

Challestend's ACM Templates

Challestend Rehtorbegnaro

Last Update: 2023 年 8 月 3 日

目录

1	数论	1
1.1	扩展欧几里得算法	1
1.2	CRT	1
1.3	Lucas 定理	2
1.4	康托展开	3
1.5	BSGS	3
1.6	高斯消元	3
1.7	Miller-Rabin 算法	4
1.8	Pollard-Rho 算法	4
1.9	杜教筛	6
1.10	min_25 筛	7
2	数论：特殊无穷数列	9
2.1	组合数	9
2.2	第二类斯特林数	9
2.3	第一类斯特林数	9
2.4	斐波那契数	9
2.5	卡特兰数	10
2.6	贝尔数	10
2.7	伯努利数	11
2.8	默慈金数	11
2.9	错排数	12
2.10	分拆数	12
3	多项式	12
3.1	FFT	12
3.2	NTT	13
3.3	FWT	22
4	其他数学	23
4.1	牛顿迭代	23
4.2	组合对象符号化公式	23
4.3	拉格朗日反演	23
4.4	Burnside 引理 & Polya 定理	24
5	数据结构	24
5.1	可持久化 WBLT	24
6	图论	28
6.1	矩阵树定理	28
6.2	最大流	28

6.3	费用流	31
6.4	带下界可行流	32
6.5	带下界最大流	32
6.6	六元组模型	32
7	字符串	32
7.1	KMP	32
7.2	AC 自动机	33
7.3	倍增 SA	34
7.4	SAM	36
7.5	manacher	37

1 数论

1.1 扩展欧几里得算法

当且仅当 $(a, b) \mid c$ 时，不定方程

$$ax + by = c$$

有整数解。其通解为

$$x = x_0 + \frac{b}{(a, b)} \cdot t$$

$$y = y_0 - \frac{a}{(a, b)} \cdot t$$

```

1 inline void exgcd(int a, int b, int &x, int &y)
2 {
3     if (b == 0)
4     {
5         x = 1;
6         y = 0;
7         return;
8     }
9     else
10    {
11        exgcd(b, a % b, y, x);
12        y -= a / b * x;
13    }
14 }

```

1.2 CRT

一元线性同余方程组

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \vdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

若 m_i 两两互质，则通解为

$$\sum_{k=1}^n a_i t_i M_i \pmod{M}$$

其中

$$M = \prod_{i=1}^n m_i$$

$$M_i = \frac{M}{m_i}$$

$$t_i = M_i^{-1} \pmod{m_i}$$

若 m_i 不保证两两互质，设有两个方程形如

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \end{cases}$$

设

$$x = pm_1 + a_1 = qm_2 + a_2$$

解不定方程

$$m_1p - m_2q = a_2 - a_1$$

即可得到原来的两个方程的解

$$x \equiv pm_1 + a_1 \pmod{[m_1, m_2]}$$

1.3 Lucas 定理

如果 p 是质数，有

$$\binom{n}{m} \equiv \binom{\lfloor \frac{n}{p} \rfloor}{\lfloor \frac{m}{p} \rfloor} \binom{n \bmod p}{m \bmod p} \pmod{p}$$

如果模数不保证是质数，将其唯一分解后对于每个 p^k 分别求解再用中国剩余定理合并。令

$$\varphi(l, r, x) = \prod_{i=l}^r [x \nmid i] i$$

$$f(n) = f\left(\left\lfloor \frac{n}{p} \right\rfloor\right) \varphi\left[\frac{n}{p^k}\right](1, p^k, p) \varphi\left(p^k \left\lfloor \frac{n}{p^k} \right\rfloor + 1, n, p\right)$$

$$g(n) = \left\lfloor \frac{n}{p} \right\rfloor + g\left(\left\lfloor \frac{n}{p} \right\rfloor\right)$$

可以得出

$$\binom{n}{m} \equiv \frac{f(n)}{f(m)f(n-m)} \cdot p^{g(n)-g(m)-g(n-m)} \pmod{p^k}$$

1.4 康托展开

$$A = \sum_{i=1}^n \text{cnt}(i) \cdot (n-i)!$$

其中 $\text{cnt}(i)$ 表示 i 右侧小于 a_i 的数有多少个。

1.5 BSGS

求解方程

$$a^x \equiv b \pmod{p}$$

要求 $a \perp p$ 。

令 $x = us - v$, 其中 $s = \lceil \sqrt{p} \rceil$, $0 \leq u, v \leq s$ 。移项得到

$$a^{us} \equiv ba^v \pmod{p}$$

1.6 高斯消元

```

1  int rank = 0;
2  double det = 1;
3  for (int i = 1; i <= n; ++i)
4  {
5      int cur = 0;
6      for (int j = 1; j <= n; ++j)
7          if (a[i][j] & !vis[j])
8              {
9                  cur = j;
10                 break;
11             }
12     if (cur == 0)
13     {
14         det = 0;
15         break;
16     }
17     else
18     {
19         ++rank;
20         det *= a[i][cur];
21     }
22     for (int j = 1; j <= n; ++j)
23         if (j != cur)

```

```

24         a[i][j] /= a[i][cur];
25     b[i] /= a[i][cur];
26     a[i][cur] = 1;
27     for (int k = 1; k <= n; ++k)
28         if (k != i)
29         {
30             for (int j = 1; j <= n; ++j)
31                 if (j != cur)
32                     a[k][j] -= a[k][cur] * a[i][j];
33             b[k] -= a[k][cur] * b[i];
34             a[k][cur] = 0;
35         }
36 }

```

1.7 Miller-Rabin 算法

要求 n 是否为质数。

多次选取底数 a 。若某个 a 不满足费马小定理或二次探测定理（的逆否命题），则 n 为合数。全部满足则 n 可能为质数。

代码见下一小节。

1.8 Pollard-Rho 算法

不是很懂。

下面的代码是 Miller-Rabin 和 Pollard-Rho，还顺便附赠了快速加，快速乘，快速幂和 gcd。

效率：Luogu P4718 11/14。

```

1 inline long long add(long long x, long long y, long long m)
2 {
3     return x + y < m ? x + y : x + y - m;
4 }
5
6 inline long long mul(long long x, long long n, long long m)
7 {
8     long long res = 0;
9     for (; n; n >>= 1, x = add(x, x, m))
10         if (n & 1)
11             res = add(res, x, m);
12     return res;
13 }
14
15 inline long long pow(long long x, long long n, long long m)

