数学

组合数

函数名	功能	复杂性
`init(ll mod)`	以 mod 为模数初始化前 10^6+10 个组合数	$\Theta(n)$
`init(int n,ll mod)`	以 mod 为模数初始化前 n 个组合数	$\Theta(n)$
`ll Inv(int x)`	求 $\frac{1}{x}$ 在模意义下的结果	$\Theta(\log \mathrm{mod})$
`ll power(ll b,ll n)`	求 b^n 对 mod 的余数	$\Theta(\log n)$
`ll C(ll x,ll y)`	求 C(x,y)$(x,y\le n)$ 对 mod 的余数	$\Theta(1)$
`ll fac[int x]`	求 $x!(x\le n)$ 对 mod 的余数	$\Theta(1)$
`ll inv[int x]`	求 $\frac{1}{x}(x\le n)$ 在模意义下的结果	$\Theta(1)$

code

线性筛

函数名	功能	复杂性
`init()`	预处理前 10^7+5 个数字	$\Theta(n)$
`init(int n)`	预处理前 $n+1$ 个数字	$\Theta(n)$
`char isPrime[int x]`	判定 x 是否是质数，注意，此处 1 被视为质数	$\Theta(1)$
`int OneFac[int x]`	如果 x 不是质数，则返回 x 的最小质因数	$\Theta(1)$
`int Prime[int x]`	返回第 $x+1$ 个质数，此处 1 不算质数	$\Theta(1)$
`vector<pair<int,int>>GetPrimeFac(int a)`	将 a 分解因数并返回，$a=\prod \mathrm{first}^{\mathrm{second}}$，满足所有 first 不相同	$O(\log a)$
`vector<int>GetFacList(int a)`	返回 a 的所有因数	$\Theta(d(a))$

code

平方根

函数名	功能	复杂性
`ll isqrt(ll x)`	返回 $\lfloor\sqrt{x}\rfloor$	$\Theta(\log w)$

```
long long isqrt(long long x){long long a=x,b=(x+1)/2;while(a>b){a=b;b=
(b+x/b)/2;}return a;}
```

PR（分解质因数）

函数名	功能	复杂性
`bool Prime(ll x)`	概率断言 x 是否为质数，此处 1 不算	$O(\log^2 x)$
`ll NonTrivialFact(ll x)`	返回任意一个 x 的非平凡因子，若没有返回 -1	期望 $\Theta(\sqrt[4]{x}\log x)$
`vector<ll> PrimeFacList(ll x)`	质因数分解 x	期望 $\Theta(\sqrt[4]{x}\log x)$

[code]

多项式乘法

函数名	功能	复杂性
`vector<int> mul(vector<int> a,vector<int> b)`	将 a 和 b 相乘（基于 NTT 的实现）	$\Theta(n\log n)$

[code]

图论

不带负权最短路

函数名	功能	复杂性
`clear(int n)`	重新建立一个节点为 1 到 n 的图，上一个将被废弃	$\Theta(n)$
`addedge_dis(int u,int v,long long w)`	添加一个从 u 到 v 的有向边，边权为 w	$\Theta(1)$
`addedge_undis(int u,int v,long long w)`	添加一个从 u 到 v 的无向边，边权为 w	$\Theta(1)$
`dijkstra(int s)`	以 s 为源点运行最短路	$\Theta(n\log n)$
`ll get(int n)`	求 s 到 n 的最短路，不连通则返回 10^{18}	$\Theta(1)$
`vector<int>path(int n)`	返回从 s 到 n 上最短路的所有节点（含 s 和 n）	$\Theta(n)$

code

强联通分量

函数名	功能	复杂性
`clear(int n)`	重新建立一个节点为 1 到 n 的图，上一个将被废弃	$\Theta(n)$
`addedge(int u,int v)`	建立一条 u 到 v 的有向边	$\Theta(1)$
`vector<vector<int>> kosaraju()`	得到图中所有的强联通分量（每个 `vector<int>` 中保存连通分量节点编号）	$\Theta(n)$

code

数据结构

猫树（强 ST 表）

函数名	功能	复杂性
`init(vector<node> x)`	初始化一个 ST 表	$\Theta(n\log n)$
`node query(int l,int r)`	返回区间为 $[l,r]$ 的合并结果	$\Theta(1)$

