EXPERIMENT 1

20CP209P - Design and Analysis of Algorithm Lab

Aim:

Implement Insertion Sort and Selection Sort and give complexity analysis

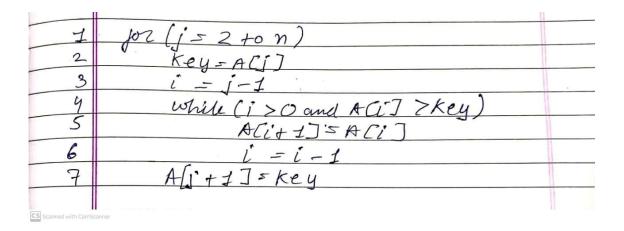
Code:

Insertion Sort:

```
#include <stdio.h>
#include <time.h>
void insertion_sort(int arr[], int len);
int main(void)
  clock_t start, end;
  int arr[] = {7,4,8,9,0,1,2,5,3,6};
  int len = sizeof(arr) / sizeof(int);
  start = clock();
  insertion_sort(arr, len);
  end = clock();
  for (int i = 0; i < len; i++)
  {
     printf("%d ", i);
  printf("\n");
  printf("time taken for execution: ", (double) (end - start));
  return 0;
}
void insertion_sort(int arr[], int len)
  for (int i = 1; i < n; i++)
     int key = arr[i];
    int j = i - 1;
     while (j \ge 0 \&\& arr[j] > key)
       arr[j + 1] = arr[j];
```

```
PS B:\sem4\23bcp153_daa\lab1> gcc insertionsort.c -o insertionsort
PS B:\sem4\23bcp153_daa\lab1> ./insertionsort
0 1 2 3 4 5 6 7 8 9
time taken for execution:
PS B:\sem4\23bcp153_daa\lab1> [
```

Algorithm:



Complexity Analysis:

16		Ti and the second
,	C - 14	
1	- Cost Home) - ()	
1	C1. 2 2	
2	C3(2) 3 3 (8) - X2) 3 1	
3	(3 n-1-10) = 2 + 11 - 11 - 11 - 11 - 11 - 11 - 11 -	
4	Cy	
5	CC 72 1.37147110 = 7 Cj	44
6	22 7 7 10000000 000	
1	2 tj-1	1.
7	C_{7} $n-1$	2
	, 1-m, 3	Ć.
	For Best case	J
1	n 2/01-2007/2010 25	7
2	1 Ce Torres (Sure-1)/2 1-19	d.
3	n-1	J
4	n-1	V
. 5	T(m) = 9(m) + (2/m-1) + (2/m-1) + (2/m) =	
6.	15 (((() (() - () ())) (() ()	
7	n-1	
	T(n) 5 ((n) + (2(n-1) + (3(n-1) + C4()	n_11
	1: (i(0) + (i(0)-+(7/n-1))-1)
	= ((+ Ce+ (3+C4+C7)) - 1(2) 1(3+C40	(1)
		CF)
	=[0(n)]	
9		
	Table 1	-
CS Scanned with Ca	amscanner -	

	Page tilo	Amainst	1 00 1	Mr. Maria	
	Ton in	rst case		1.1	
4		M	3	1110251111111	-
	2	カーユ			
<i></i>	3	n-1	(11 54 6	11)1 - t	-
	4	2+3+4+	.+n = n()	1+1-1/2 - I	
	5	n(n-1)/2	2 1 - 1	* ***	
8	6 (135	n(n-1)(2	11101	1
	7	N-10	11 / 2 6	-1) +C. (h	(n 2) 1
	7(n) 5		1-1) F(3(1)	n-1) + (4 (n	2
	+ (5 (m)	<u>n-1)</u> +46 (n(n=0) +	The (M b)	
			CHCitto	(- C-C+	(7)20
	5 (5+	(6 + (4/n2 +			
		+ (-	(2+ (-63) 7	-6-G) -	
	59h2d	bn+ (= 06)		ر .	18
	. + 4 -	1 to + 1 1 1	1. ++ .	_ 5_	15
	For au	rage cas		32	1 ?
-1	CI	9	A 500	1	٥,
2	(,	n-1	= - (4	T.	于
3	C_{2}	n-1	1 1/2		
Ч	Cy ((n/2) (n/2+	$\frac{1}{1} - \frac{2}{2}$	1779 2.3F	1
	CS	(n/2) (n/2-1)/2		1
¥	Co	(n/2)(n/2-	0/2	- (1	0
7	7	カーエ		1-16 1	P
-	7(n) 5 G	(n) +(2ln-1) + (o(n-1)	+ G (0/2)3	6/27-21
	+(s(M2)(6(n/2)(n/2-1	1/3) FC2 (V	viel)
	9 650 17		7 00	1-11	V
14-4	= (cg + C	5+6 m2 + (cit	-Ce+(3+64/4	7.5/4-6	14-67)
34. 7	8	+ (((2) + (-	(3) + (= 74	1-Cg)	
(3)	· san f	pn+c.	11 + 11 + 41		
	= 0(n2)	100	On the same	(10 (1)	
		-	-	,	
CS Scanned with 6	amScanner			11	

