高斯消元与线性空间

河南省实验中学信息技术组

2025 年 07 月 11 日

线性方程组

- 由 m 个 n 元一次方程构成的方程组被称为线性方程组。例如，三元一次方程组如下：

$$\begin{cases} x_1 + 2x_2 - x_3 = -6 \\ 2x_1 + x_2 - 3x_3 = -9 \\ -x_1 - x_2 + 2x_3 = 7 \end{cases}$$

- 求解此线性方程组的方式是通过方程之间的算术运算使得方程中的未知量尽量少 (消元)，甚至只有一个未知量 (元)，然后解出未知量少的方程，将解出的未知量带入其他方程从而解出剩余未知量，这个过程就是高斯消元。
- 线性方程组的解只有三种情况：
  1. 无解；
  2. 唯一解；
  3. 无穷多解。

高斯消元法

- 在求解上述方程的过程中，你会发现我们实际参与运算的是未知量前的系数和等号右边的常数，于是我们可以将上述方程组写成一个 3 行 4 列的一个矩阵

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 2 & 1 & -3 & -9 \\ -1 & -1 & 2 & 7 \end{array}\right]$$

  这个矩阵被称为*增广矩阵*。

- 求解线性方程组的过程可以转化为对增广矩阵的操作：
  1. 用一个非零的数乘以某一行。
  2. 把其中一行的若干倍加到另一行上。
  3. 交换两行的位置。

  上述操作被称为矩阵的*初等行变换*。

高斯消元法

- 利用矩阵的初等行变换求解线性方程组的过程如下：

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 2 & 1 & -3 & -9 \\ -1 & -1 & 2 & 7 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & -3 & -1 & 3 \\ -1 & -1 & 2 & 7 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & -3 & -1 & 3 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 1 & 1 & 1 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 0 & \frac{2}{3} & 2 \end{array}\right]$$

- 最后得到的矩阵被成为阶梯形矩阵，它的系数部分被称为上三角矩阵。此矩阵等价于：

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 0 & \frac{2}{3} & 2 \end{array}\right] \Rightarrow \begin{cases} x_1 + 2x_2 - x_3 = -6 \\ x_2 + \frac{1}{3}x_3 = -1 \\ \frac{2}{3}x_3 = 2 \end{cases}$$

此时，已知最后一个未知量的值，从下向上依次代回方程组，即可以得到每个未知量的解，但是不利于算法实现。

高斯消元法

高斯消元法
河南省实验中学信息技术组
高斯消元法
算法过程
解的情况
计算机实现
例题
球形空间产生器
开关问题
坏掉的机器人
练习

- 便于实现的高斯消元过程如下：

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 2 & 1 & -3 & -9 \\ -1 & -1 & 2 & 7 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & -3 & -1 & 3 \\ 0 & 1 & 1 & 1 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -\frac{5}{3} & -4 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 0 & \frac{2}{3} & 2 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -\frac{5}{3} & -4 \\ 0 & 1 & \frac{1}{3} & -1 \\ 0 & 0 & 1 & 3 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \end{array}\right]$$

- 最后得到的矩阵被成为简化阶梯形矩阵，此矩阵等价于：

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -2 \\ 0 & 0 & 1 & 3 \end{array}\right] \Rightarrow \begin{cases} x_1 & & & = 1 \\ & x_2 & & = -2 \\ & & x_3 & = 3 \end{cases}$$

此时，增广矩阵的最后一列即为每个未知量的解。

高斯消元法

- 通过初等行变换将增广矩阵转化为简化阶梯形矩阵的线性方程组求解算法就是高斯消元法。算法实现时的具体步骤为：
  - 当前正在处理所有方程 x_i 前面的系数，找到前 $i-1$ 个值全为 0，第 i 个值不为 0 的行，将它与当前行交换 (初等行变换)。
  - 如果找不到 $x[i]$ 前的系数不为 0 的行，那么无法针对 x_i 进行消元。
  - 将当前行的的所有元素除以 $A[][i]$，也就是将 x_i 前的系数变为 1(初等行变换)。
  - 对于其他行，通过减去当前行的若干倍的方式使得 $A[][i]$ 全部变为 0(初等行变换)。
- 如果有唯一解，简化阶梯型矩阵的最后一列即为解。

无解

- 对于如下方程

$$\begin{cases} x_1 + 2x_2 - x_3 = -6 \\ 2x_1 + x_2 - 5x_3 = -9 \\ -x_1 - x_2 + 2x_3 = 7 \end{cases} \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 2 & 1 & -5 & -9 \\ -1 & -1 & 2 & 7 \end{array}\right]$$

是无解的，那进行初等行变换时会怎么样呢?

