Assignment 1

1. 请写出下列操作的时间复杂度

(1) $a+b(a,b\in R)$ (2) 二分搜索 (3) 数组最大值

(4) (5) 选择排序 (6) 矩阵乘法

```
for ( i = 0; i < n; i++ )
  for ( j = 0; j <= i; j++ )
    sum += a[i][j];
```

2. 对一个初始为空的向量依次执行 insert(0, 4), insert(1, 2), put(0, 1), remove(1), insert(0, 8) 后的结果是_____。

3. 请自行举例分析并判断下列结论的正确性

(1) 2^n的复杂度高于n的任意系数的多项式

(2) $\log(n)$的渐进复杂度显著高于 1

4. 请判断下示程序的复杂度，并说明其功能

```
__int64 func(int n){
    return (2 > n) ? (__int64)n : func(n-1) + func(n-2);
}
```

5. 对长度为 N 的常规向量做顺序查找，其中 αN 次查找成功（$0 \le \alpha \le 1$），请计算平均查找长度。

6. 下示 mergeSort() 算法即使在最好情况下仍需要 $O(n \log n)$ 时间运行。试修改代码，使之在（子）序列已有序时仅需线性时间，并简单解释。

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