原文链接：<https://github.com/abcwuhang/Algorithms/blob/master/multiplicative/multiplicative.pdf>

今天与大家分享一些数论知识和新（？）科技（与广为人知的杜教筛相得益彰~）

感谢[zzq](https://www.cnblogs.com/zzqsblog/p/9904271.html)对pe某数论问题论坛里（我的）解法的扩充。

先介绍一下OI中常见的[积性函数](https://zh.wikipedia.org/wiki/%E7%A9%8D%E6%80%A7%E5%87%BD%E6%95%B8)([multiplicative function](https://en.wikipedia.org/wiki/Multiplicative_function))及性质。（不想看入门介绍的可直接跳到最后读正文干货☺）

对一个函数，若它满足以下几个条件，则称之为积性函数：

1、定义域为正整数集合

2、

3、对任意两个互素的正整数和，有

由唯一分解定理可知，若的因式分解为（下同），则。可见完全由其在素数幂次处的值唯一确定。

常见的积性函数包括：

1、[欧拉函数](https://en.wikipedia.org/wiki/Euler%27s_totient_function)，含义为到中与互素的数的个数

2、[约数个数函数](https://en.wikipedia.org/wiki/Divisor_function)，含义为的约数个数（即因子个数）

3、[约数和函数](https://en.wikipedia.org/wiki/Divisor_function)，含义为的所有约数之和

4、[莫比乌斯函数](https://en.wikipedia.org/wiki/M%C3%B6bius_function)，含义为：若有平方因子，则；若为奇数个素数之积，则；否则。

更进一步，完全积性函数([completely multiplicative function](https://en.wikipedia.org/wiki/Completely_multiplicative_function))的定义为：

1、定义域为正整数集合

2、

3、对任意两个正整数和（无需互素），都有

常见的完全积性函数包括：

1、全0函数

2、[全1函数](https://en.wikipedia.org/wiki/Identity_function)

3、，其中为给定常数

4、（即单位元函数)

接下来介绍一下常见的[狄利克雷卷积](https://en.wikipedia.org/wiki/Dirichlet_convolution)。定义两个函数和的狄利克雷卷积为。

狄利克雷卷积算子满足：

1、交换律 （显然）

2、分配率 （这里加法指的是两个函数单点求和：

证明：

3、结合律

证明：，而，对比式子可发现二者相同。

4、单位元 （显然）

接下来介绍一下[莫比乌斯反演](https://en.wikipedia.org/wiki/M%C3%B6bius_inversion_formula)。如果对两个定义在正整数上的函数和满足，那么有。

证明：

这里用到了莫比乌斯函数的一个性质：，即当时左边求和才，否则左边为。证明：时显然。若，那么由莫比乌斯函数定义，当含有平方因子时对求和没有贡献，非零值只在由互不相同的素因子相乘时取得，所以左边。

入门知识完毕，下面开始讲新筛子（雾）。

给一个普通积性函数，现在希望求它的前缀和：。对任一积性函数，都存在一个积性函数满足（方便起见，可以用除号表示，即）。将式子展开，有（为素数，e为正整数）。特别的，有。

记的前缀和为，那么有。如果我们能构造一个，使得相对应的在素数处的取值都为（即对任意素数，），那么我们只需要求到中所有[幂数](https://zh.wikipedia.org/wiki/%E5%86%AA%E6%95%B8)（[powerful number](https://en.wikipedia.org/wiki/Powerful_number)）处的和的值即可。（一个数是幂数，当且仅当对的所有素因子，都能整除）（在素数幂处的值可由上面表达式展开求出）

原理：观察前缀和表达式，只需要考虑不为0时的贡献即可。由于对任意素数都有，那么当且仅当的因式分解表达式为，其中所有均大于等于。由于不大于的powerful number个数至多有个，如果能在时间复杂度内求出，则求前缀和的算法时间复杂度为，其中为黎曼函数。算法的实际表现非常强劲，上界都能几秒钟出结果，复杂度常数非常低。

下面看几个（迷之）例子：

1、：取即可满足，且前缀和非常好求！总时间复杂度为

2、：取即可满足，且前缀和非常好求！总时间复杂度为

3、：取即可满足，且前缀和非常好求（套一下杜教筛）！总时间复杂度为

4、：与第二个例子相同！

5、：取即可满足，且前缀和非常好求（套一下整除分块）！总时间复杂度为

6、：与第三个例子相同！

7、：与第三个例子相同！（算会多一个log复杂度，无伤大雅）

总结：与杜教筛相得益彰，又双叒叕有一大堆能“快速求出”前缀和的积性函数了！而且算法非常简单好写，妙手偶得之。