- 对上述增广矩阵进行初等行变换如下:

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 2 & 1 & -5 & -9 \\ -1 & -1 & 2 & 7 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & -3 & -3 & 3 \\ 0 & 1 & 1 & 1 \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & -6 \\ 0 & 1 & 1 & -1 \\ 0 & 1 & 1 & 1 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 \end{array}\right]$$

无解

- 上述转化过后的矩阵等价于如下线性方程

$$\left[\begin{array}{ccc|c} 1 & 0 & -3 & -4 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 2 \end{array}\right] \Rightarrow \begin{cases} x_1 \qquad\quad - 3x_3 = -4 \\ \qquad x_2 + x_3 = 1 \\ \qquad\qquad\quad 0 = 2 \end{cases}$$

- 上述方程组显然无解。

无穷多解

- 对于如下方程

$$\begin{cases} x_1 + 2x_2 - x_3 = 3 \\ 2x_1 + 4x_2 - 8x_3 = 0 \\ -x_1 - 2x_2 + 6x_3 = 2 \end{cases} \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3 \\ 2 & 4 & -8 & 0 \\ -1 & -2 & 6 & 2 \end{array}\right]$$

  有无穷多解，那进行初等行变换时会怎么样呢?
- 对上述增广矩阵进行初等行变换如下:

$$\left[\begin{array}{ccc|c} 1 & 2 & -1 & 3 \\ 2 & 4 & -8 & 0 \\ -1 & -2 & 6 & 2 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3 \\ 0 & 0 & -6 & -6 \\ 0 & 0 & 5 & 5 \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & -1 & 3 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 5 & 5 \end{array}\right] \Rightarrow \left[\begin{array}{ccc|c} 1 & 2 & 0 & 4 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right]$$

无穷多解

- 上述转化过后的矩阵等价于如下线性方程

$$\left[\begin{array}{ccc|c} 1 & 2 & 0 & 4 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{array}\right] \Rightarrow \begin{cases} x_1 + 2x_2 & = 4 \\ & x_3 = 1 \end{cases}$$

- 上述方程组的解可以写为：

$$\begin{aligned} x_1 &= 4 - 2x_2 \\ x_3 &= 1 \end{aligned}$$

此时，x_2 可以取任意值，也可以计算出对应的 x_1，也即线性方程组有无穷多解。

- 我们将 x_1, x_3 这样的未知量称为主元，把 x_2 这样的未知量称为自由元。

线性方程组解的情况

在高斯消元完成之后，根据增广矩阵的情况

- 若存在系数全为零、常数不为零的行，则方程组无解；
- 若系数不全为零的行恰好有 n 个，则说明主元有 n 个，方程组有唯一解；
- 若系数不全为零的行有 $k < n$ 个，则说明主元有 k 个，自由元有 $n-k$ 个，方程组有无穷多解。

高斯消元法的计算机实现


```cpp
// 当前正在方程 r 中 将其他方程 x[i] 前的系数消除
for(int r = 1, i = 1; i <= n; ++i)
{
    int p = 0; // 找到 x[i] 系数不为 0 的方程 将其移动到当前行 r
    for(int k = r; k <= n; ++k)  // 在方程 r~n 中找到 x[i] 前系数不为 0 的行
        if(abs(a[k][i]) >= 1e-8) { p = k; break; }
    if(p == 0) continue; // x[i] 前的系数全部为 0 无法消元
    if(p != r) for(int j = 1; j <= n + 1; ++j) swap(a[r][j], a[p][j]);
    double d = a[r][i];
    for(int j = 1; j <= n + 1; ++j) a[r][j] /= d;     // x[i] 前系数调为 1
    for(int k = 1; k <= n; ++k)    // 其他方程都消掉 x[i] 前的系数
    {
        if(r == k) continue;
        double d = a[k][i];    // 倍数
        for(int j = 1; j <= n + 1; ++j) a[k][j] -= d * a[r][j];
    }
    ++r;  // 处理下一个方程
}
```

高斯消元法的计算机实现

```
// 解的判定
int cnt = 0;
for(int i = 1; i <= n; ++i)
{
    bool all = true;   // 系数是否全为 0
    for(int j = 1; j <= n; ++j) if(abs(a[i][j]) >= 1e-8) all = false;
    if(all && a[i][n + 1] >= 1e-8) { cnt = -1; break; }    // 无解
    if(all && a[i][n + 1] < 1e-8) ++cnt;    // 无穷多解
}
if(cnt == -1) puts(" 无解! ");
else if (cnt == 0)
    for(int i = 1; i <= n; ++i) printf("x%d=%.3lf\n", i, a[i][n + 1]);
else puts(" 无穷多解! ");
```

