U19CS076 DAA ASSIGNMENT-1

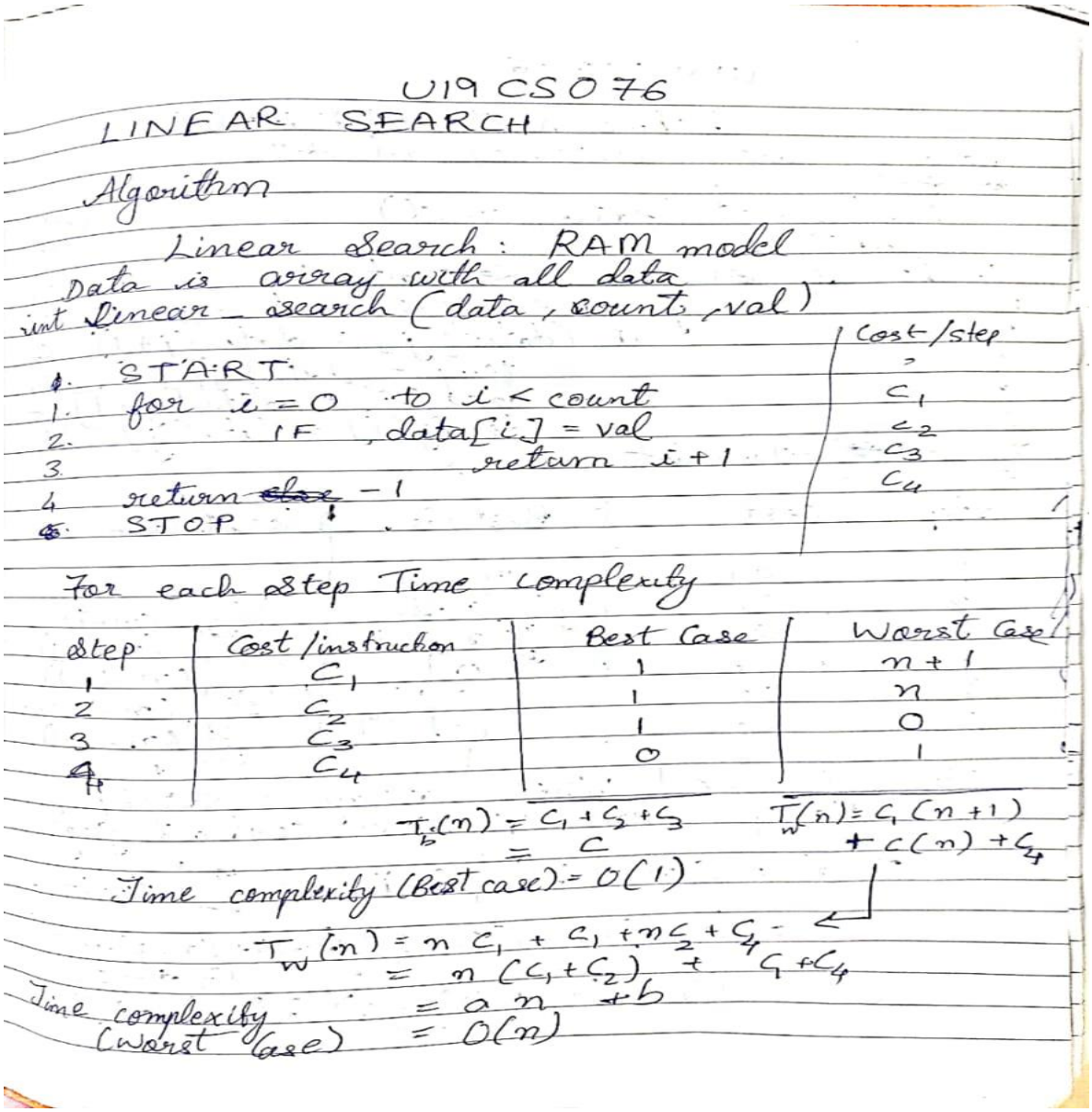
NAME:KRITHIKHA BALAMURUGAN

1.Given the following algorithms, answer the questions.

- Linear Search
- Bubble Sort
- Selection Sort

1.1.(T) Analyze the time complexity of above algorithms using the RAM model

LINEAR SEARCH



BUBBLE SORT

U19CS076

Bubble Sort - RAM Model

Data is array containing all data

```
bubble_sort (data, count)                                   Cost/step
START
1. for i = 1 to i <= count - 1                              C1
2.     for j = 1 to j < count - i - 1                       C2
3.         IF (data[j] > data[j+1])                         C3
4.             temp = data[j]                               C4
5.             data[j] = data[j+1]                          C5
6.             data[j+1] = data[j]                          C6
7.     return data
8. STOP
```

Step	Cost	Best Case	Worst Case
1	C_1	n	$\sum_{i=1}^{n} (n-i)$
2	C_2	$\sum_{j=1}^{n-1} (n-i)$	$\sum_{i=1}^{n-1} (n-i-1)$
3	C_3	$\sum_{i=1}^{n-1} n-i-1$	$\sum_{i=1}^{n-1} (n-i+1)$
4	C_4	0	$\sum_{i=1}^{n-1} (n-i-1)$
5	C_5	0 (No swaps)	$\sum_{i=1}^{n-1} (n-i-1)$
6	C_6	0	
7	C_7	1	

