Hello, Ladies and Gentlemen.

女士们先生们大家好

Bonjour, Mesdames et Messieurs.

Witajcie, Panie i Panowie.

Здравствуйте, дамы и господа.

Hallo, Damen und Herren.

Buna ziua, Doamenelor si Domnilor.

Ciao, signore e signori.

Zeyuan Zhu, Grade 12, Nanjing Foreign Language School, Jiangsu, China.

朱泽园，高三，南京市外国语学校，江苏，中国

New algorithm for Half-plane Intersection and its Practical Value

–– Thesis for Chinese Team Selecting Contest 2006

半平面交的新算法及其实用价值

-- 中国代表队 2006 年选拔赛论文

Zeyuan Zhu, Grade 12, Nanjing Foreign Language School, Jiangsu, China.

朱泽园，高三，南京市外国语学校，江苏，中国

Project Overview – 全文总揽

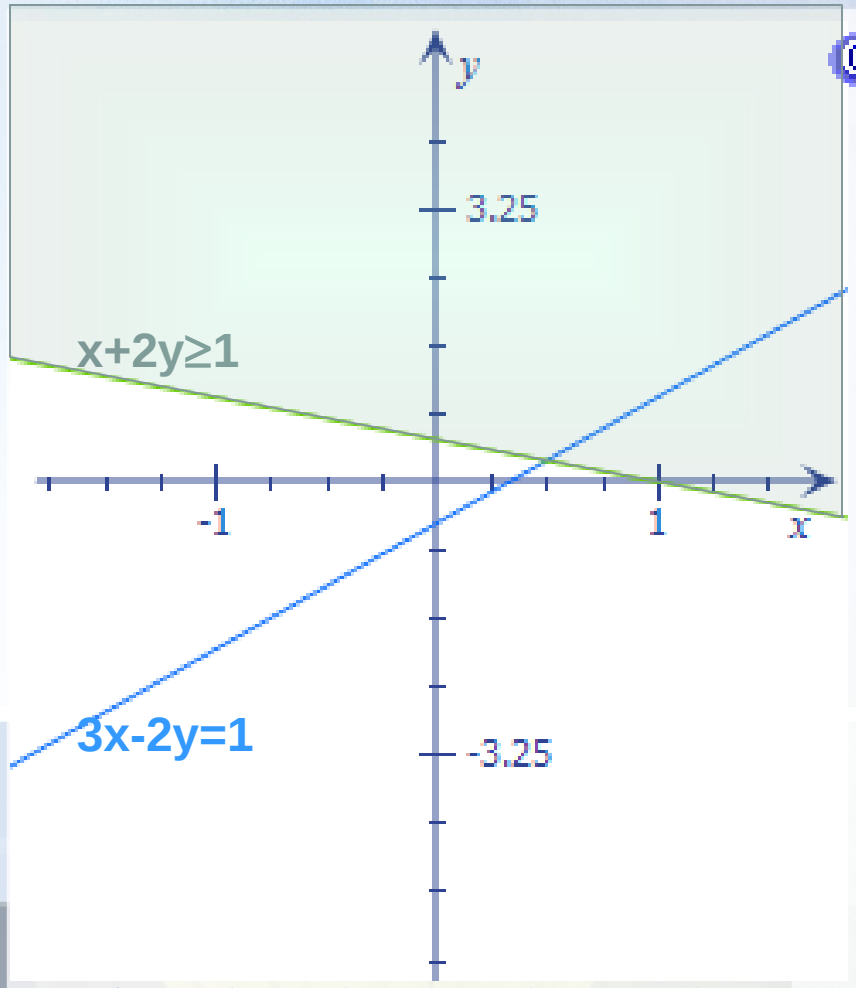
- Aim: Present a new $O(n\log n)$ algorithm for half-plane intersection (abbr. HPI), which is one of the most heatedly discussed problems in computer science; emphasize its advantages in practical application, and to some extent, reduce the complexity to $O(n)$. However, the new algorithm will be extraordinarily easy to be implemented.
- 主旨：半平面的交是当今学术界热烈讨论的问题之一，本文将介绍一个全新的 $O(n\log n)$ 半平面交算法，强调它在实际运用中的价值，并且在某种程度上将复杂度下降至 $O(n)$ 线性。最重要的是，我将介绍的算法非常便于实现。

Project Overview – 全文总揽

- §1 introduces what Half-Plane Intersection (HPI) is. 什么是半平面交 .
- §2 prepares a convex polygon intersection (CPI). 凸多边形交预备知识 .
- §3 briefly discuss a common solution for HPI – D&C. 简要介绍旧 **D&C** 算法 .
- §4 my new algorithm S&I emerges detailedly. 揭开我的新算法 **S&I** 神秘面纱 .
- §5 conclusion and discussion on further practical use. 总结和实际运用 .