```

```
16 {
17     long long res = 1;
18     for (; n; n >>= 1, x = mul(x, x, m))
19         if (n & 1)
20             res = mul(res, x, m);
21     return res;
22 }
23
24 inline long long gcd(long long a, long long b)
25 {
26     for (; b %= a ^= b ^= a ^= b; );
27     return a;
28 }
29
30 inline bool IsNotPrime(long long n, int a)
31 {
32     long long u = n - 1;
33     int t = 0;
34     for (; (u & 1) == 0; u >>= 1, ++t);
35     long long x = pow(a, u, n);
36     for (int i = 1; i <= t; ++i)
37     {
38         long long y = mul(x, x, n);
39         if (y == 1)
40             if (x == n - 1)
41                 return false;
42             else if (x != 1)
43                 return true;
44         x = y;
45     }
46     return x != 1;
47 }
48
49 inline bool MillerRabin(long long n)
50 {
51     const int test[]={2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
52     for (int i = 0; i < 12; ++i)
53         if (n == test[i])
54             return true;
55     if (n <= 40)
56         return false;
```



```

57     for (int i = 0; i < 12; ++i)
58         if (IsNotPrime(n, test[i]))
59             return false;
60     return true;
61 }
62
63 inline long long Rho(long long n)
64 {
65     long long c = rand() % (n - 3) + 1;
66     long long s = rand() % n, t = add(mul(s, s, n), c, n);
67     for (int lim = 1; s != t; lim = std::min(128, lim << 1))
68     {
69         long long val = 1;
70         for (int i = 1; i <= lim; ++i)
71         {
72             long long tmp = mul(val, std::abs(t - s), n);
73             if (tmp)
74             {
75                 val = tmp;
76                 s = add(mul(s, s, n), c, n);
77                 t = add(mul(t, t, n), c, n);
78                 t = add(mul(t, t, n), c, n);
79             }
80             else
81                 break;
82         }
83         long long d = gcd(val, n);
84         if (d > 1)
85             return d;
86     }
87     return n;
88 }

```

1.9 杜教筛

令 $f(n)$ 为一积性函数，现要计算其前缀和 $S(n)$ 。

另取一积性函数 $g(n)$ ，使得 $(f \times g)(n)$ 能够快速求前缀和，有

$$S(n) = \sum_{k=1}^n (f \times g)(k) - \sum_{k=2}^n g(k) f\left(\left\lfloor \frac{n}{k} \right\rfloor\right)$$

使用线性筛预处理出前 $n^{2/3}$ 项，则总时间复杂度取到最小值。

1.10 min_25 筛

$\text{minp}(n)$ 表示 n 的最小质因子。 p_k 表示第 k 小的质数。

第一部分。定义

$$F(n, k) = \sum_{i=1}^n [i \in \mathbb{P} \vee \text{minp}(i) > p_k] f(i)$$

要求

$$F(n, +\infty) = \sum_{i=1}^n [i \in \mathbb{P}] f(i)$$

$p_k^2 \leq n$ 时

$$F(n, k) = F(n, k-1) - f(p_k) \left(F\left(\left\lfloor \frac{n}{p_k} \right\rfloor, k-1\right) - \sum_{i=1}^{k-1} f(p_i) \right)$$

$p_k^2 > n$ 时

$$F(n, k) = F(n, k-1)$$

话说回来，代码块的注释不能写中文？不过懒得调了，反正应该能看懂。

```

1  for (int i = 2; i <= sq; ++i)
2  {
3      if (f[i] == 0)
4      {
5          g[++g[0]] = i;
6          // fill in the blank below with f(i)
7          fsum[g[0]] = fsum[g[0] - 1] + /* */;
8      }
9      for (int j = 1; j <= g[0] && i * g[j] <= sq; ++j)
10     {
11         f[i * g[j]] = 1;
12         if (i % g[j] == 0)
13             break;
14     }
15 }
16 for (int l = 1, r; l <= n; r = n / (n / l), l = r + 1)
17 {
18     w[++m] = n / l;
19     // fill in the blank below with \sum_{t=2}^{\infty} f(t)
20     F[m] = /* */;
21     if (w[m] <= sq)
22         id1[w[m]] = m;

```

```

23     else
24         id2[n / w[m]] = m;
25 }
26 for (int j = 1; j <= g[0]; ++j)
27     for (int i = 1; i <= m && w[i] >= g[j] * g[j]; ++i)
28     {
29         int id;
30         if (w[i] / g[j] <= sq)
31             id = id1[w[i] / g[j]];
32         else
33             id = id2[n / (w[i] / g[j])];
34         // fill in the blank below with f(g[j])
35         F[i] -= /* */ * (F[id] - fsum[j - 1]);
36     }

```

第二部分。定义

$$S(n, k) = \sum_{i=1}^n [\min p(i) \geq p_k] f(i)$$

要求

$$S(n, 1) + f(1)$$

有

$$S(n, k) = F(n, +\infty) - \sum_{i=1}^{k-1} f(p_i) + \sum_{i \geq k \wedge p_i^2 \leq n} \sum_{j \geq 1 \wedge p_i^{j+1} \leq n} \left(f(p_i^j) S\left(\left\lfloor \frac{n}{p_i^j} \right\rfloor, i+1\right) + f(p_i^{j+1}) \right)$$

```

1 int S(int x, int y)
2 {
3     if (x <= 1 || g[y] > x)
4         return 0;
5     else
6     {
7         int id = x <= sq ? id1[x] : id2[n / x];
8         int res = F[id] - fsum[y - 1];
9         for (int i = y; i <= g[0] && g[i] * g[i] <= x; ++i)
10             for (int p = g[i]; p * g[i] <= x; p *= g[i])
11                 // fill in the blanks below with f(p) and f(p * g[i]),
12                 // correspondingly
13                 res += /* */ * S(x / p, i + 1) + /* */;
14         return res;
15     }
16 }

```

14 }
15 }

2 数论：特殊无穷数列

2.1 组合数

递推式

$$\binom{n}{m} = \binom{n-1}{m-1} + \binom{n-1}{m}$$

$$\binom{n}{0} = \binom{n}{n} = 1$$

通项

$$\binom{n}{m} = \frac{n^{\underline{m}}}{m!}$$

2.2 第二类斯特林数

递推式

$$\left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \left\{ \begin{matrix} n-1 \\ m-1 \end{matrix} \right\} + m \left\{ \begin{matrix} n-1 \\ m \end{matrix} \right\}$$

$$\left\{ \begin{matrix} n \\ 0 \end{matrix} \right\} = [n=0]$$

2.3 第一类斯特林数

递推式

$$\left[\begin{matrix} n \\ m \end{matrix} \right] = \left[\begin{matrix} n-1 \\ m-1 \end{matrix} \right] + (n-1) \left[\begin{matrix} n-1 \\ m \end{matrix} \right]$$

