Báo cáo bài tập lớn môn học
Thuật toán ứng dụng
Đề tài: Quy hoạch động

Họ tên sinh viên: Phan Việt Hoàng
MSSV: 20180086
Lớp: KSTN-CNTT K63
Viện: Công nghệ thông tin và Truyền thông

Ngày 28 tháng 12 năm 2020

Nội dung

1. Giới thiệu đề tài
2. Phát biểu bài toán
   - English
   - Tiếng việt
3. Tóm tắt lý thuyết phương pháp quy hoạch động
   - Phương pháp chia để trị
   - Phương pháp quy hoạch động
   - Các bước giải toán tối ưu bằng quy hoạch động
4. Trình bày thuật toán giải
   - Đề xuất thuật toán
   - Code sample
5. Tài liệu tham khảo

Giới thiệu đề tài

Trong ngành khoa học máy tính, quy hoạch động (tiếng Anh: dynamic programming) là một phương pháp giảm thời gian chạy của các thuật toán thể hiện các tính chất của các bài toán con gối nhau (overlapping subproblem) và cấu trúc con tối ưu (optimal substructure).

Giới thiệu đề tài

Cấu trúc con tối ưu có nghĩa là các lời giải tối ưu cho các bài toán con có thể được sử dụng để tìm các lời giải tối ưu cho bài toán toàn cục. Ví dụ, đường đi ngắn nhất tới một đỉnh trong một đồ thị có thể được tìm thấy bằng cách: trước hết tính đường đi ngắn nhất tới đích từ tất cả các đỉnh kề nó, rồi dùng kết quả này để chọn đường đi toàn cục tốt nhất, như trong hình 1. Nói chung, ta có thể giải một bài toán với cấu trúc con tối ưu bằng một quy trình ba bước:

- Chia bài toán thành các bài toán con nhỏ hơn.
- Giải các bài toán này một cách tối ưu bằng cách sử dụng đệ quy quy trình ba bước này.
- Sử dụng các kết quả tối ưu đó để xây dựng một lời giải tối ưu cho bài toán ban đầu.

Giới thiệu đề tài

Các bài toán con được giải bằng cách chia chúng thành các bài toán nhỏ hơn, và cứ tiếp tục như thế, cho đến khi ta đến được trường hợp đơn giản dễ tìm lời giải. Nói rằng một bài toán có các bài toán con trùng nhau có nghĩa là mỗi bài toán con đó được sử dụng để giải nhiều bài toán lớn hơn khác nhau Điều này áp dụng mỗi khi có mặt các bài toán con gối nhau: một cách tiếp cận ngây thơ có thể tốn thời gian tính toán lại lời giải tối ưu cho các bài toán con mà nó đã giải.
Để tránh việc đó, ta lưu trữ lời giải của các bài toán con đã giải. Do vậy, nếu sau này ta cần giải lại chính bài toán đó, ta có thể lấy và sử dụng kết quả đã được tính toán. Hướng tiếp cận này được gọi là lưu trữ (trong tiếng Anh được gọi là memoization, không phải memorization, dù từ này cũng hợp nghĩa). Nếu ta chắc chắn rằng một lời giải nào đó không còn cần thiết nữa, ta có thể xóa nó đi để tiết kiệm không gian bộ nhớ. Trong một số trường hợp, ta còn có thể tính lời giải cho các bài toán con mà ta biết trước rằng sẽ cần đến.

Phát biểu bài toán

English

In Python, code blocks don't have explicit begin/end or curly braces to mark beginning and end of the block. Instead, code blocks are defined by indentation.
We will consider an extremely simplified subset of Python with only two types of statements.
Simple statements are written in a single line, one per line. An example of a simple statement is assignment.
For statements are compound statements: they contain one or several other statements. For statement consists of a header written in a separate line which starts with "for" prefix, and loop body. Loop body is a block of statements indented one level further than the header of the loop. Loop body can contain both types of statements. Loop body can't be empty.

Phát biểu bài toán

You are given a sequence of statements without indentation. Find the number of ways in which the statements can be indented to form a valid Python program.

Input

The first line contains a single integer N $1 \leq N \leq 5000$ - the number of commands in the program. N lines of the program follow, each line describing a single command. Each command is either "f" (denoting "for statement") or "s" ("simple statement"). It is guaranteed that the last line is a simple statement.

Output

Output one line containing an integer - the number of ways the given sequence of statements can be indented modulo $10^9 + 7$.

Phát biểu bài toán

Tiếng việt

Trong Python, các khối mã không có dấu ngoặc nhọn bắt đầu / kết thúc hoặc dấu ngoặc nhọn rõ ràng để đánh dấu phần đầu và phần cuối của khối. Thay vào đó, các khối mã được xác định bằng cách thụt lề.
Chúng ta sẽ xem xét một tập hợp con Python cực kỳ đơn giản với chỉ hai loại câu lệnh.
Các câu lệnh đơn giản được viết trên một dòng, mỗi câu một dòng. Một ví dụ về một câu lệnh đơn giản là phép gán.
Đối với câu lệnh là câu lệnh ghép: chúng chứa một hoặc một số câu lệnh khác. Câu lệnh For bao gồm một tiêu đề được viết trong một dòng riêng biệt bắt đầu bằng tiền tố "for" và nội dung vòng lặp. Nội dung vòng lặp là một khối các câu lệnh được thụt vào một cấp so với tiêu đề của vòng lặp. Nội dung vòng lặp có thể chứa cả hai loại câu lệnh. Phần nội dung vòng lặp không được để trống.

