作业回顾：非编程题

1.生活中的排序

比较常见的答案：身高，成绩，年级.....

比较有个性的答案：淘宝性剁手(性价比)，图书馆整理书籍，旗语，历史朝代，打牌，元素周期表(原子量)……

排队？：排队更多的是强调先来后到，不是典型的排序。参见标准容器std::queue

2.最牛的排序

比较符合的答案：利用概率论进行数学实验

我的答案：当算法在成千上万台机器上并行运行（云计算）

家长找到的一种答案：量子计算机

我竟无言以对的答案：n很小的时候，有钱任性的时候.....

3.渐进复杂度

1) $7n+n^2+6=O(n^2)$
2) $2n\log n+3n^3-100=O(n^3)$
3) $2^n+n^{100}+3n=O(2^n)$

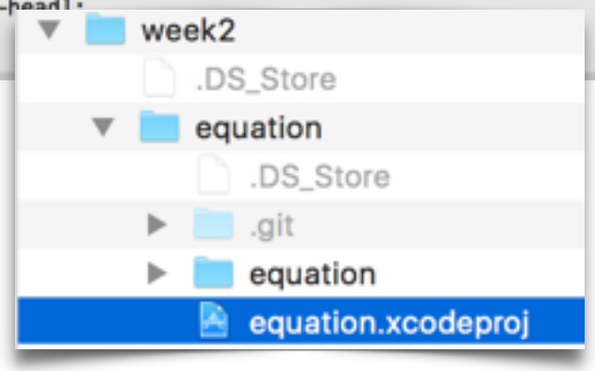
渐进趋势

指数>幂次>对数>常数

$O(a^n)>O(n^a)>O(\log n)>O(1)$

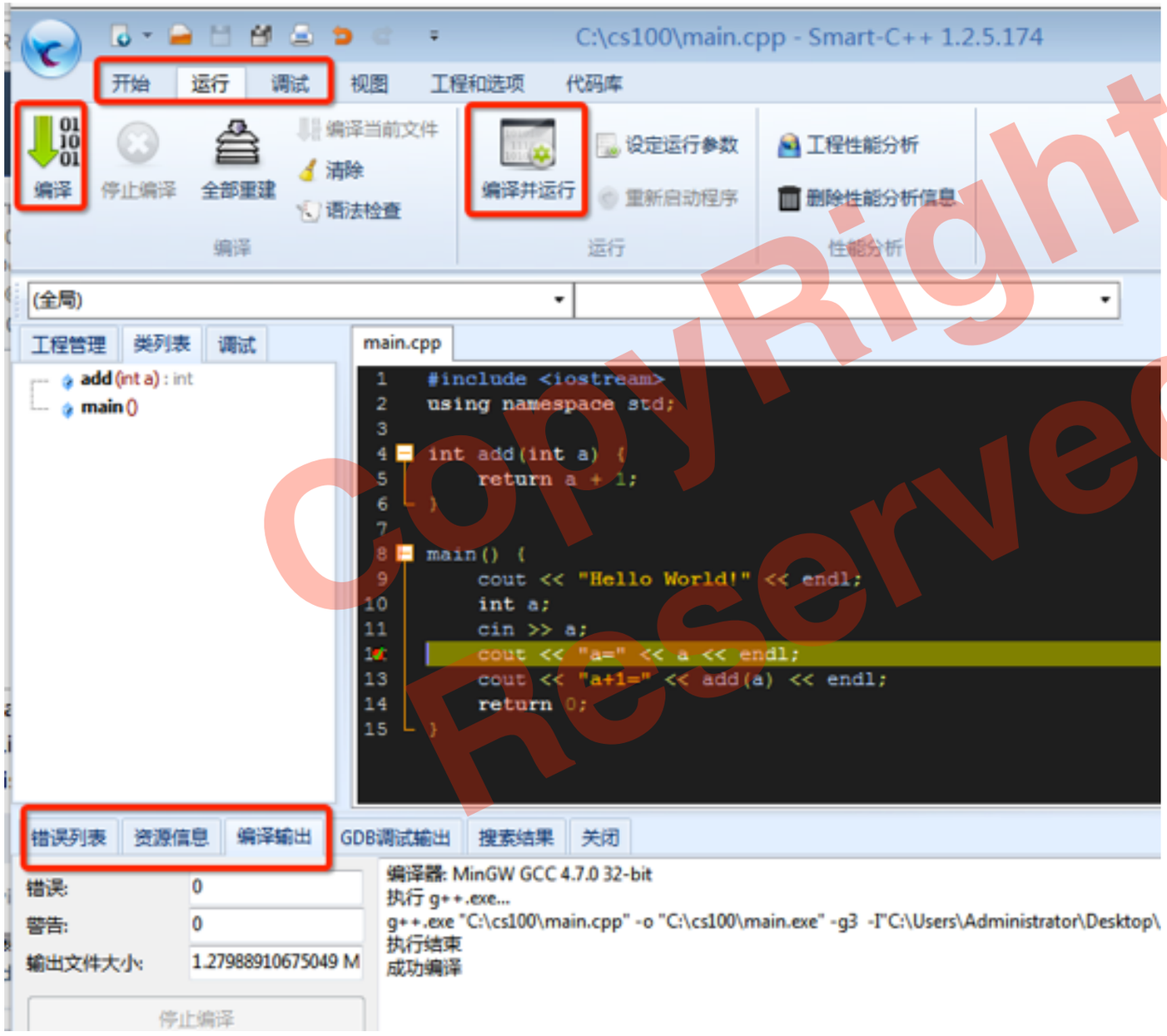
作业回顾：问题总结

```
void sortByMerge(int data[],int head,int tail){
    if (head==tail) { // 递归边界
        return;
    }
    int mid=(head+tail)/2;
    // cerr<<"head="<<head<<" mid="<<mid<<" tail="<<tail<<"\n";
    sortByMerge(data,head,mid);   // 排序左子列
    sortByMerge(data,mid+1,tail); // 排序右子列

    int left=head, right=mid+1, i=0;    // 左右两个游标
    int tmp[tail-head+1];   // 存放临时结果
    while (left<=mid || right<=tail) {  // 归并过程
        // 把左右游标中较小的一个放入tmp的当前位置
        if (right>tail|| (left<=mid && data[left]<data[right])){
            tmp[i++]=data[left++];
        } else{
            tmp[i++]=data[right++];
        }
    }
    // 将tmp（排好序的）回填到原数组的head到tail位置
    for (int j=head;j<=tail;j++){
        data[j]=tmp[j-head];
    }
}
```

