1.27 Note that $(p-1)(q-1)=352$. We can find the secret key by applying the extended euclidean algorithm to $352, 3$. This yields 235 as the multiplicative inverse. Therefore the message $M=41$ is encrypted via $41^{253} \mod 351 \equiv 153$. Furthermore $153^3 \mod 352 \equiv 41$.

1.28 For $p=11, q=7$, $(p-1)(q-1)=60$. Thus we just need to pick e such that $\gcd(e,60)=1$. Since the factorization of $60=2^2\cdot 3\cdot 5$ then one of the first prime which is not included is 11, which I will pick as my e. Now I need to find d such that $11d\equiv 1 \mod 60$. Note that $11^2=121=120+1=2*60+1$. Thus $d=11$ also works.

1.29 (a) The hash function We claim that $h_{a_1,a_2}(x_1,x_2)=a_1x_1+a_2x_2 \mod m$ is a universal hash function. Suppose a_1,a_2 are not known yet, and $(x_1,x_2),(y_1,y_2)\in[m]^2$, and wlog assume $x_2\neq y_2$. If the following 2 are equal under h then by definition $a_1(x_1-y_1)\equiv a_2(y_2-x_2) \mod m$. Under our assumption $y_2\neq x_2$, since $\mathbb{Z}_m$ is a field, then a_2 is exactly decided to be $a_1(x_1-y_1)(y_2-x_2)^{-1}\equiv a_2 \mod m$. Note that there is exactly a $\frac{1}{m}$ probability of a_2 being chosen. Thus the collision of two arbitrary elements in $[m]$ is $\frac{1}{m}$, therefore h is a universal hashing function. The number of bits required is $2*(\lfloor\log(m)\rfloor+1)$

(b) The same hash function H with $2^k=m$ is not universal since if one has $a_1=a_2=2^{k-1}$ then if $x_1+x_2=2n+1$, then we have $a_1x_1+a_2x_2=(x_1+x_2)2^k=(2n+1)2^k\equiv 2^k \mod m$, and if $x_1+x_2=2n$ then $a_1x_1+a_2x_2\equiv 0 \mod 2^k$. Additionally to specify a function one needs $2k$ bits.

(c) Note that there are $(m-1)^m$ functions $f:[m]\to[m-1]$. If we consider $x\in[m], y\in[m-1]$, and all the functions which fix $f(x)=y$ implies that they all have (x,y) in their definition. Since f is a set of m ordered pairs, and we take away one gives us $(m-1)^{m-1}$ functions with our desired ordered pair. Thus the probability that $f(x)=y$ is $\frac{(m-1)^{m-1}}{(m-1)^m}=\frac{1}{m-1}$, which is exactly the uniform probability of choosing an item in $[m-1]$. The number of bits required is exactly the number of bits to specify m distinct elements in $[m-1]$. Since an element in $[m-1]$ has up to $\lfloor\log(m-1)\rfloor+1$ bits in it's definition. Therefore we need $m(\lfloor\log(m-1)\rfloor+1)$ bits.

1.31 (a) The number of bits in $N!$ is going to be given by $\lfloor\log(N!)\rfloor+1$. We showed in a previous homework that $\log(n!)=\Theta(n\log(n))$. In the limit $\lfloor\log(N!)\rfloor+1\approx\log(N!)$. Thus the same Θ as above works.

(b) I propose the algorithm in which one computes $n!$ as $n*(n-1)!$. Therefore it takes exactly $T(n)=T(n-1)+(\log(n)+1)(\log((n-1)!)+1)$. The multiply is approximate $O(\log(n)*n\log(n))$ as $\log((n-1)!)=O(n\log n)$, shown on a previous homework. Therefore $T(n)=O((n\log n)^2)$ as $T(n)=\sum_{i=1}^n O(i(\log i)^2)\le O((\log n)^2)\sum_{i=1}^n i=O((\log n)^2)\frac{n(n+1)}{2}=O((n\log n)^2)$. Thus $T(n)=O((n\log n)^2)$.

1.31e If we have Karatsuba's algorithm then our $T(n)$ becomes

$$T(n)=T(n-1)+(\log(n)+1)^d(\log((n-1)!)+1)=T(n-1)+O(n(\log n)^{d+1})\le O(n^2(\log n)^{d+1})$$

Where we apply the same summation trick as before, and $d=0.585$.

2.4
- Algorithm A has the recurrence relation $T(n) = 5T(n/2) + O(n)$, thus we have that $\log_2(5)$ for the log term, and $d = 1$ since $O(n)$ is a first degree polynomial. Since $\log_2(5) > \log_2(2) = 2 > 1$ then by the masters theorem the programs run time is $O(n^{\log(5)/\log(2)})$.
- Algorithm B has the recurrence relation $T(n) = 2T(n-1)+O(1)$. Since the combination time is constant then there exists c which is the constant time. Therefore $T(n) = 2T(n-1) + c = 4T(n-2) + 2c + c = 8T(n-3) + 4c + 2c + c = \cdots = c\sum_{i=1}^{n} 2^i = c(2^{n+1} - 2)$. Thus the algorithms runtime is $O(2^n)$.
- Algorithm C has the recurrence relation $T(n) = 9T(n/3) + O(n^2)$. Therefore by the Master's theorem we have that $\log_3(9) = 2 = d$. Thus algorithm C has a runtime of $O(n^2 \log(n))$.

