【1】下列算法完成对一个n位二进制数加1的操作。假如无溢出，该算法在最坏情况下的时间复杂度为O(n)，试问平均情况下的时间复杂度是什么？请用数学推导或实验验证的方式证明你的结论。

```
def  Inc(A) :
    i = len(A) - 1
    while  A[i] == 1 :
        A[i] = 0;  i -=1
    A[i] = 1
```

【2】现有互不相同的n个整数，试编写算法，输出从这n个数中取出k个数的所有组合(0<k<= n)。例如：若n个数是 1，2，3，4，5，k = 3，则输出结果为：543，542，541，532，531，521，432，431，421，321

【3】请编写算法，生成任意一个集合的幂集（不计空集）。集合元素用1，2，3，…表示。例如：若集合S= { 1, 2, 3 }，则其不计空集的幂集为：

{1}, {2}, {3}, {1, 2,}, {1, 3,}, {2, 3}, {1, 2, 3}

【4】在由整型数组成的序列A={a_0, a_1, …, a_{n-1}}中，称a_i-a_j（0 <= i<j < n）为数对 a_i和 a_j之差。考虑下面的算法，它求的是序列中数对之差的最大值。

```
def  MaxDiff( A ) : #平方时间复杂度
    n = len(A)
    dmax = 0
    for  i in range(n) :
        for  j in range(n):
            if  i < j and A[i] - A[j] > dmax :
                dmax = A[i] - A[j]
    return dmax
```

请对该算法进行改进（或重写），使得其能够达到线性的时间复杂度

【5】一个（元素可重复）序列中两个元素的最短距离是两个元素所有可能的距离中的最小值。例如：

若序列为：{1,4,3,2,6,5,6,7,4,3,2,3,1,5,9,8,6,4,2,5}

则2和7的最小距离为3；

5和1的最小距离为1。

考虑下面的算法，它求的是序列中两个指定元素的最小距离。

```
def  MinD(A, x, y) : # 平方时间复杂度
    n = len(A)
    Min, d1, d2 = 1000, -1, -1
    for i in range(n) :
        for  j in range(i, n) :
            if  A[i] == x and A[j] == y or A[i] == y and A[j] == x :
                if  abs(i-j) < Min :
                    Min, d1, d2 = abs(i - j), i, j
    return Min, d1, d2
```

请对该算法进行改进（或重写），使得其能够达到线性的时间复杂度

【6】输入为 n 个整数组成的序列A，求A中其和为最大的子序列的和。例如：

若A为：{ -2, 11, -4, 13, -5, -2 }

则最大子序列和为20。

若A为：{ -6, 2, 4, -7, 5, 3, 2, -1, 6, -9, 10, -2 }

则最大子序列和为16。

考虑下面的算法，它求的是序列A中其和为最大的子序列的和。

```
def  MaxSubsequenceSum(A) :  #立方时间复杂度
    n = len(A)
    maxSum = 0
    for  i in range(n) :
        for  j in range(n) :
            thisSum = 0
            for  k in range(i, j) :
                thisSum += A[k]
            if  thisSum > maxSum :
                maxSum = thisSum
    return maxSum
```

请对该算法进行改进（或重写），使得其能够达到线性的时间复杂度