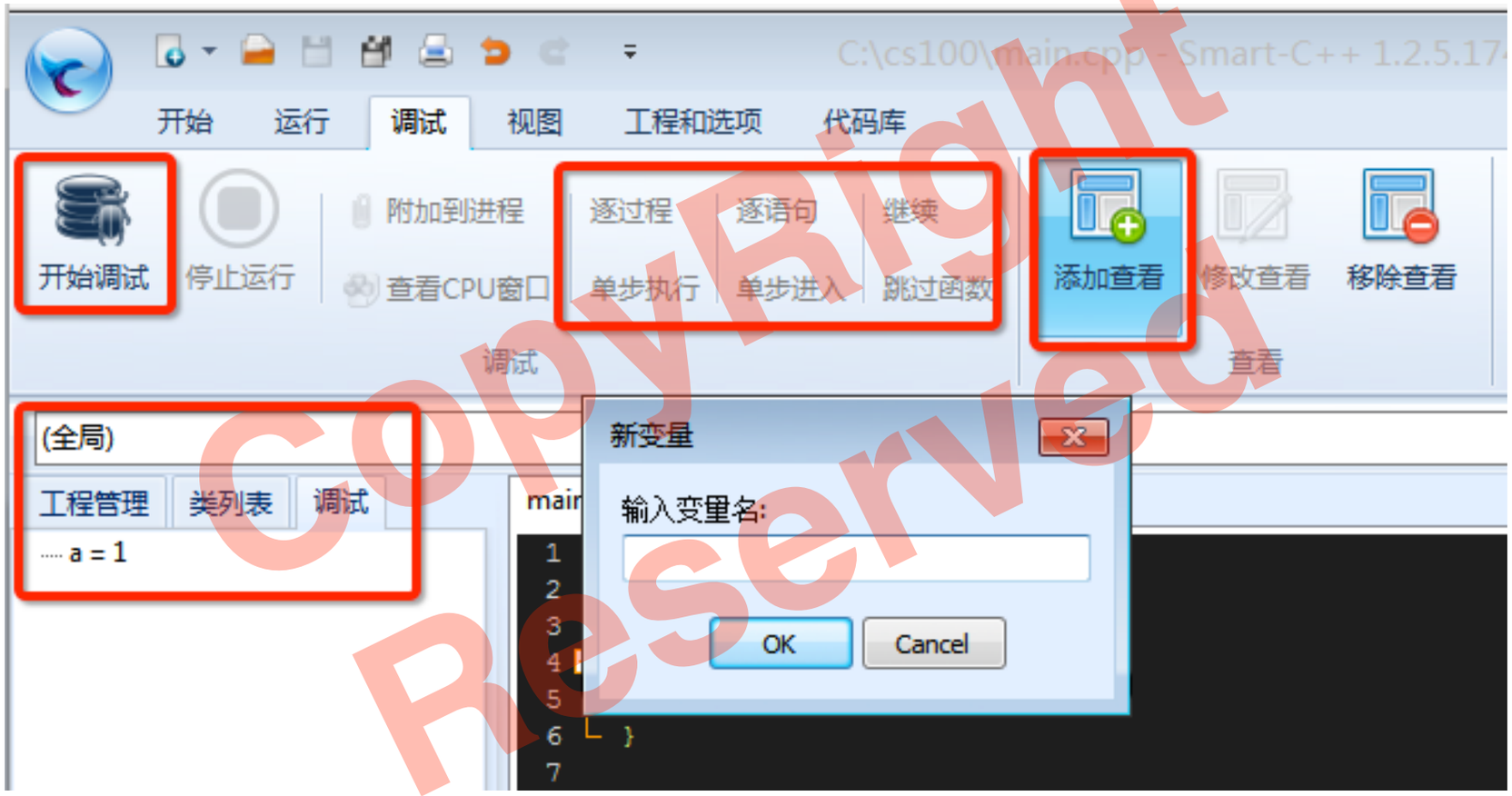


- **新语言还是怕怕?**
  - 巩固一下这些看着不太友好的语法/奇怪的符号
    - >>，<<（流式）输入输出
    - &&，||，！逻辑运算（and，or，not）
    - a++，a-- 累加/累减（相当于a=a±1）
    - a+=b，a*=b （相当于a=a+b, a=a*b）
    - i=a>b ? x:y 三元表达式（相当于if (a>b) {i=a} else {i=b}）
    - vector<int> 整数动态数组，<>里面表示数组元素的类型
    - for (int i=0;i<n;i++) i从0循环到n-1
- **读代码的问题**
  - 我们这是算法课，如果你语法真的困难很大，可以**只关注伪代码，先理解算法思路**
  - 花括号{}勾勒出程序的层次结构，按这个层次去看代码
  - PPT里只贴了核心代码，例题完整代码在cpp文件里（但是不要复制粘贴）
- **读/写代码的问题**
  - 虽然C++对缩进不敏感，还是**要求**像Python那样缩进写代码，这样能帮助理清程序结构
  - 所有复合语句（for、if、else、while）**所跟的代码块都用花括号围起来，即使只跟一句语句**

SmartC++：开发



SmartC++：调试



作业回顾：编程题

4.dupnumber.cpp（同学讲解）

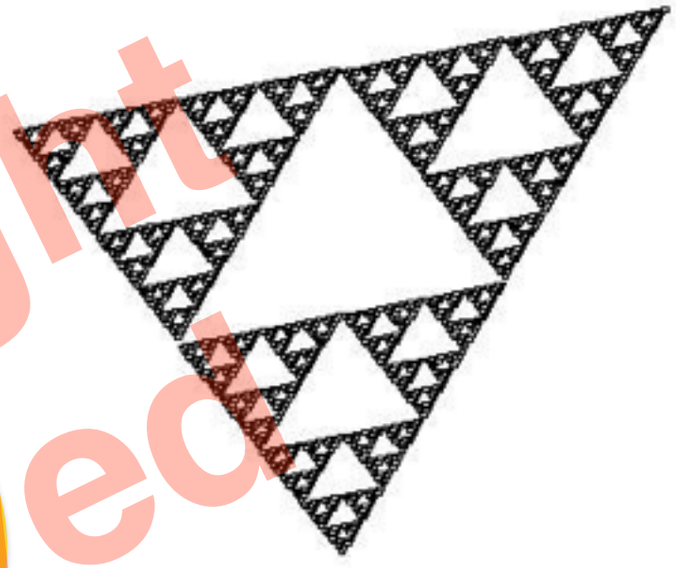
5.factorial.cpp

Welcome to Xcode

CS100 算法入门

分治法