2.5 Observe that by the proof of the masters theorem, since all of the following recurenses have an exact amount of time being added, then our solutions are exact, which satisfy bounds from both above and below.

(a) $T(n) = 2T(n/3) + 1$. Since $\log(2)/\log(3) > 0 = d$, then $T(n) = \Theta(n^{\log(2)/\log(3)})$

(b) $T(n) = 5T(n/4) + n$. Since $\log_4(5) > 1$ then $T(n) = \Theta(n^{\log_4(5)})$

(c) $T(n) = 7T(n/7) + n$. Since $\log_7(7) = 1 = d$ then $T(n) = \Theta(n\log(n))$.

(d) $T(n) = 9T(n/3) + n^2$. Since $\log_3(9) = 2 = d$ then $T(n) = \Theta(n^2\log(n))$

(e) $T(n) = 8T(n/2) + n^3$. Since $\log_2(8) = 3 = d$ then $T(n) = \Theta(n^3\log(n))$

(f) $T(n) = 49T(n/25) + n^{3/2}\log n$. Since $\log(7) < \log(8) = 3, \log(5) > \log(4) = 2$, then $\log(7)/\log(5) < 3/2$. Therefore $T(n) = \Theta(n^{3/2}\log n)$, as the dominating term will be the constant multiple of $n^{3/2}\log n$.

(g) $T(n) = T(n-1) + 2$. Since each additional n adds a single 2, then $T(n) = 2n = \Theta(n)$.

(h) $T(n) = T(n-1)+n^c$. Observe that $T(n) = \sum_{i=1}^{n} i^c \le n*n^c = n^{c+1}$. Furthermore, $\frac{n}{2}^{c+1} = \frac{n}{2}\frac{n}{2}^c < \sum_{i=n/2}^{n} i^c < T(n)$. Thus $T(n) = \Theta(n^{c+1})$.

(i) $T(n) = T(n-1) + c^n = \sum_{i=1}^{n} c^i = \frac{c^{n+1}-1}{c-1} = \Theta(c^n)$, as $\frac{c^{n+1}-1}{c-1} < \frac{c^{n+1}}{c-1} = \frac{c}{c-1}c^n$ and $\frac{c^{n+1}-1}{c-1} > c^n - \frac{1}{c} > c^n - c > c^n - \frac{c^n}{2} = \frac{1}{2}c^n$ for sufficently large n. Thus $T(n) = \Theta(c^n)$.

(j) $T(n) = 2T(n-1) + 1 = \sum_{i=1}^{n} 2^{i-1} = 2^n - 1$. Clearly $T(n) = \Theta(2^n)$. As seen before in algorithm B, problem 2.4.

(k) $T(n) = T(\sqrt{n}) + 1$. We claim that $T(n) = \Theta(\log\log n)$. Observe for 2^{2^k} that $T(2^{2^k}) = k + T(\sqrt{2}) = k + 1 + T(1) = k + 2$. Since for arbitrary n there exists k such that $2^{2^{k-1}} \le n \le 2^{2^k}$, then $k+1 = \log\log 2^{2^{k-1}} \le T(n) \le \log\log 2^{2^k} = k+2$. Thus for well chosen multiples one has $T(n) = \Theta(\log\log n)$.

2.8 (a) The appropriate choice for ω is $\omega = i$. Since $i^4 = 1$, thus making it the fourth root of unity. Thus the fourier tranform for $[1, 0, 0, 0]$ at the first phase is simply mapping $1, 0 \mapsto 1, 1$ and $0, 0 \mapsto 0, 0$. Furthermore we have the first coordinate being $1+0=1$ the second being $1+i*0=i$, the third being $1-1*0=1$, and the forth being $1-i*0=1$. Thus the fourier transform is $[1,1,1,1]$. Since the inverse FFT is $\frac{1}{n}M_n(\omega^{-1})$ and there was no interaction with the complex parts implies that multiplying by the inverse will yield the same result, times $\frac{1}{4}$, $\frac{1}{4}[1,1,1,1]$.

(b) Note that for $[1,0,1,-1]$ we have $s_0=2, s_1=0, s_0'=-1, s_1'=1$. Thus we have that $[2+(-1), 0-i(-1), 2-(-1), 0+i(-1)] = [1, i, 3, -i]$. Going in reverse we get that $\frac{1}{4}[1,-i,3,i]$ is the inverse transform of $[1,0,1,-1]$.