L. Clock Master

> 时钟设计。
> 给出一个数字：
> 分配到若干 t1 t2 .... ti.
> 产生若干组变化 (k%t1 , k%t2 , k%t3...,k%ti);
> 找出使得变化数最大的方案。
> 并且使输出该方案下，变化情况ln(sum)。

thinking

- 给定了一个具体的方案怎么快速算出来?
  - 简单的用程序进行了验证，发现就是多个数的最小公倍数。
- 转化问题，将该数字分解成若干个数，使得它们的最小公倍数最大。

事实上最后只来到了这一步。在这里就无法追求思路突破了。

思路

- 最终的问题，怎么拆分这一个数字，使得它们之间的最小公倍数最大?
  - 保证数字之间两两互质。
- 方案的前提是，两两个数互质，且所选择的数之和一个不小于预支的问题。
  - 可以感受到问题可以转化为一个背包问题。
  - 保证怎么选择可以选完所有可能的方案?
  - 于是问题就转化为了一个多重背包的问题。
- 为什么题目要求给出一个取对数的情况?
  - 藏思路?
    - 我这种啥比，不藏也发现不了。
  - 可能就是整数之间的乘法运算结果太大。无法表示
    - 为什么不引用模运算？模运算会改变所谓的到小关系，结果会出现偶然性。
    - 用一个经典的单调函数，将它转化成一个一个更小的小数。妙！！！！

```cpp
#include <iostream>
#include <cmath>
#include <iomanip>
typedef long long ll;
using namespace std;
//#define int long long

const int maxn = 3e4 + 10;

double f[maxn];  //这里表示各个情况的ans;
double lo[maxn]; //记录过程中可能用到的对数减少运算量。
bool pass[maxn];
int prime[maxn], tot;
// vector<int> g[maxn]; //估算一下复杂度。
```

```
void init()
{
    //首先找出所有的素数:
    for (int i = 2; i < maxn; i++)
    {
        if (!pass[i])
            prime[tot++] = i;
        for (int j = 0; prime[j] * i < maxn; j++)
        {
            // if (prime[j] * i > 30000)
            //     break;
            pass[i * prime[j]] = true;
            if (i % prime[j] == 0)
                break;
        }
    }
    for (int i = 1; i < maxn; i++)
        lo[i] = log(i);
}

void solve()
{
    int t;
    cin >> t;
    cout << fixed << setprecision(8) << f[t] << '\n';
}
signed main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr), cout.tie(nullptr);
    init();
    for (int i = 0; i < tot; i++) //输入处理；//找出一系列的素数组选哪机组。
        for (int j = 30000; j >= prime[i]; j--)
            for (int k = prime[i]; k <= j; k *= prime[i])
                f[j] = max(f[j], f[j - k] + lo[k]);
    int t = 1;
    cin >> t;
    while (t--)
        solve();
    return 0;
}
```

生长思考：

- 学习素数筛法减少常数的方法。什么时代了还在使用埃氏塞法。都是使用欧拉筛法了。