-scaling
unset label            # remove any previous labels

set xtic auto          # set xtics automatically
set ytic auto          # set ytics automatically
set tics font "Helvetica,10"

set title "MergeSort vs HeapSort AVG CASE"
set xlabel "Array Size"
set ylabel "Time (ms)"
#set key 0.01,100
#set label "Yield Point" at 0.003,260
#set arrow from 0.0028,250 to 0.003,280
set xr [0:125000]
set yr [0:200]
plot "hTableAVG.txt" using 1:2 title 'HeapSort' with linespoints, \
     "mTableAVG.txt" using 1:2 title 'MergeSort' with linespoints
```

```
#pltBST.p
```

```
set autoscale          # scale axes automatically
unset log              # remove any log-scaling
unset label            # remove any previous labels

set xtic auto          # set xtics automatically
set ytic auto          # set ytics automatically
set tics font "Helvetica,10"

set title "MergeSort vs HeapSort BEST CASE"
set xlabel "Array Size"
set ylabel "Time (ms)"
#set key 0.01,100
#set label "Yield Point" at 0.003,260
#set arrow from 0.0028,250 to 0.003,280
set xr [0:125000]
set yr [0:200]
plot "hTableBST.txt" using 1:2 title 'HeapSort' with linespoints, \
     "mTableBST.txt" using 1:2 title 'MergeSort' with linespoints
```

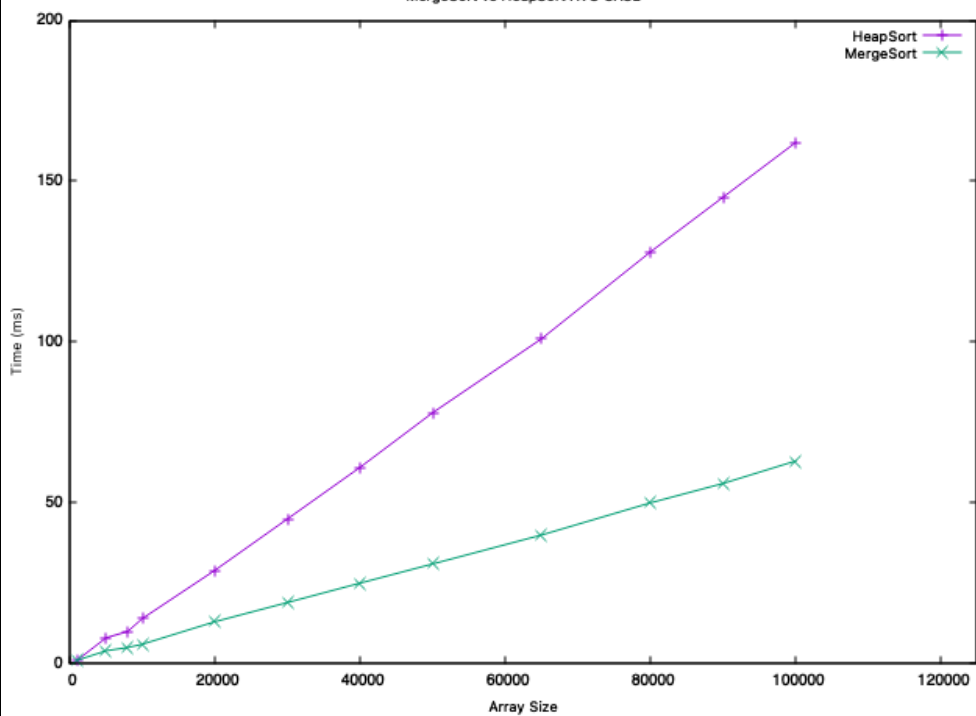
```
#pltWST.p
```

```
set autoscale          # scale axes automatically
unset log              # remove any log-scaling
unset label            # remove any previous labels

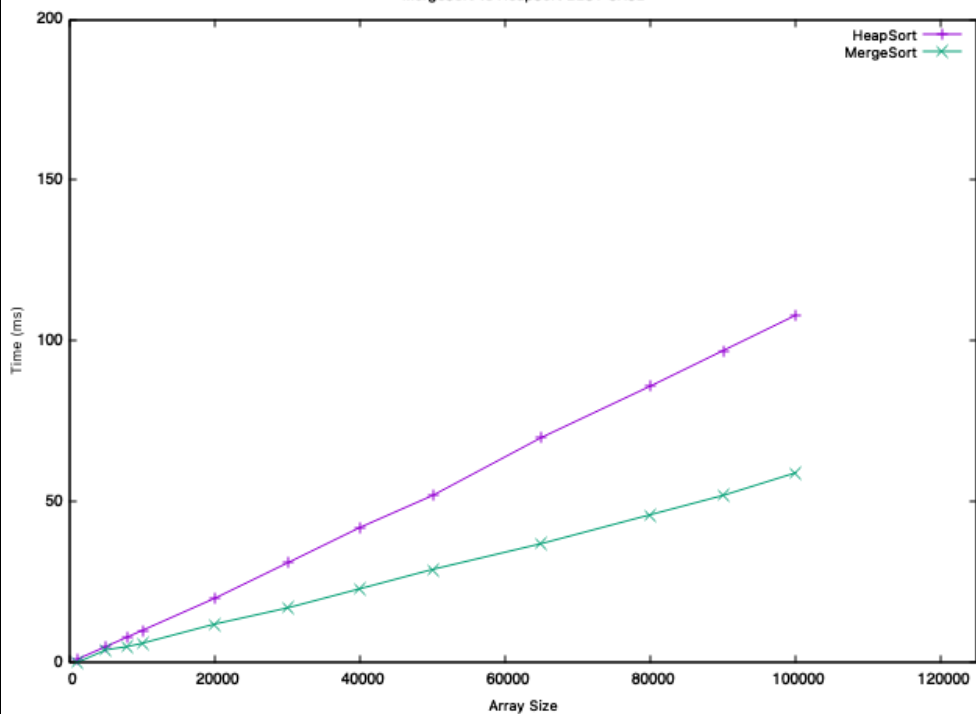
set xtic auto          # set xtics automatically
set ytic auto          # set ytics automatically
set tics font "Helvetica,10"

set title "MergeSort vs HeapSort BEST CASE"
set xlabel "Array Size"
set ylabel "Time (ms)"
#set key 0.01,100
#set label "Yield Point" at 0.003,260
#set arrow from 0.0028,250 to 0.003,280
set xr [0:125000]
set yr [0:200]
plot "hTableBST.txt" using 1:2 title 'HeapSort' with linespoints, \
     "mTableBST.txt" using 1:2 title 'MergeSort' with linespoints
```

MergeSort vs HeapSort AVG CASE



MergeSort vs HeapSort BEST CASE



MergeSort vs HeapSort WORST CASE