1. Statement of the Problem - 问题概述

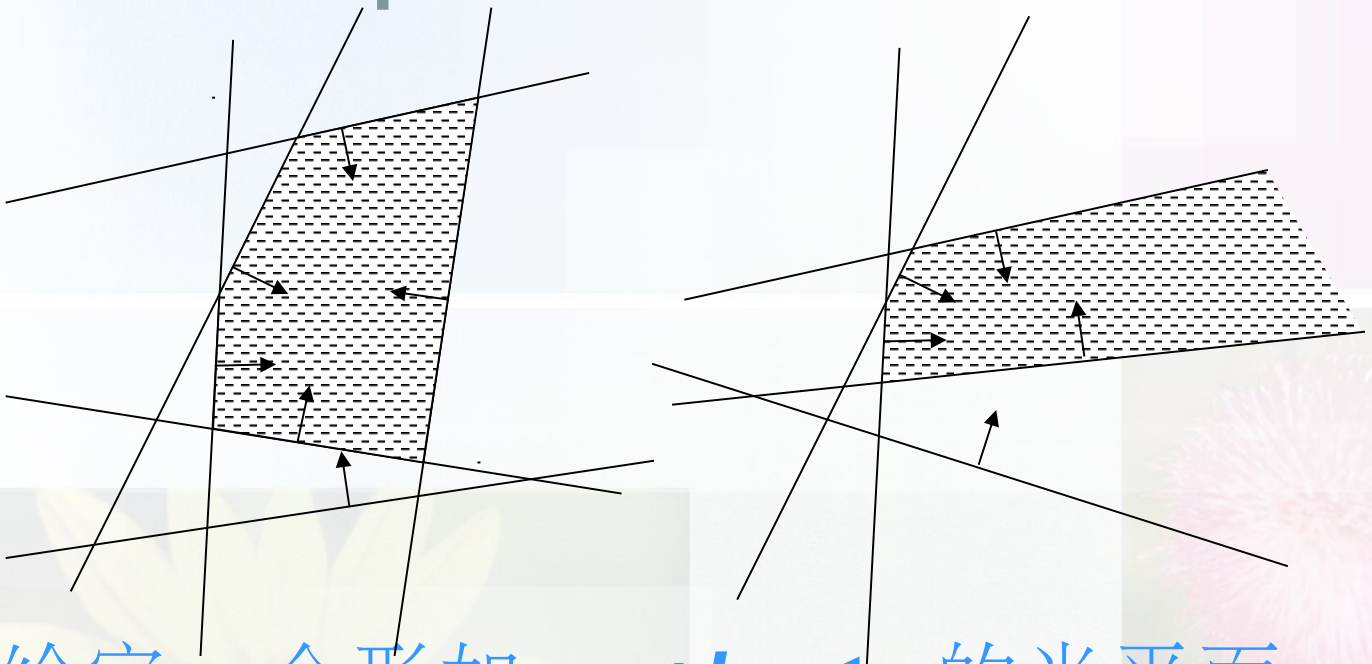
1. Statement of the Problem



- A line in plane is usually represented as ***ax+by=c***. Similarly, its inequality form ***ax+by*** ≤(≥) ***c*** represents a ***half-plane*** (also named ***h-plane*** for short) as one side of this line.
- 众所周知，直线常用 ***ax+by=c*** 表示，类似地半平面以 ***ax+by ≤(≥)c*** 为定义。

1. Statement of the Problem

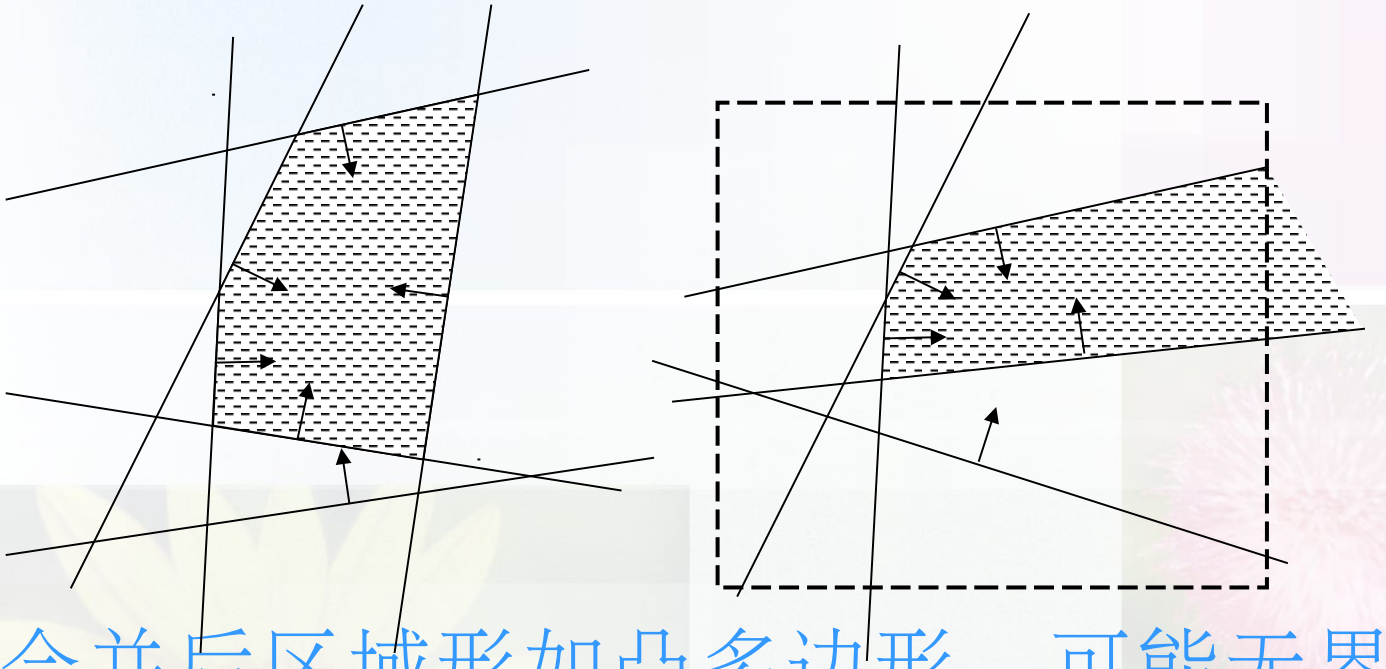
- Given *n* half-planes, $a_i x+b_i y \le c_i$ $(1 \le i \le n)$, you are to determine the set of all points that satisfying all the *n* inequations.