$$\left[\begin{matrix} n \\ 0 \end{matrix} \right] = [n=0]$$

2.4 斐波那契数

递推式

$$F_n = F_{n-1} + F_{n-2}$$

$$F_0 = 0 \quad F_1 = 1$$

一些性质

$$F_{n+m} = F_{n+1}F_m + F_nF_{m-1}$$

$$\sum_{k=0}^n F_k^2 = F_n F_{n+1}$$

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

$$\pi(p^k) = p^{k-1} \pi(p)$$

$$\pi(nm) = [\pi(n), \pi(m)] \quad (n \perp m)$$

$$\pi(2) = 3 \quad \pi(5) = 20$$

$$p \equiv \pm 1 \pmod{10} \Rightarrow \pi(p) \mid p-1$$

$$p \equiv \pm 2 \pmod{5} \Rightarrow \pi(p) \mid 2p+2$$

2.5 卡特兰数

递推式

$$C_{n+1} = \sum_{k=0}^n C_k C_{n-k}$$

$$C_0 = 1$$

通项

$$C_n = \binom{2n}{n} - \binom{2n}{n-1} = \frac{1}{n+1} \binom{2n}{n}$$

OGF

$$C(x) = \frac{1 - \sqrt{1-4x}}{2x}$$

2.6 贝尔数

递推式

$$B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$$

$$B_0 = 1$$

通项

$$B_n = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\}$$

EGF

$$B(x) = \exp(e^x - 1)$$

其他性质

$$B_{n+p} \equiv B_n + B_{n+1} \pmod{p}$$

2.7 伯努利数

递推式

$$\sum_{k=0}^n \binom{n+1}{k} B_k = 0$$

$$B_0 = 1$$

EGF

$$B(z) = \frac{z}{e^z - 1}$$

其他性质

$$\sum_{k=0}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k (n+1)^{m-k+1}$$

2.8 默慈金数

递推式

$$M_{n+1} = M_n + \sum_{k=0}^{n-1} M_k M_{n-k-1} = \frac{(2n+3)M_n + 3nM_{n-1}}{n+3}$$

$$M_0 = 0 \quad M_1 = 1$$

通项

$$M_n = \sum_{k=0}^{\lfloor n/2 \rfloor} \binom{n}{2k} C_k$$

2.9 错排数

递推式

$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

$$D_1 = 0 \quad D_2 = 1$$

通项

$$D_n = n! \sum_{k=2}^n \frac{(-1)^k}{k!} = \left\lfloor \frac{n!}{e} + \frac{1}{2} \right\rfloor$$

EGF

$$D(x) = \frac{1}{\exp(x + \ln(1-x))}$$

2.10 分拆数

OGF

$$F(x) = \prod_{k \geq 1} \frac{1}{1-x^k}$$

$$\frac{1}{F(x)} = \sum_{k=-\infty}^{+\infty} (-1)^k x^{k(3k-1)/2}$$

3 多项式

3.1 FFT

都 2022 年了谁还写 FFT 啊 (暴论)

```

1 inline void FFT(std::complex<double> F[], int N, int tp)
2 {
3     for (int i = 0; i < N; ++i)
4         if (i < (rev[i] = (rev[i] >> 1) >> 1) | ((i & 1) ? (N >> 1) : 0))
5             std::swap(F[i], F[rev[i]]);
6     for (int len = 2, p = 1; len <= N; len <<= 1, p <<= 1)
7     {
8         std::complex<double> unit = {cos(Pi / p), tp * sin(Pi / p)};
9         for (int i = 0; i < N; i += len)
10            {
11                std::complex<double> cur = {1.0, 0.0};
12                for (int j = i; j < i + p; ++j)

```

```

13         {
14             std::complex<double> x = F[j], y = F[j + p] * cur;
15             F[j] = x + y;
16             F[j + p] = x - y;
17             cur = cur * unit;
18         }
19     }
20 }
21 }

```

3.2 NTT

包含：

1. 多项式与常数的四则运算
2. 多项式快速幂，求导，不定积分，DFT/IDFT，求逆，ln 函数，exp 函数
3. 多项式相加，相减，卷积，点乘，转置乘
4. 多项式多点求值，快速插值
5. 波斯坦-茉莉算法
6. 常系数齐次线性递推第 n 项的求解

封装程度较大，可能会因为常数不够优秀而 TLE。

```

1  #include <cstdio>
2  #include <algorithm>
3  #include <set>
4  #define maxn 524288
5  #define maxlog 20
6  #define mod 998244353
7
8  int unit[2][24];
9  int inv[maxn], fac[maxn], fav[maxn];
10 int rev[maxn];
11
12 inline int add(int a, int b)
13 {
14     return a + b < mod ? a + b : a + b - mod;
15 }
16
17 inline int sub(int a, int b)

```



```

18 {
19     return a - b >= 0 ? a - b : a - b + mod;
20 }
21
22 inline int pow(int a, int n)
23 {
24     int res = 1;
25     for (; n; n >>= 1, a = 1LL * a * a % mod)
26         if (n & 1)
27             res = 1LL * res * a % mod;
28     return res;
29 }
30
31 inline int C(int n, int m)
32 {
33     if (m >= 0 && m <= n)
34         return 1LL * fac[n] * fav[m] % mod * fav[n - m] % mod;
35     else
36         return 0;
37 }
38
39 inline void InitIntUnit()
40 {
41     unit[0][23] = pow(3, 119);
42     unit[1][23] = pow(332748118, 119);
43     for(int i = 0; i < 2; ++i)
44         for(int j = 22; j >= 0; --j)
45             unit[i][j] = 1LL * unit[i][j + 1] * unit[i][j + 1] % mod;
46 }
47
48 inline void InitFac()
49 {
50     inv[1] = 1;
51     for (int i = 2; i < maxn; ++i)
52         inv[i] = sub(0, 1LL * (mod / i) * inv[mod % i] % mod);
53     fac[0] = fav[0] = 1;
54     for (int i = 1; i < maxn; ++i)
55     {
56         fac[i] = 1LL * fac[i - 1] * i % mod;
57         fav[i] = 1LL * fav[i - 1] * inv[i] % mod;
58     }

```

```

59 }
60
61 inline void Derivative(int F[], int n, int G[])
62 {
63     for (int i = 0; i < n; ++i)
64         G[i] = 1LL * (i + 1) * F[i + 1] % mod;
65     G[n] = 0;
66 }
67
68 inline void Integration(int F[], int n, int G[], int C)
69 {
70     for (int i = n + 1; i > 0; --i)
71         G[i] = 1LL * inv[i] * F[i - 1] % mod;
72     G[0] = C;
73 }
74
75 inline void NTT(int F[], int N, int tp, int A[])
76 {
77     for (int i = 0; i < N; ++i)
78         A[i] = F[rev[i]] = (rev[i >> 1] >> 1) | ((i & 1) ? (N >> 1) : 0)
79             ];
80     for (int k = 1, p = 1; p < N; ++k, p <= 1)
81         for (int i = 0; i < N; i += p << 1)
82             for (int j = i, tmp = 1; j < i + p; ++j, tmp = 1LL * tmp *
83                 unit[tp][k] % mod)
84             {
85                 int x = A[j], y = 1LL * A[j + p] * tmp % mod;
86                 A[j] = add(x, y);
87                 A[j + p] = sub(x, y);
88             }
89     if (tp == 1)
90     {
91         int v = pow(N, mod - 2);
92         for (int i = 0; i < N; ++i)
93             A[i] = 1LL * A[i] * v % mod;
94     }
95 }
96
97 inline void Prod(int F[], int n, int G[], int m, int H[])
98 {
99     static int A[maxn], B[maxn], C[maxn];

