一、数论

1. 素数

$\pi(n)$：：表示 n 以下的素数个数

$\lim\limits_{n\to\infty}\frac{\pi(n)}{\frac{n}{ln(n)}}=1,\ \pi(n)=O(\frac{n}{log(n)})$

P_n：第 n 个素数，$P_n=O(nlog(n))$

调和级数累和：$\sum\limits_{i=1}^{n}\frac{1}{i}=O(log(n))=ln(n)+O(1)$

素数倒数和：$\sum\limits_{p=2}^{n}\frac{1}{p}=O(loglog(n))$，$p$ 为素数

2. 公约数与公倍数

$a=\sum\limits_{i=1}^{k}p_i^{u_i},\ b=\sum\limits_{i=1}^{k}p_i^{v_i}$

$gcd(a,b)=(a,b)=\sum\limits_{i=1}^{k}p_i^{min(u_i,v_i)}$

$lcm(a,b)=[a,b]=\sum\limits_{i=1}^{k}p_i^{max(u_i,v_i)}$

$gcd(a,b)*lcm(a,b)=\sum\limits_{i=1}^{k}p_i^{min(u_i,v_i)+max(u_i,v_i)}=\sum\limits_{i=1}^{k}p_i^{u_i+v_i}=a*b$

$[a,b]=\frac{a*b}{(a,b)}$

3. 欧几里得算法（辗转相除法）

$(a,b)=(a-b,b)=(a\bmod b,b),\ a\bmod b=a-\lfloor\frac{a}{b}\rfloor*b$

$d|a,d|b\to d|(a-b)$

$a\bmod b\le\frac{a}{2}$，每次取模，数值减少一半

- $a\ge2*b,a\bmod b<b\le\frac{a}{2}$
- $a<2*b,a\bmod b=a-b\le\frac{a}{2}$
- 复杂度为 $O(log(n))$

裴蜀定理

$a*u+b*v=d$，u、v 有整数解当且仅当 $gcd(a,b)|d$

扩展欧几里得算法（Exgcd）

$a*u+b*v=gcd(a,b)$ 必定有解 $(a\geq b)$

$\rightarrow (a \bmod b)*u+(\lfloor \frac{a}{b} \rfloor+b)*v=gcd(a,b)$

$\rightarrow$ 令$a_1=(a \bmod b)$，令$b_1=(\lfloor \frac{a}{b} \rfloor+b)$，$a_1*u+b_1*v=gcd(a,b)$

$\rightarrow$ 持续辗转相除，递归求解

- 特解与通解

$ax+by=d$，特解 $x=x_0,y=y_0$

$\rightarrow$ 通解为 $x=x_0+\frac{b}{(a,b)}*t,y=y_0-\frac{a}{(a,b)}*t$

4. 同余

$a\equiv b\ (\mathrm{mod}\ m)\Leftrightarrow m|(a-b)$

- 性质 1

$a\equiv b\ (\mathrm{mod}\ m)$

$a\equiv b\ (\mathrm{mod}\ n)$

$\rightarrow a\equiv b\ (\mathrm{mod}\ [n,m])$

- 性质 2

$(k,m)=d$

$k*a\equiv k*b\ (\mathrm{mod}\ m)$

$\rightarrow\ \ a\equiv b\ (\mathrm{mod}\ \frac{m}{d})$

同余式两边乘除仍成立。

证明：

$\rightarrow m|(ka-kb)$

$\rightarrow d=(k,m)\Rightarrow 1=(\frac{k}{d},\frac{m}{d})$

$\rightarrow \frac{m}{d} * d \mid (\frac{k}{d}) * d * (a-b)$

$\rightarrow \frac{m}{d} \mid \frac{k}{d} * (a-b)$

$\rightarrow \frac{m}{d}$ 与 $\frac{k}{d}$ 互质，$\frac{m}{d} \mid (a-b)$，得证

解同余方程

$ax \equiv b \pmod m \Leftrightarrow ax - my = b \Leftrightarrow (a,m) | b$

使用 $Exgcd$ 进行求解。

h

5. 欧拉函数

$\varphi(n)$ —— 1 ~ n 之间与 n 互质的数的个数

- 性质 1

$\varphi(p) = p-1$，p 为质数

$\varphi(p^e) = p^e - (p \text{ 的倍数}) = p^e - p^{e-1} = (1-\frac{1}{p}) * p^e$

- 性质 2

$n = \sum_{i=1}^{k} p_i^{e^i}$，$(e_i \geq 1)$

$\varphi(n) = n * \prod_{i=1}^{k} (1-\frac{1}{p_i})$

- 性质 3 —— 欧拉定理

$(a,m) \equiv 1$，a 与 m 互质

$\rightarrow a^{\varphi(m)} \equiv 1 \pmod m$

大致证明：

$(x_i, m) \equiv 1 \ (1 \leq i \leq \varphi(m))$，则 $ax_i \equiv 1 \pmod m$

$\prod_{i=1}^{\varphi(m)} ax_i = a^{\varphi(m)} \sum_{i=1}^{\varphi(m)} x_i \equiv 1 \pmod m$

