1. 调研学习Voronoi图及相关算法，并给出至少一个应用。

Voronoi图： 在二维中，由一系列种子点的垂直平分线构成；三维中由一系列种子点的垂直平分面构成。对于给定的一组点，Voronoi 图将空间分割成由各个点为中心的区域，每个点的区域称为 Voronoi 单元，满足该点到相邻 Voronoi 单元的点的距离大于该点到其他任意点的距离。

Voronoi图相关的算法有：

1. Fortune算法：Fortune算法是Voronoi图计算的标准算法之一，时间复杂度为O(nlogn)。它是基于扫描线的思想，将每个点的切线作为扫描线，对每个点进行处理。Fortune算法在实现过程中需要用到平衡树来维护当前点集的Delunay三角剖分。
2. 基于Delaunay三角剖分的算法：Voronoi图和Delaunay三角剖分有着密切的关系，它们是互相对偶的。因此，可以通过计算Delaunay三角剖分，然后从中推导出Voronoi图。这种方法的时间复杂度也是O(nlogn)。
3. Bowyer-Watson算法：Bowyer-Watson算法也是基于Delaunay三角剖分的思想。它是一种递归算法，逐步向Delaunay三角剖分中加入新的点，并根据新的点来更新Delaunay三角剖分。Bowyer-Watson算法的时间复杂度也是O(nlogn)。、
4. Fortune-Sweepline算法：该算法将Fortune算法与空间分治算法相结合，将原始点集按照空间位置划分到不同的区域中，并在每个区域内运行Fortune算法，然后将不同区域的Voronoi图合并成整体的Voronoi图。这种方法可以在分布式计算环境中实现，时间复杂度也是O(nlogn)。
5. 层次Voronoi图算法：该算法是一种递归算法，通过将空间分成不同的层次，然后在每一层计算出Voronoi图，再将不同层次的Voronoi图合并成整体的Voronoi图。该算法可以快速计算出大规模点集的Voronoi图，时间复杂度为O(nlogn)。

应用：

1. Voronoi图在机器人避障中有广泛的应用，Voronoi图主要将机器人周围的障碍物分割为一个个几何图形，并计算出机器人能够通过的最短路径，从而避免与障碍物碰撞。
2. Voronoi图在气象学中也有广泛应用，如气象站的布局中，需要考虑到气象数据的准确性和覆盖范围，而Voronoi图可以根据气象站的位置信息生成一组覆盖范围相对均匀的区域，从而确定气象站的最佳布局。
3. 在气候模拟中，Voronoi图可以用于空间插值。例如，在一组气象站数据中，可以使用Voronoi图将空间分割成多个单元，对每个单元中的气象数据插值，从而得到全局的气象数据分布。
4. Voronoi图还应用于地貌分析，地貌分析中，Voronoi图可用于地形特征的提取和分类。例如可以使用Voronoi图将地形特征分割成不同的区域，然后对每个区域进行高程、坡度等地形特征的统计和分析，从而更好地理解地形特征的分布规律。

