**Problem1:**

Text, letter

Description automatically generated

**Problem2:**

Letter

Description automatically generated with medium confidence

**Problem 3:**

Text, letter

Description automatically generated

**public** **int** occursOnluOnce(**int**[] inputArray) {

**int**[] bucket = **new** **int**[3\*inputArray.length];

**for**(**int** i = 0; i < inputArray.length ; i++) {

bucket[inputArray[i]] += 1;

}

**for**(**int** i = 0 ; i < inputArray.length ; i++) {

**if**(bucket[inputArray[i]] == 1) {

**return** inputArray[i];

}

}

**return** -1;

}

**Running Time Analysis:**

1. Creating bucket array takes 3n -> O(n)
2. Counting occurrence of elements takes -> O(n)
3. Searching first single occurrence element takes in worst case -> O(n

Total Running Time = O(n)

**Problem4:**

1. Recursive algorithm to calculate the nth Fibonacci number

Text

Description automatically generated Graphical user interface, text

Description automatically generated

a) pseudo-code b) implementation

1. It is not straight forward to apply one of the methods discussed in class. However, we can determine the running time using by drawing the binary tree representation of the recursive calls. Analytically the running time can be represented as .

This can be evaluated as follows

T open parentheses n close parentheses space equals space T open parentheses n minus 1 close parentheses plus thin space T open parentheses n minus 2 close parentheses space plus thin space C space space space space space space space C space i s space s o m e space c o n s tan t space open parentheses thin space 5 space i n space t h i s space c a s e close parentheses
T open parentheses n close parentheses space less than thin space T open parentheses n minus 1 close parentheses space plus thin space T open parentheses n minus 1 close parentheses space space space space space space space space space space space space space space space B e c a u s e space T open parentheses n minus 1 close parentheses space greater than thin space T open parentheses n minus 2 close parentheses
equals space 2 space T open parentheses n minus 1 close parentheses space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space
equals space 2 times 2 T open parentheses n minus 2 close parentheses space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space space A p p l y space t h e space s a m e space i d e a space open parentheses thin space close parentheses
equals space 2. space 2. space 2 space T open parentheses n minus 3 close parentheses
vertical ellipsis
equals space 2 to the power of k space space T open parentheses n minus k close parentheses space comma space s o l v i n g space n space minus space k space equals space 0 space comma space k space equals space n space comma space T open parentheses n minus k close parentheses space equals space T open parentheses 0 close parentheses space equals space 1
equals space 2 to the power of n
equals space O open parentheses 2 to the power of n close parentheses space space space space space space space space space space



Fibonacci series using iterative approach

Text

Description automatically generated Text

Description automatically generated

a) pseudo-code b) implementation

1. **Running Time of the iterative approach**: O(n)