Some tips…

<https://www.geeksforgeeks.org/top-algorithms-and-data-structures-for-competitive-programming/>

BIT MANIPULATIONS

* Trees - Binary Lifting
* Fenwick Tree for Dynamic Range Problems
* Masking

PREFIX AND SUFFIX

*prefix*[i] = // prefix sum of all elements of array from index 0 to index i(inclusive)

*prefix*[0] = *a*[0]

*prefix*[1] = *a*[0] + *a*[1] = *prefix*[0] + *a*[1]

*prefix*[i] = *a*[0] + *a*[1] + .... + *a*[i - 1] + *a*[i] = *prefix*[i - 1] + *a*[i]

*suffix*[i] = // sum of all elements from index i to n - 1 (both inclusive)

*suffix*[n - 1] = *a*[n - 1]

*suffix*[i] = *suffix*[i + 1] + *a*[i]

PERMUTATIONS

* Given a permutation of the array (arrangement) means it's just a modified
* problem statement.
* Sorting is one permutation of the array.
* HackerRank question - Find Next Permutation of given array.

## Exponentiation

[PROBLEM] Find x Power n.

int ans = power(2, 100000);

* This value is outside the range of int.
* Max value of an int is INT\_MAX = 231 -1 = 2 \* 210 \* 210 \* 210 = 2 \* 103\*103\*103 ≈ 2 \* 109
* So, the problem is changed to…

[PROBLEM] Find x Power n MOD M.

0 <= x, n <= 100000

M = 109 + 7

* This value is outside the range of int

## The Power Function

TC: O(N) – Brute force…

long long power(long long x, int n, int m) {

// Multiply x ‘n’ times.

}

TC: O(logN)

long long power(long long x, int n, int m) {

if (n == 0) return 1;

long long hf = power(x, n / 2);

long long sq = ((hf % m) \* (hf % m)) % m;

if (n & 1) return x \* sq;

return sq;

}

## MATRIX EXPONENTIATION

[PROBLEM] Given a matrix, calculate matrix10 % M. Where M = 109+7

#include <iostream>

#include <vector>

#include <iomanip>

using namespace *std*;

long long int MOD = 1E9 + 7;

*vector*<*vector*<int>> identityMatrix(int r, int c) {

*vector*<*vector*<int>> im(r, *vector*<int>(c));

for (auto i = 0; i < r; i++)

im[i][i] = 1;

return im;

}

*vector*<*vector*<int>> multiplyMatrix(*vector*<*vector*<int>> a, *vector*<*vector*<int>> b) {

int rows = a.*size*();

int cols = b[0].*size*();

int rows2 = b.*size*();

*vector*<*vector*<int>> res(rows, *vector*<int>(cols));

for (auto i = 0; i < rows; i++) {

for (auto j = 0; j < cols; j++) {

res[i][j] = 0;

for (auto k = 0; k < rows2; k++) {

res[i][j] = ((a[i][k] % MOD) \* (b[k][j] % MOD)) % MOD + res[i][j] % MOD;

}

}

}

return res;

}

*vector*<*vector*<int>> powerOfMatrix(*vector*<*vector*<int>> mat, int n) {

if (n == 1) return mat;

if (n == 0) return identityMatrix(mat.*size*(), mat[0].*size*());

*vector*<*vector*<int>> halfSq = powerOfMatrix(mat, n / 2);

*vector*<*vector*<int>> sqOfHalf = multiplyMatrix(halfSq, halfSq);

if (n & 1) return multiplyMatrix(sqOfHalf, mat);

return sqOfHalf;

}

void display2DMatrix(*vector*<*vector*<int>>& matrix) {

for (auto i : matrix) {

for (auto j : i) {

*cout* << *setw*(2) << j << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

int main()

{

int n, rows, cols; *cin* >> n >> rows >> cols;

*vector*<*vector*<int>> mat(rows, *vector*<int>(cols));

for (auto i = 0; i < rows; i++) {

for (auto j = 0; j < cols; j++) {

*cin* >> mat[i][j];

}

}

auto res = powerOfMatrix(mat, n);

display2DMatrix(res);

return 0;

}

**[PROBLEM]** Given N, find the nth Fibonacci number mod M. Where M = 109+7 (big prime integer).

1 <= N <= 1018

* In mathematical terms, the sequence Fn of Fibonacci numbers is defined by the recurrence relation…

Fn = Fn-1 + Fn-2 Where F0 = 0 and F1 = 1.