【例】球形空间产生器

有一个球形空间产生器能够在 n 维空间中产生一个坚硬的球体。现在，你被困在了这个 n 维球体中，你只知道球面上 $n+1$ 个点的坐标，你需要以最快的速度确定这个 n 维球体的球心坐标，以便于摧毁这个球形空间产生器。数据保证有解，答案精确到小数点后三位。

- 球心：到球面任意一点距离都相等的点。
- 距离：设两个 n 为空间上的点 A, B 的坐标为 $(a_1, a_2, \cdots, a_n), (b_1, b_2, \cdots, b_n)$，则 AB 的距离定义为：$\sqrt{(a_1-b_1)^2+(a_2-b_2)^2+\cdots+(a_n-b_n)^2}$。

【例】球形空间产生器

【输入格式】

第一行一个整数 $n(n \leq 10)$。

接下来 $n+1$ 行，每行有 n 个实数，表示球面上一点的 n 维坐标，每个实数精确到小数点后 6 位，且绝对值不超过 20000。

【输出格式】

一行 n 个实数，表示球心的 n 维坐标，精确到小数点后 3 位。

数据保证一定有解。

【样例输入】

```
2
0.0 0.0
-1.0 1.0
1.0 0.0
```

【样例输出】

```
0.500 1.500
```

【例】球形空间产生器

- 一个球体上的所有点到球心的距离相等，因此只需要求出一个点 $(x_1, x_2, \cdots, x_n)$，使得

$$\sum_{j=1}^{n}(a_{i,j} - x_j)^2 = C$$

其中 C 为常数，$i \in [1, n+1]$，球面上第 i 个点的坐标为 $(a_{i,1}, a_{i,2}, \cdots, a_{i,n})$。

- 不过该方程组由 $n+1$ 个 n 元二次方程构成，我们可以通过两相邻连个方程作差将它转化为 n 个 n 元一次方程，同时消去常数 C：

$$\sum_{j=1}^{j=n}(a_{i,j}^2 - a_{i+1,j}^2 - 2x_j(a_{i,j} - a_{i+1,j})) = 0, (i = 1, 2, \cdots, n)$$

高斯消元法
算法过程
解的情况
计算机实现
例题
球形空间产生器
开关问题
坏掉的机器人
练习

【例】球形空间产生器

- 将变量放在左侧，常数放在右边：

$$\sum_{j=1}^{j=n} 2(a_{i,j} - a_{i+1,j})x_j = \sum_{j=1}^{n}(a_{i,j}^2 - a_{i+1,j}^2), (i = 1, 2, \cdots, n)$$

此时上述方程组是一个线性方程组，它的增广矩阵如下：

$$\begin{bmatrix} 2(a_{1,1} - a_{2,1}) & 2(a_{1,2} - a_{2,2}) & \cdots & 2(a_{1,n} - a_{2,n}) & \sum_{j=1}^{n}(a_{1,j}^2 - a_{2,j}^2) \\ 2(a_{2,1} - a_{3,1}) & 2(a_{2,2} - a_{3,2}) & \cdots & 2(a_{2,n} - a_{3,n}) & \sum_{j=1}^{n}(a_{2,j}^2 - a_{3,j}^2) \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 2(a_{n,1} - a_{n+1,1}) & 2(a_{n,2} - a_{n+1,2}) & \cdots & 2(a_{n1,n} - a_{n+1,n}) & \sum_{j=1}^{n}(a_{n,j}^2 - a_{n+1,j}^2) \end{bmatrix}$$

- 根据上述增广矩阵求出线性方程组的解即可。

【例】球形空间产生器


```c
int n; scanf("%d", &n);
for(int i = 1; i <= n + 1; ++i)
    for(int j = 1; j <= n; ++j)
        scanf("%lf", &a[i][j]);
//b 是增广矩阵
for(int i = 1; i <= n; ++i)
    for(int j = 1; j <= n; ++j)
    {
        b[i][j] = 2 * (a[i][j] - a[i + 1][j]);
        b[i][n + 1] += (a[i][j] * a[i][j] - a[i + 1][j] * a[i + 1][j]);
    }
```

