Assignment 5

11510257 彭福

Exe.1.

Solution:

Define: database A, database B, integer A(k) or B(k) is the k^{th} smallest value we want to ask.

Let k = $\lceil \frac{n}{2} \rceil$. If A($\lceil \frac{n}{2} \rceil$) >B($\lceil \frac{n}{2} \rceil$), then we know that A($\lceil \frac{n}{2} \rceil$) is larger than n-$\lceil \frac{n}{2} \rceil$+ $\lceil \frac{n}{2} \rceil$ = n numbers in two database at least, and B($\lceil \frac{n}{2} \rceil$) is less than n- $\lceil \frac{n}{2} \rceil$ + $\lceil \frac{n}{2} \rceil$ = n numbers in two database at least. Therefore, the median number we want is in the smallest $\lceil \frac{n}{2} \rceil$ number in A and in the largest $\lceil \frac{n}{2} \rceil$ number in B. And then we do the same thing as above recursively until we find only one number in A and one number in B may be the median number.

```
Function find_median(n,A_start,B_start):
        If n = 1, return min(A(A_start),B(B_start));
        If(A(A_start + celling(n/2))>B(B_start +  celling(n/2)));
            Then B_start +=  celling(n/2);
        Else
            Then A_start +=  celling(n/2);
         n= celling(n/2);
         find_median(n,A_start,b_start);
```

Exe.2.

Solution:

We can use the algorithm to counting

the inversion in the textbook with some modification. The modification is merging twice every time we merge two sequences. The first merge is the original sequence a_1, a_2, $a_3 \cdots a_n$ and b_1, b_2, $b_3 \cdots b_n$ to sorting the sequences. And the second merge is a_1, a_2, $a_3 \cdots a_n$ and $2*b_1$, $2*b_2$, $2*b_3 \cdots 2*b_n$ to counting the significant inversions.

```
Funtion merge_and_count(A,B):
   int a,b=0;
   int i = 0;
   int count = 0;
   let sequence B1={ 2*b1, 2*b2, 2*b3 … 2*bn };
   let L is the sequence after merging;
   while(a<length(A) &&b<length(B))
{
                if(A[a]<B[b])
                { a++;
                  L[i] = A[a];
                  i++;
                 }
                else
                { b++;
                   L[i] = B[b];
                   i++;
                }
}
    put the remain number into L;
    a = 0;
    b = 0;
    while(a<length(A) &&b<length(B1))
{
                if(A[a]<B1[b])
                     then a++;
                else
                { b++;
```

```
                    count+=length(A)-a;
                }
}
    return count;
```

```
Function sort_and_count(L):
    If (length(L)==1)
      then return 0;
    else
{
   let A be the first half of L;
   let B be the second half of L;
   count_A = sort_and_count(A);
   count_B = sort_and_count(B);
   count = merge_and count(A,B);
}
Return count = count_A + count_B+count;
```