早餐组合

小扣在秋日市集选择了一家早餐摊位，一维整型数组 `staple` 中记录了每种主食的价格，一维整型数组 `drinks` 中记录了每种饮料的价格。小扣的计划选择一份主食和一款饮料，且花费不超过 `x` 元。请返回小扣共有多少种购买方案。

注意：答案需要以 `1e9 + 7 (1000000007)` 为底取模，如：计算初始结果为：`1000000008`，请返回 `1`

示例 1：

输入：`staple = [10,20,5], drinks = [5,5,2], x = 15`

输出：`6`

解释：小扣有 6 种购买方案，所选主食与所选饮料在数组中对应的下标分别是：
第 1 种方案：staple[0] + drinks[0] = 10 + 5 = 15；
第 2 种方案：staple[0] + drinks[1] = 10 + 5 = 15；
第 3 种方案：staple[0] + drinks[2] = 10 + 2 = 12；
第 4 种方案：staple[2] + drinks[0] = 5 + 5 = 10；
第 5 种方案：staple[2] + drinks[1] = 5 + 5 = 10；
第 6 种方案：staple[2] + drinks[2] = 5 + 2 = 7。

示例 2：

输入：`staple = [2,1,1], drinks = [8,9,5,1], x = 9`

输出：`8`

解释：小扣有 8 种购买方案，所选主食与所选饮料在数组中对应的下标分别是：
第 1 种方案：staple[0] + drinks[2] = 2 + 5 = 7；
第 2 种方案：staple[0] + drinks[3] = 2 + 1 = 3；
第 3 种方案：staple[1] + drinks[0] = 1 + 8 = 9；
第 4 种方案：staple[1] + drinks[2] = 1 + 5 = 6；
第 5 种方案：staple[1] + drinks[3] = 1 + 1 = 2；
第 6 种方案：staple[2] + drinks[0] = 1 + 8 = 9；
第 7 种方案：staple[2] + drinks[2] = 1 + 5 = 6；
第 8 种方案：staple[2] + drinks[3] = 1 + 1 = 2；

解题思路：

对第一个数组进行排序，遍历第二个数组，依次对第一个数组使用二分法查看是否符合要求，

第二个数组中每一个元素在第一个数组内找到对应的合适的区间**[left,right]**

区间左边界以左肯定满足，与其相邻的右边界需要在判断

记录符合要求的数量 **left** 或者 **left+1** 即 **right**

暴力求解无法通过时间限制

```
/*int breakfastNumber(int* staple, int stapleSize, int* drinks, int drinksS
ize, int x){
    int cout=0;
    for(int i=0;i<stapleSize;i++)
    {
        for(int j=0;j<drinksSize;j++)
        {
            if(staple[i]+drinks[j]<=x)
            {
                cout++;
            }
        }
    }
    return cout;
}*/

int CmpInt(const void *a, const void *b)
{
    return *((int*)a) - *((int*)b);
}
int breakfastNumber(int* staple, int stapleSize, int* drinks, int drinksSiz
e, int x)
{
    long long res=0;
    qsort(staple,stapleSize,sizeof(int),CmpInt);
    for(int i=0;i<drinksSize;i++)
    {
        int left=0;
        int right=stapleSize-1;
        while(left<right)
        {
            int mid=left+(right-left)/2;
            if(staple[mid]+drinks[i]>x)
            {
                right=mid;
            }
            else if(staple[mid]+drinks[i]<=x)
```

```
            {
                left=mid+1;
            }
        }
        res+=left;//左边界以左的元素肯定满足
        if(staple[right]+drinks[i]<=x)
            res++;
    }
    res%=1000000007;
    return res;
}
```