【例】球形空间产生器

高斯消元法

河南省实验中学信息技术组

高斯消元法
- 算法过程
- 解的情况
- 计算机实现

例题
- 球形空间产生器
- 开关问题
- 坏掉的机器人

练习


```cpp
// 高斯消元
for(int r = 1, i = 1; i <= n; ++i)
{
    int p = 0;
    for(int k = r; k <= n; ++k)
        if(abs(b[k][i]) >= 1e-8) { p = k; break; }
    if(p == 0) continue;
    if(p != r) for(int j = 1; j <= n + 1; ++j) swap(b[r][j], b[p][j]);
    double d = b[r][i];
    for(int j = 1; j <= n + 1; ++j) b[r][j] /= d;
    for(int k = 1; k <= n; ++k)
    {
        if(r == k) continue;
        double d = b[k][i];
        for(int j = 1; j <= n + 1; ++j) b[k][j] -= d * b[r][j];
    }
    ++r;
}
for(int i = 1; i <= n; ++i) printf("%.3lf ", b[i][n + 1]);
```

【例】开关问题

有 n 个相同的开关，每个开关都与某些开关有着联系，每当你打开或者关闭某个开关的时候，其他的与此开关相关联的开关也会相应地发生变化，即这些相联系的开关的状态如果原来为开就变为关，如果为关就变为开。你的目标是经过若干次开关操作后使得最后 n 个开关达到一个特定的状态。对于任意一个开关，最多只能进行一次开关操作。你的任务是，计算有多少种可以达到指定状态的方法。（不计开关操作的顺序）

【输入格式】

第一行一个整数 T，表示有 T 组测试数据。
对于每组测试数据，第一行一个整数 $n(1 \leq n \leq 28)$。
接下来一行 n 个 0 或 1，表示开始时 n 个开关的状态。
接下来一行 n 个 0 或 1，表示结束时 n 个开关的状态。
接下来若干行，每行包含两个整数 x, y，表示操作开关 x，开关 y 也会变化。
每组数据以 0 0 结束。

【输出格式】

如果有可行方法，输出总数，否则输出 `Oh,it's impossible !!`。

【例】开关问题

【样例输入】

```
2
3
0 0 0
1 1 1
1 2
1 3
2 1
2 3
3 1
3 2
0 0
3
0 0 0
1 0 1
1 2
2 1
0 0
```

【样例输出】

```
4
Oh,it's impossible~!!
```

【例】开关问题

- 设 x_i 表示第 i 个开关的操作情况，$x_i=1$ 表示按了这个开关，$x_i=0$ 表示没有按。
- 设 $a_{i,j}$ 表示第 i 个开关和第 j 个开关的联系情况，$a_{i,j}=1$ 表示按下 j 会影响 i 的状态，$a_{i,j}=0$ 表示不会影响。特别地，令 $a_{i,i}=1$。
- 一个开关最后的状态 $dist_i$，取决于它的初始状态 src_i 和所有与他有联系的开关的操作情况执行异或运算得到的结果。因此，可以列出异或线性方程组：

$$\begin{cases} a_{1,1}x_1 \oplus a_{1,2}x_2 \oplus \cdots \oplus a_{1,n}x_n = src_1 \oplus dist_1 \\ a_{2,1}x_1 \oplus a_{2,2}x_2 \oplus \cdots \oplus a_{2,n}x_n = src_2 \oplus dist_2 \\ \vdots \\ a_{n,1}x_1 \oplus a_{n,2}x_2 \oplus \cdots \oplus a_{n,n}x_n = src_n \oplus dist_n \end{cases}$$

【例】开关问题

- 异或其实是不进位加法[1]。我们仍然可以写出增广矩阵，矩阵中每个值要么是 0，要么是 1。
- 但是在执行高斯消元的过程中，把加、减运算替换成异或，而且不需要执行乘法运算。最终，可以得到该异或方程组对应的简化阶梯形矩阵。
- 此时，若存在形如 $0 = 1$ 的方程，那么方程组无解；否则，因为自由元可以取 0 或 1，所有方程组的解的数量就是 2^{cnt}，其中 cnt 为自由元个数。

