1. (1) O(1)
   (2) O(logn)
   (3) O(n)
   (4) O(n^2)
   (5) O(n^2)
   (6) 一般O(n^3)
      Strassen算法 O($n^{log_2 7}$))
2. 8, 1
3. (1) 正确 随着n的增大 $2^n > k * n$必然成立，无论常数k有多大
   $$2^n > 10^{1000} * n$$
   $when \quad n > N, N为一确定常数$
   (2) 正确 随着n的增大，log(n)会越来越高于1
   $log10^{1000} = 1000 >> 1$
4. T(n) = T(n - 1) + T(n - 2)
   解得 T(n) = O(ϕ^n) \ \ $\phi = \frac{\sqrt{5}+1}{2}$
   功能 计算斐波拉契数列的第n项
5. $\frac{(1-\alpha)N*N+\alpha*N\frac{1+2+..+N}{N}}{N} = (1 - \frac{\alpha}{2}) * N + \frac{\alpha}{2}$
6. 修改后的代码：

```
    template<typename T>
    void Vector<T>::mergeSort(Rank lo, Rank hi)
    {
        if (hi - lo < 2) return;

        bool flag = true;
        for (int i = lo; i < hi - 1; i++)
            if (A[i] > A[i+1])
                flag = false, break;
        if (flag) return;

        int mi = (lo + hi)/2;
        mergeSort(lo, mi), mergeSort(mi, hi);
        merge(lo, mi, hi);
    }

    template <typename T>
    void Vector<T>::merge ( Rank lo, Rank mi, Rank hi ) {
    T* A = _elem + lo;
    int lb = mi - lo; T* B = new T[lb];
    for ( Rank i = 0; i < lb; B[i] = A[i++] );
    int lc = hi - mi; T* C = _elem + mi;
    for ( Rank i = 0, j = 0, k = 0; (j < lb)||(k < lc);) {
        if ( ( j < lb ) && ( ! ( k < lc ) || ( B[j] <= C[k] ) ) ) A[i++] =
B[j++];
        if ( ( k < lc ) && ( ! ( j < lb ) || ( C[k] <  B[j] ) ) ) A[i++] =
C[k++];
    }
    delete [] B;
    }
```

若子序列已经有序，进入mergeSort后会直接退出，此时复杂度达到线性。