For best case [When array is sorted]
Using non-optimised (without flag)

$$T(n) = C_1 n + C_2 \sum_{i=1}^{n-1} (n-i) + C_3 \sum_{i=1}^{n-1} (n-i-1) + C_7$$

$$= (C_1 - C_3) n + (C_2 + C_3)[(n-1) + (n-2) \cdots 2+1] - C_3$$

$$= (C_1 - C_3) n + \frac{(C_2 + C_3)(n^2 - n)}{2} + C_7 - C_3$$

$$= \left[\frac{C_2 + C_3}{2}\right] n^2 + \left[C_1 - 2C_3 - C_2\right] n + (C_7 - C_3)$$

Best Case

$T(n) = a n^2 + b n + C = O(n^2)$

[Non Modified]

for worst case, array is reversed

$$T(n) = C_1 n + C_2 \sum_{i=1}^{n-1} (n-i) + C_3 \sum_{i=1}^{n-1} (n-i-1)$$

$$+ C_4 \sum_{i=1}^{n-1} (n-i-1) + C_5 \sum_{i=1}^{n-1} (n-i-1)$$

$$+ C_6 \sum_{i=1}^{n-1} (n-i-1) + C_4$$

$$= C_1 n + \frac{C_2 (n)(n-1)}{2} + \left[\frac{C_3 + C_4 + C_5 + C_6}{2}\right] n^2 + 2 - 3n$$

$$= \left[\frac{C_2 + C_3 + C_4 + C_5 + C_6}{2}\right] n^2 + C_1 - \frac{C_2}{2} - \frac{3}{2}\left[C_3 + C_4 + C_5 + C_6\right]$$

