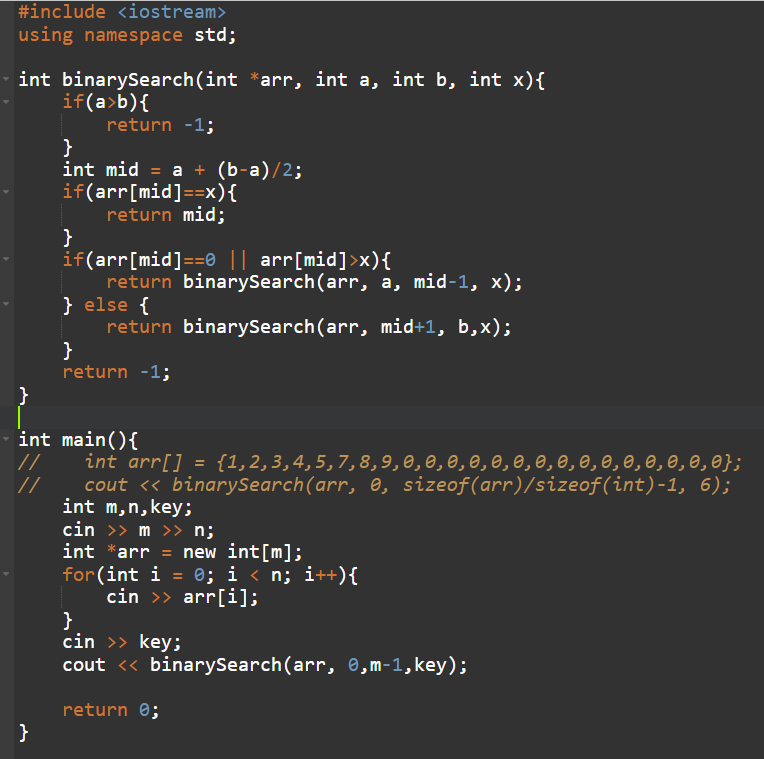
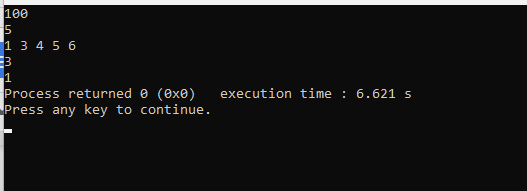
**Algorithms and Problem-Solving Lab (15B17CI471)**

