**题目描述：在数组中的两个数字，如果前面一个数字大于后面的数字，则这两个数字组成一个逆序对。输入一个数组，求出这个数组中的逆序对的总数。**

1、(辅助知识点)归并排序算法：

class Solution {

public:

int InversePairs(vector<int> data) {

int size = data.size();

if (size == 0)return 0;

int \*p = new int[size];

if (p == NULL)return false;

mergesort(data, 0, size - 1, p);

delete[] p;

return reverse\_pair;

}

void mergearray(vector<int> &a, int first, int mid, int last, int temp[]){ //合并

int i = first, j = mid + 1;

int m = mid, n = last;

int k = 0;

while (i <= m && j <= n){

if (a[i] <= a[j]) temp[k++] = a[i++]; //右边数大

else temp[k++] = a[j++]; //左边数大

}

while (i <= m)temp[k++] = a[i++];

while (j <= n)temp[k++] = a[j++];

for (i = 0; i < k; i++)a[first + i] = temp[i];

}

Void mergesort (vector<int> &a, int first, int last, int temp []) { //划分(相当于分治)

if (first < last){

int mid = (first + last) / 2;

mergesort(a, first, mid, temp); //左边有序

mergesort(a, mid + 1, last, temp); //右边有序

mergearray(a, first, mid, last, temp); //再将二个有序数列合并

}

}

int reverse\_pair = 0;

};

分析：

2、逆序对算法(在归并算法上改进):

void mergearray(vector<int> &a, int first, int mid, int last, int temp[]){

int i = first, j = mid + 1;

int m = mid, n = last;

int k = 0;

while (i <= m && j <= n){

if (a[i] <= a[j]) { //左边数小

temp[k++] = a[i++];

}

else { //右边数小

temp[k++] = a[j++];

reverse\_pair += mid - i + 1;

}

}

while (i <= m){

temp[k++] = a[i++];

}

while (j <= n){ //在左边全部加入之后再加入的右边的元素不算逆序数

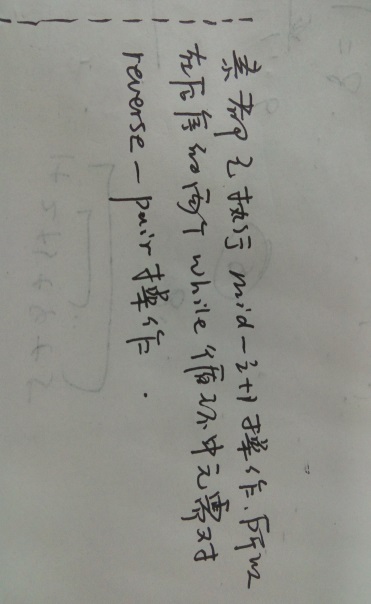
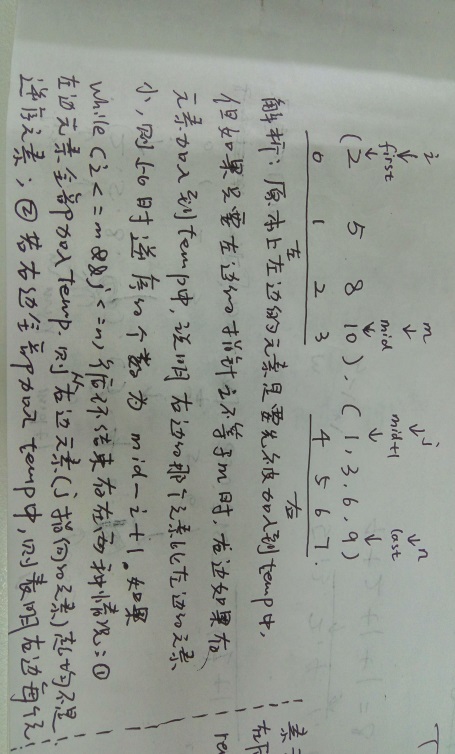
temp[k++] = a[j++];

}

for (i = 0; i < k; i++)a[first + i] = temp[i];

}

分析: 在归并算法中，每一次void mergearray时，左右两个子部分已经是有序的子数组，在每次递归分组中，左右两个子部分的大小相等或者相差1（右边比左边少1），比如（2,5,8,10），（1,3,6,9）。



**1.分治法**

分治法采用的是递归的思想来求解问题，两个分解的子问题独立求解，两者之间无任何的重叠。而上一层问题只需要对两个子问题进行一定的merge即可得到答案。即s（t）＝ s(sub1)+s(sub2)，但是s(sub1)和s（sub2）之间（看子问题）无任何重叠。

典型应用：

**2. 贪心法**

可以定义为 s（t）＝ s(t-1) ＋ selection according to certain criteria。

同样其使用了类似迭代子问题的求解方式，逐步求得全局的最优答案。而其只有一个s(t-1)，故不存在重叠求解子问题的情况。

**3. 动态规划方法**

该种方法较为复杂，但十分有用和高效，其核心性质是当前问题的答案s(t)，并不能单独由s(t-1)求得。还有可能需要使用到s(1)...s(t-1)。具体需要使用到那些，是由问题本身的性质所决定的（常常是一个约束，或变相的约束）。