Code:

Selection Sort:

```
#include <stdio.h>
#include <time.h>
void selection_sort(int arr[], int len);
int main(void)
  clock_t start, end;
  int arr[] = \{7,4,8,9,0,1,2,5,3,6\};
  int len = sizeof(arr) / sizeof(int);
  start = clock();
  selection_sort(arr, len);
  end = clock();
  for (int i = 0; i < len; i++)
  {
     printf("%d ", i);
  printf("\n");
  printf("time taken for execution: %f", (double) (end - start));
  return 0;
}
void selection_sort(int arr[], int n)
  for (int i = 0; i < n; i++)
  {
    int min = i;
     for (int j = i; j < n; j++)
       if (arr[j] < arr[min])
          min = j;
     if (min != i)
       int temp = arr[i];
       arr[i] = arr[min];
       arr[min] = temp;
    }
  }
```

```
return;
}
```

```
PS B:\sem4\23bcp153_daa\lab1> gcc selectionsort.c -o selectionsort
PS B:\sem4\23bcp153_daa\lab1> ./selectionsort
0 1 2 3 4 5 6 7 8 9
time taken for execution: 0.000000
PS B:\sem4\23bcp153_daa\lab1> [
```

Algorithm:

1	Jor (i = 0 ton)
2	min = 1
-3 U	for (1=(ton)
5	if (arraj 7< arramin] min = i
6	i' f (min 1 = i')
7	swap (arrsi], arremin]
CS Scanned with CamSo	anner

Complexity Analysis:

		*
	Cost Homes	Le constitution de la constituti
, D.	1-29 min (2) + (2) n (2) + (2) 2 - (1)	-
2	C2: (2) 1/2 (n-1 - (n-1) - (n)	dilion
3	(3) 2 (i + 1)	cheking.
4	C4 2-1, 100 c	
5	(5 5 1-1 1	
6	Ge let n-1 . seas apartir	
7	C7 N-I	
	Barri I	
	Best case 1-1/1/10/10	3
	2 (1-11)	12
3	n-1 (1-11) 18 510	9
4	12 (CC+4) (CC)	9
	11xn	T.
5	160) - ((17) + (11) - (10) - (10)	
6	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
7		
	7/20/20/20/20/20/20/20/20/20/20/20/20/20/	
1274	7(n) = (1(n) + (2(n-1) + (3 (8))	1 (n+1)-1)
	+ (y (n) + (s (n) + (5/0) + (7/0)	2
	5 (C1+(2+ C4+(2) N+(2(m2+n 1)+n	
).e	5 (01+(2+ cy+Co) n+(3(m2+n-1)+0)	

CS Scanned with CamScanner

	Page Hill Other	
A/====	= an2 +bn+c	
N	= 0 (ne)	
N	0 (1)	
W	Worst case	-
N a		-
N 2	n-1 (not 121110)	-
$N = \frac{2}{3}$	(n(n+1/2)-1	1/2
n 4	n (n = 1) 2	- 6
1 5	n(n=1) (2)	-
6	Tie 23 + 100 = 77 200 4 200 2	-
7	n-1	
V	7 725	1).
	T(n) = G(n) + (2(n) - (2 + (3(n(n+1/2-	DACY
10/10	(n(n-1)/2) + (5 (n(n-1)/2) + (6 n-(6	+ (7 n-(7
Section 1	santhing 6	
	$= O(n^2)$	1 2
	5 1-11 2	17.
	Average case.	-
1	$\gamma \qquad \vdash = \bowtie$	1 1
2	n-1	1
3	n(n+1)/2-1 3/200 7/200	
4	n/2 (n-1)	1:
75	7/2 8/(n-1)	12
6	n12 (n/2-H)/2 +0.	G
7	no -1 n-1 11 x 1: 10	11/2
	T(n) = G(n) + Gn-(2 + C3(n2 +n -)/3
	+6 (n2 -n) + (5/n2 -n) + (n)	2 -21/2
7	79 2 2 2 2 2 2 2	2112
o App	+ C7 n - C7	1
(1000)	= 162 + Cy + G +6 1 n2 + (1+(-12)+3+1-4)+t/s	2,69
-	2 2 2 4	4
	+ (2) n + (-6) + FC21)	
<i>5</i>	(n2+6n+c = 0 (n2)	
.		
CS Scanned with CarnSc	anner	

