1.2-2 On the average, the number of elements to be checked is $\frac{n^2}{2}$, where $n$ is the number of elements in the array. In the worst case, it is $n$. The average-case and worst-case running times of linear search are both $\Theta(n)$.

1.3-3 The basis case when $n = 2$ is true, since $T(2) = 2\lg 2 = 2$. Assume it is true for all $n < n_0$, we want to prove that it is true for $n = n_0$.

$$T(n_0) = 2T\left(\frac{n_0}{2}\right) + n_0 = 2\left(\frac{n_0\lg n_0}{2}\right) + n_0 = n_0(\lg n_0 - 1) + n_0 = n_0\lg n_0$$.

1.3-5 The following routine will return the position of Key found in array $A[p, \ldots, q]$, or return -1 if Key is not in array $A[p, \ldots, q]$.

**BINARY-SEARCH($A, p, q, Key$)**

1. If ($p > q$) then return -1
2. else
3. If $Key = A[\lfloor \frac{p + q}{2} \rfloor]$ then return $\lfloor \frac{p + q}{2} \rfloor$
4. else If $Key < A[\lfloor \frac{p + q}{2} \rfloor]$ then return BINARY-SEARCH($A, p, \lfloor \frac{p + q}{2} \rfloor, Key$)
5. else return BINARY-SEARCH($A, \lfloor \frac{p + q}{2} \rfloor + 1, q, Key$)

The worst-case running time of binary search is $\Theta(\lg n)$ because at each iteration the size of the array is halved. After $\lg n$ times, the size of the array is reduced to 0.

2-3 (a)

1. $(\frac{3}{2})^n, n2^n, e^n$
2. $2^{\sqrt{\ln n}}, n^3, (\sqrt{2})^{\lg n}, n$
3. $2\sqrt{2\ln n}$
4. $(\lg n)^{\lg n}$
5. $\lg^2 n$
6. $\ln \ln n$
7. $2\lg^* n$
8. $\lg^* n$
9. $\lg(\lg^* n), \lg^* \lg n$

(b) $f(n) = e^n$ if $n$ is even and $f(n) = \lg^*\lg n$ if $n$ is odd.