线性代数

2021-03-16 | 2022-10-05 | OI & ACM | 6
6.1k | 6 分钟

线性代数

幂零矩阵

如果存在正整数 k，使得 $A^k=0$，则 A 为幂零矩阵

如果 A 是幂零矩阵，那么 $I+A$ 一定是可逆矩阵

范德蒙德行列式

形如

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ a_1 & a_2 & \cdots & a_n \\ \vdots & \vdots & \ddots & \vdots \\ a_1^{n-1} & a_2^{n-1} & \cdots & a_n^{n-1} \end{vmatrix}$$

的 n 阶行列式称为范德蒙德行列式，且有 $\prod_{1\le j<i\le n}(a_i-a_j)$

线性方程组的解

对于 m 个方程，n 个未知数的线性方程组 $A_{m\times n}X_{n\times 1}=b_{m\times 1}$，其系数矩阵为 A，增广矩阵为 $B=(A\ b)$

$$\begin{cases} a_{1,1}x_1+a_{1,2}x_2+\cdots+a_{1,n}x_n=b_1 \\ a_{2,1}x_1+a_{2,2}x_2+\cdots+a_{2,n}x_n=b_2 \\ \cdots \\ a_{m,1}x_1+a_{m,2}x_2+\cdots+a_{m,n}x_n=b_m \end{cases}$$

如果 $b \neq 0$，则称为非齐次线性方程组，否则称为齐次线性方程组

非齐次线性方程组**有解**：$R(A) = R(B)$

非齐次线性方程组**无解**：$R(A) \neq R(B)$

非齐次线性方程组**有唯一解**：$R(A) = R(B) = n$

非齐次线性方程组**有无穷多解**：$R(A) = R(B) < n$

齐次线性方程组**只有零解**：$R(A) = n$

齐次线性方程组**有非零解**：$R(A) < n$

逆矩阵

代数余子式：A 的代数余子式 $A_{i,j}$ 为将 A 的第 i 行和第 j 列去掉后得到的矩阵 A' 的行列式的值

伴随矩阵：$A^为A的伴随矩阵，a^_{i,j}=A_{i,j}$

矩阵 A 可逆，则有 $A^*A = |A|I$

矩阵的秩

1. $r(AB) \leq min(r(A), r(B))$
2. A 为 $m \times s$ 矩阵，B 为 $s \times n$ 矩阵，若 $AB = 0$，则 $r(A) + r(B) \leq s$

秩为 1 的矩阵的性质：

1. 任意两行（两列）成比例
2. 秩为 $n-1$ 的 n 阶方阵 A 的伴随矩阵 $A^的秩为1，因为A的秩为n-1，所以一定存在不全为0的实数x_1,\cdots,x_n使得\sum_{i=1}^nx_ia_i=0，其中a_i表示A的第i行的行向量，同理可以得出关于列向量的y_i，可以证明A^的第i行和第j行的比例为\frac{x_i}{x_j}，列同理，那么有A_{a,b}=A_{i,j}\frac{x_ay_b}{x_iy_j}$

矩阵

矩阵乘法

矩阵乘法就是矩阵乘法吧

```cpp
struct Matrix {
    static const int n = 3;
    int a[n + 1][n + 1];

    Matrix() { clear(); }
    inline void setone() { clear(); for (int i = 1; i <= n; ++i) a[i][i] = 1; }
        inline void clear() { fill(a[0], a[0] + (n + 1) * (n + 1), 0); }
    friend Matrix operator * (const Matrix &u, const Matrix &v) {
        Matrix w;
        for (int k = 1; k <= n; ++k)
            for (int i = 1; i <= n; ++i)
                for (int j = 1; j <= n; ++j)
                    w.a[i][j] = add(w.a[i][j], 1ll * u.a[i][k] * v.a[k][j] % p);
        /*for (int i = 1; i <= n; ++i)
            for (int k = 1; k <= n; ++k) {
                if (!u.a[i][k]) continue;
                for (int j = 1; j <= n; ++j)
                    w.a[i][j] = add(w.a[i][j], 1ll * u.a[i][k] * v.a[k][j] % p);
            }*/
        return w;
    }
} ;
```

