|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| N | 0 | 1 | 2 | 3 | 4 | 5 | 6 |  |
| Fib at N | 0 | 1 | 1 | 2 | 3 | 5 | 8 |  |

Key n = index

Return value = fib(n)

Fib (n) = fib (n-1) + fib (n-2)

Base case fib(0) = 0, fib (1) = 1

fib(int n){

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

}

Fib(2) return fib(1) + fib(0)

Fib(3) return fib(1) + fib(0) + fib(1)

N = 4

fibAtN = 3

nums = [-1,0,3,5,9,12], target = 9

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| -1 | 0 | 3 | 5 | 9 | 12 | 14 |

Left\_pointer = 0

Right\_pointer = 6

While left < right {

Mid = (left + right)/2}

if nums[mid\_pointer] < target {

left\_pointer = mid\_pointer + 1;

}

Brute force

public boolean searchMatrix *(*int*[][]* matrix, int target*){* if *(*matrix.length == 0*){* return false;  
 *}* int n = matrix.length;  
 int m = matrix*[*0*]*.length;  
  
 for *(*int i = 0; i < n; i++*) {* for *(*int j = 0; j < m; j++*) {* if *(*matrix*[*i*][*j*]* == target*){* return true;  
 *}  
 }  
 }* return false;  
*}*

public class mPlusN *{* public boolean searchMatrix *(*int*[][]* matrix, int target*){* if *(*matrix.length == 0*){* return false;  
 *}* int m = matrix.length;  
 int n = matrix*[*0*]*.length;  
  
 int row = 0;  
 int col = n - 1;  
  
 while *(*col >= 0 && row < n*){* if *(*matrix*[*row*][*col*]* == target*){* return true;  
 *}* else if *(*matrix*[*row*][*col*]* > target*){* col--;  
 *}* else*{* row++;  
 *}  
 }* return false;   
 *}  
}*

isHappyNumber

Set<Integer> visited;

public boolean isHappy(int n) {

visited = new HashSet<>();

return isHappyR(n);

}

private boolean isHappyR(int n){

if (n==1){

return true;

}

int sum = 0;

while(n!=0){

sum += Math.pow((n%10), 2);

n /= 10;

}

if (visited.contains(sum)){

return false;

}

visited.add(sum);

return isHappyR(sum);

}