$$+ C_2 + C_3 + C_4 + C_5 + C_8$$

Worst Case $= a n^2 + b n + C$

$T(n) = O(n^2)$

SELECTION SORT

U19CS076

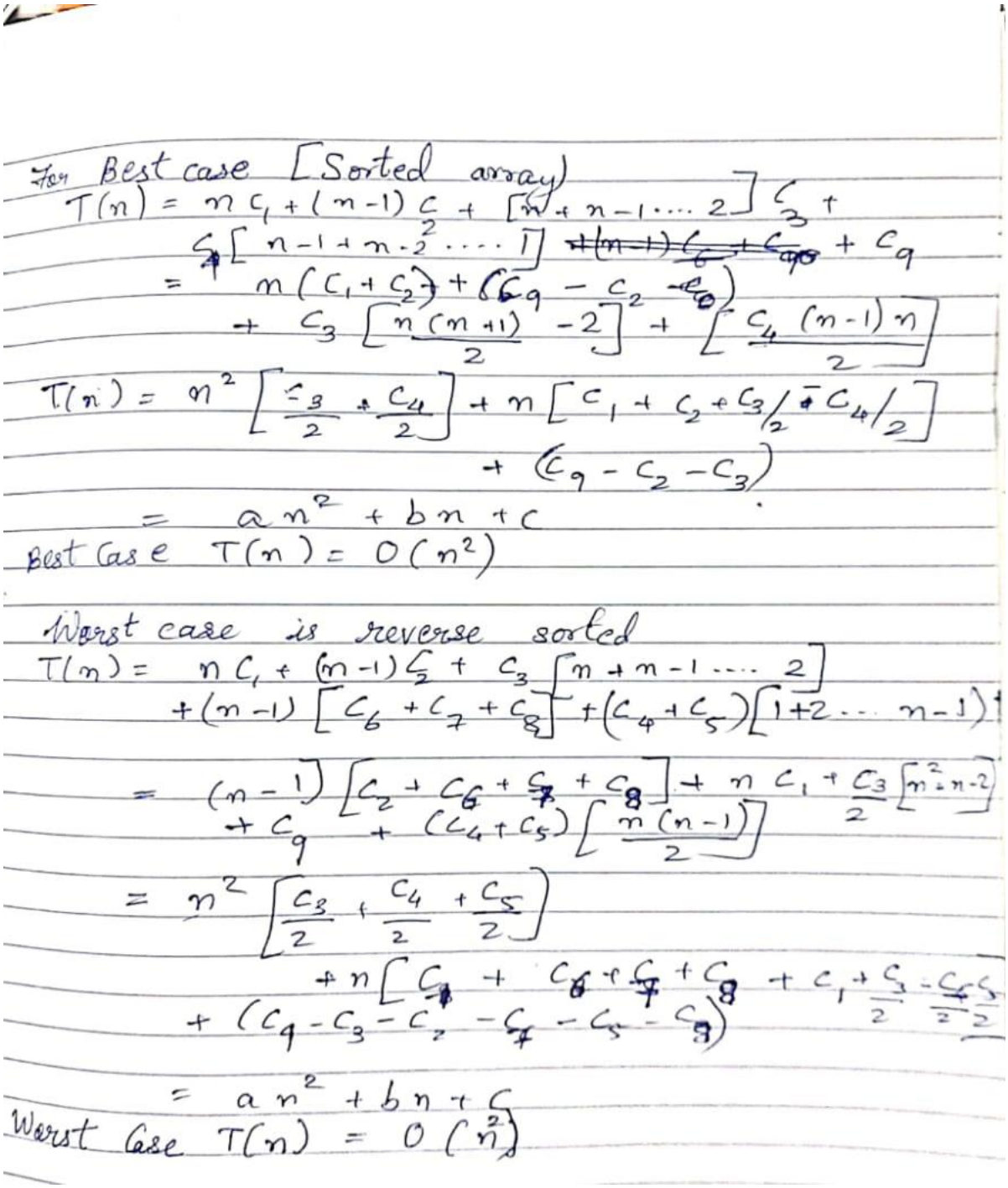
Selection Sort – RAM Model

Data [] array takes in all data of file

selection_sort (data, count)

```
1   START                                              Cost/Step
1.  for i = 1 to i <= count - 1                        C1
2.      set min = i                                    C2
3.      for j = i + 1 to j <= count                    C3
4          IF data[j] < data[min]                      C4
5              min = j                                 C5
6       temp = data[min]                               C6
7       data[min] = data[i]                            C7
8       data[i] = temp                                 C8
9.  return data                                        C9
    STOP
```

Step	Cost	Best Case	Worst Case
1	C_1	n	n
2	C_2	$(n-1)$	$n-1$
3	C_3	$\sum_{i=1}^{n-1}(n-i+1)$	$\sum_{i=1}^{n-1}(n-i+1)$
4	C_4	$\sum_{i=1}^{n-1} n-i$	$\sum_{i=1}^{n-1}(n-i)$
5	C_5	0	$\sum_{i=1}^{n-1}(n-i)$
6	C_6	0	$n-1$
7	C_7	0	$n-1$
8	C_8	0	$n-1$
9	C_9	1	1



1.2.(L) Implement the above algorithms using the programming language of your choice.

Implementation done .Code given ahead.

1.3. (L) Provide the details of Hardware/Software you used to implement algorithms and to measure the time.

Compiler	Dev C++ 5.11
OS Name	Microsoft Windows 10 Home (i5 8th Gen)
Version	10.0.19042 Build 19042
System Name	DESKTOP-BLE6CMQ
System Model	HP Pavilion x360 Convertible 14-ba1xx
System Type	x64-based PC
Processor	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, 1800 Mhz, 4 Core(s), 8 Logical Processor(s)
BIOS Version/Date	Insyde F.54, 04-12-2019
Installed Physical Memory (RAM)	8.00 GB
Total Physical Memory	7.88 GB
Available Physical Memory	1.75 GB
Total Virtual Memory	12.4 GB
Available Virtual Memory	4.59 GB
Page File Space	4.50 GB

1.4. (L) Submit the code (complete programs).

LINEAR SEARCH

```
#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#include<time.h>

clock_t begin, end;

double time_;


int linear_search(long long data[],long long count,int val)

{

        int i=0;
```

```
	for (i = 0; i<count; i++)
	{
		if(data[i]==val)
		{
			return i+1;
		}
	}
	return -1;	//if element not found
}

long long count(char file[])
{
	FILE *fp = fopen(file, "r");
	long long count = 0;
	char b[100];
		while(fscanf(fp, "%s\n", &b) == 1)
				count++;
	fclose(fp);
	return count;
}
int main()
{
	long long ele;
	long long n;
```

```
long long j;
long long pos;
long long *data;                    //array to hold data
int i;
char filename[15];
FILE *fp;
printf("********************TIME SUMMARY***************\n");
for(i=0;i<10;i++)
{
        sprintf(filename, "File %d.txt", i+1);
        n = count(filename);
        printf("-------------------------File %d.txt----------------------\n",i+1);
        printf("File %d has %lld elements\n",i+1,n);

        fp = fopen(filename, "r");
        data=(long long*)malloc(n*((long long)sizeof(long long)));
        fscanf(fp, "%lld", &ele);                       //first element taken for best case
        fseek(fp,0,SEEK_SET);                           //returns pointer to start
        for(j=0; j<n; j++)
        {
                fscanf(fp, "%lld", &data[j]);
        }
        begin= clock();
        pos=linear_search(data,n,ele);
```

```
        end = clock();
        fclose(fp);
        time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
        printf("Best case : %lf\n", time_);

        begin = clock();
        pos=linear_search(data,n,-1);
        end = clock();
        fclose(fp);
        time_= ((double)(end-begin)) / CLOCKS_PER_SEC;
        printf("Worst case  : %lf\n",time_);
        free(data);
    }
```

