# 题目

给你一个字符串 path ，表示指向某一文件或目录的 Unix 风格 绝对路径 （以 '/' 开头），请你将其转化为 更加简洁的规范路径。

在 Unix 风格的文件系统中规则如下：

一个点 '.' 表示当前目录本身。

此外，两个点 '..' 表示将目录切换到上一级（指向父目录）。

任意多个连续的斜杠（即，'//' 或 '///'）都被视为单个斜杠 '/'。

任何其他格式的点（例如，'...' 或 '....'）均被视为有效的文件/目录名称。

返回的 简化路径 必须遵循下述格式：

始终以斜杠 '/' 开头。

两个目录名之间必须只有一个斜杠 '/' 。

最后一个目录名（如果存在）不能 以 '/' 结尾。

此外，路径仅包含从根目录到目标文件或目录的路径上的目录（即，不含 '.' 或 '..'）。

返回简化后得到的 规范路径 。

示例 1：

输入：path = "/home/"

输出："/home"

解释：

应删除尾随斜杠。

示例 2：

输入：path = "/home//foo/"

输出："/home/foo"

解释：

多个连续的斜杠被单个斜杠替换。

示例 3：

输入：path = "/home/user/Documents/../Pictures"

输出："/home/user/Pictures"

解释：

两个点 ".." 表示上一级目录（父目录）。

示例 4：

输入：path = "/../"

输出："/"

解释：

不可能从根目录上升一级目录。

示例 5：

输入：path = "/.../a/../b/c/../d/./"

输出："/.../b/d"

解释：

"..." 在这个问题中是一个合法的目录名。

提示：

1 <= path.length <= 3000

path 由英文字母，数字，'.'，'/' 或 '\_' 组成。

path 是一个有效的 Unix 风格绝对路径。

# 分析

要解决“简化Unix风格绝对路径”的问题，核心思路是利用栈模拟路径的导航过程，按规则处理.、..和连续斜杠，最终拼接出规范路径。

解题思路

1、拆分路径为目录组件：

Unix路径以`/`分隔目录，首先需要将路径按`/`拆分为多个组件（如/home//foo/拆分为["home", "foo"]），过滤掉空字符串（由连续`/`产生）。

2、用栈处理目录导航：

- 遇到`.`：表示当前目录，不做处理（不压栈）。

- 遇到`..`：表示上一级目录，若栈非空则弹出栈顶元素（返回父目录）；若栈为空（已在根目录），则不做处理。

- 其他组件：视为有效目录名，压入栈中。

3、拼接栈中元素为规范路径：

- 若栈为空，直接返回"/"。

- 否则，依次弹出栈中元素，用"/"连接（注意栈是后进先出，需逆序拼接），最终结果以"/"开头，且目录间仅一个"/"，结尾无"/"。

代码：

class Solution {

public:

string simplifyPath(string path) {

vector<string> stack; // 用vector模拟栈，存储有效目录名

stringstream ss(path); // 字符串流用于拆分路径

string component; // 存储拆分后的单个目录组件

// 按'/'拆分路径，遍历所有组件

while (getline(ss, component, '/')) {

// 过滤空字符串（连续'/'产生）和当前目录'.'

if (component.empty() || component == ".") {

continue;

}

// 处理上一级目录'..'：栈非空则弹出栈顶

if (component == "..") {

if (!stack.empty()) {

stack.pop\_back();

}

} else {

// 有效目录名，压入栈

stack.push\_back(component);

}

}

// 拼接栈中元素为规范路径

if (stack.empty()) {

return "/"; // 栈为空，返回根目录

}

string result;

// 栈中元素是从根到目标的目录，需用'/'连接

for (const string& dir : stack) {

result += "/" + dir;

}

return result;

}

};

代码解释

1、路径拆分：

使用stringstream和getline(ss, component, '/')按'/'拆分路径。例如，"/home//foo/"会被拆分为["", "home", "", "foo", ""]，空字符串由连续'/'或首尾'/'产生。

2、栈操作逻辑：

- 空字符串和"."直接跳过，不影响路径。

- ".."触发“回退”操作：若栈中有目录（非根目录），则弹出栈顶（返回上一级）；若栈为空（已在根目录），则不操作（根目录无父目录）。

- 其他组件（如"home"、"..."）均为有效目录，压入栈中。

3、规范路径拼接：

- 栈为空时，返回"/"（根目录）。

- 栈非空时，遍历栈中元素，每个元素前加'/'，拼接成"/dir1/dir2/..."的形式，满足“以'/'开头、目录间单'/'、无尾'/'”的规范。

复杂度分析

- 时间复杂度：O(n)，其中n是路径长度。拆分路径、处理栈操作、拼接结果均需遍历路径中的每个字符，整体为线性时间。

- 空间复杂度：O(n)。栈最多存储所有有效目录名，最坏情况下（路径无`.`和`..`）空间与路径长度成正比。