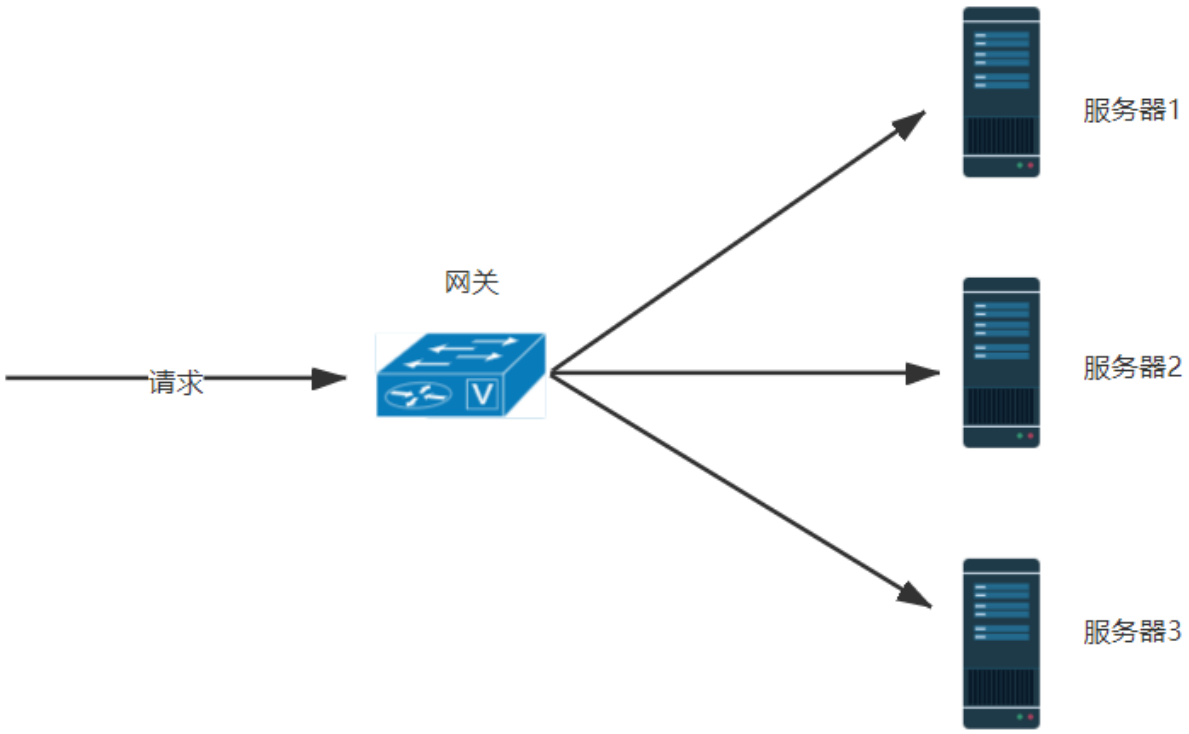
负载均衡

实现一个简单的负载均衡场景。如今的互联网公司都追求服务的高可用，所以一般一个服务会有不止一个服务器。假设某个服务有N个服务器，每个到达网关的请求都会按照一定的负载均衡算法被分配到其中的**一个**服务器上面。如下图，图中服务有三个服务器。



每个请求有如下的属性：

> tag：每个请求自身的唯一标识，string类型

每个服务器有如下的属性：

> id：服务器本身的编号，是从1到N的连续正整数

请求分配的**策略**有如下几种：

> 1: 按照从小到大，如此反复的顺序。假设N为6，则请求分配的服务器顺序为
>
> `1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6...`
>
> 2: 按照从小到大再从大到小，如此反复的顺序。假设N为6，则请求分配的服务器顺序为
>
> `1, 2, 3, 4, 5, 6, 5, 4, 3, 2, 1, 2, 3...`
>
> 3: 按照先奇数从小到大，再偶数从小到大，如此反复的顺序。假设N为6，则请求分配的服务器顺序为

> 1，3，5，2，4，6，1，3，5...
>
> 4：按照服务器的权重分配。假设N为3，id为1，2，3的服务器权重分别为3，2，1，则请求分配的服务器顺序为
>
> 1，1，1，2，2，3，1，1，1，2，2，3...

输入描述

输入的第一行为三个整数 M, N, S,空格隔开，表示有 M个请求, N个服务器,策略S可能取值为1，2，3，4,分别对应于四种策略

其中第四种策略会在第二行给出N个正整数，为a_1到a_N，空格隔开，分别对应id从1到N的服务器的权重

接下来M行每行一个字符串，表示每个请求的`tag`

输出描述

按照请求的顺序输出每个请求落在哪个服务器上。每个请求输出一行，输出两个值，分别为请求`tag`和服务器`id`

说明

$M \geq 0$,$N > 0$,`tag`为`string`类型，第四种策略中权重必为正整数

必须使用面向对象的实现方案，头文件与`cpp`文件必须分离，违者扣分

示例

```
输入:
3 2 1
a
bb
ccc

输出:
a 1
bb 2
ccc 1

解释:
三个请求分配到两个服务器，策略为第一种
按照1，2，1的顺序分配
```

```
输入:
8 3 4
3 2 1
aaaa
b
c
d
eeee
f
g
h

输出:
aaaa 1
b 1
c 1
d 2
eeee 2
f 3
g 1
h 1

解释:
八个请求分配到三个服务器，策略为第四种，服务器权值为3，2，1
分配按照1，1，1，2，2，3的顺序
```