[1] $a \oplus b \oplus c \oplus d = (a + b + c + d)\%2$

【例】开关问题


```
for(int r = 1, i = 1; i <= n; ++i)
{
    int p = 0;
    for(int k = r; k <= n; ++k) if(a[k][i]) { p = k; break; }
    if(p == 0) continue;
    if(p != r) for(int j = 1; j <= n + 1; ++j) swap(a[r][j], a[p][j]);
    for(int k = 1; k <= n; ++k)
    {
        if(r == k) continue;
        if(a[k][i] == 0) continue;
        for(int j = 1; j <= n + 1; ++j) a[k][j] ^= a[r][j];
    }
    ++r;
}
int cnt = 0;
for(int i = 1; i <= n; ++i)
{
    bool all = true;   // 系数是否全为 0
    for(int j = 1; j <= n; ++j) if(abs(a[i][j]) >= 1e-8) all = false;
    if(all && a[i][n + 1] >= 1e-8) { cnt = -1; break; }   // 无解
    if(all && a[i][n + 1] < 1e-8) ++cnt;   // 无穷多解
}
if(cnt == -1) puts("Oh,it's impossible~!!");
else printf("%d\n", 1 << cnt);
```

【例】开关问题

- 因为最多只有 28 个开关，为了简便、高效，可以把增广矩阵的每一行进行状态压缩。
- 用一个`int`类型的数据表示 $n+1$ 位二进制，其中第 0 位为增广矩阵最后一列的常数，第 $1\sim n$ 位分别为增广矩阵的第 $1\sim n$ 列的系数。
- 在进行高斯消元后，当表示一行的整数为 1 时，说明出现了形如 $0=1$ 的方程，此时无解；当表示一行的整数为 0 时，说明是自由元。

【例】开关问题

高斯消元法

河南省实验中学信息技术组

高斯消元法
算法过程
解的情况
计算机实现
例题
球形空间产生器
开关问题
坏掉的机器人
练习

```
for(int r = 1, i = 1; i <= n; ++i)
{
    for(int k = r; k <= n; ++k)
        if (a[k] >> i & 1) { swap(a[k], a[r]); break; }
    if((a[r] >> i & 1) == 0) continue;
    for(int k = 1; k <= n; ++k)
    {
        if (k == r) continue;
        if ((a[k] >> i & 1) == 0) continue;
        a[k] ^= a[r];
    }
    ++r;
}
int cnt = 0;     // 默认有一解
for(int i = 1; i <= n; ++i)
{
    if (a[i] == 1) { cnt = -1; break; }
    if (a[i] == 0) ++cnt;
}
if(cnt == -1) puts("Oh,it's impossible~!!");
else printf("%d\n", 1 << cnt);
```

【例】坏掉的机器人

给定一张 $n \times m$ 的棋盘，有一个机器人处于 (x, y) 位置。这个机器人可以进行很多轮行动，每次等概率地随机选择停在原地、向左移动一格、向右移动一格或向下移动一格 (当然机器人不能移出棋盘)。求机器人从起点走到最后一行的任意一个位置上，所需行动次数的数学期望值。

【输入格式】

第一行包含两个整数 $n, m(n, m \le 1000)$。

第二行包含两个整数 x 和 y，表示机器人的初始位置。

设定棋盘左上角为 $(1, 1)$，右下角为 (n, m)。

【输出格式】

输出一个实数，表示数学期望，结果保留四位小数。

【样例输入】

```
10 14
5 14
```

【样例输出】

```
18.0038
```

【例】坏掉的机器人

- 定义 $f(i,j)$ 表示从 (i,j) 走到最后一行的，所需行动次数的数学期望。
- 显然有 $f(n,j)=0$，目标为 $f(x,y)$。
- 当 $j=1$ 时，有

$$f(i,1)=\frac{1}{3}(f(i,1)+f(i,2)+f(i+1,1))+1$$

- 当 $j=m$ 时，有

$$f(i,m)=\frac{1}{3}(f(i,m)+f(i,m-1),f(i+1,m))+1$$

- 当 $2\le j\le m$ 时，有

$$f(i,j)=\frac{1}{4}(f(i,j)+f(i,j-1)+f(i,j+1)+f(i+1,j)+1$$

- 通过观察可以发现，第 $i+1$ 行的状态转移只能转移到第 i 行，但是列上可以向左或向右走，不满足无后效性。

【例】坏掉的机器人

- 在递推到第 i 行时，需要求解的值为 $f(i,j)$，已知的只有 $f(i+1,j)$，那么根据上页递推式，将 $f(i,j)$ 当作未知数，将 $f(i+1,j)$ 当作已知数，创建一个 m 元一次方程，方程的增广矩阵为 (以 $m=5$ 为例):

$$\begin{bmatrix} \frac{2}{3} & -\frac{1}{3} & 0 & 0 & 0 & \frac{1}{3}f(i+1,1)+1 \\ -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & 0 & \frac{1}{4}f(i+1,2)+1 \\ 0 & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & 0 & \frac{1}{4}f(i+1,3)+1 \\ 0 & 0 & -\frac{1}{4} & \frac{3}{4} & -\frac{1}{4} & \frac{1}{4}f(i+1,4)+1 \\ 0 & 0 & 0 & -\frac{1}{3} & \frac{2}{3} & \frac{1}{3}f(i+1,5)+1 \end{bmatrix}$$