BUBBLE SORT

```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
clock_t begin, end;
double time_;
void bubble_sort(long long data[], long long count)
{
   int i, j;
   long long temp;
```

```
  // int f;                  //using flags to optimize code
   for (i = 0; i < count-1; i++)
          {
          //f=0;
          for (j = 0; j < count-i-1; j++)
          {
                    if (data[j] > data[j+1])
           {
          temp=data[j];
          data[j]=data[j+1];
          data[j+1]=data[j];
    //            f=1;
                              }
                    }
          //        if(f==0)
          //        break;
          }

}
void bubble_sort_dec(long long data[], int count)
{
   int i, j;
   long long temp;
   for (i = 0; i < count-1; i++)
```

```
        {
   for (j = 0; j < count-i-1; j++)
        {
                if (data[j] < data[j+1])
         {
        temp=data[j];
        data[j]=data[j+1];
        data[j+1]=data[j];
                        }
                }
        }
}
long long count(char file[])
{
        FILE *fp = fopen(file, "r");
        long long count = 0;
        char b[100];
                while(fscanf(fp, "%s\n", &b) == 1)
                                count++;
        fclose(fp);
        return count;
}
int main()
{
```

```
long long n;
long long j;
long long *data;                    //array to hold data
int i;
char filename[15];
FILE *fp;
printf("********************TIME SUMMARY***************\n");
for(i=0;i<10;i++)
{
        sprintf(filename, "File %d.txt", i+1);
        n = count(filename);
        printf("-------------------------File %d.txt----------------------\n",i+1);
        printf("File %d has %lld elements\n",i+1,n);
        fp = fopen(filename, "r");
        data=(long long*)malloc(n*((long long)sizeof(long long)));
        for(j=0; j<n; j++)
        {
                fscanf(fp, "%lld", &data[j]);
        }
        begin= clock();
        bubble_sort(data,n);
        end = clock();
        fclose(fp);
        time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
```

```
            printf("AVERAGE case : %0.10lf\n", time_);


            sprintf(filename, "File %d_asc.txt", i+1);
            fp = fopen(filename, "r");
            for(j=0; j<n; j++)
            {
                  fscanf(fp, "%lld", &data[j]);
            }
            begin = clock();
            bubble_sort(data,n);
            end = clock();
            fclose(fp);
            time_= ((double)(end-begin)) / CLOCKS_PER_SEC;
            printf("Best case  : %0.10lf\n",time_);


            begin = clock();
            bubble_sort_dec(data,n);
            end = clock();
            fclose(fp);
            time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
            printf("Worst case  %0.10lf\n\n", time_);
            free(data);
      }
}
```

SELECTION SORT

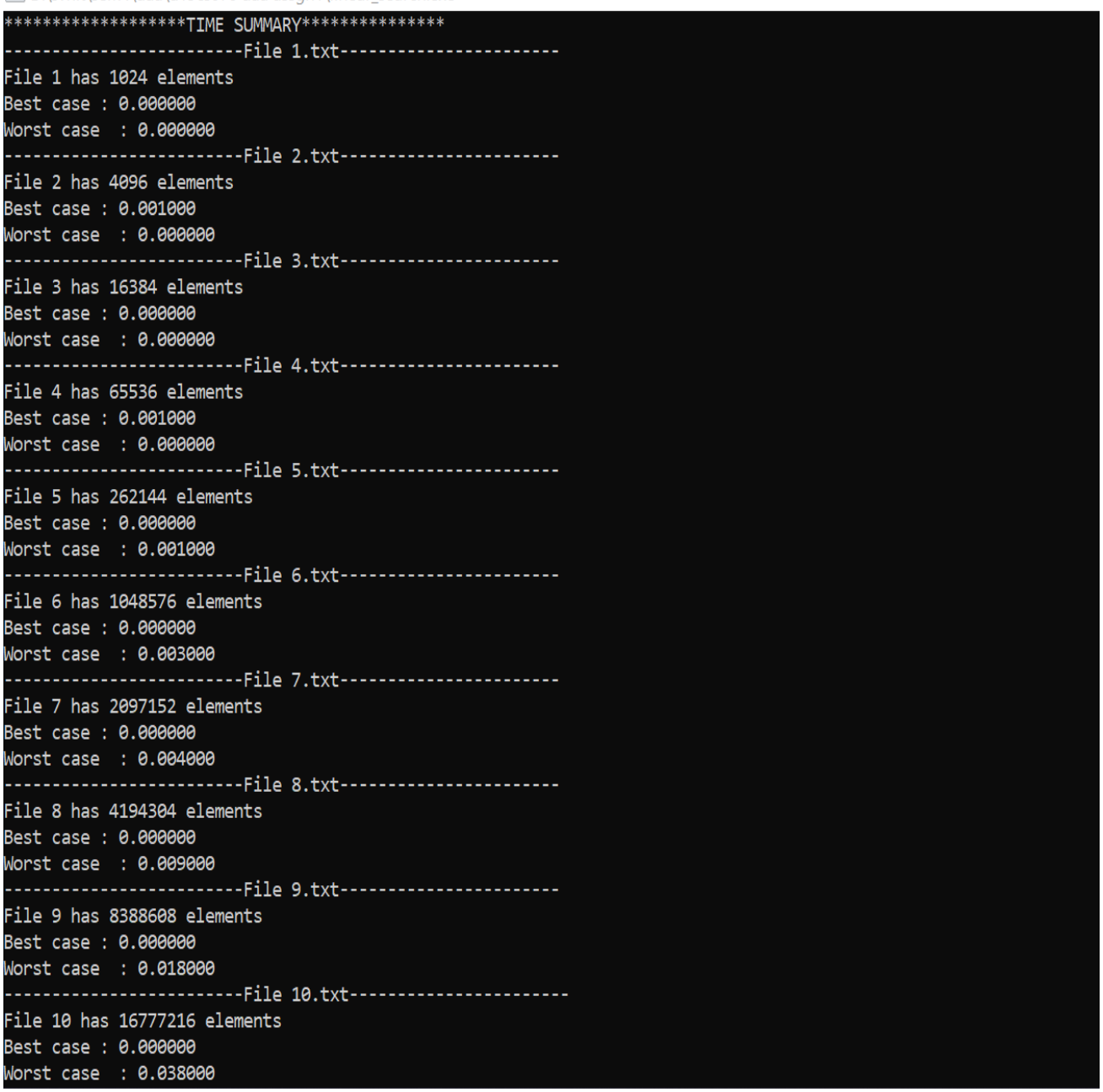
```
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
#include<time.h>
clock_t begin, end;
double time_;


void selection_sort(long long data[], long long count)
{
   int i, j, min;
  long long temp;
   for (i = 0; i < count-1; i++)
   {
      min = i;  //index of minimum element
      for (j = i+1; j < count; j++)
      {
                        if (data[j] < data[min])
                        {
                                min = j;
                    }
                }
           temp=data[min];
           data[min]=data[i];
           data[i]=temp;
   }
}


void selection_sort_dec(long long data[], int count)
{
   int i, j, min;
  long long temp;
   for (i = 0; i < count-1; i++)
```