```

```

98     int N = 1;
99     for (; N < n + m + 1; N <= 1);
100    for (int i = n + 1; i < N; ++i)
101        F[i] = 0;
102    for (int i = m + 1; i < N; ++i)
103        G[i] = 0;
104    NTT(F, N, 0, A);
105    NTT(G, N, 0, B);
106    for (int i = 0; i < N; ++i)
107        C[i] = 1LL * A[i] * B[i] % mod;
108    NTT(C, N, 1, H);
109    for (int i = n + m + 1; i < N; ++i)
110        H[i] = 0;
111 }
112
113 inline void TransProd(int F[], int n, int G[], int m, int H[])
114 {
115     static int tmp[maxn];
116     for (int i = 0; i <= m; ++i)
117         tmp[i] = G[m - i];
118     Prod(F, n, tmp, m, H);
119     for (int i = m; i <= n; ++i)
120         H[i - m] = H[i];
121     for (int i = n - m + 1; i <= n + m; ++i)
122         H[i] = 0;
123 }
124
125 inline void Inv(int F[], int n, int G[])
126 {
127     static int tmp[maxn], res[maxn];
128     res[0] = pow(F[0], mod - 2);
129     for (int i = 1; i - 1 < n; i <= 1)
130     {
131         for (int j = 0; j < 2 * i; ++j)
132             tmp[j] = F[j];
133         Prod(tmp, 2 * i - 1, res, i - 1, tmp);
134         for (int j = 0; j < 2 * i; ++j)
135             tmp[j] = sub(0, tmp[j]);
136         tmp[0] = add(tmp[0], 2);
137         Prod(tmp, 2 * i - 1, res, i - 1, res);
138     }

```

```

139     for (int i = 0; i <= n; ++i)
140         G[i] = res[i];
141 }
142
143 inline void Ln(int F[], int n, int G[])
144 {
145     static int tmp0[maxn], tmp1[maxn];
146     Derivative(F, n, tmp0);
147     Inv(F, n, tmp1);
148     Prod(tmp0, n - 1, tmp1, n, G);
149     Integration(G, n - 1, G, 0);
150 }
151
152 inline void Exp(int F[], int n, int G[])
153 {
154     static int tmp[maxn], res[maxn];
155     res[0] = 1;
156     for (int i = 1; i - 1 < 2 * n; i <= 1)
157     {
158         for (int j = 0; j < i; ++j)
159             tmp[j] = res[j];
160         Ln(tmp, i - 1, tmp);
161         for (int j = 0; j < i; ++j)
162             tmp[j] = sub(F[j], tmp[j]);
163         for (int j = i; j < 2 * i; ++j)
164             tmp[j] = F[j];
165         tmp[0] = add(tmp[0], 1);
166         Prod(tmp, 2 * i - 1, res, i - 1, res);
167     }
168     for (int i = 0; i <= n; ++i)
169         G[i] = res[i];
170 }
171
172 int _Q[maxlog][maxn], _B[maxlog][maxn];
173
174 void dfsM1(int A[], int cur, int dep, int l, int r)
175 {
176     static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
177     if (l == r)
178         _Q[dep][l] = sub(0, A[l]);
179     else

```

```

180     {
181         int mid = (l + r) >> 1;
182         dfsM1(A, cur << 1, dep + 1, l, mid);
183         dfsM1(A, cur << 1 | 1, dep + 1, mid + 1, r);
184         for (int i = l; i <= mid; ++i)
185             tmp0[i - l + 1] = _Q[dep + 1][i];
186         tmp0[0] = 1;
187         for (int i = mid + 1; i <= r; ++i)
188             tmp1[i - mid] = _Q[dep + 1][i];
189         tmp1[0] = 1;
190         Prod(tmp0, mid - l + 1, tmp1, r - mid, tmp2);
191         for (int i = l; i <= r; ++i)
192             _Q[dep][i] = tmp2[i - l + 1];
193     }
194 }
195
196 void dfsM2(int B[], int cur, int dep, int l, int r)
197 {
198     static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
199     if (l == r)
200         B[l] = _B[dep][l];
201     else
202     {
203         int mid = (l + r) >> 1;
204         for (int i = l; i <= r; ++i)
205             tmp0[i - l] = _B[dep][i];
206         for (int i = l; i <= mid; ++i)
207             tmp1[i - l + 1] = _Q[dep + 1][i];
208         tmp1[0] = 1;
209         TransProd(tmp0, r - l, tmp1, mid - l + 1, tmp2);
210         for (int i = mid + 1; i <= r; ++i)
211             _B[dep + 1][i] = tmp2[i - (mid + 1)];
212         for (int i = mid + 1; i <= r; ++i)
213             tmp1[i - mid] = _Q[dep + 1][i];
214         tmp1[0] = 1;
215         TransProd(tmp0, r - l, tmp1, r - mid, tmp2);
216         for (int i = l; i <= mid; ++i)
217             _B[dep + 1][i] = tmp2[i - l];
218         dfsM2(B, cur << 1, dep + 1, l, mid);
219         dfsM2(B, cur << 1 | 1, dep + 1, mid + 1, r);
220     }

```

```

221 }
222
223 inline void Multipoint(int F[], int A[], int n, int B[])
224 {
225     static int tmp[maxn];
226     dfsM1(A, 1, 0, 0, n);
227     for (int i = 1; i <= n + 1; ++i)
228         tmp[i] = _Q[0][i - 1];
229     tmp[0] = 1;
230     Inv(tmp, n + 1, tmp);
231     TransProd(F, 2 * n + 1, tmp, n + 1, _B[0]);
232     dfsM2(B, 1, 0, 0, n);
233 }
234
235 int _P[maxlog][maxn], _F[maxlog][maxn];
236
237 void dfsI1(int A[], int cur, int dep, int l, int r)
238 {
239     static int tmp0[maxn], tmp1[maxn], tmp2[maxn];
240     if (l == r)
241         _P[dep][l] = sub(0, A[l]);
242     else
243     {
244         int mid = (l + r) >> 1;
245         dfsI1(A, cur << 1, dep + 1, l, mid);
246         dfsI1(A, cur << 1 | 1, dep + 1, mid + 1, r);
247         for (int i = 1; i <= mid; ++i)
248             tmp0[i - 1] = _P[dep + 1][i];
249         tmp0[mid - 1 + 1] = 1;
250         for (int i = mid + 1; i <= r; ++i)
251             tmp1[i - mid - 1] = _P[dep + 1][i];
252         tmp1[r - mid] = 1;
253         Prod(tmp0, mid - 1 + 1, tmp1, r - mid, tmp2);
254         for (int i = 1; i <= r; ++i)
255             _P[dep][i] = tmp2[i - 1];
256     }
257 }
258
259 void dfsI2(int B[], int C[], int cur, int dep, int l, int r)
260 {
261     static int tmp0[maxn], tmp1[maxn], tmp2[maxn], tmp3[maxn];

