1、有个大小不同的杯子和与之匹配的*n*个杯盖, 你可以尝试一个杯子和一个杯盖是否匹配, 尝试结果有三种：（1）杯子太大；（2）匹配成功；（3）杯盖太大. 请设计一个分治算法完成所有杯子和杯盖的匹配, 算法的时间复杂性用匹配尝试的次数来衡量。请写出（1）算法设计的思路，包括边界条件、Divide、Conquer和Merge的基本过程；（2）算法的伪代码；（3）算法的时间复杂度。

2、对于两个二维数据元素和，如果(1) , 或者 (2) , ，则支配，记为。二维数据集合的定义如下：。设计基于分治的二维数据Skyline求解算法。请写出（1）算法设计的思路，包括边界条件、Divide、Conquer和Merge的基本过程；（2）算法的伪代码；（3）算法的时间复杂度。

1. 和为两个数组，每个数组中的个均已经排好序，试设计一个的分（减）治算法，找出*X*和*Y*中个数的中位数。（注：2个数的中位数是排序后第个和第个数）。请写出（1）叙述算法设计思路. （2）写出算法伪代码. （3）分析算法的时间复杂性.
2. 算法总体思路：假设N为杯盖的集合，M为杯子的集合，从M中任取一个杯子，下标为mk，把mk和N中所有元素比较，得到1）杯子太大的集合N小于，2）N大于杯子太小的集合。3）nk匹配成功的集合。
3. Divide：假设N为杯盖的集合，M为杯子的集合，从M中任取一个杯子mk，把mk和N中所有元素比较，得到：

1）杯子太大的下标集合IndexN小于。

1. 杯子太小的下标集合IndexN大于。
2. 和mk匹配成功的nq。

另外，把所求得的nk与M中所有元素比较，得到：

1）杯盖太大的下标集合IndexM小于。

2）杯盖太小的集合IndexM大于。

此时，问题转化为①IndexN小于和IndexM小于，②IndexN大于和IndexM大于两个杯子杯盖组合的匹配问题。

1. Conquer：对①IndexN小于和IndexM小于，②IndexN大于和IndexM大于两个杯子杯盖组合进行匹配。

4，Merge：无

5，边界条件：T(n)=O(1),if n<3;

6，时间复杂度为T(n)=T(n/2)+O(n)=O(nlogn)

伪代码：

Match(M,N)//M表示茶杯的数组，N是杯盖的数组

IndexM=[0,1,2,...,n-1];//n是茶杯的个数

IndexN=[0,1,2,...,n-1];

IndexM小于,IndexN小于,IndexM大于,IndexN大于=Divide(IndexM,IndexN,M,N);

Match(IndexM小于,IndexN小于,M,N);

Match(IndexM大于,IndexN大于,M,N);

Divide(IndexM,IndexN,M,N):

for i=IndexN[0] to IndexN[IndexN.size-1] Do

If M[0]>N[i]

Add i to IndexN小于;

If M[0]<N[i]

Add i to IndexN大于;

Else

j=i;

for j=IndexM[0] to IndexM[IndexM.size-1] Do

If M[j]>N[q]

Add j to IndexM大于

If M[j]<N[q]

Add j to IndexM小于

Return IndexM小于,IndexN小于,IndexM大于,IndexN大于;

1. 算法总体思路：先将所有点对都按照x轴进行排序，
2. Divide：把点对分为左右两组QL和QR。
3. Conquer：求QL和QR的skyline集合。
4. Merge：把QL的skyline点集中所有点的yk坐标依次与QR的skyline点集中最大y坐标ymax比较，若yk小于ymax，则舍去该点，否则添加到Q的skyline点集中。

时间复杂度为T(n)=T(n/2)+O(n)=O(nlogn)

5，边界条件：T(n)=O(1),if n=1;

伪代码：

Sort(Q) on x;//Q表示所有点的列表,在x轴上对Q进行排序

Skyline(Q):

用x=m把Q分为QL和QR两部分;

Res=Merge(Skyline(QL),Skyline(QR));//Res用来存放Merge后的结果

Return Res;

Merge(Q1,Q2):

for q1 in Q1 Do

if q1.x>=ymax

Add q1 to Res;

return Res;

1. 算法总体思路：二分法在X数组中寻找分界点。
2. Divide：把数组X从p=n/2处分为X左 , (X左包含X[0]~X[p-1]) , X右(X右包含X[p]~X[n-1])两部分。同理，Y数组从q=n-p处分为Y左 , Y右部分。
3. 终止条件：若满足max(X[p-1],Y[q])<min(X[p-1],Y[q])，则表示找到正确分界点，退出递归；
4. 若不满足终止条件，则有两种可能性:

1、当X[p-1]>X[q]时，分界点p=(beg+p-1)/2;同时，q=n-p；

2、X[p-1]<Y[q]时，分界点p=( (p+1)+end)/2;同时q=n-p；

5，第四步结束后，返回第三步，如此递归，直到找到合适的分界点p和q；

6，中位数即为max(X[p-1],Y[q-1])和min(X[p],Y[q])；

7，时间复杂度为:O(log(n))

main():

return Median(X,Y,n/2);

Median(X,Y,p):

Xp-1=X[p-1] if p!=0 else Integer.MIN\_VALUE;

Xp=X[p] if p!=n else Integer.MAX\_VALUE;

Yq-1=Y[q-1] if q!=0 else Integer.MIN\_VALUE;

Yq=Y[q] if q!=n else Integer.MAX\_VALUE;

if max(Xp-1,Yq-1)<min(Xp,Yq)

return max(Xp-1,Yq-1),min(Xp,Yq);

else

If Xp-1>Xp

Median(X,Y,Divide(beg,p-1));

If Xp-1<Yq

Median(X,Y,Divide(p+1,end));

Divide(beg,end):

p=(beg+end)/2;

return p;