目 录

1 Thanks

感谢Jacky007提供题目，感谢cauchy，WJH，1957等进行测试。

2 submission

- 比赛地址1:http://cs.scu.edu.cn/soj/contest/contest.action?cid=249
- 比赛地址2:http://zuojie.3322.org:88/soj/contest/contest.action?cid=249
- 请在题目列表中输入题目名字查找或者在38xx中找题目。
- 数据有问题请QQ联系。
- Baihacker's Script Language 作为校赛题目不公布数据。
- 广告：个人主页（如果题目有问题会在上面更新）

3 Queue

题目是因为我去排队，见到很多人插队，很不爽，然后yy出来的。题目是简单的递归的使用。 假定从上到下第i层有2^i个结点，i从0开始，最大为n。

对于全局位置到局部位置的转换： 对于当前的一个结点（门），前k个是来自左边，接着k个是来自右边，于是位置除以k得到一个 block值，根据block的奇偶性就可以知道来自左边还是右边，对于偶数，可以重新计算出来， 是左边的哪一个，对于奇偶，在计算出来在右边的位置的时候还要注意所处的队伍要加上一个值。

对于局部位置到全局位置的转换： 计算出所在块，根据queue的id的奇偶性可以知道在与其相关的另一个队伍中插入了多少人，也就是说 可以直接计算出来从当前门出去之后的id，于是就变成了一个子问题了。

当然，也可以把递归的写成非递归的。

PS:可以考虑队列中的人数不是无限，队列不是2的方幂个，每个门不一定从左边开始选，每个门的输入队列数不一定是2。

```
struct solver_recursive
{
int64 pos2id_impl(int64 n, int64 k, int64 global_pos, int64& q)
{
    if (n == 0)
    {
        return global_pos;
    }
    else
    {
        int64 block = global_pos / k;
        int64 off = global_pos % k;
```

```
        if (block % 2 == 0)
        {
            return pos2id_impl(n-1, k, block / 2 * k + off, q);
        }
        else
        {
            q += 1LL << (n - 1);
            return pos2id_impl(n-1, k, block / 2 * k + off, q);
        }
    }
}
int64 pos2id(int64 n, int64 k, int64 global_pos, int64& q)
{
    q = 0;
    return pos2id_impl(n, k, global_pos, q);
}

int64 id2pos(int64 n, int64 k, int64 q, int64 queue_pos)
{
    if (n == 0)
    {
        return queue_pos;
    }
    else
    {
        int64 block = queue_pos / k;
        int64 off = queue_pos % k;
        if (q % 2 == 0)
        {
            return id2pos(n-1, k, q/2, block*2*k+off);
        }
        else
        {
            return id2pos(n-1, k, q/2, (block*2+1)*k+off);
        }
    }
}
};

struct solver_normal
{
public:
int64 pos2id(int64 n, int64 k, int64 global_pos, int64& q)
{
    int64 t = global_pos % k, ans = global_pos - t;
    q = 0;
    for (; n > 0; --n)
    {
        int64 block = ans / k;
        ans = block / 2 * k;
        if (block & 1) q += 1LL << (n - 1);
    }
```

```
    return ans + t;
}
int64 id2pos(int64 n, int64 k, int64 q, int64 queue_pos)
{
    int64 t = queue_pos % k, ans = queue_pos - t;
    for (; n > 0; --n, q >>= 1)
    {
        int64 block = ans / k;
        ans = block * 2 * k;
        if (q&1) ans += k;
    }
    return ans + t;
}
};
```

4 Cube

裸的Polya，要么直接用Polya算出来，要么注意到是6个面，所以答案是和输入的6次方相关，题目 中给了很多数据了，可以待定系数出来。

$$\frac{1}{24}(k^6 + 3k^4 + 12k^3 + 8k^2) \tag{4.1}$$

要注意的是如果用中国剩余定理的话，3和被除数24不是互素的，搞起来很麻烦，最高才6次，所以直接大数。 一开始的标程是c++大数模板写的，由cauchy用java的大数进行测试，我最后又写了一个专门用于这个题的大数。此外，直接对331012005*24取模后除以24也行。 这个题时限是10s，用c++写大数写得再矬都能过吧。
PS:标程中考虑了直接计算溢出的情况，但是实际上通过计算可以知道对331012005*24取模的结果再乘以k不会溢出。

```
const int64 mod = 331012005LL*24;
inline int64 fuck(int64 x){for (;x>=mod;x-=mod); return x;}
inline int64 mul(int64 x, int64 y)
{
    int64 ans = 0;
    for (int64 i = 1; i <= x; i <<= 1, y = fuck(y<<1))
    {
        if (i&x) ans += y;
    }
    return fuck(ans);
}

int main()
{
    for (int caseID = 1, cas = Rint(); cas--; ++caseID)
    {
        int64 k = Rint();
        int64 k2 = mul(k, k);
```

```
        int64 k3 = mul(k, k2);
        int64 k4 = mul(k, k3);
        int64 ans = (mul(k3, k3) + 3 * k4 + 12 * k3 + (k2 << 3));
        printf("%lld\n", fuck(ans) / 24);
    }
    return 0;
}
```