$\rightarrow a^{\varphi(m)} \equiv 1 \pmod m$

- 性质 4 —— 费马小定理

$a^{p-1} \equiv 1 \pmod p$

- 性质 5 —— 欧拉降幂 【当 $(a,m) \neq 1$ 时】

第二、三条是广义欧拉降幂，广义降幂是通用的，$gcd(a,p)=1$ 也适用。

$$a^b = \begin{cases} a^{b \bmod \varphi(p)} & gcd(a,p)=1 \\ a^b & gcd(a,p)\neq 1, b<\varphi(p) \\ a^{b \bmod \varphi(p)+\varphi(p)} & gcd(a,p)\neq 1, b\geq \varphi(p) \end{cases} \pmod p$$

6. 逆元

费马小定理或者 $exgcd$ 求解，通常 $exgcd$ 更快。

- 求解 $1!$ ~ $n!$ 的逆元

先 $O(n)$ 求出 $inv[n!]$，$inv[(n-1)!] = inv[n!] * n$，即可 $O(n)$ 求出所有的逆元。

- 逆元递推公式

$inv[i] = (p - \lfloor \frac{p}{i} \rfloor) * inv[p \bmod i]$

证明：

$p\%p = (\lfloor \frac{p}{i} \rfloor * i + p\%i)\%p$

$\longrightarrow i * \lfloor \frac{p}{i} \rfloor = -(p\%i)\%p$

$\longrightarrow \frac{-\lfloor \frac{p}{i} \rfloor}{p\%i} = \frac{1}{i}$

$\longrightarrow inv[i] = (p - \lfloor \frac{p}{i} \rfloor) * inv[p\%i]$

7. 中国剩余定理

- 定理概述

$$\begin{cases} x \equiv a_1 (\bmod\ m_1) \\ x \equiv a_2 (\bmod\ m_2) \\ \dots \\ x \equiv a_k (\bmod\ m_k) \end{cases}$$

当 $m_1, m_2, \dots, m_k$ 两两互质，则方程组有解，解在 $(\bmod)\ m_1 * m_2 * \dots * m_k$ 下唯一。

- 增量法求解

思想：一开始只有一条方程，想办法把另一条方程加进去。

$x = m_1 * t + a_1$，代入第二条，$m_1 * t + a_1 \equiv a_2 (\%\ m_2)$，求解出 t，再代回去，即可合并两条式子。再依据这个方法不断合并其它式子，最终合并成一条式子，即可高效求解。

8. 组合数取模

$C(a, b) \% p$

- 当 a、b 都很小

利用杨辉三角，$C(n, m) = C(n - 1, m) + C(n - 1, m - 1)$

- p 是大素数，且 $p > a, b$

$C(a, b) = \frac{a!}{(a - b)!b!}$，可以 $O(n)$ 时间内预处理出所有数阶乘的逆元。

- p 是小素数（ $Lucas$ 定理）

p 是小素数，可能会出现分母与 p 不互质，导致无法求取逆元。比如：$p = 97, a = 97$，则逆元不存在，预处理逆元方法失效。

`Lucas 定理:`

$C(a, b) = C(\lfloor \frac{a}{p} \rfloor, \lfloor \frac{b}{p} \rfloor) * C(a \% p, b \% p)\ (\% p)$

$C(a, b) \equiv 0$ 当且仅当 $a < b$

`推论`

将 a 和 b 写成 p 进制，则 $a = a_0 * p^0 + a_1 * p^1 + \ldots + a_k * p^k$，
$b = b_0 * p^0 + b_1 * p^1 + \ldots + b_k * p^k$，且 $0 < a_i, b_i < p$。

则 $C(a, b) = C(a_0, b_0) * C(a_1, b_1) * \ldots * C(a_k, b_k)\ \% p$，因此预处理 $[1, p]$ 之间阶乘逆元即可。

