tags: `IM2009 Algorithms  Homework`

Algorithms Homework Assignment 5

B08705034 資管二 施芊羽

Basic Introduction

- 執行環境：Windows 10
- 使用語言：C++ 11
- Online Judger for Testing:
  - Codeforces https://codeforces.com/gym/101205/submit (https://codeforces.com/gym/101205/submit)
  - Kattis https://icpc.kattis.com/problems/fibonacci (https://icpc.kattis.com/problems/fibonacci)

Problem Description

Given a function

$$F(n) = \begin{cases} 0 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ F(n-1) + F(n+1) & \text{if } n \geq 2 \end{cases}$$

while $+$ denotes the concatenation of string. With input an integer $n(0 \leq n \leq 100)$ and a pattern p(has a length of at most $100,000$ characters) find out the occurences of p in $F(n)$.

First Glimpse

To solve this problem, if there are no time and memory limits, we can using the following way for sure

```
void main(int argc, char* argv[]){
    int n = 0;
    string p = "";
    int currentcase = 0;
    while(cin >> n >> p){
        currentcase++;
        vector<string> Ftable;
        Ftable.push_back("0"); \\F(0)
        Ftable.push_back("1"); \\F(1)
        for(int i = 2 ; i < n ; i++){
            Ftable.push_back(F(i - 1) + F(n - 2));
        } \\construct the Fibonacci table

        size_t found = Ftable[n].find(pattern, 0);
        int count = 0;
        while(found != std::string::npos){
            count++;
            found = Ftable[n].find(pattern , found + 1);
        }
        cout << "Case " << currentcase << ": " << count;
    }
}
```

Using this way, we just construct the whole table of $F(n)$ to the given input n and search the occurences of p in $F(n)$ directly. However, the following problems will occur:

1. Memory Limit Exceeds: The construction of `Ftable` will cost a lot of memory space when n is large. (i.e. The length of $F(28) = 514229$)
2. Time Limit Exceeds: The search of occurence in $F(n)$ takes a lot of time, the using of `find` to count the occurences of p has an average time of $O(n)$ while the worst case is $O(m \times n)$ where m is the length of p. As 1. mentioned, the length of $F(n)$ may be long so if we search for p in a longer $F(n)$ will have a huge cost of time.

Hypothesis

To use the way **"Divide and Conquer"**, we'd like to search for the patterns occur in $F(i)$ for different $0 \leq i \leq n$. Let $Count(n)$ stands for the occurences of p in $F(n)$ will there be any relationship with $Count(n)$ and the equation $F(n) = F(n-1) + F(n-2)$?

It seems that $Count(n) = Count(n-1) + Count(n-2) + Intersection(n)$ while $Intersection$ is the number of occurences of p in the concatenation of $F(n-1) + F(n-2)$ (which means the p in this section will be constructed by part of $F(n-1)$ and another part of $F(n-2)$).

Observation

The Head(Begin) of $F(n)$

By observation, we can find that the head(begin) of $F(n)$ will be the same of different length from $F(1)$ since $F(n) = F(n-1) + F(n-2)$. We can say that $F(n)$ and $F(n-1)$ will have the same head of length equal to the length of $F(n-1)$ for $n \geq 2$ because $F(n)$ is a concatenation of $F(n-1)$ and $F(n-2)$
as the parts highlightened on the following image.

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	10110101101101011010101
8	1011010110110101101010110110101101101011010101
9	101101011011010110101101101011011010110101101101011010101101101011010101

The Tail(End) of $F(n)$

$F(n) = F(n-1) + F(n-2)$, we can figure out that as what we found in the relationship of the head, the tail of $F(n)$ might have a same tail as $F(n-2)$ of the length of $F(n-2)$. Which means that every $F(n)$ and $F(n-2)$ will have the same tails, and there will be two types of the

patterns of tails as the pink- and blue- highlightened parts on the following image.

n	$F(n)$
0	0
1	1
2	10
3	101
4	10110
5	10110101
6	1011010110110
7	101101011011010110101
8	10110101101101011010110110101101011011
9	1011010110110101101011011010110110101101011011010110101

$Intersection$ and Heads & Tails

$Intersection(n)$ we've just be defined we'll be the occurences of pattern p in $Tail(n-1)+Head(n-2)$ while each length of the two parts might be at most the length of $p-1$[1].

[1] *If the $Head$ or $Tail$ is longer than the length of p it'll be calculated in $Count(n-1)$ or $Count(n-2)$.*

Thinking

Can I find k that the $Intersection(k)$ will be two constants(as there are two types of tails) after $n > k$?

If I find the k where I can find $Intersection$ will be two kinds of constants, $Count(n) = Count(n-1) + Count(n-2) + c$ where c will be either $c1$ or $c2$ since there'll only be two types of $Intersection(n)$.

Solution

Definition

- For the following description, I use i to represent the index of $F(i)$ to prevent from the misunderstanding between input n and $F(n)$.
- `m` is the length of input p.

Function `searchPattern`

```
int searchPattern(string target, string pattern){
        size_t found = target.find(pattern, 0);

        int count = 0;
        while(found != std::string::npos){
                count++;
                found = target.find(pattern , found + 1);
        }
        return count;
}
```

I use this function to count for the occurences of pattern in target.

Input

```
int n = 0;
string p = "";
while(cin >> n >> p)
...
```

I use an `int` to store n and a `string` to get p. The using of `while` is for the case that'll have several inputs.