```

```

262     if (l == r)
263         _F[dep][l] = 1LL * B[l] * pow(C[l], mod - 2) % mod;
264     else
265     {
266         int mid = (l + r) >> 1;
267         dfsI2(B, C, cur << 1, dep + 1, l, mid);
268         dfsI2(B, C, cur << 1 | 1, dep + 1, mid + 1, r);
269         for (int i = l; i <= mid; ++i)
270             tmp0[i - l] = _F[dep + 1][i];
271         for (int i = mid + 1; i <= r; ++i)
272             tmp1[i - mid - 1] = _P[dep + 1][i];
273         tmp1[r - mid] = 1;
274         Prod(tmp0, mid - l, tmp1, r - mid, tmp2);
275         for (int i = mid + 1; i <= r; ++i)
276             tmp0[i - mid - 1] = _F[dep + 1][i];
277         for (int i = l; i <= mid; ++i)
278             tmp1[i - l] = _P[dep + 1][i];
279         tmp1[mid - l + 1] = 1;
280         Prod(tmp0, r - mid - 1, tmp1, mid - l + 1, tmp3);
281         for (int i = l; i <= r; ++i)
282             _F[dep][i] = add(tmp2[i - l], tmp3[i - 1]);
283     }
284 }
285
286 inline int Unique(int A[], int B[], int n)
287 {
288     std::set<int> S;
289     S.clear();
290     for (int i = 0; i <= n; ++i)
291         if (!S.count(A[i]))
292         {
293             S.insert(A[i]);
294             A[S.size() - 1] = A[i];
295             B[S.size() - 1] = B[i];
296         }
297     return S.size() - 1;
298 }
299
300 inline void Interpolation(int A[], int B[], int n, int F[])
301 {
302     static int tmp[maxn];

```

```

303     n = Unique(A, B, n);
304     dfsI1(A, 1, 0, 0, n);
305     for (int i = 0; i <= n; ++i)
306         tmp[i] = _P[0][i];
307     tmp[n + 1] = 1;
308     Derivative(tmp, n + 1, tmp);
309     Multipoint(tmp, A, n, tmp);
310     dfsI2(B, tmp, 1, 0, 0, n);
311     for (int i = 0; i <= n; ++i)
312         F[i] = _F[0][i];
313 }
314
315 inline int BostanMori(int F[], int G[], int k, int n)
316 {
317     static int tmp0[maxn], tmp1[maxn];
318     for (; n; n >>= 1)
319     {
320         for (int i = 0; i <= k; ++i)
321             tmp0[i] = (i & 1) ? G[i] : sub(0, G[i]);
322         Prod(F, k, tmp0, k, tmp1);
323         for (int i = (n & 1), j = 0; i <= 2 * k; i += 2, ++j)
324             F[j] = tmp1[i];
325         Prod(G, k, tmp0, k, tmp1);
326         for (int i = 0, j = 0; i <= 2 * k; i += 2, ++j)
327             G[j] = tmp1[i];
328     }
329     return 1LL * F[0] * pow(G[0], mod - 2) % mod;
330 }
331
332 inline int Recurrence(int F[], int A[], int k, int n)
333 {
334     static int tmp[maxn];
335     F[0] = 1;
336     for (int i = 1; i <= k; ++i)
337         F[i] = sub(0, F[i]);
338     Prod(F, k, A, k, tmp);
339     tmp[k] = 0;
340     return BostanMori(tmp, F, k, n);
341 }

```


3.3 FWT

FFT，但是是位运算。

按位或

$$\tilde{F} = \begin{cases} (\tilde{F}_0, \tilde{F}_1 + \tilde{F}_0) & (t > 0) \\ F & (t = 0) \end{cases}$$

$$\epsilon = (1, 0, \dots, 0, 0)$$

$$(F^{-1})_0 = F_0^{-1}$$

$$(F^{-1})_1 = -F_1 \vee F_0^{-1} \vee (F_0 + F_1)^{-1}$$

按位与

$$\tilde{F} = \begin{cases} (\tilde{F}_0 + \tilde{F}_1, \tilde{F}_1) & (t > 0) \\ F & (t = 0) \end{cases}$$

$$\epsilon = (0, 0, \dots, 0, 1)$$

$$(F^{-1})_1 = F_1^{-1}$$

$$(F^{-1})_0 = -F_0 \wedge F_1^{-1} \wedge (F_0 + F_1)^{-1}$$

按位异或

$$\tilde{F} = \begin{cases} (\tilde{F}_0 + \tilde{F}_1, \tilde{F}_0 - \tilde{F}_1) & (t > 0) \\ F & (t = 0) \end{cases}$$

$$\epsilon = (1, 0, \dots, 0, 0)$$

$$(F^{-1})_0 = \frac{(F_0 + F_1)^{-1} + (F_0 - F_1)^{-1}}{2}$$

$$(F^{-1})_1 = \frac{(F_0 + F_1)^{-1} - (F_0 - F_1)^{-1}}{2}$$

4 其他数学

4.1 牛顿迭代

给定形式幂级数 $G(x)$ 。设

$$G(F_0(x)) \equiv 0 \pmod{x^n}$$

令

$$F(x) \equiv F_0(x) - \frac{G(F_0(x))}{G'(F_0(x))} \pmod{x^{2n}}$$

则

$$G(F(x)) \equiv 0 \pmod{x^{2n}}$$

4.2 组合对象符号化公式

无标号集合的 Sequence 构造

$$B(x) = \frac{1}{1 - A(x)}$$

无标号集合的 Multiset 构造（完全背包）

$$B(x) = \text{Exp}(A(x)) = \exp \sum_{n \geq 1} \frac{A(x^n)}{n}$$

无标号集合的 Powerset 构造（01 背包）

$$B(x) = \overline{\text{Exp}}(A(x)) = \exp \sum_{n \geq 1} \frac{(-1)^{n-1} A(x^n)}{n}$$

