

数理知识总结

位运算

判断一个数是不是 2^k 的非负整数次幂

```
bool isPowerOfTwo(int n) {
    return n>0&&(n&(n-1))==0;
}
```

对2的非负整数次幂取模

$2^k \bmod 2^k$ 代表 2^k

```
int modPowerOfTwo(int x,int mod) {
    return x&(mod-1);
}
```

取绝对值

在某些机器上 效率比 $n>0?n:-n$ 高

```
int Abs(int n) {
    return (n^(n>>31))-(n>>31);
}
long long Abs(long long n) {
    return (n^(n>>63))-(n>>63);
}
```

取两个数的最大/最小值

在某些机器上 效率比 $a>b?a:b$ 高 如果 $a \geq b, (a-b) \gg 31$ 为 0 ，否则为 -1

```
int max(int a, int b) { return b & ((a - b) >> 31) | a & (~(a - b) >> 31); }
int min(int a, int b) { return a & ((a - b) >> 31) | b & (~(a - b) >> 31); }
```

模拟集合操作

差集是 $a \& (\sim b)$ 对称差是 $a \oplus b$

子集遍历

可以遍历一个数代表的集合中所有非空子集包括自己

这个复杂度为 $O(2^{\text{popcount}(u)})$ 的复杂度遍历 u 的子集

进而可以在 $O(3^n)$ 的时间复杂度内遍历大小为 n 的集合的每个子集的子集

```
int n=13;
for(int i=n;i;i=(i-1)&n){}
//遍历子集的子集
for(int i=n;i;i=(i-1)&n) {
    for(int j=i;j;j=(j-1)&i) {}
}
```

GCC内建函数

这些函数经过编译器的高度优化，运行速度很快，且省事

```
int __builtin_ffs(int x)
```

返回 x 的二进制中最后一个为 1 的位置，最低位的编号为 1 ，当 x 为 0 时返回 0

```
int __builtin_clz(unsigned int x)
```

返回 x 的二进制的前导 0 的个数。也就是可以确定二进制从左到右第一个为 1 的位置

```
int __builtin_popcount(unsigned int x)
```

返回 x 的二进制中 1 的个数

```
int __builtin_parity(unsigned int x)
```

判断 x 的二进制中 1 的个数的奇偶性

矩阵的操作

例题1：三维空间中 n 个点 p_i 要求将 m 个操作都应用与这些点，操作种类有三种：

- 1.沿向量移动
- 2.按比例缩放坐标
- 3.绕某个坐标轴旋转

这些操作都可以重复 k 次。

要求在尽可能低于 $O(nmk)$ 的时间内输出最终结果

对于三种操作，我们都可以理解为对三维坐标的线性变换。前两种的 k 次很好解决。

看起来第二种很好搞，将 3×3 的矩阵的对角分别写上三个维度的缩放比例，在进行矩阵乘法的时候就可以直接乘了。

但是这样对于第一种操作，我们首先肯定至少需要 3×3 的矩阵，对角线是 1 ，因为要加上新输入的值，但新输入的值却没有地方放，只能将矩阵扩大，比如变成 4×4 的。为了让三个维度的增量不相互影响，需要把新的增量写在新的一列，比如我们让 $AX=X'$ $x'=x+dx$ 其中 X 和 X' 都是 4×1 的矩阵，于是前三行都已经确定，都是由对角线的 1 和最后的增量构成 X 和 X' 最后一行也可以确定为 1 ，因为要乘上增量，于是 A 的最后一行也是对角线那里是 1 才可以保证 X' 的最后一行为 1 。

处理好了第一种操作，回头再看第二种操作。只需要在最后一行对角线处加一个 1 就正确了。

现在只剩下第三种操作。比如绕 z 轴旋转 z 坐标不变，所以显然下面的两行只有对角线元素是 1 。对于上面的矩阵，只需要考虑最左上角的 2×2 即可。然后将坐标变为复平面就可以推出坐标变换之后和原坐标关系，矩阵就能求出来了，然后对于 x, y 轴同理。