EXPERIMENT 2

20CP209P - Design and Analysis of Algorithm Lab

Aim:

Implement Merge Sort and Quick Sort and give complexity analysis

Code:

```
Merge Sort:
```

```
#include <stdio.h>
#include <time.h>
void merge_sort(int arr[], int low, int high);
void merge(int arr[], int low, int mid, int high);
int main(void)
  clock t start, end;
  int arr[] = {7, 4, 8, 9, 0, 1, 2, 5, 3, 6};
  int len = sizeof(arr) / sizeof(int);
  int low = 0;
  int high = len - 1;
  start = clock();
  merge_sort(arr, low, high);
  end = clock();
  for (int i = 0; i < len; i++)
     printf("%d ", arr[i]);
  printf("\n");
  printf("Time taken for execution: %f seconds\n", (double)(end - start) / CLOCKS_PER_SEC);
  return 0;
}
void merge_sort(int arr[], int low, int high)
  if (low < high)
     int mid = (low + high) / 2;
     merge_sort(arr, low, mid);
     merge_sort(arr, mid + 1, high);
     merge(arr, low, mid, high);
  }
```

```
}
void merge(int arr[], int low, int mid, int high)
  int i = low;
  int j = mid + 1;
  int k = 0;
  int arrB[high - low + 1];
  while (i \leq mid && j \leq high)
  {
     if (arr[i] <= arr[j])</pre>
       arrB[k] = arr[i];
       i++; k++;
     }
     else
       arrB[k] = arr[j];
       j++; k++;
     }
  }
  if (i > mid)
     while (j <= high)
       arrB[k] = arr[j];
       j++; k++;
     }
  }
  else if (j > high)
     while (i <= mid)
       arrB[k] = arr[i];
       i++; k++;
     }
  }
  for (int x = 0; x < (high - low + 1); x++)
     arr[low + x] = arrB[x];
  return;
}
```

```
PS B:\sem4\23bcp153_daa\lab2> code mergesort.c
PS B:\sem4\23bcp153_daa\lab2> ./mergesort
0 1 2 3 4 5 6 7 8 9
Time taken for execution: 0.000000 seconds
PS B:\sem4\23bcp153_daa\lab2> [
```

Algorithm:

	0.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1.1	71.20
	Agorthin	, 0
	1.	. 5
1	i-low 1	
2	15 mid +1 1 +18	
3	K=OC & B	+
4	while (i<=mid and j (=high) if (arr[i] <= wrr(j])	
5	(arrsi T = arr 27	
6	BERTT = evt 1+17	-65
7	else else	6-7
8	B[K++] = arn [i++]	6
q	U (i) mid)	21)
10		- 0
10	While (j <= high) B[K++] = arr(j++]	6,3
-12	DIKTES = arr(jet)	112
12	elle ly (j > high)	100
13	while (< <= mid)	513
14	BCK++7 = arritit	(11)
15	for (x from 0 to high - low +1)	3
16	arr [ww+x]-BCx)	
CS connect wi	th CaroConnor	

Complexity Analysis:

Comple	inity Analysis.
t	1 3 5 7 9 1011 2 4 6 8 12 13 14 Page No. Date: 1 1
	Cost
1	Ci
2	
3	C_3 $\frac{1}{4}$
4	Cy n+1
5	n/2
6	(m/a) - 1
78	11/2
9	$ \begin{array}{c} $
10	
11	m+I m511/x
(2	Cia Wi XEE
13	
14	C14 0
15	C1.5 n+1
16	C16
7(n)	Cit C2 + (3 + (y(n+1) + (s(n)) + (s(n-1))
	+ (7(n/) + (8(n-1)+(a+ 210(m+1))2
	$C_1 + C_2 + C_3 + C_4(n+1) + C_5(n_1) + C_6(n_1) + C_6(n_1) + C_{10}(n+1) + C_{10}(n$
	Cyt(s+(s+C1+18+(10+(15+(16))
	+ (Cy + (-(6)+ (-(8)+ 40+ 4+ +(15)
	- anth_
	5 [O(n)]
CS Scanned w	vith CamScanner

