**Esercizio 1**

#include <iostream>

#include <vector>

#include <algorithm>

**using** **namespace** std;

**void** fillVector(**int**\*&,**int**);

**void** printVector(**int**\*,**int**);

**int** countNumberOfOccurrence(**int**\*,**int**,**int**,**int**);

**int** countOccurrence(**int**\*,**int**,**int**,**int**);

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** test\_cases;

**int** \*array;

**int** dim;

**int** k;

cout<<"Test cases: ";

cin>>test\_cases;

**while**(test\_cases){

cin>>k;

cin>>dim;

fillVector(array, dim);

cout<<countNumberOfOccurrence(array, 0, dim-1,k)<<endl;

test\_cases--;

**delete**[] array;

}

**return** 0;

}

**void** fillVector(**int**\*& array,**int** n){

array = **new** **int**[n];

cout<<"Inserisci i numeri"<<endl;

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

cout<<"[ "<<i<<" ]: ";

cin>>array[i];

}

}

**void** printVector(**int**\* array,**int** n){

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

cout<<array[i]<<" ";

}

}

**int** countNumberOfOccurrence(**int**\* array,**int** start,**int** end,**int** k){

**if**(start == end){

**if**(array[start] == k){

**return** 1;

}**else**{

**return** 0;

}

}

**int** mid = start +(end-start)/2;

**int** n\_occ\_1 = 0,n\_occ\_2 = 0;

n\_occ\_1 = countNumberOfOccurrence(array, start, mid,k);

n\_occ\_2 = countNumberOfOccurrence(array, mid+1, end,k);

**return** n\_occ\_2 + n\_occ\_1;

}

T(n) = 2T(n/2) + O(1) ———-> Caso 1 teorema dell’esperto

**COMPLESSITA’:** theta(n) oppure O(nlogn)

**Esercizio 2**

#include <iostream>

**using** **namespace** std;

**bool** isPrime(**int** n);

**bool** isSafe(**int** n, **int** \*output, **int** k);

**bool** primeBacktrack(**int** N, **int** P, **int** S, **int** \*output, **int** k,**int** number);

**bool** isASolution(**int** \*output, **int** N, **int** S);

**void** PrintSolution(**int** \*output,**int** N);

**int** main(**int** argc, **const** **char** \* argv[]) {

**int** N, P, S;

**int** \*output;

**int** test\_cases;

**int** test\_Case\_count = 0;

cout<<"Inserisci numero di casi di test"<<endl;

cin>>test\_cases;

**while**(test\_Case\_count < test\_cases){

cin>> S;

cin>> N;

cin>> P;

output = **new** **int**[N];

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

output[i] = 0;

}

cout<<"CASO DI TEST "<<test\_Case\_count+1<<endl;

primeBacktrack(N, P, S, output, 0,P);

**delete**[] output;

test\_Case\_count++;

}

**return** 0;

}

**bool** isPrime(**int** n){

**bool** is\_prime = **true**;

**if** (n == 0 || n == 1) {

is\_prime = **false**;

}

**for** (**int** i = 2; i <= n/2; ++i) {

**if** (n % i == 0) {

is\_prime = **false**;

**break**;

}

}

**return** is\_prime;

}

**bool** isSafe(**int** n, **int** \*output, **int** k){

**if**( !isPrime(n) ){

**return** **false**;

}

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

**if**(output[i] == n){**return** **false**;}

}

**return** **true**;

}

**bool** primeBacktrack(**int** N, **int** P, **int** S, **int** \*output, **int** k, **int** number){

**if**( k == N ){

**if**(isASolution(output, N, S)){

PrintSolution(output, N);

**return** **true**;

}**else**{

**return** **false**;

}

}

**bool** res = **false**;

**for**(**int** i=number+1; i<=S ; i++ ){

**if**(isSafe(i,output,k)){

output[k]= i;

res = primeBacktrack(N, P, S, output, k+1,i) || res;

output[k] = 0;

}

}

**return** res;

}

**bool** isASolution(**int** \*output, **int** N, **int** S){

**int** sum = 0;

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

sum += output[i];

}

**if**(sum == S){

**return** **true**;

}**else**{

**return** **false**;

}

}

**void** PrintSolution(**int** \*output,**int** N){

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

cout<<output[i]<<" ";

}

cout<<endl;

}

**NB:** Ho usato una funzione che controlla se il numero che sto considerando è primo, questo perché faccio la scrematura direttamente nella funzione di backtracking, senza allocare un vettore con tutti i numeri primi nell’intervallo (P,S]

**COMPLESSITA’:** O(2^(S-P))