于是就把三种矩阵处理好了，分别求 k 的快速幂，这样复杂度可以看成是 $O(m \log k)$ 最后对于 n 个点，分别乘矩阵，总复杂度为 $O(n + m \log k)$

例题2：定长路径计数

给一个有向图，边权都为 1 ，求任意两点 u, v 间从 u 到 v 长度为 k 的路径的条数。

套路题，将邻接矩阵求 k 次幂即可。

如果是小于等于 k 则对于每个点加一个权值为 1 的自环，这样就可以原地踏步，包括所有情况了。

例题3：树上询问

有一颗 n 结点的树，根为 1 号结点。每个结点有两个权值 k_i, t_i 初始值均为 0 。

有三种操作：

1. 将 x 到根的路径上所有点的 k_i 变成 $k_i + d$
2. 将 x 到根的路径上所有点的 t_i 变成 $t_i + d \times k_i$
3. 查询点 t_x

我们用矩阵维护两种操作

每个点的矩阵都设为 $\begin{bmatrix} k & t \\ 1 & 1 \end{bmatrix}$ 末尾的 1 的理由上面说过了

于是两种操作写一下就很显然能知道转移矩阵了。

组合数

从排成一排的 n 个球里选出 r 个球互不相邻的方案种数是 $C(n-r+1, r)$

从围成一圈的 n 个球里选出 r 个球互不相邻的方案种数是 $C(n-r, r) \times \frac{n}{n-r}$

（第二个结论相当于分类讨论选不选一号球（自己规定的）然后转化成两个第一个结论的问题）

$\sum_{i=0}^n i \times C(n, i) = n \times 2^{n-1}$ 把组合数都提出来个 n 然后二项式定理显然。

同理, 有 $\sum_{i=0}^n i^2 \times c(n,i) = n(n+1)2^{n-2}$

$$\sum_{i=0}^n c(l,k) = c(n+1,k+1)$$

$$c(n,r)c(r,k) = c(n,k)c(n-k,r-k)$$
 通过组合意义证明。

上指标反转 (其实在我这里变成了下指标反转 $c(r,k) = (-1)^k c(k-r-1,k)$, k 是整数, 对于任意的整数 n 都成立!!! 所以对于“下指标”和 (-1) 同时出现的, 要提高警惕。

$$(-1)^m c(-n-1,m) = (-1)^n c(-m-1,n)$$
 整数 $m, n \geq 0$ 因为两边都等于 $c(m+n,m)$

$$\sum_{k \leq m} c(r,k) (-1)^k = c(r,0) - c(r,1) + \dots + (-1)^m c(r,m) = \sum_{k \leq m} c(k-r-1,k) = c(m-r,m) = (-1)^m c(r-1,m)$$

有意思的关系式 $\sum_{k \leq m} c(m+r,k) x^k y^{m-k} = \sum_{k \leq m} c(-r,k) (-x)^k (x+y)^{m-k}$, m 为整数, 用数学归纳法可以证明, 此处省略。

$$\text{令 } x = -1 \text{ 并令 } y = 1 \text{ 则有 } \sum_{k \leq m} c(m+r,k) (-1)^k = c(-r,m)$$

$$\text{令 } x = 1 \text{ 并令 } y = 1, r = m+1 \text{ 则有 } \sum_{k \leq m} c(2m+1,k) = \sum_{k \leq m} c(m+k,k) 2^{m-k}$$