Step 1 Build the Fibonacci Table

```
vector<string> Ftable;
Ftable.push_back("0"); \\F(0)
Ftable.push_back("1"); \\F(1)
while(Ftable.size() < 28){
    int cur = Ftable.size();
    Ftable.push_back(Ftable[cur - 1] + Ftable[cur - 2]);
}
```

To begin with, I use a `vector<string>` to store the $F(i)$ I need. I use a `while` loop to construct the Fibonacci table before any inputs.

Why only construct the table to $F(27)$ while the input n satisfied $1 \leq n \leq 100$?

As we calculate the length of $F(n)$, we can know $F(27)$ will have $317,811$ characters, which we can get $F(26)$ and $F(25)$ might also have more than $100,000$ characters. Thus, even the longest pattern p can be given as the input(Max length $= 100,000$), the k we want to find will

still be smaller than or equal to 27. Therefore, we can get all the $Head$s and $Tail$s we want from the `Ftable` we constructed.

Step 2 Basic Search of $Count(i)$

Basic Search stands for getting $Count(i)$ through rough search of the occurences of p instead of using the formula $Count(i) = Count(i-1) + Count(i-2) + Intersection$.

```
vector<long long int> count;
if(p == "0")
    count.push_back(1);
else
    count.push_back(0);
if(p == "1")
    count.push_back(1);
else
        count.push_back(0);
for(int i = 2 ; Ftable[i - 2].length() < m && i <= n ; i++){
        if(m > Ftable[i].length())
                count.push_back(0);
    else
                count.push_back(searchPattern(Ftable[i] , p));
                }
```

I use `vector<>` to memorize $Count(i)$. And for $Count(0)$ and $Count(1)$ we use direct `if-else` to determine whether the pattern exists in $F(0)$ and $F(1)$. For $i \geq 2$, I'll first determine whether $F(i)$ is longer than the pattern p: if the length of $F(i)$ is shorter than p, then $Count(i) = 0$; else I'll use `searchPattern` to search for the occurences of p.

When to End Basic Search

To determine whether the basic search should stop, I use `Ftable[i - 2].length() < m && i <= m` (`m` is the length of `p`) to judge. Since if the length of $F(i-2)$ is longer than p then I will be able to get the two $Intersection$ constants I want.

Step 3 Get the $Intersection$

```
int p1, p2;
if(count.size() <= n){
    string s1 = Ftable[count.size() -1].substr(Ftable[count.size() - 1].length() - m + 1)
                + Ftable[count.size() - 2].substr(0, m - 1);
    p1 = searchPattern(s1, p);
    string s2 = Ftable[count.size() - 2].substr(Ftable[count.size() - 2].length() - m + 1
                +  Ftable[count.size() - 1].substr(0, m - 1);
    p2 = searchPattern(s2, p);
}
```

First, I'll determine whether the `Count` is filled up with $n+1$ or not to prevent from Runtime Error. Then I'll use `substr` to get the heads and tails I want and `+` to connect head and tail. Therefore, I use `searchPattern` again to finally get the two constant $Intersection$s which are label as `p1` and `p2` in the program.

Step 4 Complete all $Count(n)$

```
int a = count.size();
for(int i = count.size() ; i <= n ; i++){
        if((i - a) % 2 == 0)
                count.push_back(count[i - 1] + count[i - 2] + p1);
        else
                count.push_back(count[i - 1] + count[i - 2] + p2);
}
```

I use a `for` loop to get the rest of $Count(i)$. `a` is for memorizing where we start to use the equation $Count(i) = Count(i-1) + Count(i-2) + Intersection$ to solve the problem. `if((i - a) % 2 == 0)` is used for know $p1$ or $p2$ to use.

Output

```
cout << "Case " << cnt << ": " << count[n] << endl;
```

`cnt` is the variable I use to count for the number of cases, it'll be add up every time the input loop starts. Finally, we'll have `count[n]` which is the number of occurences of p in $F(n)$.

Difficulty

1. Wrote Programs less lately ->
   At first, I want to use pointer and write another function to construct the current `Ftable` ;

however, I found out that perhaps is because that I've not written c++ programs lots lately, I kept making wrong and couldn't get any output I'd like to get. Therefore, I gave up those thoughts and just put the rest function into the main function.
It also reminds me to spend some time reviewing programming design.

2. The bug I got stucked ->
   `string s2 = Ftable[count.size() - 2].substr(Ftable[count.size() - 2].length() - m + 1) + Ftable[count.size() - 1].substr(0, m - 1);`
   I spent a lot of time debuging and find out that at first I typed wrong for the $head$ to add. I use `Ftable[count.size() - 3]` to cut for the substring to be the $head$, but it'll be wrong if $F(count.size() - 3)$ is not long enough. As mentioned in the Step 2, we can make sure `Ftable[count.size() - 2]` must be longer than `m`; however, we cannot know if `Ftable[count.size() - 3]` is long enough to get a substring of length `m - 1`. Hence, after correcting this tiny small wrong, my code finally got accepted on online judger.

Special Thanks

- My high school classmate Yi Hung who's currently studying at Dept. of CSIE in NTU provided some hints to me and stayed up late with me when I'm debugging.
- My classmates in IM discussed and shared their observations in this problem made me have more ideas towards this problem.