5 Distribute The Apples IV

题目是srm455DIVI level2中，可能用到的。似乎这个题也可以根据那么多的样例暴出来。

解法1：用dp[i]表示A = i的时候的解。那么对于dp[i]解分为两部分，一部分是

$$x + y < i, x + z < i, y + z < i \tag{5.1}$$

的，另一部分是至少有一个等号成立的。 对于全部成立，显然，只有i为偶数的时候才有一种。对于至少两个成立，不妨设是前两个取等，于是可以计算出来z的取法有 (i + 2) / 2种。 同样的，对于至少一个成立的，可以计算出来根据i的奇偶性有，t * (t + 1)或 (t + 1) * (t + 1)种，其中t = (i + 1) / 2。 于是打表计算就行了。

```
struct runner
{
int64 fun(int64 x)
{
    int64 t = (x + 1) >> 1;
    return x & 1 ? t * (t + 1) % mod : (t + 1) * (t + 1) % mod;
}
void init()
{
    dp[0] = 1;
    for (int i = 1; i <= 100000; ++i)
    dp[i] = ((int64)dp[i-1] + fun(i) * 3 - ((i + 2) >> 1) * 3 + ((i&1)
        ? 0 : 1)) % mod;
}
void run()
{
    int cas;scanf("%d", &cas);
    init();
    while (cas--)
    {
        int k;scanf("%d", &k);
        assert(k>=0&&k<=100000);
        printf("%d\n", dp[k]);
    }
}
```

```
};
```

解法2： 分别求

$$0 \le x + y + z \le A + i \tag{5.2}$$

解数。 i取零的时候比较好计算。 i非零的时候，对固定的i，作图可以发现是一个正三角形，对i等于1，边上有 n - 1个点，正三角形内的点可以计算出来。同样的，i=2的时候，边上有n-3个点。 于是分n的奇偶性可以分别计算。

```
int64 sum1(int64 n)
{
    return n * (n + 1) >> 1;
}
int64 sum2(int64 n)
{
    return n * (n + 1) * (2*n + 1) / 6;
}
int main()
{
    for (int caseID = 1, cas = Rint(); cas--; ++caseID)
    {
        const int64 n = Rint();
        int64 ans = (sum2(n) + sum1(n) * 3 + ((n + 1) << 1)) >> 1;
        const int64 s = n - 1;
        if (s&1)
        {
            const int64 t1 = sum2(n) + sum1(n), t2 = (2 + n) * n >>
         1;
            ans += (t1 - t2) >> 2;
        }
        else
        {
            const int64 t1 = sum2(s) + sum1(s), t2 = (2 + s) * s >> 1;
            ans += (t1 + t2) >> 2;
        }
        printf("%lld\n", ans % 331012005);
    }
    return 0;
}
```

其它解法还相当多。

```
long long square(long long n)
{
    return n * n;
}

long long cube(long long n)
{
    return n * n * n;
```

```
}

long long simplex(long long n)
{
    return n * (n + 1) * (n + 2) / 6;
}

long long f(long long A)
{
    if ( A < 1 ) return 1;
    long long S = simplex(A + 1);
    long long D = cube(A + 1) - simplex(A) * 4;
    long long V = 1 - (1 & A);
    long long E = (A - 1 - V) / 2;
    long long F = (square(A - 1) - V - E * 4) / 4;

    return S + (D + V * 3 + 8 + E * 2 * 4 + (A - 1) * 6 + F * 6) / 4
        - (A + 1) * (A + 2) / 2;
}
```

6 Ordered Tree

对于一个有序树，如果去掉根，就变成一个森林，而森林的形状和二叉树是有天然的对应关系的。所以，一个n个点的有序树 和有n-1个顶点二叉树也是一一对应的。于是对于n，答案就是第n-1个catalan数。分析见《计算机程序设计艺术》卷1，国防 工业出版社，367页。