- 给定 **n** 个形如 $a_i x+b_i y \le c_i$ 的半平面，找到所有满足它们的点所组成的点集

1. Statement of the Problem

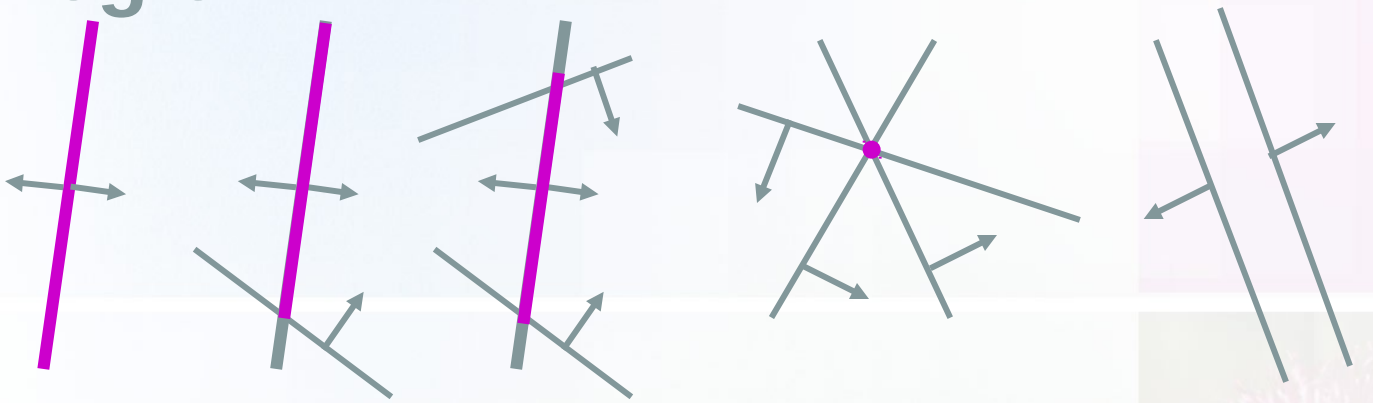
- Feasible region forms a shape of convex hull possibly unbounded. Add four h-planes forming a rectangle, to make the intersection area finite.



- 合并后区域形如凸多边形，可能无界
  此时增加 **4** 个半平面保证面积有限

1. Statement of the Problem

- Pay attention that intersection sometimes yields a line, a ray, a line-segment, a point or an empty region.