先讲一点哲学：自相似



分治法：
大事化小
小事化了

比排序多一点：逆序对问题

逆序对问题

输入一个整数序列a（长度<=1000000），元素各不相同

输出其中逆序对的总数。逆序对指这样的数对(a[i],a[j])，满足i<j && a[i]>a[j]

样例输入：

10

13 21 11 31 32 22 12 33 23 99

样例输出：

12

注：所有的逆序对为

(13,11),(13,12),(21,11),(21,12),(31,22),(31,12),(31,23),(32,22),(32,12),(32,23),(22,12),(33,23)

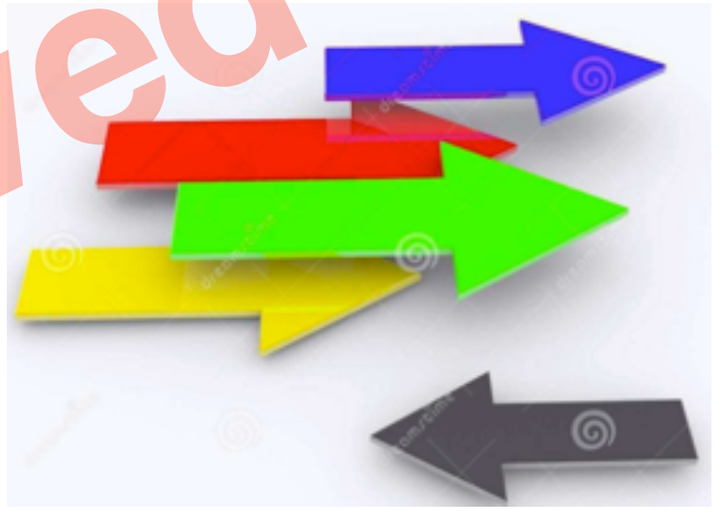
百度面试真题

逆序对问题：解答

```
int count(a[head:tail]){
  mid=(head+tail)/2, result=0
  result+=count(a,head,mid)
  result+=count(a,mid+1,tail)
  设置游标left,right和临时数组tmp
  while (没有排序完) {
    if (左边小) {
      left++进入tmp
    } else {
      right++进入tmp
      result+=(mid+1-left)
    }
  }
  tmp回填到a
  return result
}
```

两重循环肯定不行的
O(nlogn)是比较可行的

跟归并排序很像？bingo！
其实排序就是消除逆序对的过程



代码参见inversepair.cpp

DO IT!