快速乘

$(long\ long)\ *\ (long\ long)\ \%\ (long\ long)$

$x * y\ \%\ mod = (x * y - (ll)(((long\ double)x * y) / mod) * mod)\ \%\ mod$

在 $mod = 10^{18}$ 时可行，当 $mod = 2 * 10^{18}$ 时，$long\ double$ 精度可能撑不住，会出问题。

9. $Pollard\ Rho$ 算法

$O(n^{\frac{1}{4}})$ 时间复杂度内对一个数进行唯一分解。

10. 筛

- 埃氏筛

时间复杂度：$O(nloglogn)$

- 线性筛

核心思想：$v[i]$ 表示 i 的最小质因子，每个 $v[i]$ 只会被修改一遍，因此是线性的。

时间复杂度：$O(n)$

11. 积性函数

f 定义在正整数域上，对于 $(a,b)=1$，有 $f(ab)=f(a)*f(b)$，则 f 为积性函数。

- $\varphi(n)$

$\varphi(n)=n*\prod_{i=0}^{k}(1-\frac{1}{p_i})$，$1\sim n$ 中与 n 互质的个数。

$n=ab$ 且 $(a,b)=1$，则 n 唯一分解的质数分布在 a、b 之间，不会有交集，则 $\varphi(n)=\varphi(a)*\varphi(b)$。

- $d(n)$

$n=p_1^{e_1}*p_2^{e_2}*...*p_k^{e_k}$，则 $d(n)=(e_1+1)*(e_2+1)*...*(e_k+1)$，$n$ 个因子个数。

$n=ab$ 且 $(a,b)=1$，则 n 唯一分解的质数分布在 a、b 之间，不会有交集，则 $d(n)=d(a)*d(b)$。

二、组合数学

1. 盒子放小球问题

n 个盒子，m 个球。

- n 个小球有区别，m 个盒子有区别

1. `允许空盒。` 方案数为 m^n 。
2. `不允许空盒。` 考虑为 m 个盒子相同时的方案数再乘以 $m!$ ，答案即为 $S(n,m)*m!$，S 代表第二类斯特林数。

- n 个小球有区别，m 个盒子无区别

1. 允许空盒。 枚举放了 k 个盒，方案数为 $\sum_{k=1}^{m} S(n,k)$。
2. 不允许空盒。 $S(n,m)$

- n 个小球无区别，m 个盒子有区别

1. 允许空盒。 “隔板法”，$C(n+m-1,n-1)$。
2. 不允许空盒。 “插板法”，$C(n-1,m-1)$。

- n 个小球无区别，m 个盒子无区别

1. 允许空盒。 划分数问题。$dp[i][j]$ 表示 i 个球，j 个盒子的方案数。转移方程为 $dp[i][j]=dp[i-j][j]+dp[i][j-1]\ (i\geq j)$。如果 $n<m$，则答案为 $dp[n][n]$，否则为 $dp[n][m]$。
   - 当 $i\leq j$ 时，可以分为两种情况。第一种情况，至少有一个盒子空着，则 $dp[i][j]=dp[i][j-1]$。
   - 第二种情况，所有盒子都有球，我们可以从每个盒子中拿掉一个球而不影响总的放法。$dp[i][j]=dp[i-j][j]$。
2. 不允许空盒。 即在每个盒子中放入一个球，然后转成上述 $n-m$ 个小球，m 个盒子的情况。

1. 数学

1.1 莫比乌斯反演

$u(N)$ 取值

- $u(1)=1$
- N中所有质因子各不相同，若N有偶数个质因子，$u(N)=1$，若N有奇数个质因子，$u(N)=-1$
- 其他情况$u(N)=0$
- $\sum\limits_{d|n}\frac{u(d)}{d}=\frac{\varphi(n)}{n}$
- $$\sum_{d|n}u(d)=\begin{cases}1 & (n=1)\\ 0 & (n>1)\end{cases}$$

反演形式

形式一（约数反演）：$F(n)=\sum\limits_{d|n}f(d)\Longrightarrow f(n)=\sum\limits_{d|n}u(d)F(\frac{n}{d})$

形式二（倍数反演）：$F(n)=\sum\limits_{n|d}f(d)\Longrightarrow f(n)=\sum\limits_{n|d}u(\frac{d}{n})F(d)$

积性函数

- $f(xy)=f(x)f(y)$, 当 $gcd(x,y)=1$ 时成立, 则$f(n)$为积性函数。[$\varphi(n)$、$u(n)$ 均为积性函数]
- $f(xy)=f(x)f(y)$, 对于任意 x 和 y 均成立, 则$f(n)$为完全积性函数。

