机密 ★ 启用前

2024 周期测试

测试时长：150 分钟

题目名称	求和	组队出行	选修课	城市规划	氪佬的胜利
题目类型	传统型	传统型	传统型	传统型	传统型
输入文件名	sum.in	group.in	lesson.in	city.in	skin.in
输出文件名	sum.out	group.out	lesson.out	city.out	skin.out
每个测试点时限	1.0 秒	1.0 秒	1.0 秒	1.0 秒	2.0 秒
内存限制	128 MiB	128 MiB	256 MiB	256 MiB	256 MiB
测试点数目	20	20	20	20	10
测试点是否等分	是	是	是	是	是

提交源程序文件名

对于 C++ 语言	sum.cpp	group.cpp	lesson.cpp	city.cpp	skin.cpp

编译选项

对于 C++ 语言	-O2 -std=c++14 -static

注意事项：

1. 本场考试只允许使用 C++ 语言。

2. 文件名 (程序名和输入输出文件名) 必须严格按照题目要求。

3. C++ 中函数 main() 的返回值类型必须是 int，程序正常结束时的返回值必须是 0。

4. 在机房考试的同学首先在**不会重置的盘符**上建一个考试文件夹，文件夹名称为**姓名**，并将所有的程序文件放到该文件夹内。

5. 因违反以上四点而出现的错误或问题，申诉时一律不受理。

6. 若无特殊说明，结果的比较方式为全文比较 (即过滤行末空格及文末回车)。

7. 考试结束时**将文件夹压缩**后提交到教师机上，选手需注意自己提交的程序文件大小并与监考人员确认，确认无误后方可离开。

8. 程序可使用的栈空间内存限制与题目的内存限制一致。

9. 若无特殊说明，题目中一行内有多个输入或输出时，默认使用空格间隔。

求和（sum）

【题目描述】

给你两个二进制字符串 A 和 B ，以二进制字符串的形式返回它们的和。

【输入格式】

从文件 `sum.in` 读入数据。

输入的第一行包含 2 个字符串 A 和 B ，代表等待求和的两个二进制数；

【输出格式】

输出到文件 `sum.out` 中。

输出一行，包含 1 个字符串，代表两个数和的二进制形式。

【样例 1 输入】

```
11 1
```

【样例 1 输出】

```
100
```

【样例 2 输入】

```
1010 1011
```

【样例 2 输出】

```
10101
```

【数据范围】

- 对于 30% 的数据：
  - $1 \leqslant A$ 和 B 的长度 $\leqslant 50$；
- 对于 100% 的数据：
  - $1 \leqslant A$ 和 B 的长度 $\leqslant 10^4$；
- A 和 B 仅由字符 `0` 或 `1` 组成；
- 字符串如果不是”0”，就不含前导零。

组队出行（group）

【题目描述】

春天到了，W 老师准备带各位同学参加春游，但 W 老师很担心班上 N 名同学的安全，叮嘱大家一定要结伴出行，班上同学听到后决定跟自己的小伙伴组队。

现在有 N 名同学，有的已经结伴，有的还未结伴，如果 a 同学与 b 同学已经结伴，b 同学与 c 同学已经结伴，那么我们认为 a 同学和 c 同学也已经结伴，他们三人属于同一个队伍。

W 老师用一个邻接矩阵 `G[N][N]` 来保存班上的组队情况，他希望知道现在班上已经存在的队伍数量。

【输入格式】

从文件 `group.in` 读入数据。

输入的第一行包含一个整数 N，代表班上一共有 N 个同学；

接下来 N 行，每行包含 N 个整数，由 0 和 1 构成，用来构成邻接矩阵。

【输出格式】

输出到文件 `group.out` 中。

输出一行，包含一个整数，代表班上已有的队伍数量。

【样例 1 输入】

```
3
1 1 0
1 1 0
0 0 1
```

【样例 1 输出】

```
2
```

【样例 1 解释】

连接如图所示，一共有两个队伍。

【样例 2 输入】

```
4
1 0 1 0
0 1 0 0
1 0 1 0
0 0 0 1
```

【样例 2 输出】

```
3
```

【样例 2 解释】

连接如图所示，一共有三个队伍。

【数据范围】

- `G[i][j] == 1`代表 i 同学和 j 同学已经结伴，`G[i][j] == 0`代表 i 同学和 j 同学没有结伴；
- `G[i][j] == G[j][i]`；
- `G[i][i] == 1`；
- 对于 30% 的数据：
  - $1 \leqslant N \leqslant 20$；
- 对于 100% 的数据：
  - $1 \leqslant N \leqslant 200$。

选修课（lesson）

【题目描述】

新的学期到了，你这个学期必须选修 N 门课程，每科课程的序号为 0 到 N - 1 。

这些课程中有部分课程存在先修课。在选修某些课程之前需要一些先修课程。先修课程以一对整数 A B 给出，表示如果要学习课程 A 则**必须**先学习课程 B。例如，先修课程 0 1 表示：想要学习课程 0 ，你需要先完成课程 1 。