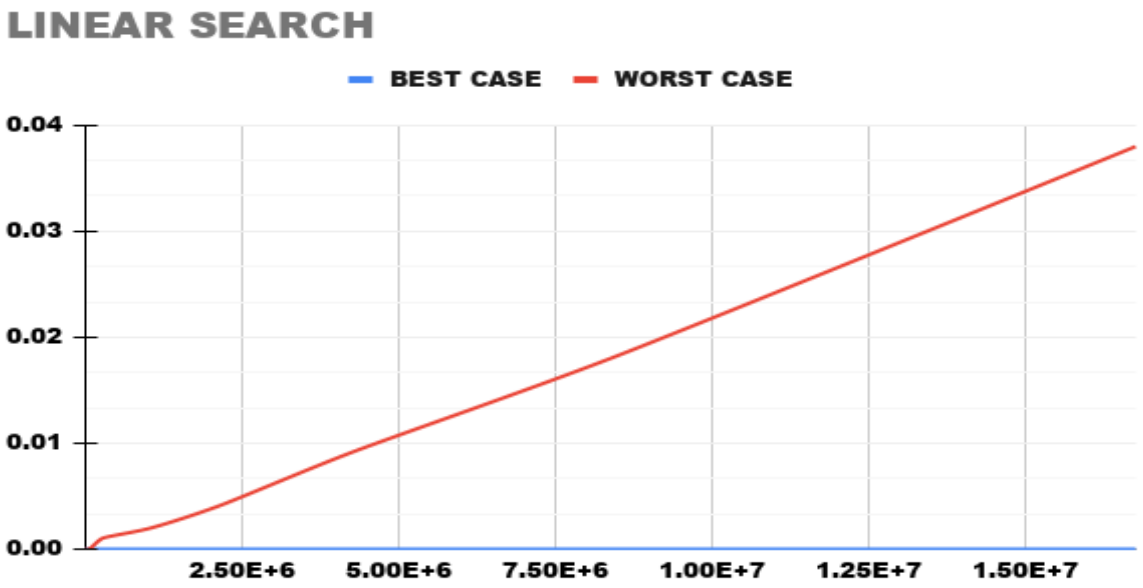
```
   {
      min = i;  //index of minimum element
      for (j = i+1; j < count; j++)
      {
               if (data[j] > data[min])
                {
                                 min = j;
                 }
      }
          temp=data[min];
          data[min]=data[i];
          data[i]=temp;
   }
}




long long count(char file[])
{
     FILE *fp = fopen(file, "r");
     long long count = 0;
     char b[100];
               while(fscanf(fp, "%s\n", &b) == 1)
                                 count++;
     fclose(fp);
     return count;
}


int main()
{
     long long n;
     long long j;
     long long *data;                    //array to hold data
     int i;
     char filename[15];
     FILE *fp;
```

```
printf("********************TIME SUMMARY***************\n");
for(i=0;i<10;i++)
{
	sprintf(filename, "File %d.txt", i+1);
	n = count(filename);
	printf("------------------------File %d.txt----------------------\n",i+1);
	printf("File %d has %lld elements\n",i+1,n);

	fp = fopen(filename, "r");
	data=(long long*)malloc(n*((long long)sizeof(long long)));
	for(j=0; j<n; j++)
	{
		fscanf(fp, "%lld", &data[j]);
	}
	begin= clock();
	selection_sort(data,n);
	end = clock();
	fclose(fp);
	time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
	printf("AVERAGE case : %0.10lf\n", time_);

	sprintf(filename, "File %d_asc.txt", i+1);
	fp = fopen(filename, "r");
	for(j=0; j<n; j++)
	{
		fscanf(fp, "%lld", &data[j]);
	}
	begin = clock();
	selection_sort(data,n);
	end = clock();
	fclose(fp);
	time_= ((double)(end-begin)) / CLOCKS_PER_SEC;
	printf("Best case  : %0.10lf\n",time_);


	begin = clock();
	selection_sort_dec(data,n);
	end = clock();
```

```
            fclose(fp);
            time_ = ((double)(end-begin)) / CLOCKS_PER_SEC;
            printf("Worst case  %0.10lf\n\n", time_);

            free(data);
        }
    }
```