总体原则

- 对于一些函数 $f(n)$，如果我们很难直接求出它的值，而容易求出倍数和或约数和 $F(n)$，那么我们可以通过莫比乌斯反演求得 $f(n)$ 的值。

例题

- $x\in[1,n],\ y\in[1,m]$ 求存在多少对 <x,y>，使得 $gcd(x,y)=k$ 。[HDU-1695]

$f(n)$ 表示有多少对 <x,y> 使得 $gcd(x,y)=n$，$F(n)$ 表示有多少对 <x,y> 使得 $gcd(x,y)=n$ 的倍数。

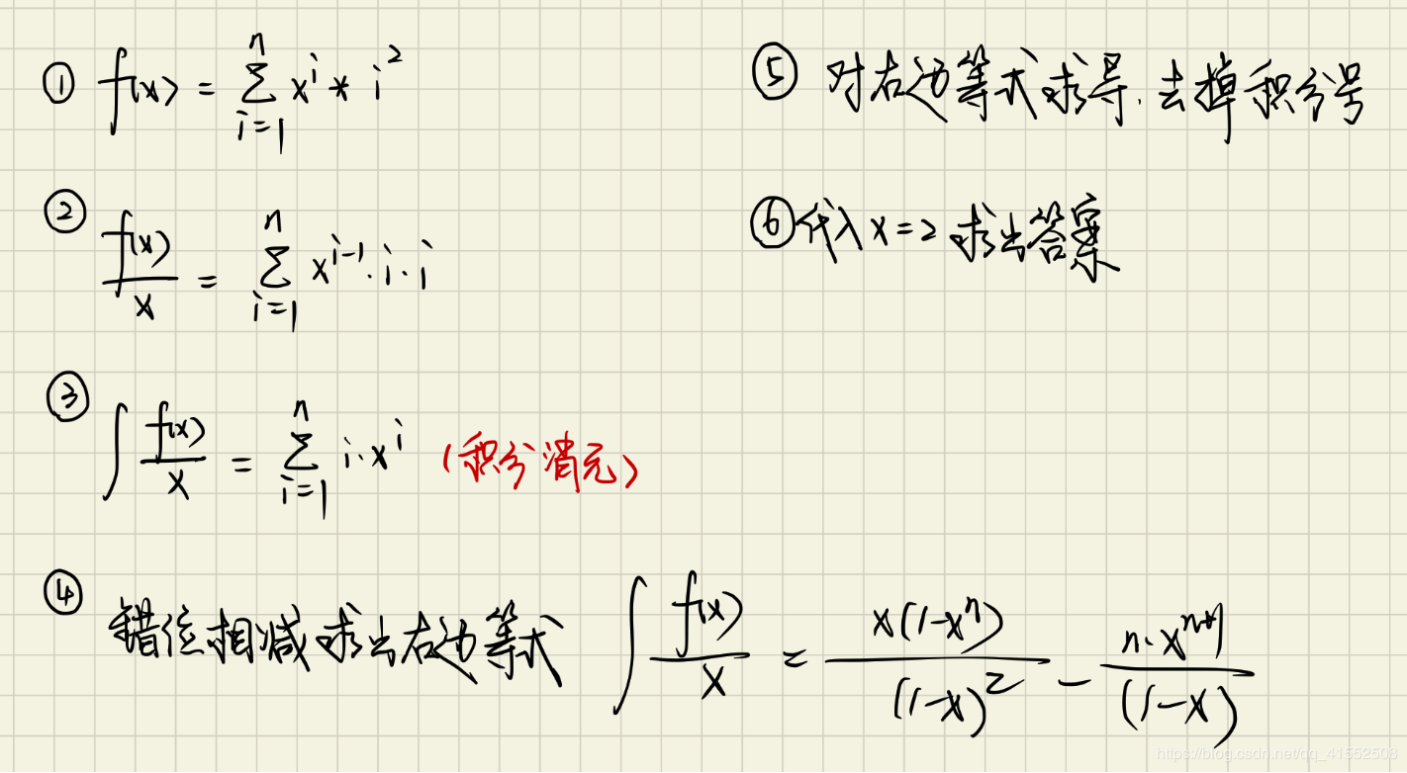
1.2 常见公式推导

- $F(n)=\sum_{x=1}^{n} x*2^x$

错位相减，直接求出答案 $F(n)=(n-1)*2^{n+1}+2.$

- $F(n)=\sum_{x=1}^{n} 2^x*x^2$

先构造 $f(x)=\sum_{i=1}^{n} x^i*i^2$，即直接求出$x$的通式，然后再将$x$替换成$2$。利用高数计算无穷级数的方法，先构造$\frac{f(x)}{x}$，然后先积分再求导，得出答案
$F(n)=2^{n+1}*(n^2-2*n+3)-6$。具体过程见下图。