又因为左侧是求了一半的组合数, 所以是 $2^{2m+1-1} = 2^{2m}$ 所以这个式子等价于 $2^m = \sum_{k \leq m} c(m+k,k) 2^{-k}$

$$\sum_k c(l,m+k) c(s+k,n) (-1)^k = (-1)^{l+m} c(s-m,n-l)$$
 整数 $l \geq 0, m, n$ 为整数, 数学归纳法, 先拆后合可证, 这里省略。

$$\sum_{k \leq l} c(l-k,m) c(s,k-n) (-1)^k = (-1)^{l+m} c(s-m-1,l-m-n)$$
 整数 $l, m, n \geq 0$

$$\sum_{-q \leq k \leq l} c(l-k,m) c(q+k,m) = c(l+q+1,m+n+1)$$
 整数 $m, n \geq 0$, 整数 $l+q \geq 0$

例题

$$1. \text{求 } \frac{\sum_{k=0}^m c(m,k) c(n,k)}{c(n,m)}$$
 整数 $n \geq m \geq 0$

我们自然是希望统一分母, 貌似没有选择, 统一成 $c(n,m)$ 会好一些, 又因为我们有 $c(n,m)c(m,k) = c(n,k)c(n-k,m-k)$ 所以原式等于 $\frac{\sum_{k=0}^m c(n-k,m-k)}{c(n,m)}$ 分子用 $t = m-k$ 换个元, 再带公式, 就知道是 $c(n+1,m)$ 和分母分别写成阶乘的形式, 最后结果是 $\frac{n+1}{n+1-m}$

斐波那契数列

$$\sum_{i=0}^n c(n-i,i) = F(n+1)$$
 其中 $F(n)$ 表示斐波那契数列的第 n 项, $F(0) = F(1) = 1$ 数学归纳法加组合数公式可证。

$$F_n = \frac{((\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n)}{\sqrt{5}}$$

斐波那契二倍项和一倍相邻项有关

$F_{2k} = F_k(2F_{k+1} - F_k)$ 这里的是 $F_1 = F_2 = 1$ 的。

$$F_{2k+1} = \{F_{k+1}\}^2 + \{F_k\}^2$$

这两个式子可以用数学归纳法（螺旋）一直证上去就结束了。

斐波那契数列中间项平方和相邻项乘积有关

$F_{n-1}F_{n+1} - \{F_n\}^2 = (-1)^2$ 设 $T(n) = F_{n-1}F_{n+1} - \{F_n\}^2$ $T(2) = 1$ 且 $T(n+1) = -T(n)$ 可证，所以原式成立

$F_{n+k} = F_k F_{n+1} + F_{k-1} F_n$ 数学归纳法照证不误（拆项）。

令 $k=n$ 我们得到 $F_{2n} = F_n(F_{n+1} + F_{n-1})$ 所以 $F_{2n} = (F_{n+1} - F_{n-1})(F_{n+1} + F_{n-1}) = F_{n+1}^2 - F_{n-1}^2$

$$(F_m, F_n) = F_{(m,n)}$$

卢卡斯数列 $L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2}$ 前几项为 $2, 1, 3, 4, 7, 11, 18 \dots$

卢卡斯数列通项公式 $L_n = \left(\left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n\right)$

事实上，我们有 $\left(\frac{L_n + F_n}{\sqrt{5}}\right)^2 = \left(\frac{1+\sqrt{5}}{2}\right)^{2n}$

$$\{L_n\}^2 - 5\{F_n\}^2 = (-4)^n$$

$$2L_{m+n} = 5F_m F_n + L_m L_n$$

$$2F_{m+n} = F_m L_n + L_m F_n$$

$$L_{2n} = L_n^2 - 2(-1)^n$$

$$F_{2n} = F_n L_n$$

考虑模 p 意义下的斐波那契数列，有一个结论是，周期不会超过 $6p$ 且只有在满足 $p = 2 \times 5^k$ 的形式时才取等号。

From:
<https://wiki.cvbbacm.com/> - CVBB ACM Team

Permanent link:
https://wiki.cvbbacm.com/doku.php?id=2020-2021:teams:legal_string:%E7%8E%8B%E6%99%BA%E5%BD%AA%E6%95%B0%E7%90%86%E7%9F%A5%E8%AF%86

Last update: 2021/08/10 23:17