此模板不依赖于合并规则的交换律（你可以用它解决 GSS1）。

code

单点修改区间查询 线段树

函数名	功能	复杂性
`init(int n,node y)`	重新建立一个下标为 1 到 n 的树，上一个将被废弃，初始全为 y	$\Theta(n)$
`change(int x,node y)`	将下标为 x 的值修改为 y	$\Theta(\log n)$
`node query(int l,int r)`	返回区间为 $[l,r]$ 的合并结果	$\Theta(\log n)$

请注意，你需要自定义 node 结构体和合并规则（为 `node merge(node x,noe y)`），此模板不依赖于合并规则的交换律（你可以用它解决 GSS1）。

zkw实现（常数优秀，三倍空间，默认实现）

递归压缩空间实现（常数较大，两倍空间）

区间修改区间查询 线段树

函数名	功能	复杂性
`init(int n,typename node::info y)`	重新建立一个下标为 1 到 n 的树，上一个将被废弃，初始全为 y	$\Theta(n)$
`change(int l,int r,node::tag y)`	将下标为 l 到 r 的值施加 `tag` y	$\Theta(\log n)$
`node::info query(int l,int r)`	返回区间为 $[l,r]$ 的合并结果	$\Theta(\log n)$

请注意，你需要自定义 node 结构体和合并规则（为 `merge`），此模板 `tag` 和 `info` 都不依赖于合并规则的交换律（你可以用它解决 GSS1，或区间加，区间赋值）。

code

最近公共祖先

函数名	功能	复杂性
`TreeLCA(int n)`	初始化一颗树	$\Theta(n)$
`addedge(int u,int v)`	建立一个 u 到 v 的边	$\Theta(1)$
`init(int Root)`	以 Root 为根初始化所有内容（同时将父节点移到最后，重儿子移到最前）	$\Theta(n)$
`int lca(int u,int v)`	返回 u 和 v 的最近公共祖先	$\Theta(\log n)$
`int dis(int u,int v)`	返回 u 到 v 路径上经过边的数量	$\Theta(\log n)$
`int nxt(int u,int v)`	返回在 u 到 v 路径上，除了 u 以外离 u 最近的点	$\Theta(\log n)$
`path inter(path a,path b)`	求两个路径的交（若不交，则返回 `{-1,-1}`）	$\Theta(\log n)$
`path diam()`	求树的直径	$\Theta(n)$
`pair<int,int> gravity_center()`	求树的重心（如果第二个不存在会用 0 占位）	$\Theta(n)$

指针-前向星 实现（常数优秀，默认实现）

vector 实现（常数较大）

静态树剖（倍增）

函数名	功能	复杂性
`hld<node,cat_tree>`	以猫树为基础建立一个 `node` 的树剖	$\Theta(n)$
`setvalue(int x,node y)`	将节点编号为 x 的值赋予 `node` y	$\Theta(1)$
`init(int x)`	初始化树剖，以 x 为根	$\Theta(n\log n)$
`node query(int l,int r)`	返回从 l 到 r 路径上的合并值	$\Theta(\log n)$

此模板不依赖于合并规则的交换律（你可以用它解决 GSS1），同时它由 LCA 继承得到。

code

（目前仅支持点权和套猫树）

树哈希

函数名	功能	复杂性
`typedef pair<ull,ull> hvalue`	双哈希值，可供比较	/
`htree(int n)`	建立一个 n 大小的树	$\Theta(n)$
`void addedge(int u,int v)`	添加一条从 u 到 v 的边	$\Theta(n)$
`void init(int root)`	以 `root` 为根初始化哈希	$\Theta(n)$
`hvalue hash[i]`	求以 i 为根的树哈希（只有根有标号）	$\Theta(1)$
`int tree[0].second`	树的大小	$\Theta(1)$

code

字符串

字符串哈希

函数名	功能	复杂性
`typedef pair<int,ll> hvalue`	`first` 表示长度，`second` 表示哈希值	/
`hstring(string x)`	以 x 处理一个字符串哈希（你可以直接将 `string` 替换为 `vector<int>` 而没有任何后果）	$\Theta(n)$
`hvalue interval(int l,int r)`	以 `hvalue` 的形式返回 x 中 $[l,r]$ 的字符串	$\Theta(1)$
`hvalue operator +(hvalue a,hvalue b)`	连接两个 `hvalue`	$\Theta(1)$
`bool operator ==(hvalue a,hvalue b)`	判断两个 `hvalue` 是否相同	$\Theta(1)$

code

待办：

1. 懒标记的 zkw 线段树
2. 树剖套线段树
3. 网络流