- 常见组合数公式
  - $\sum_{i=1}^{n} 2^i i^2=(n^2-2n+3)2^{n+1}-6$
  - $\sum_{i=1}^{n} 2^i i=(n-1)2^{n+1}+2$
  - $1^2+2^2+3^2+\cdots+n^2=\frac{n(n+1)(2*n+1)}{6}$
  - $1^3+2^3+3^3+\cdots+n^3=\frac{n^2(n+1)^2}{4}$
  - $m*C_n^m=n*C_{n-1}^{m-1}$
  - $C_n^m=C_{n-1}^{m-1}+C_{n-1}^m$
  - $\sum_{i=0}^{k} C_{n+i}^n=C_{n+k+1}^{n+1}$
  - $\sum_{i=1}^{n} iC_n^i=n*2^{n-1}$
  - $\sum_{i=1}^{n} i^2C_n^i=n*(n+1)*2^{n-2}$

- $\sum_{i=1}^{n}(-1)^{i-1}\frac{C_n^i}{i}=\sum_{i=1}^{n}\frac{1}{i}$
- $\sum_{i=0}^{n}(C_n^i)^2=C_{2n}^n$
- $(r+1)^k=\sum_{i=0}^{k}C(k,i)*r^i$
- $\sum_{i=0,i\ is\ odd}^{n}C(n,i)=\sum_{i=0,i\ is\ even}^{n}C(n,i)=2^{n-1}$
- $\sum_{i=0}^{n}C(i,k)=C(n+1,k+1)$
- $\sum_{i=0}^{k}C(n+i,i)=C(n+k+1,k)$
- $\sum_{k=0}^{n}C(n,k)^2=C(2n,n)$
- $a+b+c=z$, z 为常数，一共有 $C(z+2,2)$ 种 <a,b,c> 分配方案。

1.3 范德蒙恒等式

$$C_{m+n}^k=\sum_{i=0}^{k}C_m^iC_n^{k-i}=\sum_{i=0}^{k}C_m^iC_n^{m-k+i}$$

感性理解，即在 $m+n$ 选取 k 个。有时候式子会变成后面那个样子，需要进行判别。

2. 图论

2.1 最短路

2.1.1 传递背包

求出图中任意两点的连通性

```
bool dis[N][N];

void Floyd(int n){
    rep(k,1,n)
        rep(i,1,n)
            rep(j,1,n)
                if(dis[i][k] && dis[k][j])
                    dis[i][j] = true;
}
```

2.1.2 负环

找出负环上的所有点

- $spfa$ 过程中每个点记录一个前驱
- 若某一个点入队 n 次，即存在负环，将该点取出并退出循环
- 按前驱回溯，第一个出现的环即为负环，环上的点即为负环中的点

2.2 匹配问题

2.2.1 图的最大匹配

- **定义**：从图中选出一些边构成一个边集，边集中的任意两条边都没有公共顶点，这个边集的最大数目即为最大匹配。
- **解法**：每个点拆成 V_x、V_y, 如果有一条边 $A \to B$, 则加边 $A_x \to B_y$ 。在这个二分图上跑最大匹配即可。

2.2.2 图的最小点覆盖

- **定义**：从图中选取一个点集，使得图中任意一条边的两个端点至少有一个在这个点集中。选择最少的点覆盖所有的边即为最小点覆盖。
- **解法**：最小点覆盖 $=$ 图的最大匹配
- **简要证明**：求出图的最大匹配之后，则一定不存在一条边没有被覆盖，因此 图的最大匹配 $\geq$ 最小点覆盖。同理，已知最小点覆盖的点集之后，一定不存在一条边其两个端点不在这个点集中，因此 最小点覆盖 $\geq$ 图的最大匹配。因此 最小点覆盖 $=$ 图的最大匹配。

2.2.3 图的最小边覆盖

- **定义**：从图中选取一个边集，使得图中任意一个点均被这个边集中的边覆盖。使得这个边集最小即为最小边覆盖。
- **解法**：最小边覆盖 $=$ 顶点数 $-$ 图的最大匹配 (最小点覆盖)
- **简要证明**： 设最大匹配数为 m，总顶点数为 n。为了使边集最小，优先选取最大匹配中的边，一条边可以覆盖两个点。对于剩下没被最大匹配覆盖的点，则每个点需要一条边进行覆盖，设这些点数量为 a。则 $2 * m + a = n$，因此 最小边覆盖 $= m + a =$ 顶点数 $-$ 图的最大匹配。