请你判断是否可能完成所有课程的学习？如果可以，返回 Yes ；否则，返回 No 。

【输入格式】

从文件 lesson.in 读入数据。

输入的第一行包含 2 个整数 N 和 M，分别代表要选修的课程数和存在先修课程的课程对数；

接下来 M 行，每行包含 2 个整数 A 和 B，代表要学习课程 A 则**必须**先学习课程 B 。

【输出格式】

输出到文件 lesson.out 中。

输出一行，包含一个字符串。

【样例 1 输入】

```
2 1
1 0
```

【样例 1 输出】

```
Yes
```

【样例 1 解释】

总共有 2 门课程。学习课程 1 之前，你需要完成课程 0 。

由于课程 0 没有先修课，所以这是可行的。

【样例 2 输入】

```
2 2
1 0
0 1
```

【样例 2 输出】

```
No
```

【数据范围】

- 对于 30% 的数据：
  - $1 \leqslant N \leqslant 100$；
- 对于 100% 的数据：
  - $1 \leqslant N \leqslant 2000$；
- $0 \leqslant M \leqslant N$
- $0 \leqslant$ `A,B` $\leqslant N-1$；
- 所有课程对 **互不相同**。

城市规划（city）

【题目描述】

某国有 N 个城市，为了使得城市间的交通更便利，该国打算在城市之间修一些高速公路，由于经费限制，这个国家打算第一阶段先在部分城市之间修一些**单向**的高速公路。

现在，该国收到了一个修高速公路的计划。

看了计划后，该国发现，有些城市之间可以通过高速公路直接（不经过其他城市）或间接（经过一个或多个其他城市）到达，而有的却不能。

如果城市 A 可以通过高速公路到达城市 B，而且城市 B 也可以通过高速公路到达城市 A ，则这两个城市被称为便利城市对。

该国想知道，在这个计划中，有多少个便利城市对。

【输入格式】

从文件 `city.in` 读入数据。

输入的第一行包含两个整数 N, M ，分别表示城市和单向高速公路的数量;

接下来的 M 行包含每行两个整数 A, B，表示城市 a 有一条单向的高速公路连向城市 b。

【输出格式】

输出到文件 `city.out` 中。

输出一行，包含一个整数，表示便利城市对的数量。

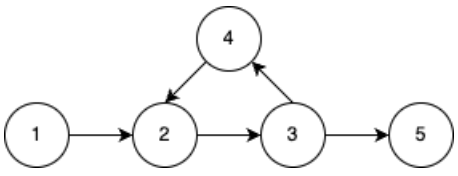
【样例 1 输入】

```
5 5
1 2
2 3
3 4
4 2
3 5
```

【样例 1 输出】

```
3
```

【样例 1 解释】



城市间的连接如图所示，有 3 个便利城市对，它们分别是 $(2,3),(2,4),(3,4)$，请注意 $(2,3)$ 和 $(3,2)$ 看成同一个便利城市对。

【数据范围】

- 对于 30% 的数据：
  - $1 \leqslant N \leqslant 100$
  - $1 \leqslant M \leqslant 1000$；
- 对于 100% 的数据：
  - $1 \leqslant N \leqslant 10^4$；
  - $1 \leqslant M \leqslant 10^5$；

氪佬的胜利（skin）

【题目描述】

时间到了 2024 年，万众瞩目的毒奶粉手游终于上市，艾坤同学迫不及待下载试玩。

经过一段时间的游玩，艾坤沉迷上了这款游戏，但他玩的很菜，为了变强了，他变强的方法就是：买时装。

艾坤球只练了 N 个职业，因此，他也只准备给这 N 个职业买时装，并且决定，以后只玩有时装的职业。

这 N 个职业中，第 i 个职业有 K_i 款时装，价格是每款 C_i `ikun` 币，同一个职业的时装价格相同。

为了让自己看起来高大上，艾坤决定给同学们展示一下自己的时装，展示的思路是这样的：对于有时装的每一个职业，随便选一个时装给同学看。

比如，艾坤共有 5 个职业，这 5 个职业分别有 $0, 0, 3, 2, 4$ 款时装，那么，艾坤就有 $3 \times 2 \times 4 = 24$ 种展示的策略。

现在，艾坤希望自己的展示策略能够至少达到 M 种，请问，艾坤至少要花多少钱呢?

【输入格式】

从文件 `skin.in` 读入数据。

输入的第一行包含两个整数 N 和 M;

输入的第二行包含 N 个整数，分别表示每个英雄的皮肤数量 K_i;

输入的第三行包含 N 个整数，分别表示每个英雄皮肤的价格 C_i。

【输出格式】

输出到文件 `skin.out` 中。

输出一行，包含一个整数，表示艾坤达到目标最少的花费。

【样例 1 输入】

```
3 24
4 4 4
2 2 2
```

【样例 1 输出】

```
18
```

【数据范围】

- 第 i 组数据满足：$\mathrm{N} \leqslant \max(5, \log_2^4 i)$
- 100% 的数据：$\mathrm{M} \leqslant 10^{17}, 1 \leqslant K_i \leqslant 10, 1 \leqslant C_i \leqslant 199$。