Phát biểu bài toán

Bạn được cung cấp một chuỗi các câu lệnh không có thụt đầu dòng. Tìm số cách mà các câu lệnh có thể được thụt lề để tạo thành một chương trình Python hợp lệ.

Input

Dòng đầu tiên chứa một số nguyên N $1 \leq N \leq 5000$ - số lượng lệnh trong chương trình. N dòng của chương trình tiếp theo, mỗi dòng mô tả một lệnh duy nhất. Mỗi lệnh là "f" (biểu thị "cho câu lệnh") hoặc "s" ("câu lệnh đơn giản"). Nó được đảm bảo rằng dòng cuối cùng là một câu lệnh đơn giản..

Output

Xuất ra một dòng chứa một số nguyên - số cách mà dãy câu lệnh đã cho có thể được thụt lề theo modulo $10^9 + 7$

Ví dụ

Examples

input
4 s f f s
output
1

input
4 f s f s
output
2

Tóm tắt lý thuyết phương pháp quy hoạch động

- Phương pháp chia để trị
  - "Chia để trị" là việc tách bài toán ban đầu thành các bài toán con độc lập, sau đó giải các bài toán con này rồi tổ hợp dần lời giải từ bài toán con nhỏ nhất đến bài toán ban đầu. Phương pháp chia để trị là phương pháp thông dụng nhất trong Tin học.
  - Phương pháp chia để trị thường được áp dụng cho những bài toán có bản chất đệ quy (bài toán P có bản chất đệ quy thì bài toán P có thể được giải bằng lời giải của bài toán P' có dạng giống như P. Tuy nhiên, chúng ta cần lưu ý rằng: P' tuy có dạng giống như P nhưng theo một nghĩa nào đó P' phải nhỏ hơn P, dễ giải hơn P và việc giải nó không cần dùng đến P).
  - Giải thuật dùng để giải bài toán có bản chất đệ quy gọi là giải thuật đệ quy.

Tóm tắt lý thuyết phương pháp quy hoạch động

- Phương pháp quy hoạch động
  - Phương pháp quy hoạch động (Dynamic Programming) là một kỹ thuật nhằm đơn giản hóa việc tính toán các công thức truy hồi bằng cách lưu toàn bộ hay một phần kết quả tính toán tại mỗi bước trước đó với mục đích sử dụng lại.
  - Như vậy, Quy hoạch động = Chia để trị + Mảng (lưu lại kết quả).
  - Quy hoạch động dùng để giải quyết bài toán tối ưu theo nguyên lý "chia để trị" nhưng thực chất là một phương pháp cải tiến hơn của phương pháp giải quyết bài toán theo hướng đệ quy. Quy hoạch động làm giảm độ phức tạp, giảm thời gian giải quyết bài toán
  - Quy hoạch động thường tiếp cận theo hướng từ dưới lên (Bottom – up)

Tóm tắt lý thuyết phương pháp quy hoạch động

- Các bước giải toán tối ưu bằng quy hoạch động
  - Lập công thức truy hồi .
  - Tổ chức dữ liệu và chương trình
  - Truy vết, tìm nghiệm của bài toán

Trình bày thuật toán giải

- Đề xuất thuật toán.
  - Đặt dp[i][j] là số chương trình lùi dòng j lần ở dòng lệnh thứ i
  - Khi đó, đáp số của bài toán là:
    $\sum_{k=0}^{n-1}(d[n][k])$
  - Khi ký tự thứ i là f thì dòng thứ i+1 chắc chắn phải thụt dòng
    => d[i+1][j+1] = d[i][j].
  - Khi ký tự thứ i là s thì dòng thứ i+1 chắc chắn phải thụt dòng ít hơn hoặc bằng ở dòng thứ i
    => d[i+1][j]= $\sum_{k=j}^{n-1}(d[i][k])$
  - Độ phức tạp của thuật toán do đó là $O(n^2)$

Lời giải

```
#include <bits/stdc++.h>
using namespace std;

const int mx = 6e3;
long long mod = 1e9 + 7;
long long dp[mx][mx];

int main()
{
    int n, i, j;
    long long sum;
    char c;
    cin >> n;
    getchar();
    dp[1][0] = 1;
```

Code sample

```
for (i = 1; i <= n; i++)
{
    cin >> c;
    if (c == 'f')
    {
        for (j = 0; j < n; j++)
        {
            dp[i + 1][j + 1] = dp[i][j];
        }
    }
    else
    {
        sum = 0;
        for (j = n - 1; j >= 0; j--)
        {
            sum = (sum + dp[i][j]) % mod;
            dp[i + 1][j] = sum;
        }
    }
}
```

Lời giải

```
    sum = 0;
    for (j = 0; j < n; j++)
        sum = (sum + dp[n][j]) % mod;
    cout << sum << endl;
}
```

Tài liệu tham khảo

- https://vi.wikipedia.org/wiki/Quy_hoạch_động
- http://tailieuso.udn.vn/bitstream/TTHL_125/4481/3/Tomtat.pdf
- https://www.programmersought.com/article/27356537272/