**★★★☆☆**

**題組：Problem Set Archive** **with Online Judge**

**題號：10007: Count the Trees**

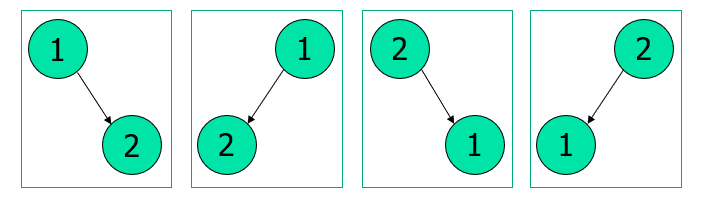
**解題者：黃建廷**

**解題日期：2022年05月19日**

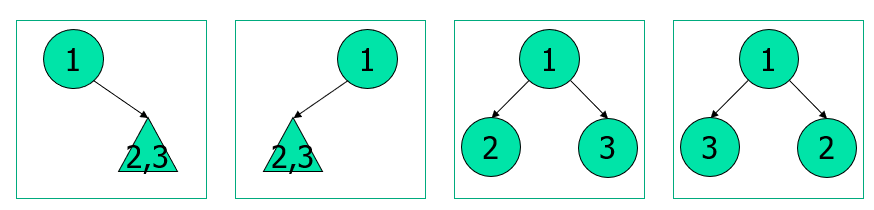
**題意：**

有n個node的binary tree，請問有此tree有幾種排列組合方式？

**題意範例：**

Input 2:

Output: 4

Input: 3

Output: 30

**解法：**

將問題分解為二：

(1) 不管node之編號，單純算n個node可有幾種tree之排列

Dynamic Programming

大數加法/乘法

(2) 將node之編號帶入各個tree的排列

排列組合：n!

大數乘法

(3) 將(1)與(2)相乘

若有範例，則列出。若沒有，則填寫「無」。

**解法範例**：

Input: 2 - (1)

tree有幾種排列：

1個root node

一張含有 文字, 名片, 向量圖形 的圖片

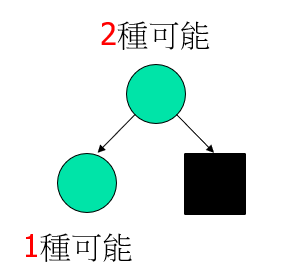
自動產生的描述一張含有 桌 的圖片

自動產生的描述1個other node

Tree之排列：2

Input: 2 - (2)

將編號填入tree中

每一種tree之排列數：2!

Output: 2 \* 2! = 4

Input: 3 - (1)

tree有幾種排列：

1個root node

一張含有 文字, 名片, 螢幕擷取畫面, 向量圖形 的圖片

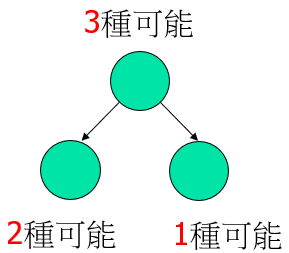
自動產生的描述一張含有 桌 的圖片

自動產生的描述2個other node

Tree之排列：5

Input: 3 - (2)

將編號填入tree中

每一種tree之排列數：3!

Output: 5 \* 3! = 30

**討論：**

(1) 將計算過之 n! 值儲存，可提升下筆輸入之運算速度

(2) 大數乘法時，需注意可能是兩個大數之相乘

**程式：**

#include <iostream>

#include <cstring>

#include <cstdlib>

#include <vector>

#include <fstream>

//#define TEST

using namespace std;

typedef struct NODE{

vector<char> tree;

vector<char> fact;

}NODE;

void init();

void print(vector<char> str);

void tree\_count(short int n);

void fact\_count(short int n);

void add(vector<char>& d, vector<char> s);

vector<char> multiply(vector<char> a, vector<char> b);

NODE record[301];

ofstream file;

int main(void)

{

short int choice = 0;

vector<char> result;

