# 题目

你有一个只支持单个标签页的 浏览器 ，最开始你浏览的网页是 homepage ，你可以访问其他的网站 url ，也可以在浏览历史中后退 steps 步或前进 steps 步。

请你实现 BrowserHistory 类：

BrowserHistory(string homepage) ，用 homepage 初始化浏览器类。

void visit(string url) 从当前页跳转访问 url 对应的页面  。执行此操作会把浏览历史前进的记录全部删除。

string back(int steps) 在浏览历史中后退 steps 步。如果你只能在浏览历史中后退至多 x 步且 steps > x ，那么你只后退 x 步。请返回后退 至多 steps 步以后的 url 。

string forward(int steps) 在浏览历史中前进 steps 步。如果你只能在浏览历史中前进至多 x 步且 steps > x ，那么你只前进 x 步。请返回前进 至多 steps步以后的url 。

示例：

输入：

["BrowserHistory","visit","visit","visit","back","back","forward","visit","forward","back","back"]

[["leetcode.com"],["google.com"],["facebook.com"],["youtube.com"],[1],[1],[1],["linkedin.com"],[2],[2],[7]]

输出：

[null,null,null,null,"facebook.com","google.com","facebook.com",null,"linkedin.com","google.com","leetcode.com"]

解释：

BrowserHistory browserHistory = new BrowserHistory("leetcode.com");

browserHistory.visit("google.com"); // 你原本在浏览 "leetcode.com" 。访问 "google.com"

browserHistory.visit("facebook.com"); // 你原本在浏览 "google.com" 。访问 "facebook.com"

browserHistory.visit("youtube.com"); // 你原本在浏览 "facebook.com" 。访问 "youtube.com"

browserHistory.back(1); // 你原本在浏览 "youtube.com" ，后退到 "facebook.com" 并返回 "facebook.com"

browserHistory.back(1); // 你原本在浏览 "facebook.com" ，后退到 "google.com" 并返回 "google.com"

browserHistory.forward(1); // 你原本在浏览 "google.com" ，前进到 "facebook.com" 并返回 "facebook.com"

browserHistory.visit("linkedin.com"); // 你原本在浏览 "facebook.com" 。 访问 "linkedin.com"

browserHistory.forward(2); // 你原本在浏览 "linkedin.com" ，你无法前进任何步数。

browserHistory.back(2); // 你原本在浏览 "linkedin.com" ，后退两步依次先到 "facebook.com" ，然后到 "google.com" ，并返回 "google.com"

browserHistory.back(7); // 你原本在浏览 "google.com"， 你只能后退一步到 "leetcode.com" ，并返回 "leetcode.com"

提示：

1 <= homepage.length <= 20

1 <= url.length <= 20

1 <= steps <= 100

homepage和 url 都只包含 '.' 或者小写英文字母。

最多调用 5000 次 visit， back 和 forward 函数。

# 分析

## 方法一：栈

**思路：**

使用一个栈记录浏览历史，使用一个pos记录当前网页在栈中的位置。每次back和forward操作都只更新pos。因为visit操作会把浏览历史前进的记录全部删除，所以每次visit先根据pos更新下栈顶指针，然后再将url入栈。

**代码：**

class BrowserHistory {

public:

int pos;

int top;

string history[5001];

BrowserHistory(string homepage) : pos(-1), top(0) {

visit(homepage);

}

void visit(string url) {

pos ++;

top = pos;

history[top++] = url;

}

string back(int steps) {

if(steps > pos) {

steps = pos;

}

pos -= steps;

return history[pos];

}

string forward(int steps) {

steps = min(steps, top - pos - 1);

pos += steps;

return history[pos];

}

};

或：

class BrowserHistory {

public:

BrowserHistory(string homepage) {

history.push(homepage);

current = history.top();

}

void visit(string url) {

// 当访问新页面时，清空当前页面之后的历史记录

while (!future.empty()) {

future.pop();

}

history.push(url);

current = history.top();

}

string back(int steps) {

while (steps > 0 && history.size() > 1) {

future.push(history.top());

history.pop();

current = history.top();

steps--;

}

return current;

}

string forward(int steps) {

while (steps > 0 && !future.empty()) {

history.push(future.top());

future.pop();

current = history.top();

steps--;

}

return current;

}

private:

stack<string> history; // 历史记录栈

stack<string> future; // 前进记录栈

string current; // 当前页面

};

上述性能比较差，继续优化：

class BrowserHistory {

public:

BrowserHistory(string homepage) {

history.push\_back(homepage);

current = 0;

}

void visit(string url) {

// 当访问新页面时，直接更新当前位置

history.resize(current + 1);

history.push\_back(url);

current = history.size() - 1;

}

string back(int steps) {

current = max(0, current - steps);

return history[current];

}

string forward(int steps) {

current = min(static\_cast<int>(history.size()) - 1, current + steps);

return history[current];

}

private:

vector<string> history; // 历史记录数组

int current; // 当前位置

};

时间复杂度：O(n)，n为操作次数

## 方法二：双向链表