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

fib(0) = O

fib(1) = 1

* What is the minimum time complexity to find nth Fibonacci Number?

|  |  |
| --- | --- |
| int fib(int n) {  if (n <= 1)  return n;  return fib(n - 1) + fib(n - 2);  } | T(N) = T(N-l) + T(N-2)  <= T(N-l) + T(N-1)  <= 2T(N-1)  **T(N) = 2N**  **TLE (Time Limit Exceeded)**  **Space Complexity**: Depth of recursion call stack: N |
| int fib(int n){  *vector*<int> f(n);  f[0] = 0; f[1] = 1;  for (auto i = 2; i <= n; i++) {  f[i] = f[i - 1] + f[i - 2];  }  return f[n];  } | TC: n-1  SC: 1+ (n+1) => 1018  **TLE (Time Limit Exceeded)**  **MLE (Memory Limit Exceeded)** |
| long long fib(int n) {  if (n <= 1) return n;  long long a = 0, b = 1, c;  for (int i = 2; i <= n; i++) {  c = a + b;  a = b;  b = c;  }  return 0;  } | TC: n-1  SC: 3  **TLE (Time Limit Exceeded)** |

Will log2N be an acceptable solution?

210 ~= 103

log2(1018) = log2(260) = 60

* Let’s use “MATRIX EXPONENTIATION” to solve the above problem.

=

=

= X

= X

= X

….

= X

Base case…

* Fn-k -> F1 when n-k = 1, k = n-1.

= X

* So, to calculate the nth Fibonacci number, we raise the matrix to (n-1) and take the first number.
* That is… in the below matrix is the answer!!!

= X

#include <iostream>

#include <vector>

#include <iomanip>

using namespace *std*;

//long long int MOD = 1E9 + 7;

*vector*<*vector*<int>> identityMatrix(int r, int c) {

*vector*<*vector*<int>> im(r, *vector*<int>(c));

for (auto i = 0; i < r; i++)

im[i][i] = 1;

return im;

}

*vector*<*vector*<int>> multiplyMatrix(*vector*<*vector*<int>> a, *vector*<*vector*<int>> b) {

int rows = a.*size*();

int cols = b[0].*size*();

int rows2 = b.*size*();

*vector*<*vector*<int>> res(rows, *vector*<int>(cols));

for (auto i = 0; i < rows; i++) {

for (auto j = 0; j < cols; j++) {

res[i][j] = 0;

for (auto k = 0; k < rows2; k++) {

res[i][j] = ((a[i][k] /\*% MOD\*/) \* (b[k][j] /\*% MOD\*/)) /\*% MOD\*/

+ res[i][j] /\*% MOD\*/;

}

}

}

return res;

}

*vector*<*vector*<int>> powerOfMatrix(*vector*<*vector*<int>> mat, int n) {

if (n == 1) return mat;

if (n == 0) return identityMatrix(mat.*size*(), mat[0].*size*());

*vector*<*vector*<int>> halfSq = powerOfMatrix(mat, n / 2);

*vector*<*vector*<int>> sqOfHalf = multiplyMatrix(halfSq, halfSq);

if (n & 1) return multiplyMatrix(sqOfHalf, mat);

return sqOfHalf;

}

void display2DMatrix(*vector*<*vector*<int>>& matrix) {

for (auto i : matrix) {

for (auto j : i) {

*cout* << *setw*(2) << j << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

int fib(int n) {

if (n <= 1) return n;

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

}

long long nthFib(int n) {

*vector*<*vector*<int>> mat = { {1,1},{1,0} };

auto res = powerOfMatrix(mat, n-1);

return res[0][0];

}

int main() {

int n; *cin* >> n;

*cout* << "Nth Fibonacci number (Using Recursion): " << fib(n) << "\n";

*cout* << "Nth Fibonacci number (Using Matrix Exponentiation): " << nthFib(n) << "\n";

return 0;

}

MATRIX EXPONENTIATION

MY TUTORIAL:

https://www.youtube.com/playlist?list=PLQ9cQ3JqeqU8dR5O3BaN4azRzqBp\_M4im

TUTORIAL: https://www.youtube.com/watch?v=eMXNWcbw75E

CONTEST:

https://codeforces.com/gym/102644

CONTEST-SOLUTIONS

https://www.youtube.com/watch?v=kQuCOFzWoa0

https://www.youtube.com/watch?v=RA\_SpxP2t54

REFERENCE:

https://codeforces.com/blog/entry/80195