2.2.4 最小路径覆盖

- **DAG的最小不相交路径覆盖**
    - **定义**：每一条路径经过的顶点各不相同
    - **建图**：每个点拆成 V_x、V_y, 如果有一条有向边 $A \to B$, 则加边 $A_x \to B_y$
    - **解法**：最小路径覆盖 = 原图的节点数 - 新图的最大匹配
    - **简要证明**：每个点只能被匹配一次，每次匹配相当于两条路径合并成了一条路径，因此每多匹配一次，相当于路径覆盖数减少 1。

- **DAG的最小可相交路径覆盖 (POJ 2594)**
- **定义**：每一条路径经过的顶点可以相同
- **解法**：
  i. Floyd 求出原图的传递闭包
  ii. 每个点拆成 V_x、V_y, 如果 $A \to B$ 可达 , 则加边 $A_x \to B_y$，原题转化为最小不相交路径覆盖问题
  iii. 最小路径覆盖 = 原图的节点数 − 新图的最大匹配
  - **简要证明**：因为路径可相交，因此只要 $A \to B$ 两点可达，在确定覆盖路径时就不会受到路径上中间节点的影响，即相当于在原图增加一条$A \to B$ 的路径，然后跑最小不相交路径覆盖。

2.2.3 欧拉路

- **性质**
  - **无向图**：有且仅有两个点度数为奇数则有**欧拉路**，所有点度数均为偶数则有**欧拉回路**。
  - **有向图**：所有点 入度 = 出度 则有**欧拉回路**。有且仅有两个点入度不等于出度，且起点出度比入度大 1 ，终点入度比出度大 1 则有**欧拉路**。
- Hierholzer 算法
  - 解决无向图、有向图、欧拉路、欧拉回路问题
  - **具体过程**：选一个点 x 为起点，存在边 <x, y>，则删去边 <x, y>，若为无向图还需删除 <y, x>。若无边可走，则将x加入结果栈。最后输出结果栈即可。
- **图上的性质**
  - 无向图中，度数为奇数的点的个数一定为偶数
  - 任何一个无向图，都存在一种方案，将图中所有边改为有向边之后，每个原来度数为偶数的顶点现在 入度 = 出度

3. 小操作

3.1 运行时间计算

```
#include <ctime>
```

```
clock_t startTime,endTime;
startTime = clock();//计时开始

endTime = clock();//计时结束
cout << "The run time is: " <<(double)(endTime - startTime) / CLOCKS_PER_SEC << "
s" << endl;
```

KMP

字符串匹配算法 —— 线性复杂度

1. KMP求解过程

next数组 —— 最长前缀后缀匹配长度，需要小于串长度

- $ababa$，$next[1]=0, next[2]=0, next[3]=1, next[4]=2, next[5]=3$

求解问题：模式串在匹配串的哪些位置出现了，模式串是小串。

```
//预处理next数组，这里注意不要反复求strlen，这个函数是O(n)复杂度的，反复求会将复杂度变成O(n^2)
next[0] = -1;
for(int i = 1, j; i <= n; i++){
  for(j = next[i-1]; j >= 0 && s1[j+1] != s1[i]; j = next[j]);
  next[i] = j+1;
}
//next[j] <= next[j-1]+1, 不会比前一位+1更大
//匹配过程
for(int i=1, j=0; i <= m; i++){
  for(; s2[i] != s1[j+1] && j >= 0; j = next[j]);
  j++;
  if(j==n) ans++;
}
```

2. 例题

例1. 求取num数组，表示最长的，不超过一半的，前缀与后缀相同的长度。$(1 \le n \le 10^6)$

- 一个显然的性质：$num[i] \le num[i-1]+1$
- 因此求取 $num[i]$ 时，先令 $num[i]=num[i-1]+1$，如果 $num[i]$ 超过一半的长度或者在 $num[i]$ 处发生失配，则直接跳 $next$ 数组。

例2. 给出 n 个串，每个串的长度为 m_i，询问这 n 个串的最长公共子串。此处两个子串相同的条件为，长度相同且其中一个子串加上一个数即可得到另一个子串。$(1 \le n \le 1000, 1 \le m_i \le 100)$