矩阵乘法的变种

矩阵乘法除了能使用乘法和加法意外，我们还可以将乘法换成加法，加法变成取最大值，可以证明这样仍然满足结合律

这样的单位矩阵是对角线为 0，其它位置是 $-\infty$

```cpp
struct Matrix {
    static const int n = 2;
    int a[n + 1][n + 1];

    Matrix() { clear(); }
    inline void clear() { fill(a[0], a[0] + (n + 1) * (n + 1), -INF); }
    inline void setone() { clear(); for (int i = 1; i <= n; ++i) a[i][i] = 0; }
    friend Matrix operator * (const Matrix &u, const Matrix &v) {
```

```
        Matrix w;
        for (int k = 1; k <= n; ++k)
            for (int i = 1; i <= n; ++i)
                for (int j = 1; j <= n; ++j)
                    w.a[i][j] = max(w.a[i][j], u.a[i][k] + v.a[k][j]);
        return w;
    }
};
```

高斯消元

求解线性方程组

就是消成单位矩阵，然后就没了

模板，注意判断无解和无穷多解

```
double g[maxn][maxn];
void Gauss() {
    bool F1 = 0, F2 = 0;
    for (int i = 1; i <= n; ++i) {
        int l = i;
        for (int j = i + 1; j <= n; ++j)
            if (fabs(g[j][i]) > fabs(g[l][i])) l = j;
        swap(g[l], g[i]); if (fabs(g[i][i]) < eps) { F1 = 1; continue; }
        for (int j = i + 1; j <= n + 1; ++j) g[i][j] /= g[i][i]; g[i][i] = 1;
        for (int j = 1; j <= n; ++j) {
            if (i == j) continue;
            for (int k = i + 1; k <= n + 1; ++k) g[j][k] -= g[j][i] * g[i][k];
            g[j][i] = 0;
        }
    }
    for (int i = 1; i <= n; ++i) {
        double s = 0;
        for (int j = i; j <= n; ++j) s += g[i][j];
        if (s < eps && g[i][n + 1] > eps) F2 = 1;
    }
    if (F2) cout << "No Solutions\n";
    else if (F1) cout << "Many Solutions\n";
    else for (int i = 1; i <= n; ++i) cout << g[i][n + 1] << "\n";
}
```

矩阵求逆

然后这东西还能用来矩阵求逆

```
int g[maxn][2 * maxn];
void Gauss() {
    for (int i = 1; i <= n; ++i) g[i][n + i] = 1;
    for (int i = 1; i <= n; ++i) {
        int pos = -1;
        for (int j = i; j <= n; ++j) if (g[j][i]) pos = j;
        if (pos == -1) return cout << "No Solution\n", void();
        swap(g[pos], g[i]); ll inv = pow_mod(g[i][i], p - 2);
        for (int j = i; j <= 2 * n; ++j) g[i][j] = g[i][j] * inv % p;
        for (int j = 1; j <= n; ++j) {
            if (j == i) continue;
            for (int k = i + 1; k <= 2 * n; ++k)
                g[j][k] = (g[j][k] - (ll) g[j][i] * g[i][k]) % p;
            g[j][i] = 0;
        }
    }
    for (int i = 1; i <= n; ++i, cout << "\n")
        for (int j = n + 1; j <= 2 * n; ++j) cout << (g[i][j] + p) % p << " ";
}
```

求解行列式

浮点数计算

```
double g[maxn][maxn];
double Gauss(int n) {
    double res = 1;
    for (int i = 1; i <= n; ++i) {
        int pos = -1;
        for (int j = i; j <= n; ++j)
            if (fabs(g[j][i]) > fabs(g[pos][i])) pos = j;
        if (pos == -1 || fabs(g[pos][i]) < eps) return 0;
        swap(g[i], g[pos]); res *= pos == i ? 1 : -1;
        res *= g[i][i];
        for (int j = i + 1; j <= n; ++j)
            for (int k = n; k >= i; --k)
                g[j][k] -= g[j][i] / g[i][i] * g[i][k];
    } return res;
}
```