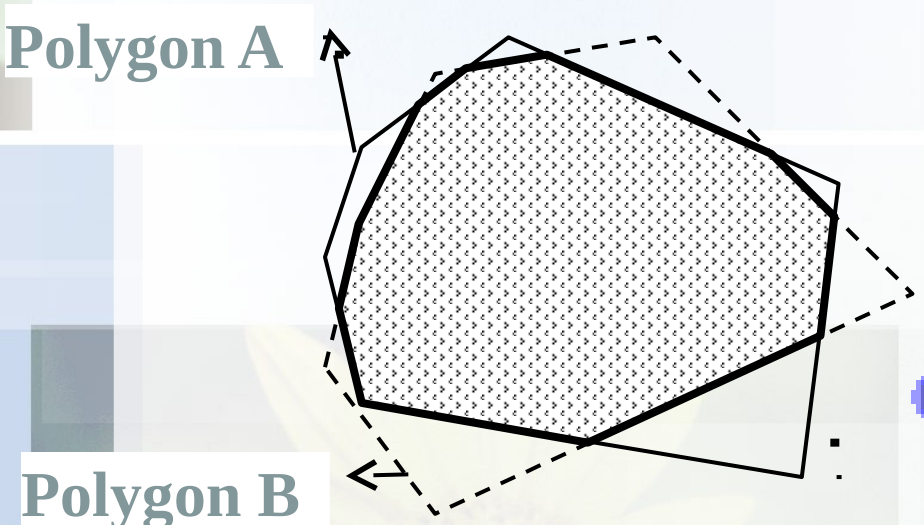
- 注意相交后的区域，有可能是一个直线、射线、线段或者点，当然也可能是空集。

2. Convex Polygon Intersection – CPI

凸多边形交预备知识

2. Convex Polygon Intersection

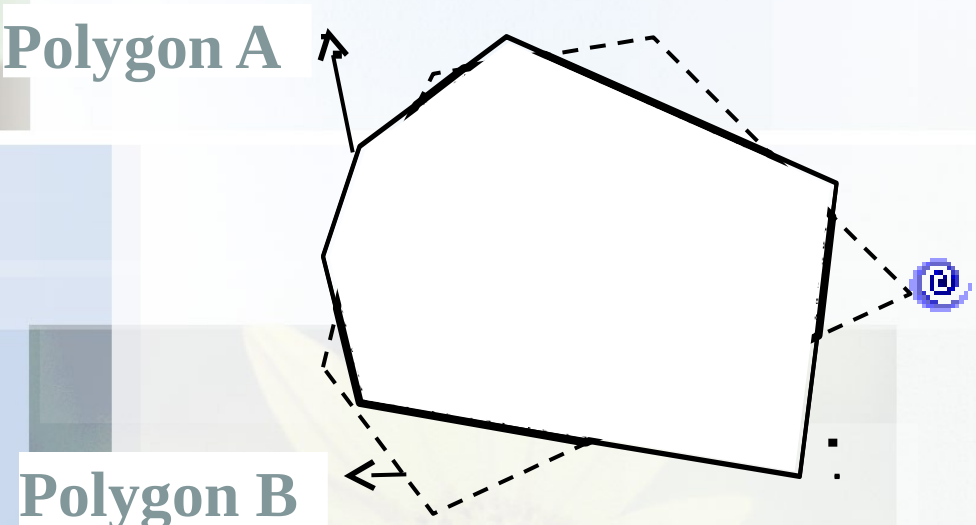
- Intersecting two convex polygons A and B into a single one.
- We will sketch out an efficient way, named ***plane sweep method***.



- 求两个凸多边形 **A** 和 **B** 的交（一个新凸多边形）。
- 我们描绘一个平面扫描法。

2. Convex Polygon Intersection

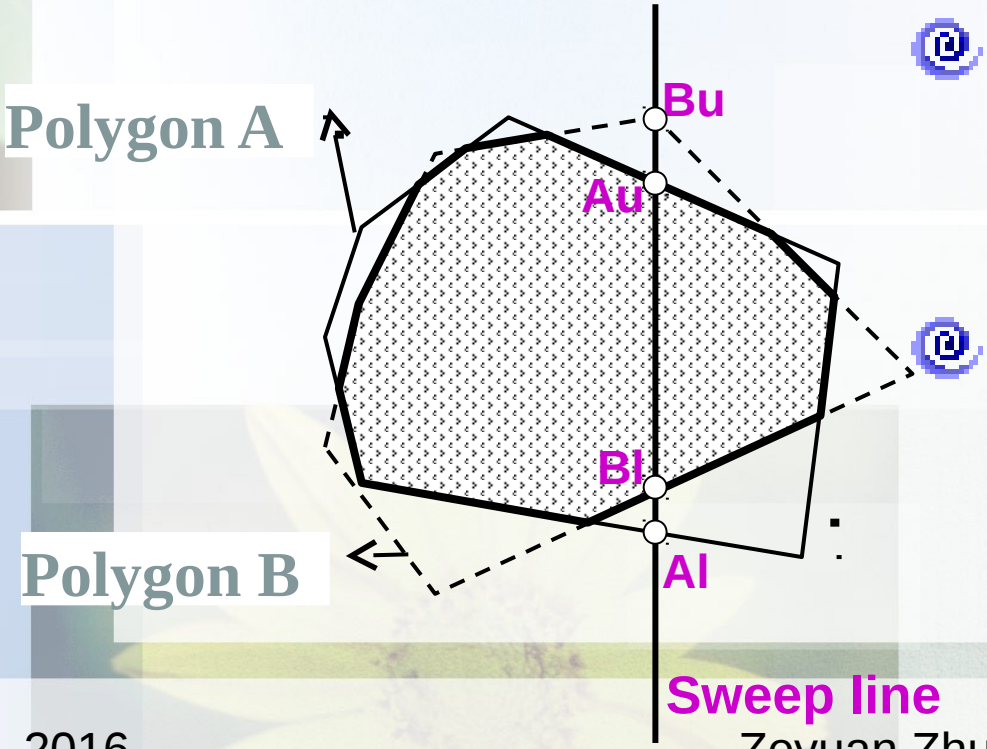
- Main idea: Regard intersections of edges as cutting points, and break boundaries of A and B, into ***outer edges*** and ***inner edges***.
- Segments of ***inner edges*** establish ties to each other, and form a polygon. (in bold)



