Hamza Ben Terdayet

Algorithme 1 :

class Solution {

public int[] solution(int N, int[] A) {

int[] compteurs = new int[N];

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

compteurs [i] = 0;

}

int min = 0;

int max = 0;

int M = A.length;

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

if (A[k] <= N) {

if (compteurs [A[k] - 1] < min + 1) {

compteurs [A[k] - 1] = min + 1;

} else {

compteurs [A[k] - 1]++;

}

if (compteurs [A[k] - 1] > max) {

max = compteurs [A[k] - 1];

}

} else {

min = max;

}

}

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

if (compteurs [i] < min) {

compteurs [i] = min;

}

}

return compteurs ;

}

}

Algorithme 2:

class solution {

public int[] solution(int[] A, int[] B) {

int L = A.length;

int max = 0;

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

max = Math.max(A[i], max);

}

int[] fibonacci = new int[max+1];

for(int i=2; i<= max; i++){

fibonacci[i] = (fibonacci[i-1] + fibonacci[i-2]) % (pow(2,30));

int[] steps = new int[L];

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

steps[i] = fibonacci[A[i]] % (pow(2, B[i]));

}

return steps;

}

}

Algorithme 3 :

class Solution {

public int solution(int[] A) {

int N = A.length;

int debut = 0;

int fin = N - 1;

int minAbsSum = Math.abs(A[debut] + A[fin]);

while (debut <= fin) {

int currentSum = A[debut] + A[fin];

minAbsSum = Math.min(minAbsSum, Math.abs(currentSum));

if (currentSum <= 0)

debut++;

else

fin--;

}

return minAbsSum;

}

}