GENERALIZED EULER TOTIENT FUNCTION AND INTEGER FACTORIZATION

DENIS XAVIER CHARLES

ABSTRACT. It is known that computing the Euler's totient function is polynomial time equivalent to integer factorization under the generalized Riemann hypothesis. We show that a natural generalization of Euler's totient function is *almost* polynomial time equivalent to integer factorization. We also consider a somewhat artificial function related to the Euler's function that is polynomial time equivalent to integer factorization without any hypothesis.

Let $\varphi(N,k) = |\{m \le k \mid \gcd(N,m) = 1\}|$. The function $\varphi(N,k)$ is a generalization of Euler's totient function $\varphi(N) = \varphi(N,N)$.

Theorem 0.1. *Integer Factorization reduces in polynomial time (Cook reduction) to computing the generalized Euler function.*

Proof : Given N we will find a non-trivial factor using oracle access to the function $\varphi(\cdot,\cdot)$. First N is prime iff $\varphi(N,N) = N-1$. So we can assume N is not prime. Thus $\varphi(N,N-1) < N-1$. By binary search we find a $k > 1$ such that $\varphi(N,k-1) = k-1$ and $\varphi(N,k) = k-1$. Such a k necessarily satisfies $1 < \gcd(N,k) < N$, and so $\gcd(N,k)$ is a non-trivial divisor of N. $\square$

Unfortunately, for the other direction we do not have a polynomial time reduction yet.
Given N and k now we need to compute $\varphi(N,k)$, and we are given oracle access to Integer factorization. First factor N into its prime factors, and let its distinct prime divisors be $p_1, p_2, \cdots, p_l$. Now

$$\varphi(N,k) = k - \sum_i \left\lfloor \frac{k}{p_i} \right\rfloor + \sum_{i \ne j} \left\lfloor \frac{k}{p_i p_j} \right\rfloor - \cdots + (-1)^l \left\lfloor \frac{x}{p_1 p_2 \cdots p_l} \right\rfloor.$$

Each term of each sum can be evaluated in polynomial time. But we have $2^{\omega(N)}$ sums, where $\omega(N)$ is the number of distinct prime factors of N (l in our notation). Using the bound $\omega(N) \ll \frac{\log N}{\log\log N}$ from elementary number theory, we can estimate the time taken for this reduction. Thus this reduction is $2^{o(\log N)}$ time, since the encoding length of N is $\log N$, this gives us a sub-exponential time reduction to integer factorization.

Remark 0.2. It is known that $\omega(N) \approx \log\log N$ for most N. This would imply that the above argument actually gives a polynomial time reduction of computing $\varphi(N,\cdot)$ to integer factorization for most values of N. So in some sense computing $\varphi(\cdot,\cdot)$ and integer factorization are polynomial time equivalent on average.

Now consider the following (artificial) function:

$$\varphi_B(N,k) = \begin{cases} 1, & \text{if } \varphi(N,k) < k \\ 0, & \text{otherwise.} \end{cases}$$

It is clear that if we have oracle access to this function $\varphi_B(\cdot,\cdot)$ we can still do integer factorization in polynomial time, since the previous proof still works. Furthermore, we can compute $\varphi_B(N,k)$ easily if we oracle access to factoring. Compute the prime factorization of N, and say p is the smallest prime factor of N. Then $\varphi_B(N,k) = 0$ iff $k < p$.

Date: Nov 20, 2001.