- 在上述方程中进行高斯消元解方程，注意每次消元时只需要消 $2\sim3$ 次，所以每次解方程的时间复杂度为 O(M)，整体的时间复杂度为 O(NM)。
- 注意，当 $m=1$ 时，$f(i,1)=\frac{1}{2}(f(i,1)+f(i+1,1)+1$，可以推出 $f(i,1)=f(i+1,1)+2$，那么答案为 $2(n-x)$。

【例】坏掉的机器人

```
if(m == 1) { printf("%.4lf", 2 * (n - x)); return 0; }
for(int c = n - 1; c >= x; --c)
{
    memset(a, 0, sizeof(a));
    a[1][1] = 2.0 / 3, a[1][2] = -1.0 / 3, a[1][m + 1] = f[c + 1][1] / 3 + 1;
    for(int i = 2; i <= m - 1; ++i)
        a[i][i - 1] = -0.25, a[i][i] = 0.75,
        a[i][i + 1] = -0.25,
        a[i][m + 1] = 0.25 * f[c + 1][i] + 1;
    a[m][m - 1] = -1.0 / 3, a[m][m] = 2.0 / 3, a[m][m + 1] = f[c + 1][m] / 3 + 1;
    for(int i = 1; i <= m - 1; ++i)
    {
        a[i][i + 1] /= a[i][i], a[i][m + 1] /= a[i][i], a[i][i] = 1;
        a[i + 1][i + 1] -= a[i + 1][i] * a[i][i + 1];
        a[i + 1][m + 1] -= a[i + 1][i] * a[i][m + 1];
        a[i + 1][i] = 0;
    }
    a[m][m + 1] /= a[m][m], a[m][m] /= 1;
    for(int i = m; i >= 2; --i)
        a[i - 1][m + 1] -= a[i - 1][i] * a[i][m + 1],a[i - 1][i] = 0;
    for(int i = 1; i <= m; ++i) f[c][i] = a[i][m + 1];
}
```

练习

- 解方程组 (COGS 721)
- 球形空间产生器 (COGS 1845)
- 开关问题 (COGS 2879)
- 随机程序 (COGS 1495)
- 乘积是平方数 (COGS 1498)
- 小部件工厂 (COGS 2649)
- 坏掉的机器人 (COGS 3543)

向量

- 由 n 个有顺序的数 $a_1, a_2, \cdots, a_n$ 所组成的

$$(a_1, a_2, \cdots, a_n) \text{ 或 } \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{pmatrix}$$

被称为 n 维行向量或 n 维列向量，记为 $\vec{a}$ 或 $\boldsymbol{a}$。

- n 维行向量可以看作 $1 \times n$ 的行矩阵，n 维列向量可以看作 $n \times 1$ 的列矩阵。

线性相关与线性无关

- 设有若干个向量 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$，若存在一组数 $k_1, k_2, \cdots, k_m$ 使得向量

$$\vec{b} = k_1\vec{a}_1 + k_2\vec{a}_2 + \cdots + k_m\vec{a}_m$$

  则称向量 $\vec{b}$ 是向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 的线性组合，或者说向量 $\vec{b}$ 可由向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 线性表示。
- 在向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 如果存在一个向量能被其他向量线性表示，则称这些向量线性相关，否则称这些向量线性无关。

线性空间

- 由向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 线性表示的所有向量构成一个线性空间，向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 被称为这个线性空间的生成子集。
- 线性空间的极大线性无关子集被称为线性空间的基底，简称基。例如，平面直角坐标系中的所有向量构成的线性空间，它的一个基是 $\{(0,1),(1,0)\}$。
- 线性空间的所有基底包含的向量个数相等，而且是线性空间所有生成子集中向量个数最少的。
- 对于任意的向量组 $\vec{a}_1, \vec{a}_2, \cdots, \vec{a}_m$ 构成的线性空间，如何求这个线性空间的基底?
  - 可以将这 m 个 n 维行向量看作一个 $m \times n$ 的矩阵。
  - 将这个 $m \times n$ 的矩阵进行高斯消元得到一个简化阶梯型矩阵。
  - 简化阶梯型的所有非零行向量线性无关，也是线性空间的一个基底。