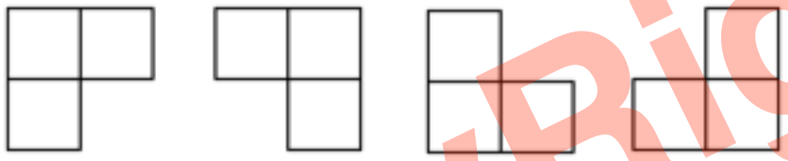
使用分治算法的一般步骤

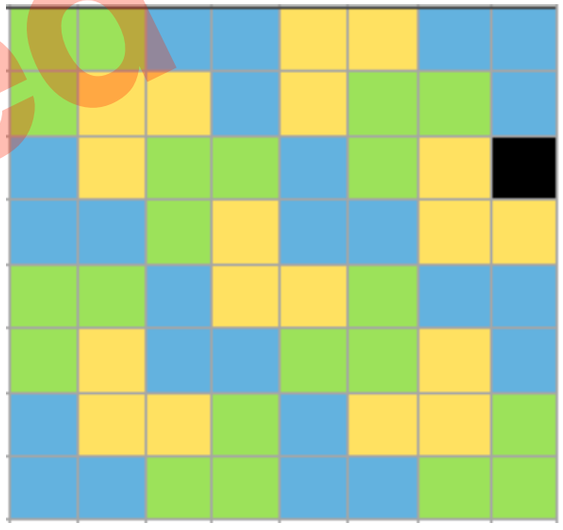
1.将问题分解为较小规模的多个子问题（一般是一分为二）
2.递归求解子问题（最简单，自己调用自己）
3.注意递归边界（通常很简单，当n=1的时候，啥事都不是事😝）
4.将子问题的解合并为原问题的解（关键！一般需要充分利用子问题已解决带来的额外条件）

创造分治条件：棋盘覆盖问题

棋盘覆盖问题

一个n*n的棋盘（n是2的幂次，n<=1024），其中有一格(x,y)是涂黑的。使用如下的3格小部件将剩余棋盘铺满。输入n,x,y，输出铺填的方案





样例输入：

8 2 7

样例输出：

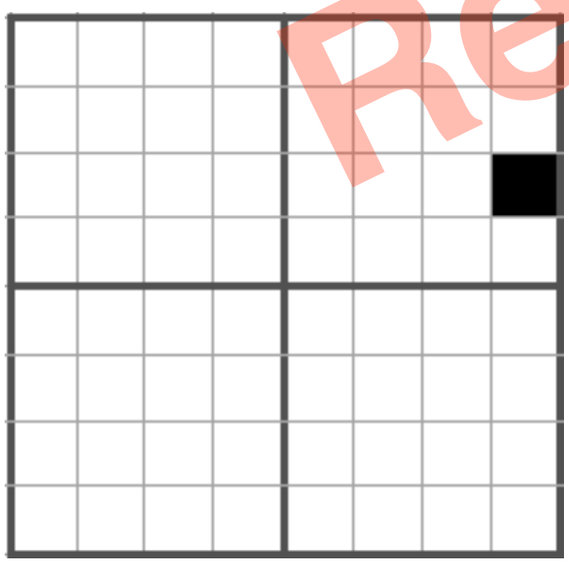
```
3  3  6  6  18 18 21 21
3  2  2  6  18 17 17 21
4  2  5  5  19 17 20  0
4  4  5  1  19 19 20 20
8  8 11  1   1 13 16 16
8  7 11 11  13 13 12 16
9  7  7 10  14 12 12 15
9  9 10 10  14 14 15 15
```

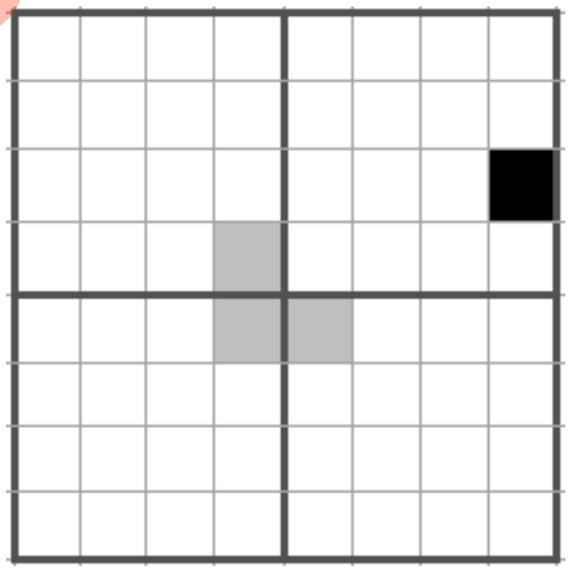
棋盘覆盖问题：思路

void cover(){
  将棋盘划分成n/2*n/2的4个子棋盘
  在中央放置一个部件，形成自相似
  递归覆盖四个子棋盘
  合并？这题不需要合并，哈哈，子棋盘拼起来就是整个解了
}