- 在差分序列上进行 kmp 匹配。
- 这里由于数据范围比较小，因此可以直接枚举后缀进行 kmp 匹配求最小值。
- 但是多个串的最长公共子串其实是一个经典问题，可以用后缀数组或后缀自动机解决。

AC自动机

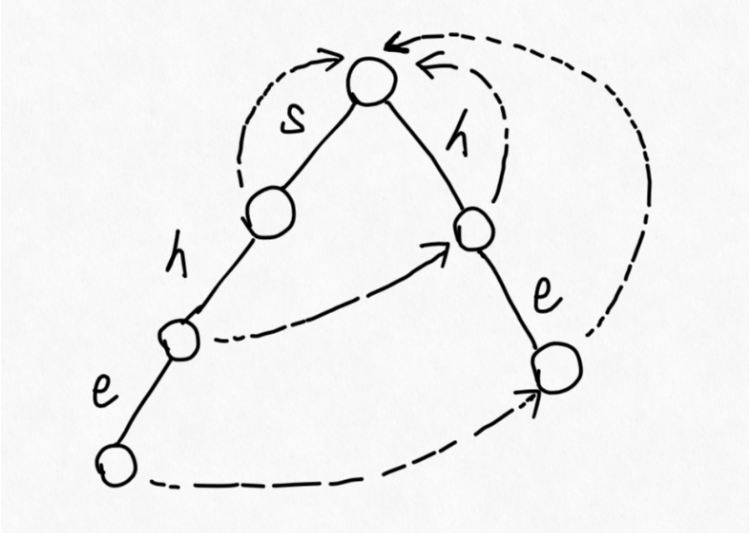
多模式串的匹配算法 —— 包含了 kmp 的功能

1.AC自动机构造过程

- 构造一颗 $Trie$ 树，作为 AC自动机的搜索数据结构。
- 构造 $fail$ 指针，使当前字符失配时跳转到具有最长公共前后缀的字符继续匹配。如同 kmp 算法一样，AC自动机在匹配时如果当前字符匹配失败，那么利用 $fail$ 指针进行跳转。由此可知如果跳转，跳转后的串的前缀，必为跳转前的模式串的后缀并且跳转的新位置的深度（匹配字符个数）一定小于跳之前的节点。所以我们可以利用 bfs 在 $Trie$ 上面进行 $fail$ 指针的求解。
- $fail$ 指针

最长的可匹配后缀，$fail$ 指向点 p，表示从根节点到点 p 是当前匹配位置的最长后缀。

按深度 bfs 整颗字典树，得到当前节点的 $fail$ 指针需要查看其父亲的 $fail$ 指针是否有自己这个儿子，如果没有，需要继续跳 $fail$ 指针。



- 常数优化：最后我们在 AC 自动机上跑字符串的时候，我们在失配时，通过 $fail$ 求出匹配点，用该点更新失配节点对应儿子，优化常数。

假如 $son[x][i] == 0$，表示节点 x 没有 i 这个节点，因此我们会不断跳 x 的 $fail$ 指针，直到找到一个节点 u，使得 $son[u][i]! = 0$，然后我们顺便设置 $son[x][i] = son[u][i]$，优化常数。

- 注意点。给出 n 个模式串以及 1 个匹配串，询问每个模式串在匹配串中出现的次数。在 AC 自动机上直接进行匹配时被匹配到的每个节点的 $fail$ 节点也都被匹配到了，不能忘记计算其 $fail$ 节点被匹配的次数。

2.例题

例1. 给出 n 个串，然后给出一篇文章，问这 n 个串有多少个在文章中出现过。$(\sum len \leq 10^6)$

- 模板题。需要注意被匹配到的节点的 $fail$ 节点也都被匹配到了。

例2. **[bzoj-1030]** 给出一些字符串，求长为 m 的字符串中包含这些的一共有多少个，字符集 $A-Z$。

- 我们转化成求有多少个长为 m 的字符串不包含这些字符串，然后在 AC 自动机上 dp。
- $f[i][j]$ 表示文本第 i 位，在 AC 自动机上匹配到第 j 个节点，并且之前没有出现过完整单词的方

案数。
- 如果一个点的 $fail$ 指针是单词结尾，则这个节点不能走。

例3. **[bzoj-2938]** 给出若干个 01 病毒串，问是否存在一个无限长的安全代码，不包括这些病毒串。$(\sum len \leq 30000)$

- 首先我们建一个完整版的 AC 自动机【补全每个点的儿子，用 $fail$ 指针补全】，之后将 AC 自动机上为病毒代码结尾的节点删去，在剩下的有向图中找是否存在环。