- 主要思想：以两凸边形边的交点为分界点，将边分为内、外两种。
- 内边互相连接，成为所求多边形（图中粗线条）

2. Convex Polygon Intersection

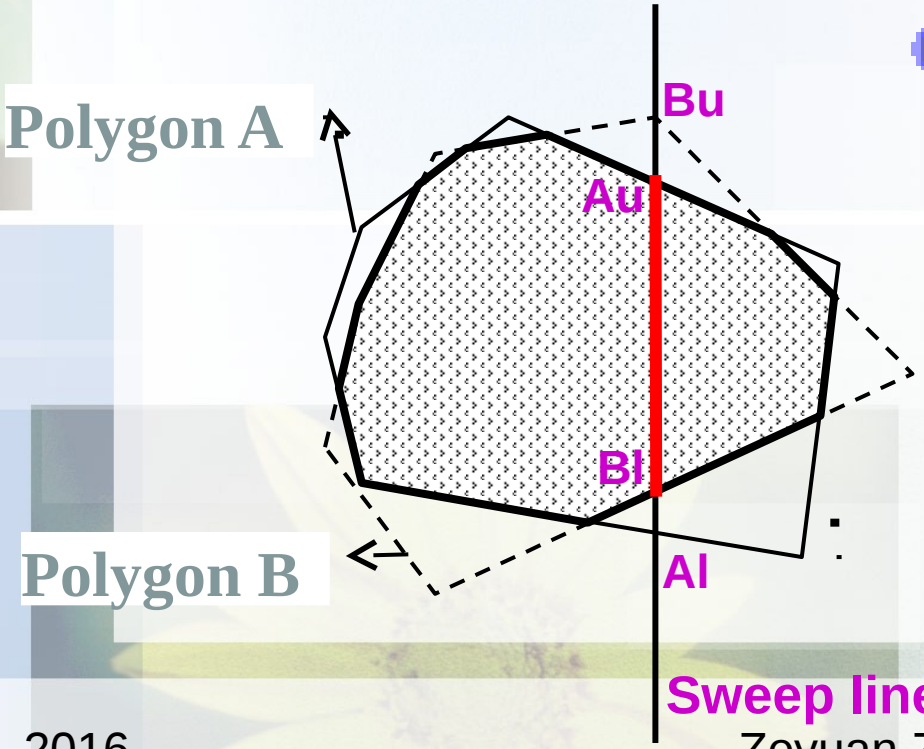
- Suppose there is a vertical sweep line, performing left-to-right sweep.
- At anytime, there are at most four intersections from sweep line to either given polygon.



- 假设有一个垂直的扫描线，从左向右扫描
- 任何时刻，扫描线和两个多边形最多 **4** 个交点

2. Convex Polygon Intersection

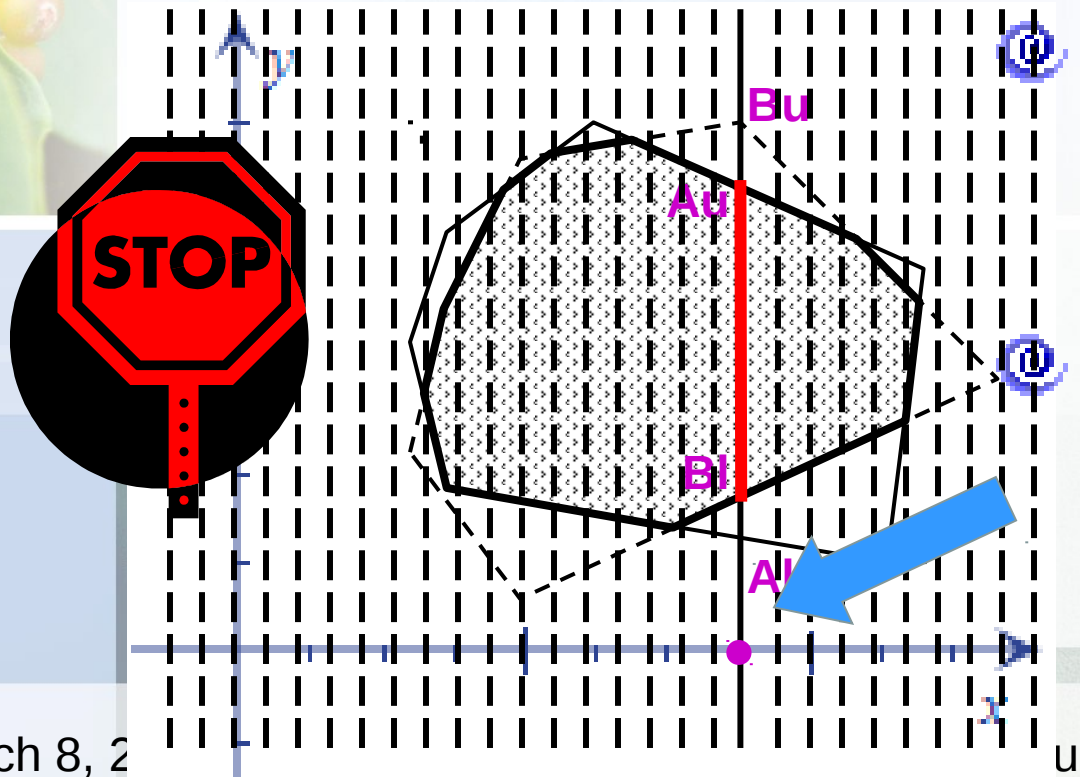
- the lower one between Au and Bu, and the upper one between Al and Bl, form an interval of the current inner region – the red segment in bold.