题目都说了是2的幂次摆明了就是让分治嘛😑
其实这题不用分治也没辙，暴力枚举都没地方下手

就这么愉快的解决了？慢着！
原问题是缺一格的，但子棋盘有3个并没有缺，没有形成自相似
so？我们就人为抹掉一格，就这么简单！





棋盘覆盖问题：解答

```
vector<vector<int>> map;  // 二维动态数组
int id; // 全局变量，表示当前放入的小组件编号

/*
 * @param n 棋盘边长
 * @param x,y 缺陷的位置
 * @param sx,xy 当前小棋盘左上角在整个大棋盘里的位置（为了填充小组件用）
 */
void cover(int n,int x,int y,int sx,int sy) {
    if (n==1) { // 递归边界，1*1的棋盘，自然不用填充了
        return;
    }

    id++;   // 中间一块的id
    int n2=n/2;    // 小棋盘的尺寸

    if (x<n2 && y<n2){  // 缺一块在左上子棋盘
        // 放置中间位置的小组件，让三个完整的小棋盘都缺一块
        map[sx+n2][sy+n2-1]=map[sx+n2][sy+n2]=map[sx+n2-1][sy+n2]=id;
        // 递归填充4个小棋盘
        cover(n2, x, y, sx, sy);
        cover(n2, 0, n2-1, sx+n2, sy);
        cover(n2, 0, 0, sx+n2, sy+n2);
        cover(n2, n2-1, 0, sx, sy+n2);

    }else if (x>=n2 && y<n2){   // 缺一块在右上子棋盘
        map[sx+n2][sy+n2]=map[sx+n2-1][sy+n2]=map[sx+n2-1][sy+n2-1]=id;
        // 后几个分支只要细心一点就行
        cover(n2, n2-1, n2-1, sx, sy);
        cover(n2, x-n2, y, sx+n2, sy);  // 这个带原始缺陷的小棋盘，注意第2，3个参数的变化
        cover(n2, 0, 0, sx+n2, sy+n2);
        cover(n2, n2-1, 0, sx, sy+n2);
    }else if (x>=n2 && y>=n2){...}else if (x<n2 && y>=n2){...}
}
```

→ 这里有一种连环赋值的写法，你也可以用，可以节省行数。纯属语法技巧（有人称为语法糖）

代码参见mapcover.cpp

→ 因为篇幅有限，后面两个if我折叠起来了，你可以课后去看代码。这也是ide提供的一个辅助功能（特别是代码比较长看不清结构的时候）

棋盘覆盖问题：解答（怪长的。。）

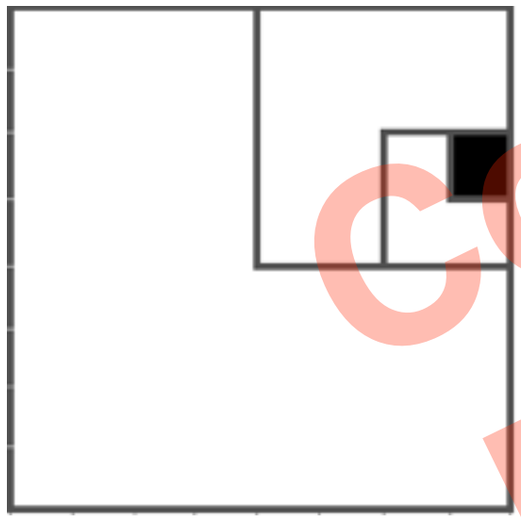
```cpp
int main(int argc, const char * argv[]) {
    /* 8 2 7 */
    int n,x,y;  // 输入
    scanf("%d%d%d",&n,&x,&y);

    // 初始化棋盘
    map=vector<vector<int>>(n);
    for(int i=0; i<n ; i++){
        map[i]=vector<int>(n);
    }
    id=0;

    // 输出
    cover(n, x, y, 0, 0);
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            cout<<setw(8)<<map[i][j];
        }
        cout<<endl;
    }
    return 0;
}
```

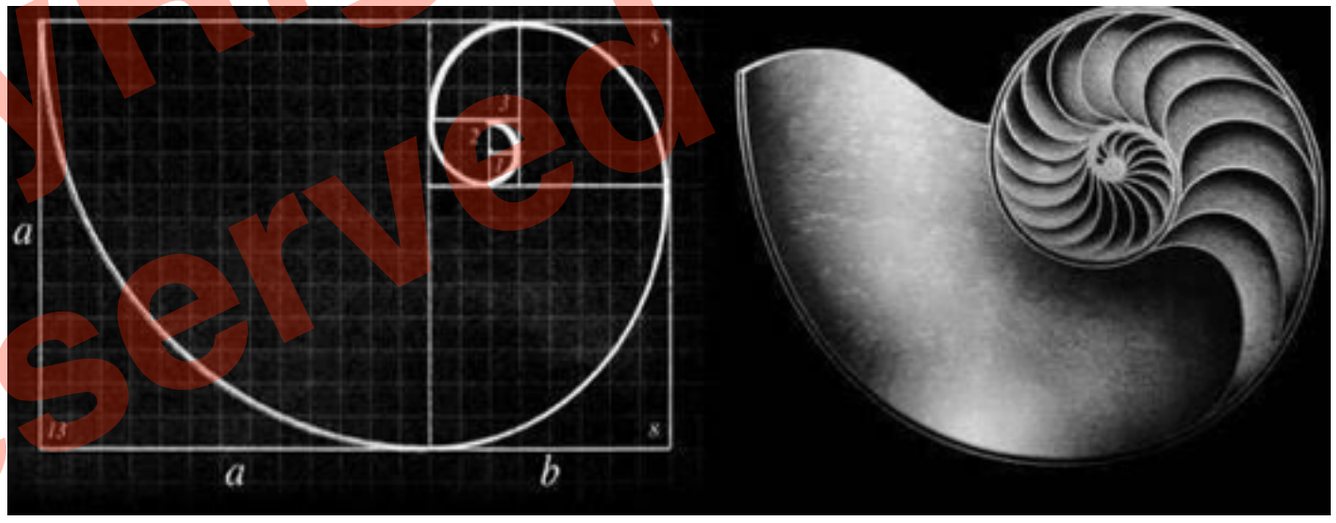
```
8 2 7
       3       3       6       6      18      18      21      21
       3       2       2       6      18      17      17      21
       4       2       5       5      19      17      20       0
       4       4       5       1      19      19      20      20
       8       8      11       1       1      13      16      16
       8       7      11      11      13      13      12      16
       9       7       7      10      14      12      12      15
       9       9      10      10      14      14      15      15
Program ended with exit code: 0
```

DO IT!