有标号集合的 Sequence 构造

$$B(x) = \frac{1}{1 - A(x)}$$

有标号集合的 Set 构造

$$B(x) = \exp A(x)$$

4.3 拉格朗日反演

设形式幂级数 $F(x), G(x)$ 互为复合逆，有

$$[x^n]G(x) = \frac{1}{n} [x^{n-1}] \left(\frac{x}{F(x)} \right)^n$$

设另有一个形式幂级数 $H(x)$ ，有

$$[x^n]H(G(x)) = \frac{1}{n}[x^{n-1}]H'(x) \left(\frac{x}{F(x)} \right)^n$$

4.4 Burnside 引理 & Polya 定理

有如下 Burnside 引理

$$|X/H| = \frac{1}{|H|} \sum_{g \in H} |X^g|$$

人话版本：集合 X 在群 $\langle H, \times \rangle$ 作用下的等价类数量等于 H 中所有元素作用在集合 X 上时的不动点数量的算术平均值。

如果 X 是使用 m 中颜色对一个大小为 n 的集合进行染色的方案数， H 是 n 阶置换群。有如下 Polya 定理

$$|X/H| = \frac{1}{|H|} \sum_{g \in H} m^{\#(g)}$$

其中 $\#(g)$ 表示 g 的轮换数。

5 数据结构

5.1 可持久化 WBLT

写的非常丑陋。估计常数巨大。

空间开销至少 20 倍 $n \log n$ 。

```

1  const double alpha = 0.293, beta = 0.707;
2
3  class WBLT
4  {
5      public:
6          struct node
7          {
8              node *lc, *rc;
9              int min, max, size;
10         } mempool[80 * maxn + 5], *memtop, *root[maxn + 5];
11
12         WBLT()
13         {
14             memtop = mempool;
15             root[0] = NewNode(NewNode(-2147483647), NewNode(2147483647));
16         }
17
18         inline node *NewNode(int val)

```

```

19     {
20         node *p = memtop++;
21         p->lc = NULL;
22         p->rc = NULL;
23         p->min = val;
24         p->max = val;
25         p->size = 1;
26         return p;
27     }
28
29     inline node *NewNode(node *p, node *q)
30     {
31         node *r = memtop++;
32         r->lc = p;
33         r->rc = q;
34         r->min = p->min;
35         r->max = q->max;
36         r->size = p->size + q->size;
37         return r;
38     }
39
40     std::pair<node *, node *> SplitByRank(node *p, int rank)
41     {
42         if (p == NULL)
43             return std::make_pair((node *)NULL, (node *)NULL);
44         else if (rank == 0)
45             return std::make_pair((node *)NULL, p);
46         else if (rank == p->size)
47             return std::make_pair(p, (node *)NULL);
48         else if (rank < p->lc->size)
49         {
50             std::pair<node *, node *> ans = SplitByRank(p->lc, rank);
51             ans.second = Merge(ans.second, p->rc);
52             return ans;
53         }
54         else
55         {
56             std::pair<node *, node *> ans = SplitByRank(p->rc, rank - p
                    ->lc->size);
57             ans.first = Merge(p->lc, ans.first);
58             return ans;

```

```

59     }
60 }
61
62 std::pair<node *, node *> SplitByVal(node *p, int val)
63 {
64     if (p == NULL)
65         return std::make_pair((node *)NULL, (node *)NULL);
66     else if (val < p->min)
67         return std::make_pair((node *)NULL, p);
68     else if (val >= p->max)
69         return std::make_pair(p, (node *)NULL);
70     else if (val < p->lc->max)
71     {
72         std::pair<node *, node *> ans = SplitByVal(p->lc, val);
73         ans.second = Merge(ans.second, p->rc);
74         return ans;
75     }
76     else
77     {
78         std::pair<node *, node *> ans = SplitByVal(p->rc, val);
79         ans.first = Merge(p->lc, ans.first);
80         return ans;
81     }
82 }
83
84 node *Merge(node *x, node *y)
85 {
86     if (x == NULL)
87         return y;
88     else if (y == NULL)
89         return x;
90     else if (std::max(x->size, y->size) <= beta * (x->size + y->size))
91         return NewNode(x, y);
92     else if (x->size > y->size)
93     {
94         if (x->lc->size > alpha * (x->size + y->size))
95             return NewNode(x->lc, Merge(x->rc, y));
96         else
97             return Merge(Merge(x->lc, x->rc->lc), Merge(x->rc->rc, y));
98     }
99 }

```

```

98     }
99     else
100    {
101        if (y->rc->size > alpha * (x->size + y->size))
102            return NewNode(Merge(x, y->lc), y->rc);
103        else
104            return Merge(Merge(x, y->lc->lc), Merge(y->lc->rc, y->rc
105                        ));
106    }
107
108    inline void Insert(int id, int val, int cur)
109    {
110        std::pair<node *, node *> pair = SplitByVal(root[id], val);
111        root[cur] = Merge(Merge(pair.first, NewNode(val)), pair.second);
112    }
113
114    inline void Erase(int id, int val, int cur)
115    {
116        std::pair<node *, node *> pair1 = SplitByVal(root[id], val);
117        std::pair<node *, node *> pair2 = SplitByRank(pair1.first, pair1
118            .first->size - 1);
119        pair1.first = val == pair2.second->max ? pair2.first : Merge(
120            pair2.first, pair2.second);
121        root[cur] = Merge(pair1.first, pair1.second);
122    }
123
124    inline int QueryRank(int id, int val)
125    {
126        node *x = SplitByVal(root[id], val - 1).first;
127        return x->size + 1;
128    }
129
130    inline int QueryKth(int id, int rank)
131    {
132        node *x = SplitByRank(root[id], rank).first;
133        node *y = SplitByRank(x, rank - 1).second;
134        return y->max;
135    }
136 } wblt;

```

6 图论

6.1 矩阵树定理

设图 G 无自环。令图 G 的度数/出度/入度矩阵分别为 $D(G), D^{\text{out}}(G), D^{\text{in}}(G)$ ，邻接矩阵为 $A(G)$ ，生成树数量为 $t(G)$ ，以点 k 为根的根向/叶向树形图数量分别为 $t^{\text{root}}(G, k), t^{\text{leaf}}(G, k)$ 。

$$t(G) = \det [D(G) - A(G)] \begin{pmatrix} 1, 2, \dots, i-1, i+1, \dots, n \\ 1, 2, \dots, i-1, i+1, \dots, n \end{pmatrix}$$

$$t^{\text{root}}(G, k) = \det [D^{\text{out}}(G) - A(G)] \begin{pmatrix} 1, 2, \dots, k-1, k+1, \dots, n \\ 1, 2, \dots, k-1, k+1, \dots, n \end{pmatrix}$$

$$t^{\text{leaf}}(G, k) = \det [D^{\text{in}}(G) - A(G)] \begin{pmatrix} 1, 2, \dots, k-1, k+1, \dots, n \\ 1, 2, \dots, k-1, k+1, \dots, n \end{pmatrix}$$