1.5.(L) Measure the best-case time and worst-case time of linear search for all ten files. Plot a graph.





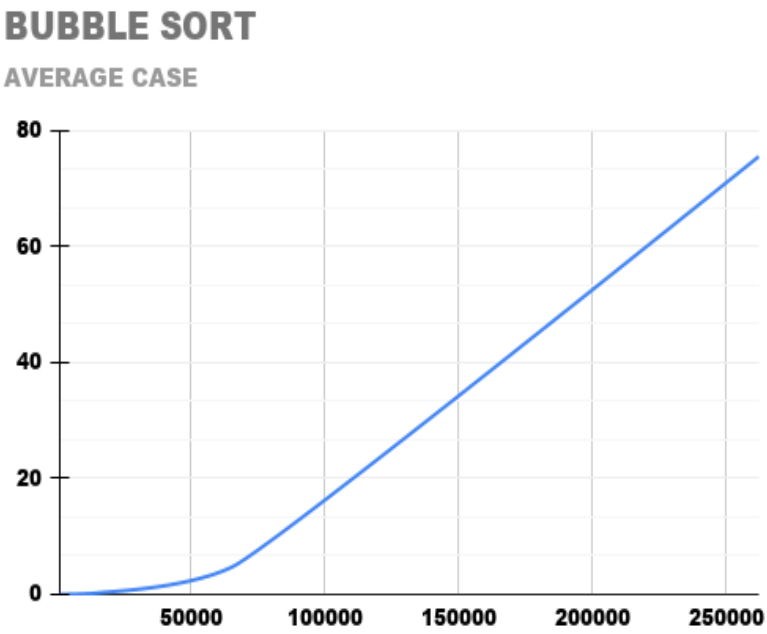
1.6. (L) Measure the average-case time (considering current data of ten files) of bubble sort, and selection sort for all ten files. Plot a graph.

BUBBLE SORT

FILE	N	AVG CASE	BEST CASE	WORST CASE
1	1024	0.002	0.001	0.003
2	4096	0.023	0.02	0.024
3	16384	0.304	0.3	0.497
4	65536	4.766	4.753	4.784
5	262144	75.2	74.4	76.23

SELECTION SORT

FILE	N	AVG CASE	BEST CASE	WORST CASE
1	1024	0	0	0.002
2	4096	0.022	0.020	0.027
3	16384	0.309	0.309	0.323
4	65536	4.955	4.949	4.955
5	262144	79.365	79.263	145.276
6	1048576	1533	1200.5	1711.9