2. 前面已经学习在一维序列中某元素的查找算法，现若查找是在一个二维矩阵中进行，并且矩阵中行列均为升序排列，请给出相应查找算法并进行分析。

法一：暴力求解方法，序列从始到末遍历寻找。

法二：利用二分查找的思想，将二维数组arr[m][n]分割为m个一维数组，在每个数组中实现二分查找;未查找到则在下一个数组中查找。

```
//二维矩阵查找算法，行列递增

#include<iostream>
```

```cpp
#include<ctime>
using namespace std;

#define M 15
#define N 15

//一维数组的二分查找
bool Binary_search(int* arr,int* n,int value);
//二维矩阵二分查找算法，行列递增
bool Binary_search_s(int** arr,int* m,int* n,int value);
//二维矩阵查找算法，行列递增,返回位置
bool find_in_matrix(int** arr,int* m,int* n,int value);

int main(){
    int arr[M][N];
    srand((unsigned)time(NULL));
    cout<<"数组元素为："<<endl;

    arr[0][0]=rand()%3;
    cout<<arr[0][0]<<"\t";
    //构造随机行列递增二维数组
    for(int i=0;i<M;i++){
        for(int j =0;j<N;j++){

            if(i==0){
                if(j==0){
                    continue;
                }
                else{
                    arr[i][j]=arr[i][j-1]+rand() % 2 + 1;//生成1-3的随机数
                    cout<<arr[i][j]<<"\t";
                    continue;
                }
            }
            if(j==0){
                arr[i][j]=arr[i-1][j]+rand()%2+1;
                cout<<arr[i][j]<<"\t";
                continue;
            }

            arr[i][j]=max(arr[i][j-1],arr[i-1][j])+rand()%2+1;
            cout<<arr[i][j]<<"\t";
        }
        cout<<endl;
    }
    cout<<endl;

    int value = rand()%10;
    cout<<"The random key is:"<<value<<endl;
    int m =-1;
    int n =-1;

    if(Binary_search_s((int**)arr,&m,&n,value)){
        cout<<"arr["<<m<<"]["<<n<<"]="<<value<<endl;
    }
    else{
        cout<<"Not found "<<value<<"!"<<endl;
    }
```

```
    return 0;
}
//二分查找
bool Binary_search(int* arr,int* n,int value){
    int low = 0;
    int high = N-1;
    int mid = 0;
    while(low<=high){
        mid = (low+high)/2;
        if(arr[mid]==value){
            *n=mid;
            return true;
        }
        else if(arr[mid]>value){
            high = mid-1;
        }
        else{
            low = mid+1;
        }
    }
    return false;
}
bool Binary_search_s(int** arr,int* m,int* n,int value){
    for(int i=0;i<M;i++){
        //cout<<*((int*)arr + 3)<<endl;    //对二维数组访问方式的不熟悉
        if(Binary_search((int*)arr+M*i,n,value)){
            *m=i;
            return true;
        }
    }
    return false;
}
```

法三：由于该序列的行列升序特性可知，待查找元素value小于当前元素时，需要向右或下继续查找；待查找元素value大于当前元素时，需要向左或上继续查找。
而我们若从右上角开始查找，则可以规避一半问题，当前元素如果大于value，则向左查找，如果小于value，向下查找。不需要向右和上方查找的原因是二维数组最初从右上角开始查找。

```
int main(){
    .....


    if(find_in_matrix((int**)arr,&m,&n,value)){
        cout<<"arr["<<m<<"]["<<n<<"]="<<value<<endl;
    }
    else{
        cout<<"Not found "<<value<<"!"<<endl;
    }


    return 0;
}
//总体思路是从右上角开始查找，如果大于value，则列减一，如果小于value，则行加一
bool find_in_matrix(int** arr,int* m,int* n,int value){
    //二维矩阵查找算法，行列递增
```

```
    int row = 0;//行,从第一行最后一个元素开始
    int col = N-1;
    while(row<M && col>=0){
        if(*((int*)arr+M*row+col)==value){
            *m=row;
            *n=col;
            return true;
        }

        else if(*((int*)arr+M*row+col)>value){
            col--;
        }
        else{
            row++;
        }
    }
    return false;
}
```

运行结果如下：

```
1       3       5       7       9       10      12      13      15      17      18      20      22      24      25
2       5       7       8       11      13      15      16      18      19      20      22      23      25      26
4       7       9       11      13      14      17      19      20      22      24      26      27      29      30
5       9       11      12      15      16      19      20      22      23      26      27      29      31      32
7       10      13      15      17      18      20      21      24      26      27      29      30      33      35
9       12      15      16      18      20      21      23      26      27      29      31      32      35      36
10      14      17      18      19      22      24      25      27      29      31      32      33      36      37
11      16      19      21      23      24      26      27      28      31      33      35      36      38      40
13      17      21      23      25      27      29      30      32      34      35      37      38      40      42
15      18      22      25      27      28      31      33      35      37      38      39      41      43      45
17      20      24      26      28      29      32      35      36      38      40      42      44      45      47
18      22      26      27      30      32      34      36      37      40      41      43      45      46      49
19      23      27      28      32      34      36      37      38      41      42      44      46      47      51
20      25      28      30      33      35      37      39      41      42      44      46      48      49      52
22      27      30      31      34      36      38      41      43      45      47      49      51      52      53

The random key is:9
arr[0][4]=9
arr[0][4]=9
```