字符串 $hash$

1.$hash$ 算法概述

- 有的时候我们需要对两个很长的字符串快速进行比较，而一些匹配算法都不适用时，我们可以使用字符串 $hash$ 算法。
- 字符串 $hash$ 方法有很多种，在算法竞赛中最常用的一种就是将字符串变成一定进制数之后进行取模。这个算法的优点在于支持修改并且速度较快，不容易出现冲突情况。

2.原根

- 对于任意 $x < p-1$，满足 $a^x \ \%p \ \neq \ 1$，则 a 是 p 的原根。
- 原根满足，对于任意 $x, y < p-1$，不存在 $a^x = a^y$，即不同位置的不同字符 $hash$ 值不同。

3.生日悖论

- x 个位置，有 $\sqrt{x}$ 个人，则有两个人位置相同的概率大于 50%。

4.模数以及进制的选择

- 10^9+7 或者自然溢出。
- 进制尽量选择原根。

5.Bkdr $hash$ 算法

- $abcdef$
- $hash = 1*131^6 + 2*131^5 + 3*131^4 + 4*131^3 + 5*131^2 + 6*131^1$
- 可以 $O(1)$ 更改其中一个字符，求一个子串 $hash$ 值以及合并两个字符串。

6.例题

例1. **[bzoj-3555]** 给出若干个字符串，询问有多少对字符串相似。字符串相似当且仅当字符串长度相等，且恰好只有一位不同。

- 首先预处理每个字符串每个前缀和后缀的 $hash$ 值。
- 之后枚举字符串的每一位，利用预处理的 $hash$ 值求出每个字符串删去这位后的值，将这些值排序，统计答案即可。

例2. **[bzoj-2084]** 对于一个 01 字符串，如果将这个字符串 0 和 1 取反后，再将整个串反过来和原串一样，就称为 “反对称” 字符串。现给出一个长度为 n 的 01 字符串，求它有多少个子串是反对称的。$(n \leq 500000)$

- 我们发现串长一定是偶数。假如串 $[l,r]$ 是合法的，那串 $[l+1,r-1]$ 一定也是合法的。
- 枚举中心进行二分求解。因此只需要求出正的 $hash$ 值和反的 $hash$ 值即可。

例3. **[bzoj-1014]** 定义函数 $LCQ(x,y)$，表示：该字符串中第 x 个字符开始的子串，与该字符串中第 y 个字符开始的子串，两个子串的公共前缀的长度。现在可以更改字符串中某一个字符的值，也可以在字符串中的某一个位置插入一个字符，并且支持询问 $LCQ(x,y)$。

- 删除，用 $spaly$ 维护

Manacher 算法

1.Manacher 算法概述

- $Manacher$ 充分利用了回文的性质，从而达到线性时间。
- 首先先加一个小优化，就是在每两个字符（包括头尾）之间加没出现的字符（如*），这样所有字符串长度就都是奇数了，方便了很多。
  - $abcde \longrightarrow *a*b*c*d*e*$
- 记录 p_i 表示 i 能向两边推（包括 i）的最大距离，如果能求出 p 数组，那每一个回文串就都确定了。
- 我们假设 $p[1\sim i]$ 已经求好了，现在要求 $p[i]$：假设之前能达到的最右边为 R，对应的中点为 pos，j 是 i 的对称点。
  - 第一种情况，$i+p[j]<R$，即 $p[i]=p[j]$。

  - 第二种情况，$i+p[j]\geq R$，先设 $p[i]=R-i$，然后再继续增加 $p[i]$，并令 $pos=i$，更新 R。
  - 由于 R 一定是递增的，因此时间复杂度为 $O(n)$，可以发现一个串本质不同的回文串最多有 n 个，因此只有 R 增加的时候才会产生本质不同的回文串。

```
int len, tot, R = 0, pos = 0;
//对字符串加'#'号
len = strlen(s1+1);
s2[0] = '$';
s2[ln=1]='#';
for(int i = 1; i <= len; i++)
  s2[++ln] = s1[i], s2[++ln] = '#';
```

```
//求p[i]数组
for(int i = 1; i <= ln; i++){
  if(R >= i) p[i] = min(p[pos*2-i],R-i+1);
  if(p[i] + i > R){
    for(; s2[R+1] == s2[i*2-R-1] && R+1 <= ln && i*2-R-1 <= ln; R++); //小心多组数据
    pos = i;
    p[i] = R-i+1;
  }
}
```