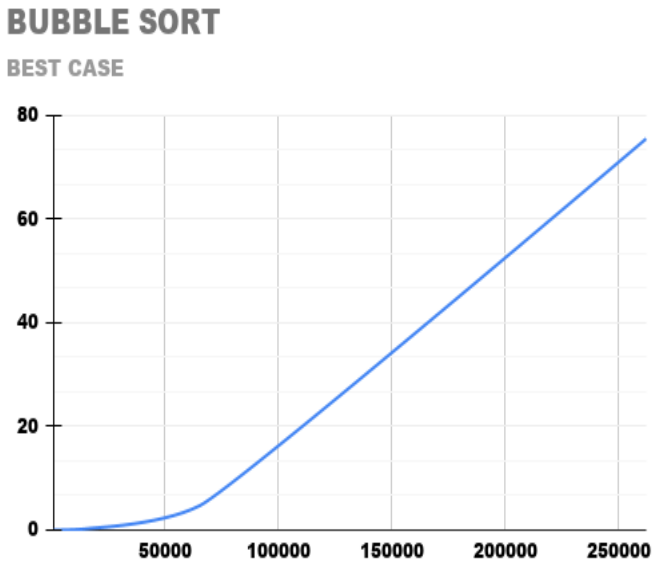


SELECTION SORT

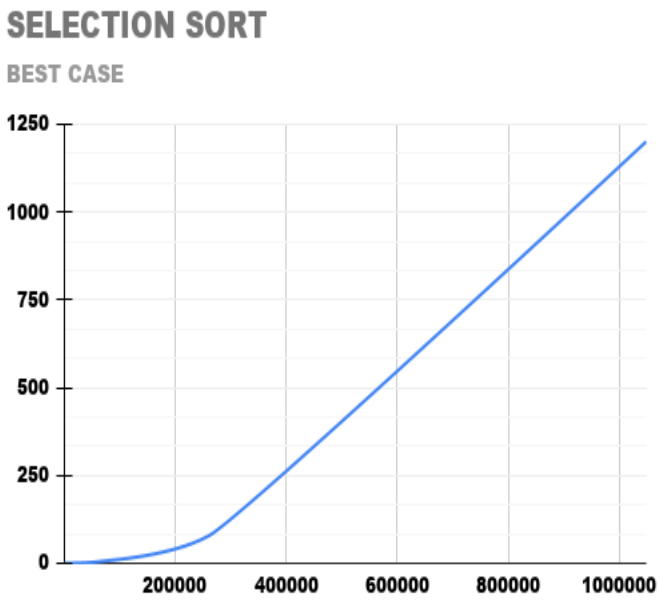
1.7. (L) Measure the best-case time of bubble sort, and selection sort for all ten files. Plot a graph.

BUBBLE SORT

This bubble sort is coded without optimization (without using flags for pre-sorted input)

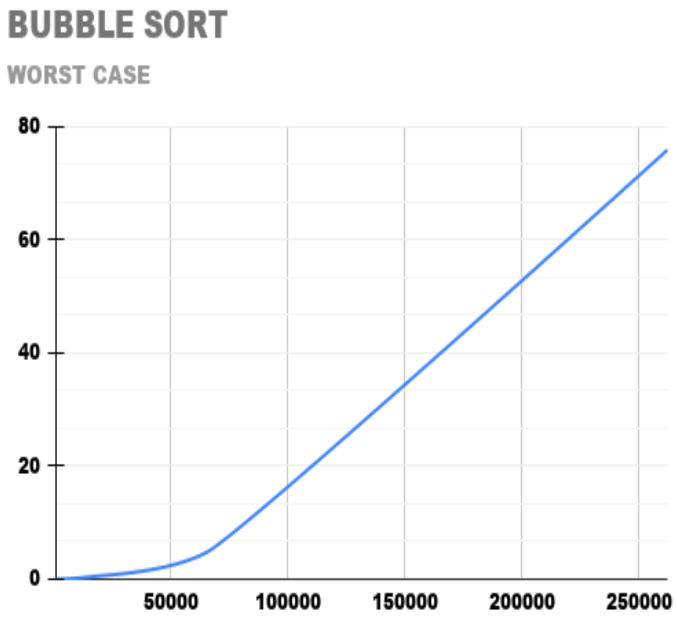


SELECTION SORT

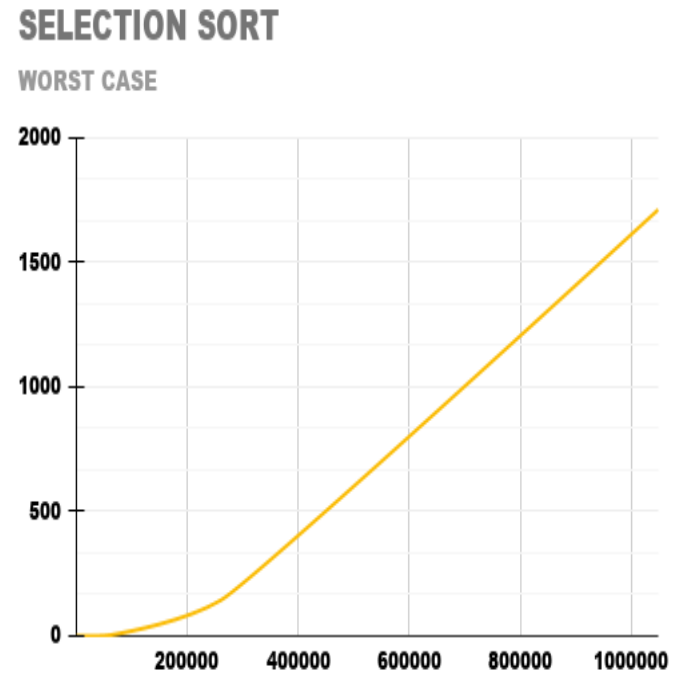


1.8. (L) Measure the worst-case time of bubble sort, and selection sort for all ten files. Plot a graph.