注意到n是可能有很多位的，但是当n足够大的时候答案就是0了。所以，只要把前导零去掉，数一下剩下的数字，如果太多 就直接输出0。标程中用的是数字大于等于10个。

取模的不是素数，所以分别取模中国剩余定理就行了。标程中组合数取余用的是以前写的小东西：一类数论问题

PS:题目中说的n的数字不超过1000个，并不是说n不超过1000。

```
const int64 mod = 331012005;
template<int64 P>
struct moder
{
    static int64 power(int64 x, int64 n)
    {
        int64 result = 1;
        x %= P;
        for (; n; n >>= 1, x = x * x % P) if (n&1) result = result * x
         % P;
        return result;
    }
    static int64 inv(int64 x)
    {
```

```
        return power(x, P-2);
    }
    static int64* fac;
    static void init_fac ()
    {
        fac = new int64[P];
        fac[0] = 1;
        for (int i = 1; i < P; ++i) fac[i] = fac[i-1] * i % P;
    }
    static void destroy_fac()
    {
        delete[] fac;
    }
    static int64 s(int x) {return x & 1 ? -1 : 1;}
    static int64 comb(int64 m, int64 n)
    {
        int64 pp = 0;
        int64 dist = m - n;
        for (int64 x = P; x <= m; x *= P) pp += m / x - n / x -
         dist / x;
        if (pp) return 0;
        int64 l = 1, r = 1;
        for (int64 x = m; x; x /= P) l = l * s(x/P) * fac[x%P] % P;
        for (int64 x = n; x; x /= P) r = r * s(x/P) * fac[x%P] % P;
        for (int64 x = dist; x; x /= P) r = r * s(x/P) * fac[x%P] % P;
        l = (l + P) % P;
        r = (r + P) % P;
        int64 t = (inv(r) * l % P + P) % P;
        return t;
    }
};
template<int64 P>
int64 * moder<P>:: fac;
int64 china_mod_331012005(int64 a, int64 b, int64 c, int64 d)
{
    if (a == 0 && b == 0 && c == 0 && d == 0) return 0;
    const int64 t1 = mod / 3;
    const int64 t2 = mod / 5;
    const int64 t3 = mod / 307;
    const int64 t4 = mod / 71881;
    const int64 a1 = (a * moder<3>::inv(t1)) % mod * t1 % mod;
    const int64 a2 = (b * moder<5>::inv(t2)) % mod * t2 % mod;
    const int64 a3 = (c * moder<307>::inv(t3)) % mod * t3 % mod;
    const int64 a4 = (d * moder<71881>::inv(t4)) % mod * t4 % mod;
    return (a1+a2+a3+a4) % mod;
}

#include <cassert>
#define ACTION(y) moder<3>::y();moder<5>::y();moder<307>::y();
         moder<71881>::y()

char buff[1024];
```

```
struct runner
{
void run()
{
    ACTION(init_fac);
    int cas; scanf("%d", &cas);
    while (getchar() != '\n');
    while (cas--)
    {
        gets(buff);
        int t = strlen(buff);
        assert(t >= 1 && t <= 1000);
        int curr = 0;
        while (buff[curr] == '0') ++curr;
        int last = curr;
        while (buff[curr] && curr - last < 10) ++curr;
        if (curr - last >= 10) {puts("0");continue;}
        int64 k;
        sscanf(buff+last, "%lld", &k);
        --k;
        int64 k2 = k * 2;
        int64 a = moder<3>::comb(k2, k);
        int64 b = moder<5>::comb(k2, k);
        int64 c = moder<307>::comb(k2, k);
        int64 d = moder<71881>::comb(k2, k);
        if (k)
        {
            a = ((a - moder<3>::comb(k2, k-1)) % 3 + 3) % 3;
            b = ((b - moder<5>::comb(k2, k-1)) % 5 + 5) % 5;
            c = ((c - moder<307>::comb(k2, k-1)) % 307 + 307) %
     307;
            d = ((d - moder<71881>::comb(k2, k-1)) % 71881 +
     71881) % 71881;
        }
        int64 ans = china_mod_331012005(a, b, c, d);
        printf("%lld\n", ans % mod);
    }
    ACTION(destroy_fac);
}
};
```

7 Path Resolution

题目是很简单的模拟，要求把路径简化，而且路径总是以/结尾。用result表示当前已解析出来的串，如果串的第i个字符是/那么prev[i]表示上一个/的位置。用done表示已经解析出来的串的最后一个位置。那么从done+1开始，复制字符串，直到遇到/为止 再反向检查，如果是..那么done就向前跳，如果是/则done不变，否则就记录一个新的解析出来的目录。

