

Permutation

时刻限制: 1×10^{10} 刻

内存限制: 256 字节

注意本题不同寻常的内存限制.

一个排列 p 是一个包含从 1 到 n 的每个整数恰好一次的序列.

对于一个排列 $p = (p_1, p_2, \dots, p_n)$, 它的逆排列 $q = (q_1, q_2, \dots, q_n)$ 定义为: 若 $p_i = j$, 则 $q_j = i$.

在本题中, 你将获得一个 1 到 n 的排列. 你的任务是计算它的逆排列.

实现方式

与常规的题目不同, 你不需要编写一个完整的程序 (即不需要实现 `main` 函数). 取而代之的是, 你需要在一个 C/C++ 源文件中包含头文件 `perm.h`, 并实现以下函数:

```
void perm_work(int n);
```

评测系统将会调用你实现的 `perm_work` 函数恰好一次. 参数 n 是排列的大小.

你无法直接访问存储排列的数组. 你需要通过以下由评测系统提供的函数来与排列进行交互:

- `int perm_get(int x)`: 获取排列中第 x 个元素的值, 即 p_x . 这里的 x 是基于 1 的索引, 满足 $1 \leq x \leq n$. 如果 x 超出范围, 函数将返回 -1 .
- `void perm_set(int x, int y)`: 将排列中第 x 个元素的值设置为 y , 即令 $p_x = y$. 这里的 x 是基于 1 的索引. 你需要保证 $1 \leq x, y \leq n$, 否则函数将无效果.

你的 `perm_work` 函数应该通过调用 `perm_get` 和 `perm_set` 来就地修改排列, 使其变为其自身的逆排列.

特别地, 你提交的程序第一行可以包含一个形如 `/* stack = S */` 的注释, 用于指定程序栈的大小. S 应当是一个正整数, 表示程序栈的大小 (以字节为单位). 如不指定, 评测系统将使用默认值 128 字节.

环境限制

你的程序能使用的总线性内存 **只有 256 字节**. 你可以将其理解为你 C/C++ 代码能直接访问的全部内存空间. 这块内存被划分为以下几个部分:

这 256 字节的内存被划分为几个部分:

- **程序栈**: S 字节. 这部分内存由编译器在线性内存中划出, 用于存储函数调用时的局部变量、返回地址等. 例如, 你在 `perm_work` 函数中声明的 `int i;` 就会占用这部分空间. 这与你通常理解的 C/C++ 程序中的栈是同一个概念.
- **静态空间**: $256 - 16 - S$ 字节. 这部分空间用于存储程序中所有的全局变量、静态变量和常量 (例如字符串字面量). 所有定义在函数之外的变量都将占用这部分空间.
- **系统保留**: 16 字节.

除了上述线性内存外, WebAssembly 运行时本身还需要一个独立的执行栈 (Runtime Stack) 来管理程序的执行流程. 这个栈不在线性内存中, 选手无法直接访问. 它的主要作用是处理 WebAssembly 指令层面的函数调用.

- **执行栈**: 本题中其大小设置为 1024 字节. 对于大多数程序, 你并不**需要** 关心执行栈的占用情况. 你主要需要关心的是「程序栈」是否会溢出. 但对于极深层次的函数递归, 也有可能耗尽执行栈空间, 从而导致运行时错误.

为了保证程序可在严苛的内存限制下编译运行, 本题存在一些特殊限制:

- **禁止使用标准库**: 你 **不能** 使用任何 C/C++ 标准库.
- **禁用部分 C++ 特性**: 对于 C++ 选手, 禁用标准库也意味着 `new`, `delete` 操作符以及异常处理 (exceptions) 功能都无法使用.

评测方式

你的 C/C++ 源代码将被编译成一种名为 WebAssembly (WASM) 的格式. 评测系统会执行这个 WASM 模块并与之交互. 测评时仅会计算你编写的程序的 WASM Tick 和内存, 而不会计算交互库的 WASM Tick 和内存.

为了兼容 UOJ 的 API, 本题测评时, 时间 (毫秒) 显示为 WASM Tick 数除以 10^6 下取整. 空间实际的 1 byte 显示为 1 kb (UOJ 上的「kb」实际上是 KiB, 故相当于以 1024 倍显示).

自测方式

为方便选手做题, 本题提供两种自测方式:

你可以使用 UOJ 的「自定义测试」功能来测试你的代码. 自定义测试输入格式为: 第一行一个整数 n , 接下来一行 n 个整数 p_1, p_2, \dots, p_n . 运行输出为修改后的排列. 自测会按照与实际评测相同的流程进行, 只是不会检查答案的正确性.

你也可以在本地编译运行你的程序. 下载本题的附件包, 里面包含 `perm.h` 和 `user_grader.c` 两个文件, 其中 `user_grader.c` 是一个同时符合 C 和 C++ 语言标准的程序. 把这两个文件和你的代码放在同一个目录下, 使用 `<gcc | g++> 你的代码 user_grader.c -o prog` 编译, 之后运行 `./prog` (或 `prog.exe`), 然后按照上文「自定义测试」所用的输入格式输入数据即可. 注意使用这种方法测试时无法得到 WASM Tick 数, 也无法得到准确的内存使用量.

样例 1

input	output
5 3 1 4 2 5	2 4 1 3 5

样例 2

input	output
10 6 7 9 8 10 4 3 2 5 1	10 8 7 6 9 1 2 4 3 5

样例 3

input	output
10 1 2 3 4 5 6 7 8 9 10	1 2 3 4 5 6 7 8 9 10

样例 4

input	output
10 3 9 6 10 7 1 2 8 5 4	6 7 1 10 9 3 5 8 2 4

样例 5

input	output
10 10 2 7 4 5 6 8 3 9 1	10 2 8 4 5 6 3 7 9 1

特征

- **n_le_10**: $n \leq 10$.
- **n_le_1e4**: $n \leq 10^4$.

子任务

#	分数	n_le_10	n_le_1e4
1	0.1	✓	?
2	0.5	?	✓
3	0.4	?	?

说明

限制与约定

$1 \leq n \leq 3 \times 10^5$.

WASM 时刻限制： 10^{10} 刻.

空间限制：256 字节.

编译器信息

- 编译器套件: wasi-sdk 28_judge.0 (基于 LLVM/Clang 21.1.4).
- 详细版本信息:
 - wasi-libc: 47e504e7c024.
 - llvm: 222fc11f2b8f.
 - config: f992bcc08219.
- 更多信息可参考: <https://github.com/aberter0x3f/wasi-sdk>.

编译参数

你的代码将使用以下关键参数进行编译:

- -nostdlib, -nostdinc, -ffreestanding: 禁用标准库.
- -fno-exceptions: 禁用 C++ 异常处理.
- -O0: 禁用优化.
- -Wl, -z, stack-size=S: 设置栈大小为 S 字节.
- -Wl, --max-memory=256: 设置总可用线性内存为 256 字节.
- -Wl, --export=perm_work: 使你的 perm_work 函数对评测系统可见.

- `-Wl,--no-entry`: 表明程序没有 `main` 入口点.