BUBBLE SORT

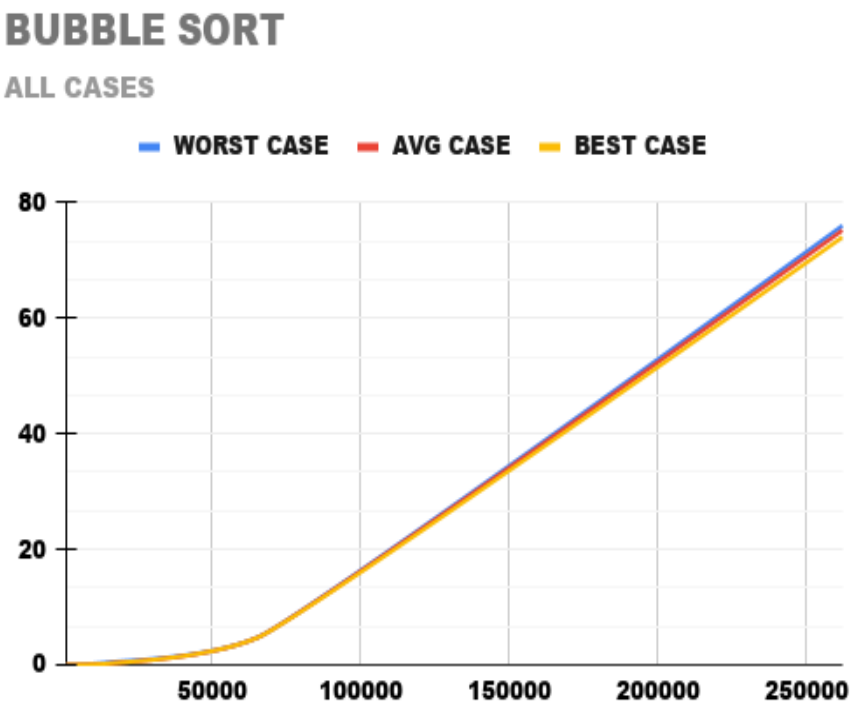


SELECTION SORT

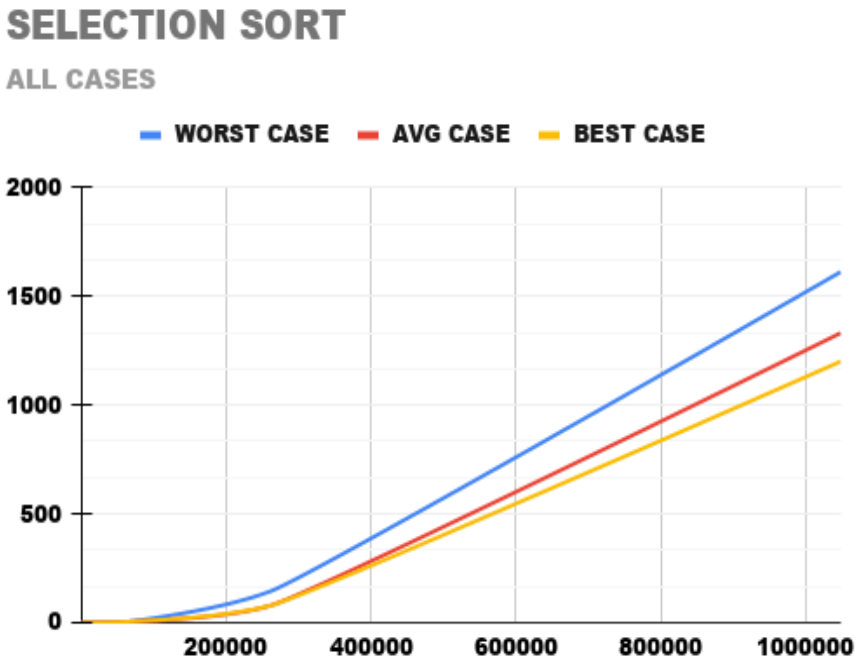


ALL CASES COMBINED

BUBBLE SORT



SELECTION SORT



1.9. Assume that you don't know the time complexity of above algorithms.

1.9.1. Can you predict the same based on your implementation of above algorithms?

By observing Graphs of 3 cases for each algorithm we can conclude:

LINEAR SEARCH

Best case is a constant straight line -> Constant C1 =O(1)

Worst Case is a straight line -> An+b=O(n)

BUBBLE SORT-WITHOUT OPTIMIZATION (WITHOUT USING FLAG)

Best case is a Parabolic graph ->An^2+Bn+C = $\Omega(n^2)$ (Without using flag)

Best case is a Parabolic graph ->An+B= $\Omega(n)$ (Using flag)

Average Case is a Parabolic graph -> An^2+Bn+C = $\theta(n^2)$

Worst Case is a Parabolic graph -> An^2+Bn+C = $O(n^2)$

SELECTION SORT

Best case is a Parabolic graph ->An^2+Bn+C= $\Omega(n^2)$

Average Case is a Parabolic graph -> An^2+Bn+C = $\theta(n^2)$

Worst Case is a Parabolic graph -> An^2+Bn+C = $O(n^2)$

So we can predict values if we know the input N by using extrapolating graphs.

1.9.2. Do they match with theoretical time complexity? Yes/No.

Yes, they would match with theoretical values too.

1.9.3. If yes, then write the time complexity of each algorithm. If no, then write the difference.

Linear Search –

Best Case:- -> Constant C = $\Omega(1)$

Worst Case:- ->An+B= $O(n)$

Bubble Sort –

Average Case:- -> $An^2+Bn+C= \theta(n^2)$

Best Case:- -> $An^2+Bn+C= \Omega(n^2)$(Without flag)

Best Case:- ->An+B= $\Omega(n)$(With optimized flag)

Worst Case:- -> $An^2+Bn+C= O(n^2)$

Selection Sort –

Average Case:- -> $An^2+Bn+C= \theta(n^2)$

Best Case:- -> $An^2+Bn+C= \Omega(n^2)$

Worst Case:- -> $An^2+Bn+C= O(n^2)$