棋盘覆盖问题：美学

你可以把大棋盘看做不同size的小部件拼成的





右边这个图形称为Fibonacci螺旋线

是不是还挺漂亮的?

此处插播硬广告：类

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

class Point {    // 平面上的点
public:
    int x=0,y=0;
    Point(int x0,int y0){ //构造函数（初始化）
        x=x0; y=y0;
    }

    void print(){    // 打印点的信息（坐标）
        cout<<"("<<x<<","<<y<<")"<<endl;
    }
};

class Circle {  // 平面上的圆
public:
    Point ctr = Point(0, 0);  // 圆心
    int r=0;//半径

    Circle(const Point& p,int r0) { //构造函数
        ctr=p;
        r=r0;
    }

    double area(){  // 求面积函数
        return 3.1415926*r*r;
    }

    void print(){    // 打印圆的信息（坐标）
        cout<<"centre=("<<ctr.x<<","<<ctr.y<<") radius="<<r<<endl;
    }
};
```

→ 类(Class)对多个变量和函数的组合，组成更有逻辑意义的复合类型（可以理解为自定义的数据类型）

→ 你们见过的string，vector，其实都是class

为什么要使用类?

→ 为了封装逻辑上一体的程序模块，使程序结构更清晰

→ 使用一些复杂数据结构时，不封装成类会很困难（下两节课会看到）

→ 类中的变量有的语言称为属性(Field)，类中的函数有的语言称为方法(Method)。只是名词不同而已

→ 类其实是一个很深刻的概念，很多语言都有类的概念（包括Python）。感兴趣的可以去查一下什么是面向对象编程(Object Oriented Programming, OOP)

充分利用子问题的解（选学）

最近点对问题

输入平面上n个点的坐标（n<=1000000)

求相距最近的两点之间的距离（精确到3位小数）

样例输入：

3
0 0
1 1
2 3

样例输出：

1.414

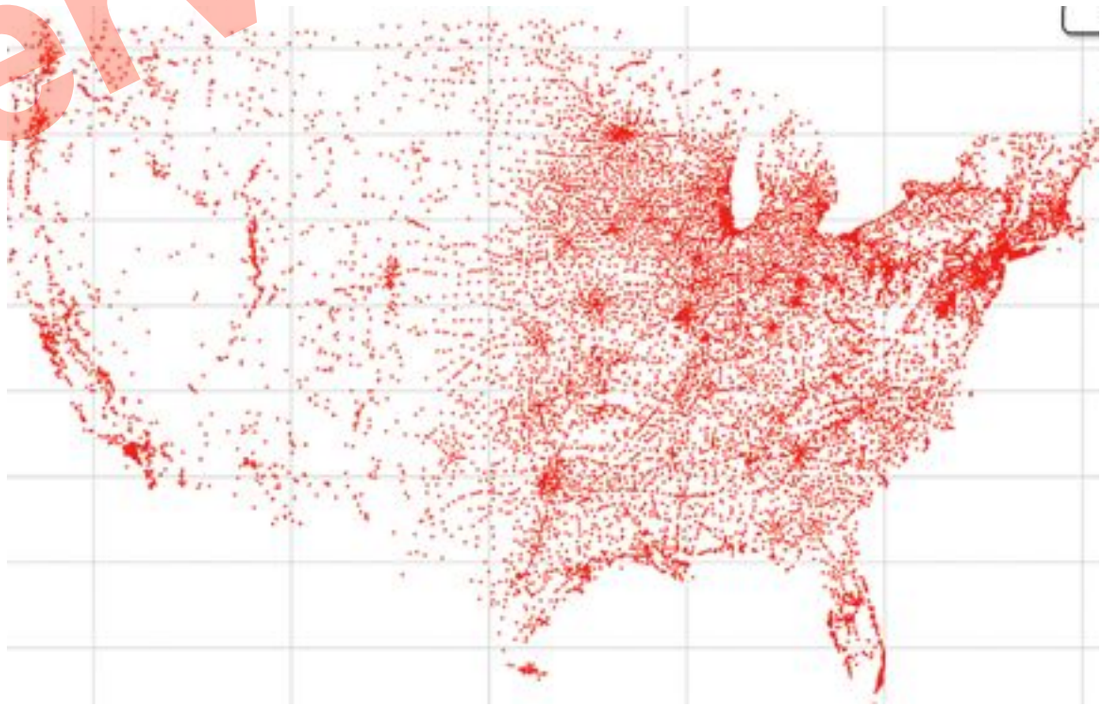
最近点对问题：思路

输入所有点的坐标，记为points
double calc(points[], left,right){
  mid=(left+right)/2
  double d=min(calc(a,left,mid),calc(points,mid+1,right))
  枚举a[left:mid]和a[mid+1,right]中所有点对(points[i],points[j]) {
      d=min(d,dis(a[i],a[j]))
  }
  return d
}

→ **两重循环咱就不用说了，来看看分治法**

→ **这算法复杂度是多少?**
实际上两两点的距离都算了一遍，所以就是$O(n^2)$
这是个形式上的分治法，其实与简单枚举没有区别，并没有体现分治的优势

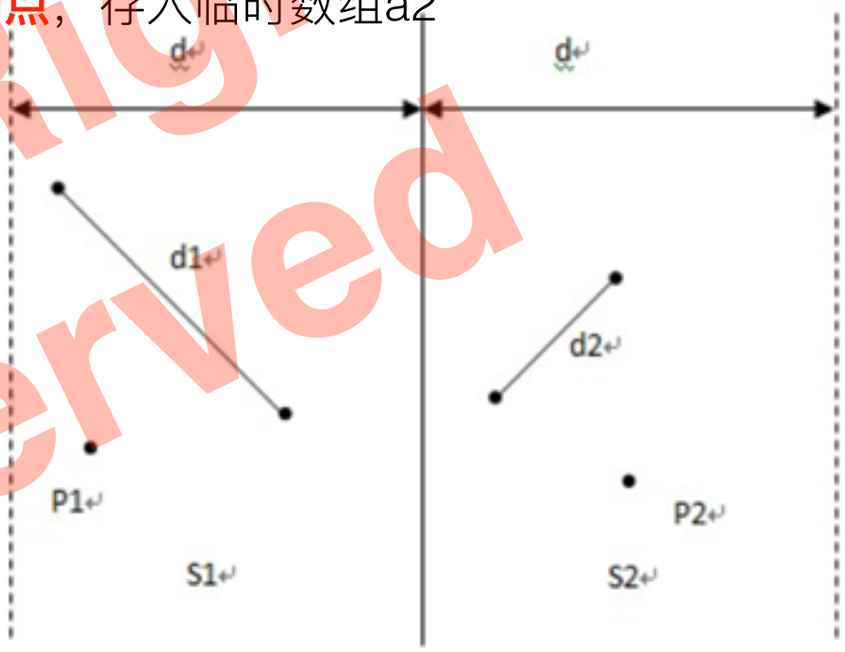
关键：没有利用子问题已解决带来的额外条件



最近点对问题：优化

将a对x坐标排序