模数不是质数，直接对两行做辗转相除，时间复杂度 $O(n^3)$

```
int g[maxn][maxn];
int Gauss() {
    int res = 1;
    for (int i = 1; i <= n; ++i) {
        int pos = -1;
        for (int j = i; j <= n; ++j) if (g[j][i]) pos = j;
        if (pos == -1) return 0; swap(g[i], g[pos]); res *= pos == i ? 1 : -1;
        for (int j = i + 1; j <= n; ++j) {
            if (g[j][i] < g[i][i]) swap(g[j], g[i]), res *= -1;
            while (g[i][i]) {
                int d = g[j][i] / g[i][i];
                for (int k = i; k <= n; ++k) g[j][k] = (g[j][k] + 1ll * (p - d) *
                swap(g[i], g[j]), res *= -1;
            } swap(g[i], g[j]), res *= -1;
        }
        res = 1ll * res * g[i][i] % p;
    } return res;
}

```

模数是质数

```
ll g[maxn][maxn];
int Gauss(int n) {
    ll res = 1;
    for (int i = 1; i <= n; ++i) {
        int pos = -1;
        for (int j = i; j <= n; ++j) if (g[j][i]) pos = j;
        if (pos == -1) return 0; swap(g[i], g[pos]); res *= pos == i ? 1 : -1;
        res = res * g[i][i] % p; ll inv = pow_mod(g[i][i], p - 2);
        for (int j = i + 1; j <= n; ++j)
            for (int k = n; k >= i; --k)
                g[j][k] = (g[j][k] - g[j][i] * inv % p * g[i][k]) % p;
    } return res;
}
```

求解异或方程组

异或本质就是模 2 意义下的加法，所以解的情况和一般的线性方程组一样，其中对于整行异或我们可以使用 $bitset$，时间复杂度 $O(\frac{n^3}{w})$

```cpp
int g[maxn][maxn];
void Gauss() {
    for (int i = 1; i <= n; ++i) {
        int p = -1;
        for (int j = i; j <= n; ++j) if (g[j][i]) p = j;
        if (p == -1) continue; swap(g[p], g[i]);
        for (int j = 1; j <= n; ++j) {
            if (i == j || !g[j][i]) continue;
            for (int k = i; k <= n + 1; ++k) g[j][k] ^= g[i][k];
        }
    }
    for (int i = 1; i <= n; ++i) {
        int s = 0;
        for (int j = i; j <= n; ++j) s |= g[i][j];
        if (!s && g[i][n + 1]) return (void) (cout << "No Solution\n");
    }
    cout << "Y\n";
}
```

例题

1. 简要题意：给出一张完全图以及从每个点到达另一个点的概率 $p_{i,j}$，另外每个点 i 都有 $p_{i,i}$ 的概率在点 i 停下，对于所有 (i,j)，求从 i 出发停在 j 的概率

   $n \leq 300$

   简要题解：

   构造矩阵 A，满足 $A_{i,j}=p_{i,j},A_{i,i}=0$，那么 $A^1+A^2+\cdots A^\infty=B$，$B_{i,j}$ 即为 i 走了若干步到 j 且中间没有停下来的概率，那么我们要求的答案就是 $B_{i,j}\times p_{i,i}$，容易得到 $B=\frac{I}{I-A}$

   2021杭电多校5 E Random Walk 2

线性基

不同于异或中常说的线性基，这里的线性基是用来维护线性最大无关组的

不过需要注意的是，这里求解线性基的算法是将向量都看成行向量的

```
for (int i = 1; i <= n; ++i)
    for (int j = 1; j <= m; ++j) {
        if (fabs(a[i][j]) < eps) continue;
        if (!p[j]) p[j] = i;
        double t = a[i][j] / a[p[j]][j];
        for (int k = j; k <= m; ++k) a[i][k] -= t * a[p[j]][k];
    }
```

这样我们在插入的时候就已经维护出一个行阶梯矩阵了

例题

Luogu P3265 [JLOI2015]装备购买

------------- 本文结束 感谢阅读 -------------

Tech # 线性代数