- **Au**、**Bu** 中靠下的，和 **Al**、**Bl** 中靠上的，组成了当前多边形的内部区域

2. Convex Polygon Intersection

- Let us call the x-coordinates to be swept *x-events*.
- Obviously, the sweep line may not go through all the x-event with rational coordinates!



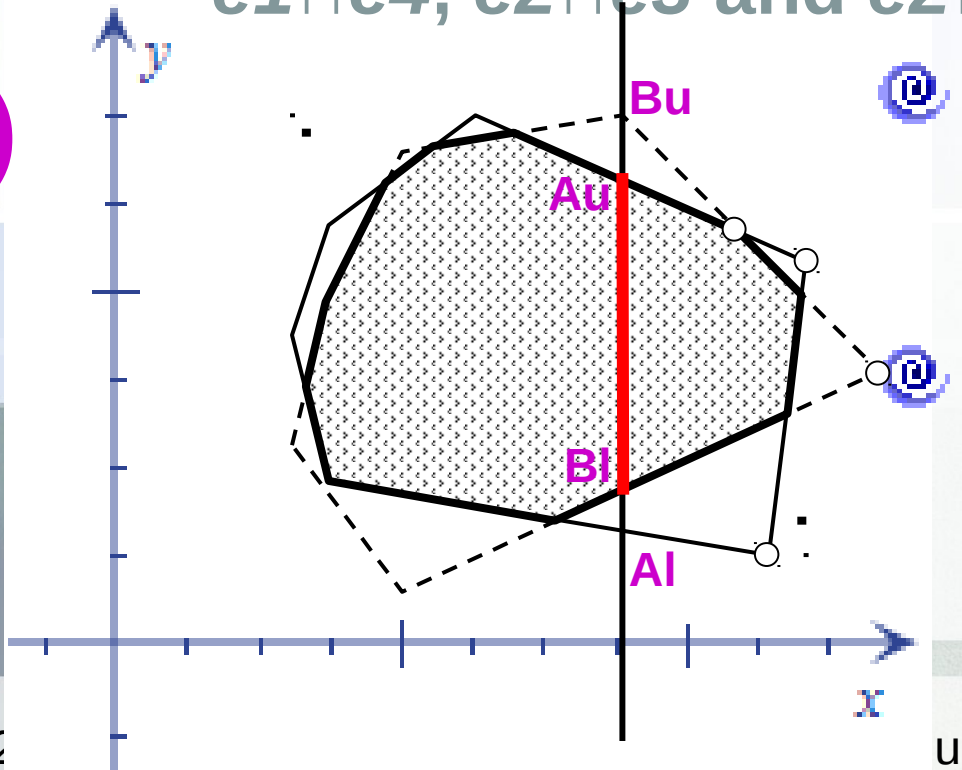
我们称被扫描线扫描到的 **x** 坐标叫做 **x** 事件。

当然，我们不能扫描所有有理数！

2. Convex Polygon Intersection

- Call the edges where Au, Al, Bu and Bl are: *e1, e2, e3* and *e4* respectively.
- Next x-event should be chosen among four endpoints of *e1*, *e2*, *e3* and *e4*, and four potential intersections: *e1*∩*e3*, *e1*∩*e4*, *e2*∩*e3* and *e2*∩*e4*.

O(n)



- 称 **Au, Al, Bu, Bl** 所在的边叫做 **e1,e2,e3,e4**
- 下一个 **x** 事件将在这四条边的端点，以及两两交点中选出

3. Common solution: Divide-and-Conquer Algorithm – D&C
通常的分治解法

3. Divide-and-Conquer Algorithm

- **Divide: Partition the n h-planes into two sets of size n/2.**
- **Conquer: Compute feasible region recursively of both two subsets.**
- **Combine: Compute intersection of two convex region, by CPI§2**
- 分：将 **n** 个半平面分成两个 **n/2** 的集合.
- 治：对两子集合递归求解半平面交.
- 合：将前一步算出来的两个交 **(** 凸多边形 **)** 利用第 **2** 章的 **CPI** 求解.

3. Divide-and-Conquer Algorithm

- The total time complexity of the solution can be calculated via recursive equation.
- 总时间复杂度可以用递归分析法.