【例】装备购买

脸哥最近在玩一款神奇的游戏，这个游戏里有 n 件装备，每件装备有 m 个属性，用向量 $z[i]=(a_1,a_2,\cdots,a_m)$ 表示，每件装备需要花费 c_i。

现在脸哥想买一些装备，但是脸哥很穷，所以总是盘算着怎样才能花尽量少的钱买尽量多的装备。对于脸哥来说，如果一件装备的属性能用购买的其他装备组合出（也就是说脸哥可以利用手上的这些装备组合出这件装备的效果），那么这件装备就没有买的必要了。

严格的定义是，如果脸哥买了 $z[i_1],z[i_2],\cdots,z[i_p]$ 这 p 件装备，并且不存在实数 $b_1,b_2,\cdots,b_p$ 使得 $z[k]=b_1z[i_1]+b_2z[i_2]+\cdots+b_pz[i_p]$，那么脸哥就会买 $z[k]$，否则 $z[k]$ 对脸哥就是无用的。

脸哥想要在买下最多数量的装备的前提下花最少的钱，你能帮他算一算他能购买的最多装备数以及最小花费吗?

【例】装备购买

【输入格式】

第一行两个整数 $n, m (1 \le n, m \le 500)$。

接下来 n 行，每行 m 个数，其中第 i 行描述装备 i 的各项属性值。

接下来一行 n 个数，其中第 i 个数 c_i 表示购买第 i 件装备的花费。

【输出格式】

一行两个整数，第一个表示能购买的最多装备数量，第二个数表示在购买最多数量装备的情况下的最小花费。

【样例输入】

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

【样例输出】

```
2 2
```

【样例解释】

选择装备 $1,2$、$1,3$、$2,3$ 均可，其中选择装备 $1,2$ 的花费最小，最小为 2。

【例】装备购买

- 把 n 件装备看作 n 个长度为 m 的向量，根据题意，购买的装备对应的向量应该是线性无关的。
- 要买下最多数量的装备，就是求这 n 个向量表出的线性空间的基底。
- 将这 n 个长度为 m 的向量看作一个 $n \times m$ 的矩阵，然后进行高斯消元法转化为简化阶梯型，非零行的数目就是能购买的最多装备数。
- 因为要求花费最小，那么在高斯消元的过程中应当使用贪心策略。对于每个主元 x_i，在前 $i-1$ 列为 0、第 i 列不为 0 的行向量中选择价格最低的一个，对其他行进行高斯消元。

【例】装备购买

```cpp
for(int r = 1, i = 1; i <= n; ++i)
{
    int p = 0;
    for(int k = r; k <= n; ++k)
        if(abs(a[k][i]) >= 1e-8 && (p == 0 || c[p] > c[k])) p = k;
    if(p == 0) continue;
    swap(c[p], c[r]);
    for(int j = 1; j <= m; ++j) swap(a[r][j], a[p][j]);
    long double d = a[r][i];
    for(int j = 1; j <= m; ++j) a[r][j] /= d;
    for(int k = 1; k <= n; ++k)
    {
        if(k == r) continue;
        long double d = a[k][i];
        for(int j = 1; j <= m; ++j) a[k][j] -= d * a[r][j];
    }
    ++r;
}
// 查找的非 0 行即可
```

【例】XOR

有 n 个整数 $a_1, a_2, \cdots, a_n$ 和 q 个询问，每个询问给出一个整数 k，求从 $a_1, a_2, \cdots, a_n$ 中选出若干个数执行异或运算能够得到的整数集合中 (去掉重复的数)，第 k 小的整数是多少。$1 \le n, q \le 10000, 1 \le a_i, k \le 10^{18}$。

【输入格式】

第一行包含整数 T，表示共有 T 组测试数据。
对于每组测试数据，第一行包含整数 n。
第二行包含 n 个整数，表示完整的整数序列。
第三行包含整数 q，表示询问的次数。
第四行包含 q 个整数 $k_1, k_2, ..., k_q$，表示 q 个询问对应的 k。

【输出格式】

对于每组测试数据，第一行输出 `Case #C:`，其中 C 为顺序编号 (从 1 开始)。
接下来 q 行描述 q 次询问的结果，每行输出一个整数，表示第 i 次询问中第 k_i 小的结果。
如果能得到的不同结果的总数少于 k_i，则输出 -1。