#ifdef TEST

file.open("file.txt", ios::out | ios::trunc ); //why cannot put in global

#endif

init();

do{

cin >> choice;

if(!choice) continue;

tree\_count(choice);

//cout << "tree:"; print(record[choice].tree); cout << "---------\n";

fact\_count(choice);

//cout << "fact:"; print(record[choice].fact); cout << "---------\n";

result = multiply(record[choice].tree, record[choice].fact);

print(result);

}while(choice);

return 0;

}

void init()

{

record[0].tree.push\_back('1');

record[0].fact.push\_back('1');

return;

}

void print(vector<char> str)

{

for(int i = str.size()-1; i >= 0; i--){

#ifndef TEST

cout << str[i];

#endif

#ifdef TEST

file << str[i];

#endif

}

#ifndef TEST

cout << endl;

#endif

#ifdef TEST

file << endl;

#endif

return;

}

void tree\_count(short int n)

{

if(!record[n].tree.empty()) return;

vector<char> temp;

for(short int i = 0; i < n>>1; i++){ //i.e., i < (n-1)/2 +(n-1)%2

tree\_count(i);

tree\_count(n-1-i);

add(record[n].tree, multiply(record[i].tree, record[n-1-i].tree) ); //may have problem

//cout << \_\_LINE\_\_ << ":"; print(record[n].tree); //

}

vector<char> multiplicand(1, '2');

record[n].tree = multiply(record[n].tree, multiplicand);

//cout << \_\_LINE\_\_ << ":"; print(record[n].tree); //

if(n%2){

tree\_count(n>>1);

temp = multiply(record[n>>1].tree, record[n>>1].tree);

//cout << \_\_LINE\_\_ << ":"; print(temp); //

add(record[n].tree, temp);

//cout << \_\_LINE\_\_ << ":"; print(record[n].tree); //

temp.clear();

//add(record[n].tree, multiply(record[n>>1].tree, record[n>>1].tree) ); //may have problem

}

return;

}

void fact\_count(short int n)

{

if(!record[n].fact.empty()) return;

fact\_count(n-1);

vector<char> multiplicand;

short int t = n;

while(t){

multiplicand.push\_back('0' + (t%10));

t /= 10;

}

record[n].fact = multiply(record[n-1].fact, multiplicand);

return;

}

void add(vector<char>& d, vector<char> s) //add s to d, do not allocate

{

//cout << "d.capacity: " << d.capacity() << endl << "s.capacity: " << s.capacity() << endl;

if(d.size() < s.size()){

for(int i = d.size(); i < s.size(); i++) d.push\_back('0');

}

else if(d.size() > s.size()){

for(int i = s.size(); i < d.size(); i++) s.push\_back('0');

}

//cout << "d.capacity: " << d.capacity() << endl << "s.capacity: " << s.capacity() << endl;

char carry = 0, temp = 0;

for(short int i = 0; i < d.size(); i++){

temp = d[i] + s[i] - 2\*'0' + carry;

carry = temp/10;

d[i] = '0' + (temp%10);

}

if(carry) d.push\_back('0' + carry);

//cout << "d.capacity: " << d.capacity() << endl << "s.capacity: " << s.capacity() << endl;

return;

}

vector<char> multiply(vector<char> a, vector<char> b) //returning muliple of a and b, allocate

{

vector<short int> temp; //may need int

vector<char> result;

temp.assign(a.size() + b.size(), 0);

result.assign(a.size() + b.size() - 1, '0');

for(int i = 0; i < a.size(); i++){

for(int j = 0; j < b.size(); j++){

temp[i+j] += (a[i]-'0') \* (b[j]-'0'); //add comparison with 0 to speedup?

}

}

short int carry = 0;

for(int i = 0; i < temp.size() - 1; i++){

temp[i] += carry;

result[i] = '0' + (temp[i]%10);

carry = temp[i]/10;

}

if(carry) result.push\_back('0' + carry);

//cout << "result.capacity: " << result.capacity() << endl;

temp.clear();

return result;

}