注意到结果不会比输入长，于是可以不用附加一个result数组。

```
struct runner
{
char* modify(char* buff)
{
    result[0] = '/';
    prev[0] = -1;
    int done = 0;
    for (int i = 1; buff[i]; ++i)
    {
        int j = done+1;
        while ((result[j] = buff[i]) != '/') ++i, ++j;
        if (buff[i-1] == '.' && buff[i-2] == '.')
        {
            if (prev[done] != -1) done = prev[done];
        }
        else if (buff[i-1] == '.')
        {
            ;
        }
        else
        {
            prev[j] = done;
            done = j;
        }
    }
    result[done+1] = 0;
    return result;
}
void run()
{
    int cas; scanf("%d", &cas);
    while (getchar() != '\n');
    while (cas--)
    {
        gets(buff);
        int l = strlen(buff);
        assert(l >= 1);
        assert(l <= 1000);
        assert(buff[0] == '/');
        assert(buff[l-1] == '/');
        puts(modify(buff));
    }
}
};
```

8 Dice Game

掷8次色子，确定4个两位数，掷出的数字放在哪一个数上并不重要，所以只需 确定放在十位上还是个位上即可。注意到最大得分为66*4，所

以$K \geq 264$的 时候结果一样。用d[i][j][k]表示(当前和，十位占用，个位占用)时的解, 枚举一下掷出的数字,求期望的最大值即可。

```
const int M = 4;
const int N = 264;
double d[N + 1][M + 1][M + 1];

int main()
{
    for (int T = Rint(); T--;)
    {
        int n = Rint();
        assert(n>=0&&n<=1000000000);
        if (n > N) n = N;
        for (int i = 0; i <= n; ++i) d[i][M][M] = i;
        for (int i = n; i >= 0; --i) for (int j = 0; j <= M; ++j)
        for (int k = 0; k <= M; ++k) if (j < M || k < M)
        {
            double expected = 0, e1, e2;
            for (int x = 1; x <= 6; ++x)
            {
                if (i + x * 10 > n || j == M) e1 = 0;
                else e1 = d[i + x * 10][j + 1][k];
                if (i + x > n || k == M) e2 = 0;
                else e2 = d[i + x][j][k + 1];
                expected += (e1 > e2 ? e1 : e2) / 6;
            }
            d[i][j][k] = expected > n ? 0 : expected;
        }
        printf("%.6f\n", d [0][0][0]) ;
    }
    return 0;
}
```

9 Guess number

被猜测的数X在1到n之间，每一次询问Y后不能得到关于Y与X的任何关系，只有 在下一次询问时才能得到Y与X的关系，第i次猜测的数Yi，不妨设$Y1 < Y2, 1...Y1...Y2...n$ 如果回答$X < Y1$，第二次猜测就浪费了，如果回答$X > Y1$，则Y2有效, 设d[i]表示i次猜测可以确定的最大区间长度，有$d[i] = d[i-2](X < Y1) + d[i-1](X > Y1) + 1(X = Y1)$； 寻找i满足 $d[i-1] < n \leq d[i]$即可。

PS:这个递推的结果就是指定高度的AVL树的最少结点数，见soj1970。

```
inline int cal(int n)
{
    if (n < 3) return n;
    int x = 1, y = 2, ret = 2, t;
```

```
    for (; y < n; ++ret) t = y, y = x + y + 1, x = t;
    return ret;
}

int main()
{
    for (int T = Rint(); T--; )
    {
        int n = Rint();
        printf("%d\n", cal(n));
    }
    return 0;
}
```

10 Lambda-Calculus

这个题来源是lambda演算,参考资料是<<类型与程序设计语言>>. 实际上程序的要求就是实现lambda演算,测试例子为church布尔式,church数值的运算对复杂表达式的解析要求不高.

- 递归下降生成语法树;
- 按要求的条件进行归约,由于限制,只要是出现application的地方就可以进行归约,并得到相同的结果,所以任何时候都可以进行归约;
- 归约时,只要对所有的application中用右端替换L.所限定的变量即可,注意变量同名的时候(Lx.Lx x) y,内部变量会隐藏外部的,所以不能进行;
- 输出时注意要求.

11 Baihacker's Script Language

这个题没啥说的，校赛个人赛的时候用来恶心人的，不过相信写了之后，不会把C++中运算符优先级搞错了。相对于Lambda，这个题可能对复杂表达式的解析比较高。 数据全部是人肉检查的，再次感谢1957。

下面是部分数据

```
test = 255;
bitcnt = 0;
while (test, sequence(bitcnt = bitcnt + 1, test = test & (test - 1)));
print(bitcnt);

test = 32768;
bitcnt = 0;
while (test, sequence(bitcnt = bitcnt + 1, test = test & (test - 1)));
print(bitcnt);
```

```
print

(

224488

)

;

ite = 1;
while (ite <= 12, sequence(fuck = fuck * ite, print(fuck), ite = ite +
    1));

result = 1;
while (n, sequence(select(n&1, result = result * x, result), x = x * x,
    n = n >> 1));
print(result);

print(-1);
print(---------333);
print(-+-+-+333);
print(-(-(1)));
print(~-333);
print(~-~-333);
print(~-~-~-333);
```