6.2 最大流

```

1  int n, m, s, t, ec = 1, maxHeight, curHeight;
2  int des[2 * MAXM + 5], suc[2 * MAXM + 5], cap[2 * MAXM + 5], last[MAXN + 5];
3  int que[MAXN + 5], height[MAXN + 5], cnt[MAXN + 5], extra[MAXN + 5];
4  std::list<int> list[MAXN + 5];
5  std::list<int>::iterator pos[MAXN + 5];
6  std::vector<int> vector[MAXN + 5];
7
8  inline void connect(int u, int v, int c)
9  {
10     des[++ec] = v;
11     suc[ec] = last[u];
12     cap[ec] = c;
13     last[u] = ec;
14 }
15
16 inline void PushEdge(int u, int e)
17 {
18     int flow = std::min(extra[u], cap[e]);
19     extra[u] -= flow;
20     extra[des[e]] += flow;
21     cap[e] -= flow;
22     cap[e ^ 1] += flow;
23     if (extra[des[e]] > 0 && extra[des[e]] <= flow)
24         vector[height[des[e]]].push_back(des[e]);
25 }
```

```

26
27 inline void BFS()
28 {
29     for (int i = 1; i <= n; ++i)
30         height[i] = n + 1;
31     height[t] = 0;
32     int head = 0, tail = 0;
33     que[++tail] = t;
34     for (; head < tail; )
35     {
36         int u = que[++head];
37         for (int i = last[u]; i; i = suc[i])
38             if (height[des[i]] == n + 1 && cap[i ^ 1] > 0)
39             {
40                 height[des[i]] = height[u] + 1;
41                 ++cnt[height[des[i]]];
42                 que[++tail] = des[i];
43             }
44     }
45     for (int i = 0; i <= n; ++i)
46     {
47         vector[i].clear();
48         list[i].clear();
49     }
50     for (int i = 1; i <= n; ++i)
51         if (height[i] < n + 1)
52         {
53             pos[i] = list[height[i]].insert(list[height[i]].begin(), i);
54             if (extra[i] > 0) vector[height[i]].push_back(i);
55         }
56     maxHeight = height[que[tail - 1]];
57     curHeight = maxHeight;
58 }
59
60 inline void Push(int u)
61 {
62     int nextHeight = n + 1;
63     for (int i = last[u]; i; i = suc[i])
64         if (cap[i] > 0)
65         {
66             if (height[des[i]] + 1 == height[u])

```



```

67         {
68             PushEdge(u, i);
69             if (extra[u] == 0) return;
70         }
71         else
72             nextHeight = std::min(nextHeight, height[des[i]] + 1);
73     }
74     if (cnt[height[u]] == 1)
75     {
76         for (; maxHeight >= height[u]; list[maxHeight].clear(), --
maxHeight)
77             for (int v : list[maxHeight])
78             {
79                 --cnt[height[v]];
80                 height[v] = n + 1;
81             }
82     }
83     else
84     {
85         --cnt[height[u]];
86         list[height[u]].erase(pos[u]);
87         height[u] = nextHeight;
88         if (height[u] == n + 1) return;
89         ++cnt[height[u]];
90         pos[u] = list[height[u]].insert(list[height[u]].begin(), u);
91         curHeight = height[u];
92         maxHeight = std::max(maxHeight, curHeight);
93         vector[height[u]].push_back(u);
94     }
95 }
96
97 inline int HLPP()
98 {
99     // height[s] = n;
100     extra[s] = INF;
101     extra[t] = -INF;
102     for (int i = last[s]; i; i = suc[i])
103         PushEdge(s, i);
104     BFS();
105     for (; curHeight >= 0; )
106         if (vector[curHeight].empty())

```

```

107         --curHeight;
108     else
109     {
110         int u = vector[curHeight].back();
111         vector[curHeight].pop_back();
112         Push(u);
113     }
114     return extra[t] + INF;
115 }

```

6.3 费用流

```

1  inline bool spfa()
2  {
3      for (int i = 1; i <= n; ++i)
4          dist[i] = 2E9;
5      dist[s] = 0;
6      flow[s] = 2E9;
7      ex[h[1] = s] = true;
8      for (int head = 0, tail = 1; head < tail; )
9      {
10         int x = h[++head];
11         ex[x] = false;
12         for (int i = last[x]; i >= 0; i = suc[i])
13             if (cap[i] > 0 && dist[des[i]] > dist[x] + cost[i])
14             {
15                 dist[des[i]] = dist[x] + cost[i];
16                 flow[des[i]] = min(flow[x], e[i].cap);
17                 pre[des[i]] = i;
18                 if (!ex[des[i]])
19                     ex[h[++tail] = des[i]] = true;
20             }
21     }
22     return dist[t] < 2E9;
23 }
24
25 inline void addflow()
26 {
27     for (int cur = t; cur != s; cur = des[pre[cur] ^ 1])
28     {
29         cap[pre[cur]] -= flow[t];

```

```

30     cap[pre[cur] ^ 1] += flow[t];
31 }
32 maxflow += flow[t];
33 mincost += dist[t] * flow[t];
34 }

```

6.4 带下界可行流

令

$$\text{cap}(e) = \text{upper}(e) - \text{lower}(e)$$

$$w(u) = \sum_{(v,u) \in E} \text{lower}(v,u) - \sum_{(u,v) \in E} \text{lower}(u,v)$$

如果 $w(u) > 0$ ，连附加边 $(S, u, w(u))$ ；如果 $w(u) < 0$ ，连附加边 $(u, T, -w(u))$ 。
如果这些附加边没有满载则无解。

6.5 带下界最大流

在可行流的基础上连边 $(T, S, +\infty)$ 。

6.6 六元组模型

连边 $(S, i, a_i), (i, T, b_i), (S, x_j, D_j), (S, y_j, E_j), (x_j, T, F_j), (y_j, T, G_j), (x_j, y_j, P_j), (y_j, x_j, Q_j)$
满足

$$\begin{cases} D_j + E_j &= (c_{1,1})_j \\ F_j + G_j &= (c_{2,2})_j \\ D_j + G_j + Q_j &= (c_{1,2})_j \\ E_j + F_j + P_j &= (c_{2,1})_j \end{cases}$$