CPI

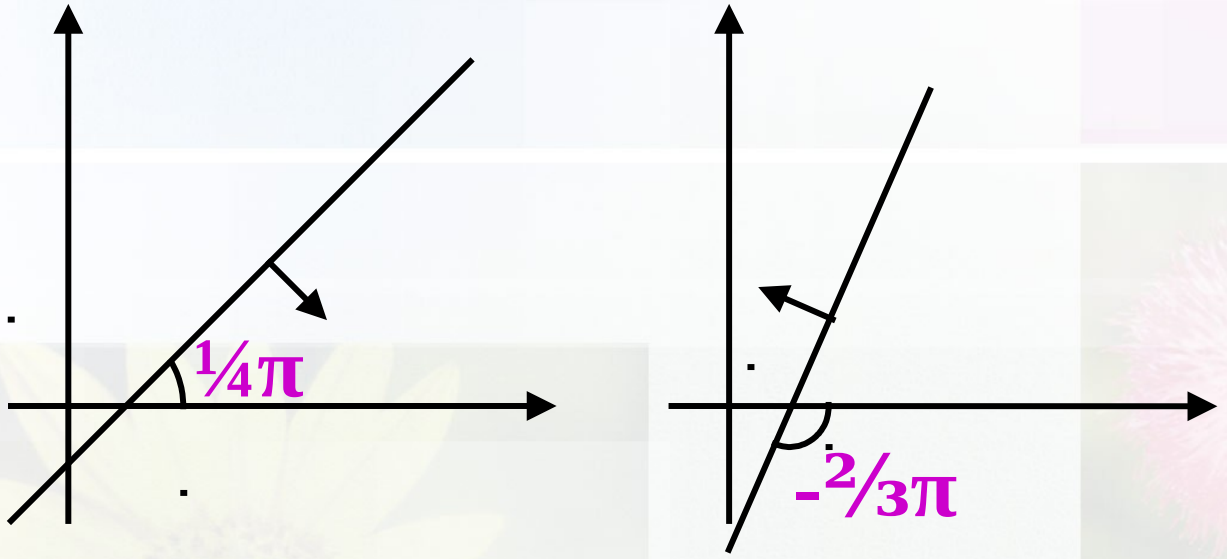
$$T(n) = 2T\left(\frac{n}{2}\right) + O(n)$$

$$T(n) = O(n \log n)$$

4. My New Solution: Sort-and-Incremental Algorithm – S&I
我自创的排序增量算法

4. Sort-and-Incremental Algorithm

- **Definition of h-plane's polar angle: for h-plane like *x-y≥constant*, we define its polar angle to ¼π.**
- 半平面的极角定义：比如 ***x-y≥*** 常数的半平面，定义它的极角为 **¼π.**

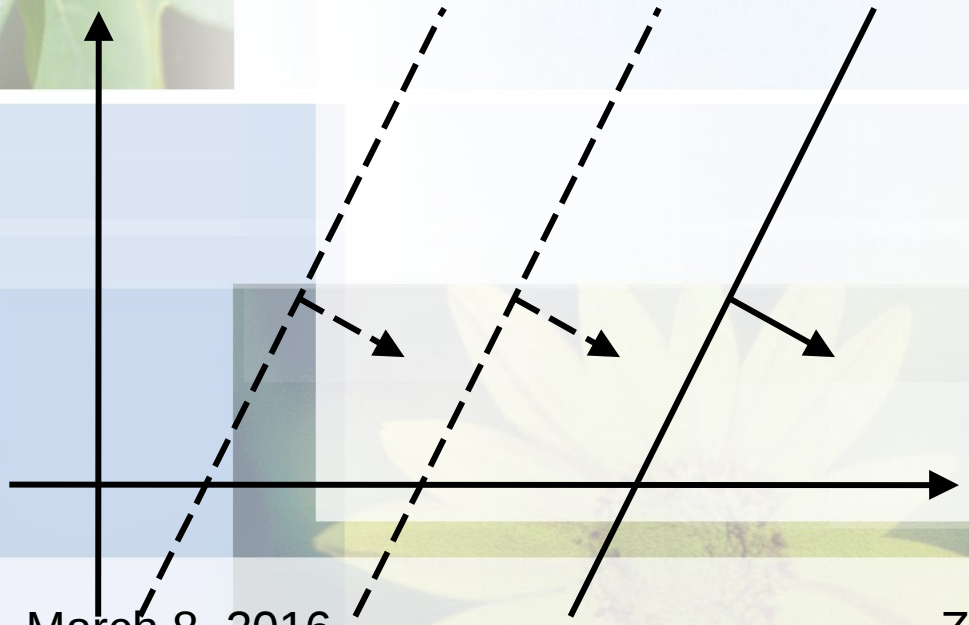


4. Sort-and-Incremental Algorithm

- **Step 1: Separate the h-planes into two sets. One has polar angles of (-½π, ½π], the other has those of (-π, -½π]∪(½π, π].**
- **Step 1:** 将半平面分成两部分，一部分极角范围 **(-½π, ½π]**，另一部分范围 **(-π, -½π]∪(½π, π]**

4. Sort-and-Incremental Algorithm

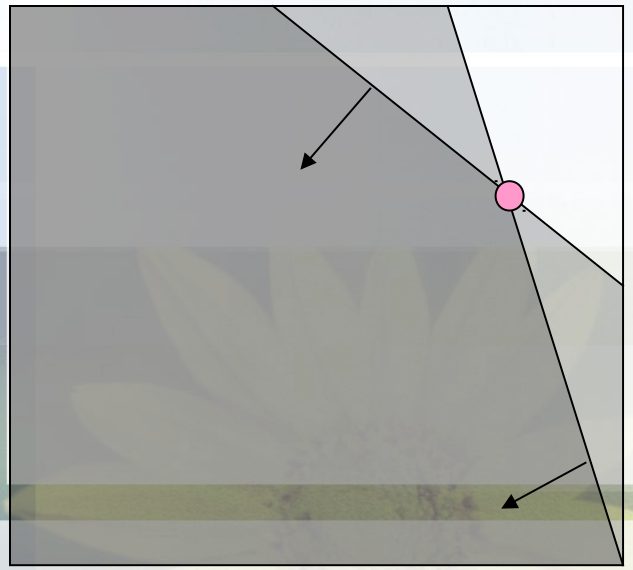
- **Step 2: Consider the set of h-planes in (-½π, ½π] (the other set should also go through step 3 and 4 similarly). Sort them by the polar angle. For the h-planes with the same polar angle, we can keep only one down (delete all others) according to the constant of these h-planes**