**EVEN 2022**

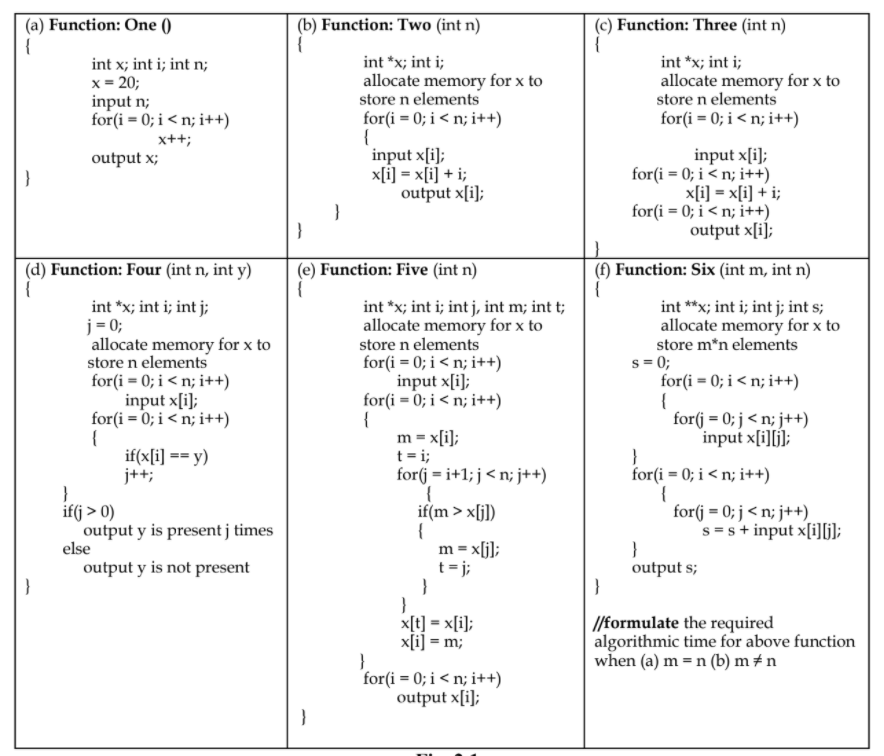
**Week -2 (14 Feb - 19 Feb 2022)**

1. You are given an array A[m] where first n cells contain integers in sorted order and the rest of the cells are filled with 0. Here assumes m>>n and value of n is unknown. Implement an algorithm that takes an integer x as input and finds a position in the array containing x, if such a position exists, in O (log n) time.





1. Find the complexity of the following code snippets:



A. O(n)

B. O(n)

C. O(n)

D. O(n)

E. O(n2)

F. O(m2), O(m\*n)

1. Implement the recursive algorithms for (a) Tower of Hanoi and (b) Fibonacci Number computation and analyse the space and time requirements of both the algorithms.

#include <bits/stdc++.h>

using namespace std;

void towerOfHanoi(char *from*, char *to*, char *aux*, int *n*){

if(*n*==0){

return;

}

towerOfHanoi(*from*, *aux*, *to*, *n*-1);

cout << "Move " << *n* << " from " << *from* << " to " << *to* << "\n";

towerOfHanoi(*aux*, *to*, *from*, *n*-1);

}

int fibonacci(int *n*){

if(*n*==1){

return 0;

} else if(*n*==2){

return 1;

} else {

return fibonacci(*n*-1) + fibonacci(*n*-2);

}

}

int main()

{

towerOfHanoi('a', 'c', 'b', 3);

for (int i = 1; i < 10; i++)

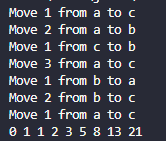
{

cout << fibonacci(i) << " ";

}

return 0;

}



5. Implement the algorithm (Algo\_1) presented below and discuss which task this algorithm performs. Also, analyse the time complexity and space complexity of the given algorithm. Further, implement the algorithm with following modification: replace m = ⌈2n/3⌉ with m = ⌊2n/3⌋, and compare the tasks performed by the given algorithm and modified algorithm.

Algo\_1(A [0 ... n-1])

{

if n = 2 and A[0] > A[1]

swap A[0] ↔ A[1]

else if n > 2

m = ⌈2n/3⌉

Algo\_1 (A [0 .. m − 1])

Algo\_1 (A [n – m .. n − 1])

Algo\_1 (A [0 .. m − 1])

}

#include <bits/stdc++.h>

using namespace std;

void algo1(int \**arr*, int *a*, int *b*){

int n = *b*-*a*+1;

if(n==2 && *arr*[*a*]>*arr*[*a*+1]){

int temp = *arr*[*a*];

*arr*[*a*] = *arr*[*a*+1];

*arr*[*a*+1] = temp;

}

if(n>2){

int m = ceil((2\*n)/3.0);

algo1(*arr*, *a*, *a*+m-1);

algo1(*arr*, *a*+n-m, *b*);

algo1(*arr*, *a*, *a*+m-1);

}

}

void algo2(int \**arr*, int *a*, int *b*){

int n = *b*-*a*+1;

if(n==2 && *arr*[*a*]>*arr*[*a*+1]){

int temp = *arr*[*a*];

*arr*[*a*] = *arr*[*a*+1];

*arr*[*a*+1] = temp;

}

if(n>2){

int m = floor((2\*n)/3.0);

algo2(*arr*, *a*, *a*+m-1);

algo2(*arr*, *a*+n-m, *b*);

algo2(*arr*, *a*, *a*+m-1);

}

}

int main()

{

int n;

cin >> n;

int \*arr = **new** int[n];

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

cin >> arr[i];

}

algo2(arr, 0, n-1);

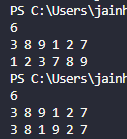
for(int i = 0; i < n; i++){

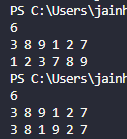
cout << arr[i] << " ";

}

return 0;

}





Algo 1 does sorting of array, Algo 2 does not do any significant task on the array.

Time Complexity of algorithm is