```
double calc(a[], left,right){
  mid=(left+right)/2
  double d=min(calc(a,left,mid),calc(a,mid+1,right))
  筛选出x坐标位于a[mid].x-d和a[mid].x+d之间的点，存入临时数组a2
  枚举a2中所有点对(a2[i],a2[j]) {
      d=min(d,dis(a[i],a[j]))
  }
  return d
}
```

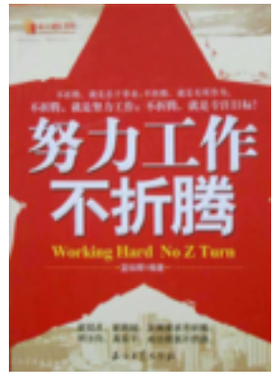


→ 比d更优的解两点的x,y坐标差值不会大于d

→ (x方向)如果有跨越左右的更优解，点对一定在位于中线左右宽2d的条带状区域之间

有多大改进?

a2的尺寸不会大于a[left:right]，得到一定优化，但最坏情况下仍然是$O(n^2)$

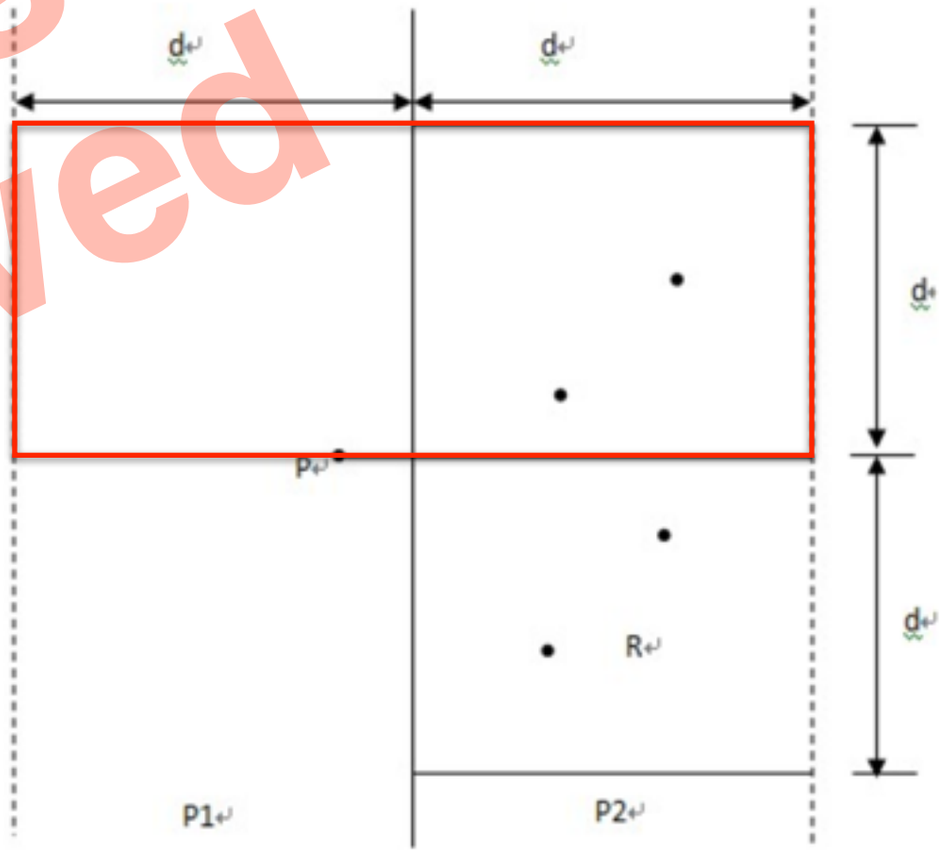


白折腾了?

关键：还是没有充分利用子问题的解，y方向还没利用

最近点对问题：继续优化