D_j 和 F_j ， E_j 和 G_j 可以同时加上一个数调整成非负数。

$$P_j = Q_j = \frac{(c_{1,2})_j + (c_{2,1})_j - (c_{1,1})_j - (c_{2,2})_j}{2}$$

这个东西非负则可做。

7 字符串

7.1 KMP

```
1 for (int i = 2, p = 0; i <= len2; ++i)
2 {
3     for (; p > 0 && s2[p + 1] != s2[i]; p = kmp[p]);
4     if (s2[p + 1] == s2[i])
5         ++p;
6     kmp[i] = p;
7 }
8 for (int i = 1, p = 0; i <= len1; ++i)
9 {
10    for (; p > 0 && s2[p + 1] != s1[i]; p = kmp[p]);
11    if (s2[p + 1] == s1[i])
12        ++p;
13    if (p == len2)
14        p = kmp[p];
15 }
```

7.2 AC 自动机

```
1 class AC
2 {
3     public:
4     int size;
5     int ch[maxn + 5][26], fail[maxn + 5];
6
7     inline void init()
8     {
9         size = 0;
10    }
11
12    inline int newNode()
13    {
14        int u = ++size;
15        for (int i = 0; i < 26; ++i)
16            ch[u][i] = 0;
17        fail[u] = 0;
18        return u;
19    }
20
21    inline void insert(char s[])
22    {
23        int len = strlen(s + 1);
```

```

24     for (int i = 1, u = 0; i <= len; ++i, u = ch[u][s[i] - 'a'])
25         if (ch[u][s[i] - 'a'] == 0)
26             ch[u][s[i] - 'a'] = newNode();
27     }
28
29     inline void prework()
30     {
31         std::queue<int> que;
32         for (; !que.empty(); que.pop());
33         for (int i = 0; i < 26; ++i)
34             if (ch[0][i])
35                 que.push(ch[0][i]);
36         for (; !que.empty(); )
37         {
38             int x = que.front();
39             que.pop();
40             for (int c = 0; c < 26; ++c)
41                 if (ch[x][c] > 0)
42                 {
43                     fail[ch[x][c]] = ch[fail[x]][c];
44                     que.push(ch[x][c]);
45                 }
46                 else
47                     ch[x][c] = ch[fail[x]][c];
48         }
49     }
50 };

```

7.3 倍增 SA

```

1  for (int i = 1; i <= n; ++i)
2      ++tnk[fst[i] = a[i]];
3  for (int i = 1; i <= max; ++i)
4      tnk[i] += tnk[i - 1];
5  for (int i = n; i >= 1; --i)
6      sa[tnk[fst[i]]--] = i;
7  for (int k = 1; ; k <= 1)
8  {
9      int cnt = 0;
10     for (int i = n - k + 1; i <= n; ++i)
11         snd[++cnt] = i;

```

```

12     for (int i = 1; i <= n; ++i)
13         if (sa[i] > k)
14             snd[++cnt] = sa[i] - k;
15     for (int i = 1; i <= max; ++i)
16         tnk[i] = 0;
17     for (int i = 1; i <= n; ++i)
18         ++tnk[fst[i]];
19     for (int i = 1; i <= max; ++i)
20         tnk[i] += tnk[i - 1];
21     for (int i = n; i >= 1; --i)
22         sa[tnk[fst[snd[i]]]--] = snd[i];
23     for (int i = 1; i <= n; ++i)
24         snd[i] = fst[i];
25     fst[sa[1]] = cnt = 1;
26     for (int i = 2; i <= n; ++i)
27         fst[sa[i]] = snd[sa[i]] == snd[sa[i - 1]] && snd[sa[i] + k] ==
28             snd[sa[i - 1] + k] ? cnt : ++cnt;
29     if (cnt == n)
30         break;
31     else
32         max = cnt;
33 }
34 for (int i = 1; i <= n; ++i)
35     rnk[sa[i]] = i;
36 for (int i = 1, j = 0; i <= n; ++i)
37     if (rnk[i] > 1)
38     {
39         j && (--j);
40         int pos = sa[rnk[i] - 1];
41         for (; i + j <= n && pos + j <= n && a[i + j] == a[pos + j]; ++j)
42             ;
43         het[rnk[i]][0] = j;
44     }
45 for (int j = 1, p = 2; p <= n; ++j, p <= 1)
46     for (int i = 1; i + p - 1 <= n; ++i)
47         het[i][j] = std::min(het[i][j - 1], het[i + (p >> 1)][j - 1]);
48 for (int i = 2; i <= n; ++i)
49     lg[i] = lg[i >> 1] + 1;

```

注意事项:

1. sa[i] 的含义是排名为 i 的后缀的位置

2. $\text{rnk}[i]$ 的含义是位置为 i 的后缀的排名
3. $\text{het}[i]$ 的含义是 $\text{LCP}(sa[i], sa[i-1])$
4. rnk 数组需要开到 2 倍大小

7.4 SAM

```
1 class SAM
2 {
3     public:
4     int size, last;
5     int ch[2 * maxn + 5][26], fail[2 * maxn + 5], len[2 * maxn + 5];
6
7     inline void init()
8     {
9         size = last = 1;
10        for (int i = 0; i < 26; ++i)
11            ch[1][i] = 1;
12        fail[1] = len[1] = 0;
13    }
14
15    inline int newNode()
16    {
17        int u = ++size;
18        for (int i = 0; i < 26; ++i)
19            ch[u][i] = 1;
20        fail[u] = 1, len[u] = 0;
21        return u;
22    }
23
24    inline int append(int c)
25    {
26        int cur = newNode(), u = last;
27        len[cur] = len[u] + 1;
28        for (; u > 0 && ch[u][c] == 1; ch[u][c] = cur, u = fail[u]);
29        if (u == 0)
30            fail[cur] = 1;
31        else
32        {
33            int v = ch[u][c];
34            if (len[v] == len[u] + 1)
```

```

35         fail[cur] = v;
36     else
37     {
38         int clone = newNode();
39         for (int i = 0; i < 26; ++i)
40             ch[clone][i] = ch[v][i];
41         fail[clone] = fail[v];
42         len[clone] = len[u] + 1;
43         fail[v] = fail[cur] = clone;
44         for (; u > 0 && ch[u][c] == v; ch[u][c] = clone, u =
45             fail[u]);
46     }
47     return last = cur;
48 }
49 };

```

7.5 manacher

```

1  for (int i = n; i >= 0; --i)
2  {
3      S[i << 1] = S[i];
4      S[i << 1 | 1] = '#';
5  }
6  int m = n << 1 | 1;
7  for (int i = 1, maxRight = 0, mid = 0; i <= m; ++i)
8  {
9      ans[i] = std::max(std::min(ans[(mid << 1) - i], maxRight - i + 1),
10         1);
11     for (; i - ans[i] >= 1 && i + ans[i] <= m && S[i - ans[i]] == S[i +
12         ans[i]]; ++ans[i]);
13     if (i + ans[i] - 1 > maxRight)
14     {
15         maxRight = i + ans[i] - 1;
16         mid = i;
17     }
18 }

```