Code:

Quick Sort:

```
#include <stdio.h>
#include <time.h>
void quick_sort(int arr[], int low, int high);
int partition(int arr[], int low, int high);
int main(void)
  clock_t start, end;
  // int arr[] = {7, 4, 8, 9, 0, 1, 2, 5, 3, 6};
  int arr[] = {9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
  int len = sizeof(arr) / sizeof(int);
  int low = 0;
  int high = len - 1;
  start = clock();
  quick_sort(arr, low, high);
  end = clock();
  for (int i = 0; i < len; i++)
     printf("%d ", arr[i]);
  printf("\n");
  printf("Time taken for execution: %f seconds\n", (double)(end - start) / CLOCKS_PER_SEC);
  return 0;
}
void quick_sort(int arr[], int low, int high)
  if (low < high)
  {
     int location = partition(arr, low, high);
     quick_sort(arr, low, location - 1);
     quick_sort(arr, location + 1, high);
  }
}
int partition(int arr[], int low, int high)
  int pivot = arr[low];
  int i = low; // i is start in lab algo
  int j = high; // j is end in lab algo
```

```
while (i < j)
{
    while (arr[i] <= pivot && i <= high)
        i++;
    while (arr[j] > pivot && j >= low)
        j--;

    if (i < j)
    {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
}

arr[low] = arr[j];
return j;
}</pre>
```

```
PS B:\sem4\23bcp153_daa\lab2> code quicksort.c
PS B:\sem4\23bcp153_daa\lab2> ./quicksort
0 1 2 3 4 5 6 7 8 9
Time taken for execution: 0.000000 seconds
PS B:\sem4\23bcp153_daa\lab2> [
```

Algorithm:

	NA.	
4	pluot = avi Clow-J	
-	i = 10, 3 1 1 1 1	N Sales
2	$i = i \circ i$	12
ъ	1 - night	1
4	while (i)	62.1
5	while (work = pwot) & ! <=	rugh
6	L-chitt	
7	white fang [] & pivot D&	25 00
. 8	me trum j	1 3
9	(1. (1. < 1)	
10	swap (aver COJ aver [3])	
(1)	ann Clow]=annCiJ	5
(2	an I'T = prot	177
13	return i	5)
	60	3
CS Scanned with Cam!	Scanner	

Complexity Analysis:

-		entia.
	Cost Thomas	1 1 1 1
	COVICE	1
	C2 1	
	C3 1	
	Gu ni id	-
	1	
	6 n/2	
	C2 n	
	C8 N/2	
	$Cq \qquad m/2$	
H	$\frac{C/0}{C1}$ $\frac{m/2-1}{4}$	
	C12 I	
	C13 1	
7G	Cg(n/2) + (2+G +(4(n/2+1). Cg(n) + (6(n/2) + C7(n) + (8(n/2)). Cg(n/2) + (0 (n/2-1) + (11(1)).	12)4
	t(13(I))	+(12(1)
	$ \begin{array}{c} + (13(1)) \\ - (\frac{C_4}{2} + \frac{C_5}{3} + \frac{C_6}{6} + \frac{C_7}{2} + \frac{C_8}{2} + \frac{C_9}{3} + \frac{C_{10}}{2}) n \end{array} $	
	t(13 (I)	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
3	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	
	$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	

Time Analysis of all sorting algorithms:

```
    PS B:\sem4\23bcp153_daa\lab2> gcc avgtimeall.c mysortlib.c -o avgtimeall
    PS B:\sem4\23bcp153_daa\lab2> ./avgtimeall
        Average execution time over 100 iterations:
        Quick Sort: 0.000750 seconds
        Merge Sort: 0.001050 seconds
        Selection Sort: 0.042600 seconds
        Insertion Sort: 0.026930 seconds
        PS B:\sem4\23bcp153_daa\lab2> []
```

Selection takes most time

Quick Sort takes least time