Homework 5 Solutions

1. (a) Depending on your choice of starting values, you might find any one of the factors 13, 17 or 23. In theory you might find a factor that is a product of two of these but that is somewhat unlikely. (b) You should find yourself with one prime and one composite factor, and after further factoring, get $5083 = 13 \cdot 17 \cdot 23$.

2. Let $r$ be a prime factor of $n$, and let $p = r^k$ be its highest power that divides $n$. Then $q = n/p$ does not have $r$ as a prime factor, so $p$ and $q$ are relatively prime. If $n$ is not prime or a power of a prime, then $q \neq 1$, so $n = pq$ is a proper factorization into relatively prime factors.

3. (a) Suppose $n$ has $d$ binary digits, and let $r = \lceil d/k \rceil$. Then $n < 2^d$, so $\sqrt[n]{n} < 2^r$. To do a binary search for $\sqrt[n]{n}$, set an initial upper limit $y = 2^r$ and lower limit $x = 1$. Repeat the following steps: compute $z = \lceil (x + y)/2 \rceil$ and compare $z^k$ with $n$. If $z^k = n$, we have found the $k$-th root. If $z^k < n$, set a new lower limit $x = z + 1$. If $z^k > n$, set a new upper limit $y = z - 1$. If this gives new upper and lower limits with $x > y$, then $\sqrt[n]{n}$ is not an integer.

The test number $z$ is always less than $2^r$, so $z^k$ is less than $2^{kr}$, which has approximately as many binary digits as $n$. The search range is cut in half at each step, so the algorithm takes at most $r = d/k$ steps, which is $O(\log n)$.

(b) Again let $d$ be the number of binary digits of $n$. Then $n < 2^d$, so $\sqrt[n]{n} < 2^r$. Hence $\sqrt[n]{n}$ cannot be an integer for $k > d$, so we only need to try $k = 2, 3, \ldots, d - 1$. If you want to be more clever, it is enough to try only prime values of $k$, since if $n$ is a $k$-th power, and $p$ is a prime factor of $k$, then $n$ is also a $p$-th power.

Overall the algorithm runs $O(\log n)$ steps for each $k$, and we need only try $O(\log n)$ different $k$ values, for a total of $O((\log n)^2)$ steps. (That's assuming the cost of computing $z^k$ in each step is constant, which isn't necessarily a realistic assumption.)

2.7 #14: In the formula $\sum_j a_{ij}b_{jk}$ for the $(i, k)$ entry of the product $AB$, each term $a_{ij}b_{jk}$ is zero unless $i = j = k$, since $A$ and $B$ are diagonal. Therefore the whole sum is zero if $i \neq k$, which shows that $AB$ is diagonal, and the $(i, i)$ diagonal entry of $AB$ is just $a_{ii}b_{ii}$.

2.7 #24 (a) Computing $(A_1A_2)A_3$ takes 18000 multiplication operations. Computing $A_1(A_2A_3)$ takes 60000 operations. The first way is more efficient.