Lab-1: Introduction to Data Structures, Algorithm Analysis & Arrays

Part A:

1. Write algorithms (pseudo-code) for the following problems:
2. Find the maximum and minimum element in a list of n integers.

Needed : arr[10],I,n,min,max

* Start
* Take input for ‘n’ from the user for the size of the array
* Using for loop insert elements into array ‘arr’
* Assign min = arr[0]
* Compare min with every other element of the array:

For(I = 0;I < n;i++) {

If(min > arr[i]) {

min = arr[i];

}

}

* Assign max = arr[0]
* Compare max with every other element of the array:

For(I = 0;I < n;i++) {

If(max < arr[i]) {

max = arr[i];

}

}

* Print ‘min’ and ‘max’
* End

1. Count the number of odd and even numbers in an array of size n.

Needed : arr[10],I,n,even,odd

* Start
* Take input for ‘n’ for the size of the array from user
* Insert elements into array ‘arr’ using for loop
* Assign even and odd = 0
* Using for loop and if condition, count even and odd numbers of the array

For(I = 0;I < n;i++) {

If(arr[i] % 2 == 0) {

Even++;

}

Else {

Odd++;

}

}

* Print the count of even and odd numbers
* end

1. Reverse a given array of integers of size n.

Needed : arr[10],rev[10],I,n,p

* Start
* Insert size of the array into n from user
* Insert elements into array arr using for loop
* Assign p = n-1
* Using for loop, reverse the array arr into rev

For(I = 0;I < n;i++) {

Arr[i] = rev[p];

p--;

}

* Print both original and reversed arrays using loop
* End

1. A Priori Analysis: For each algorithm above, determine:
2. The time complexity (Big O notation).
   * 1. min/max numbers : O(n)
     2. even/odd numbers : O(n)
     3. reverse array : O(n)
3. The space complexity.
4. min/max numbers : O(1)
5. even/odd numbers : O(1)
6. reverse array : O(n)
7. Discuss whether the algorithm is optimal or can be improved.
8. min/max numbers : optimal
9. even/odd numbers : optima
10. reverse array : can be improved and reduced to one loop
11. Given the following code snippet, determine its time complexity and justify your answer:

* for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

printf("\*");

}

}

Time complexity = O(n2)

* void func(int n) {

if (n == 1)

return;

func(n/2);

func(n/2);

}

Time complexity = O(n)

* int i = 1;

while (i < n) {

printf("%d ", i);

i = i \* 2;

}

Time complexity = O(log n)

* for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

printf("\*");

}

}

}

Time complexity = O(n3)

* int fib(int n) {

if (n <= 1)

return n;

return fib(n - 1) + fib(n - 2);

}

Time complexity = O(2n)

* printf("Hello World");

time complexity = O(1)

* for (int i = 0; i < n; i++) {

printf("\*");

}

for (int j = 0; j < n; j++) {

for (int k = 0; k < n; k++) {

printf("#");

}

}

Time complexity = O(n2)