【例】XOR

【样例输入】

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

【样例输出】

```
Case #1:
1
2
3
-1
Case #2:
0
1
2
3
-1
```

异或空间

- 类比线性空间，异或空间是一个关于异或运算封闭的非负整数集合。
- 若整数 b 能被整数 $a_1, a_2, \cdots, a_m$ 经过异或运算得出，则称 b 能被 $a_1, a_2, \cdots, a_m$ 表出。
- 整数 $a_1, a_2, \cdots, a_m$ 能表出的所有整数构成一个异或空间，$a_1, a_2, \cdots, a_m$ 被称为异或空间的生成子集。
- 整数 $a_1, a_2, \cdots, a_m$，如果存在一个整数 a_i 能被其他整数表出，则称这些整数线性相关，否则称这些整数线性无关。
- 异或空间的基底或基就是异或空间的极大线性无关子集。

异或空间

- 给定 m 个 $0 \sim 2^n - 1$ 之间的整数 $a_1, a_2, \cdots, a_m$，如果求出这 m 个整数表出的异或空间的基底?
  - 将这 m 个数看作 n 位二进制，写成一个 m 行 n 列的 01 矩阵，矩阵中第 i 行从左到右依次是 a_i 的第 $n-1, n-2, \cdots, 1, 0$ 位。
  - 把 01 矩阵作为系数矩阵，利用高斯消元法将其转化为简化阶梯型矩阵，每一行实际上是一个整数，可以直接利用异或运算进行消元。
  - 简化阶梯型矩阵的每一行是一个整数，所有非 0 整数一起构成异或空间的基底。

【例】XOR

- 利用高斯消元法求出 $a_1, a_2, \cdots, a_n$ 构成的异或空间的基。不妨设这个基由 $b_1, b_2, \cdots, b_t$ 构成，其中 $b_1 > b_2 > \cdots > b_t$。从基底中选取若干个整数，显然有 2^t 种取法，因此这个异或空间种共有 2^t 个整数，与这 2^t 种一一对应。
- 在高斯消元过程中，共有 t 个主元，分别是 $b_1, b_2, \cdots, b_t$ 的最高位，设它们的最高位分别是 $c_1, c_2, \cdots, c_t$ 位，显然有 $c_1 > c_2 > \cdots > c_t$。
- 在高斯消元过程中，主元所在列中，除了主元所在行为 1 外，其他行都为 0。所以，对于由 $b_1, b_2, \cdots, b_t$ 组成的数中，选择 b_1 的数一定比不选择 b_1 的数更大，因为 b_1 的的最高位位置 c_1 最大。同理两个数若都选择了 b_1，则选择 b_2 的数一定比不选择 b_2 的数大。

【例】XOR

- 对于询问的第 k 大，我们只需要对 $k-1$ 进行二进制分解 (最小的数是第 0 大)，如果 $k-1$ 的第 j 位等于 1，就选 b_{t-j}，最后把所有选出的 b 异或起来即为答案。特别地，若 $k > 2^t$，则无解。
- 有一个边界问题，是否存在第 0 大的数 (整数 0)? 题目中不允许 $a_i \oplus a_i$，但是线性空间中允许，所以线性空间中一定会存在整数 0，但是从 $a_1, a_2, \cdots, a_n$ 选出几个不同的数异或可能得不到 0。
- 实际上，只需要检查简化阶梯型中是否存在零行:
  - 如果存在零行，那么选出不同的数异或会得到 0，因此对 $k-1$ 进行二进制拆分;
  - 如果不存在零行，说明无法得到 0，因此对 k 进行二进制拆分。

【例】XOR

```cpp
int zero = 0, t = n;
for(int i = 1; i <= n; ++i)
{
    for(int j = i + 1; j <= n; ++j) if(a[j] > a[i]) swap(a[j], a[i]);  // 从大到小
    if(a[i] == 0) { zero = 1, t = i - 1; break; }  // 0 行
    for(int k = 63; k >= 0; --k)
    {
        if((a[i] >> k & 1) == 0) continue;
        for(int j = 1; j <= n; ++j)
        {
            if(i == j) continue;
            if((a[j] >> k & 1) == 0) continue;
            a[j] ^= a[i];
        }
        break;
    }
}
```

【例】XOR

```
int q; scanf("%d", &q);
while(q--)
{
    unsigned long long k; scanf("%llu", &k);
    if(zero) --k; // 存在 0 行 可以构成数字 0
    if(k >= 1llu << t) puts("-1");
    else
    {
        unsigned long long ans = 0;
        for(int i = t - 1; i >= 0; --i)
            if(k >> i & 1) ans ^= a[t - i];
        printf("%llu\n", ans);
    }
}
```

练习

- 装备购买 (COGS 2786)
- XOR(COGS 3544)
- 最大异或和路径 (COGS 2912)
- 新 Nim 游戏 (COGS 1818)