```
double calc(a[], left,right){
  mid=(left+right)/2
  double d=min(calc(a,left,mid),calc(a,mid+1,right))
  筛选出x坐标位于a[mid].x-d和a[mid].x+d之间的点，存入临时数组a2
  将a2按照y坐标优先排序
  for (i=0;i<a2.size;i++) {
      for (j=i+1;j<=i+7 && j<a2.size;j++){
          d=min(d,dis(a[i],a[j]))
      }
  }
  return d
}
```



→ (y方向)如果有更优解一定，在2d*d的矩形区间内

→ 2d*d的矩形中至多有8个点，否则必然有一侧存在两个点距离小于d，与d是子问题最优解矛盾（抽屉原理）

→ 这个算法的复杂度降到了O(n*logn*logn)

怎么算出来的？计算过程留作选作作业

→ 代码在nearestpoint2d.cpp

这个代码有点长，有兴趣的同学可以去看看。这个题目只要理解伪代码就可以了😊

作业

1.我在baidu地图标记了我想去的地方，你也画一下。提交截图

2.（选做）带重逆序对数（inversepairwithdup.cpp）

与例题相同，但输入中可能有重复数据，输出逆序对数（相同的两个数不算逆序对）

样例输入：

6

2 3 5 3 2 1

样例输出：

9

（提示：只需要例题上改一点点就可以了）

3.一元三次方程（cubicequation.cpp）

输入三个实数b,c,d（绝对值不超过100）

求方程$x^3+bx^2+cx+d=0$的一个根（不需要所有的），精确到3位小数

样例输入：

2 3 4

样例输出：

-1.650

（提示：1.可以通过分治法去猜解的数值；

2.怎么证明一定有解？参见扩展阅读-**介值定理**）

3.不要直接用求根公式

4.（选做）写一个类表示一个矩形，包含如下信息（squareclass.cpp）

a.矩形长/宽的值：w/h

b.一个返回矩形面积的函数：double area()

c.一个返回矩形是否为正方形的函数：int isSquare()

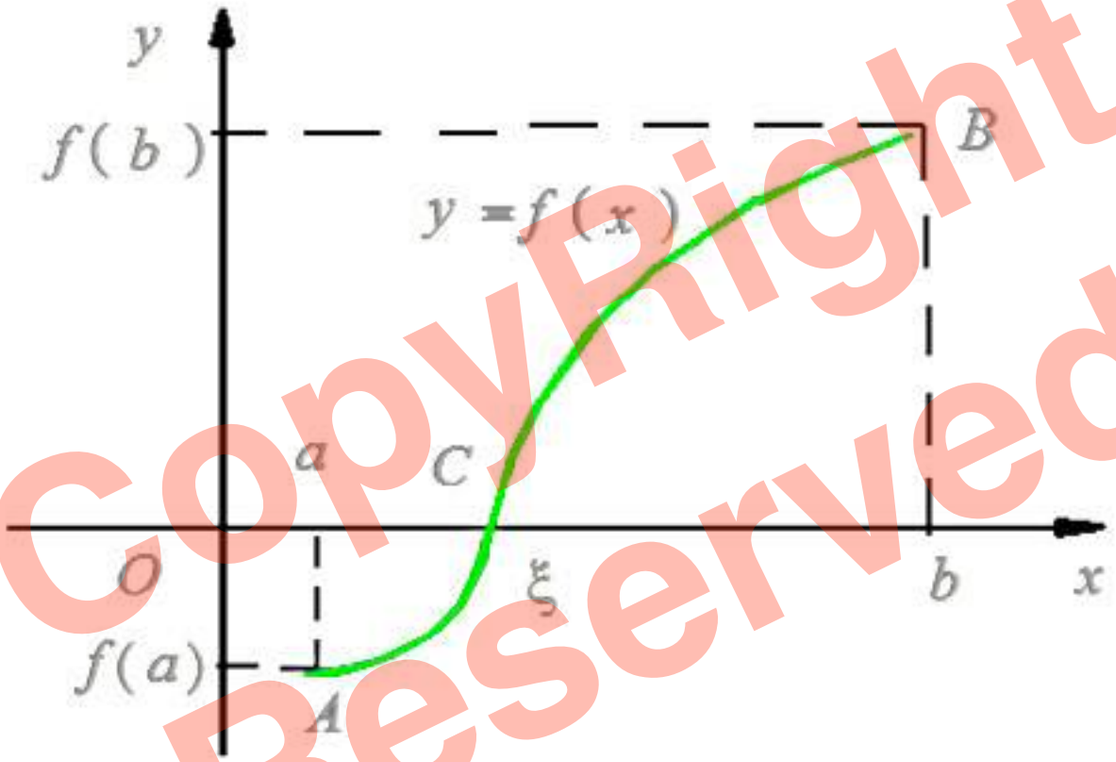
NOTE：

1.如果C++实在有困难，可以用Python写；

2.Python也是在有困难，可以写伪代码，但要写的详细

3.上交代码都要求带正确缩进（不论C++、Python、伪代码）

扩展阅读：介值定理



→ 一个连续函数f(x)，如果f(a)<0且f(b)>0（a<b），那么一定存在一点c，满足

a<c<b且f(c)=0