- 考虑 **(-½π, ½π]** 的半平面 **(** 另一个集合类似地做 **Step3/4)**，将他们极角排序。对极角相同的半平面，根据常数项保留其中之一。

4. Sort-and-Incremental Algorithm

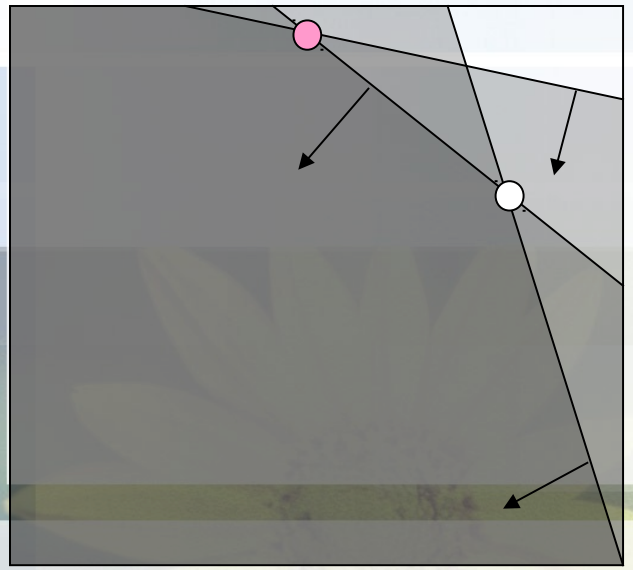
- **Step 3: Starting from two h-planes with the least polar angle, compute their intersection. Push them two into a stack.**



•从排序后极角最小两个半平面开始，求出它们的交点并且将他们押入栈。

4. Sort-and-Incremental Algorithm

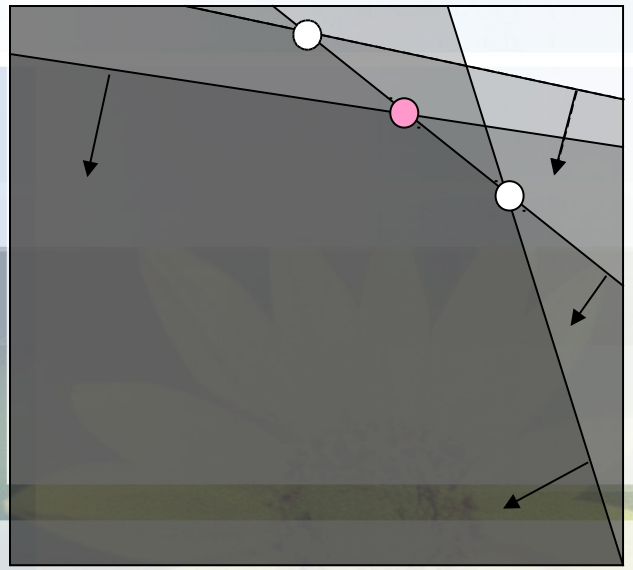
- **Step 3: Each time, add one more h-plane by increasing order of polar angles, and calculate the intersection of it and the top h-plane in stack.**



•每次按照极角从小到大顺序增加一个半平面，算出它与栈顶半平面的交点。

4. Sort-and-Incremental Algorithm

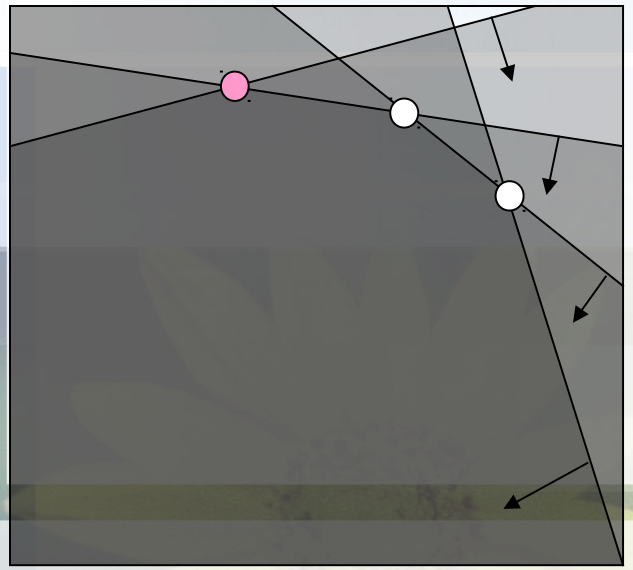
- **Step 3: If this intersection is to the right of the intersection of top two h-planes in stack, we pop the stack once.**



•如果当前的交点在栈顶两个半平面交点的右边，出栈（**pop**）。

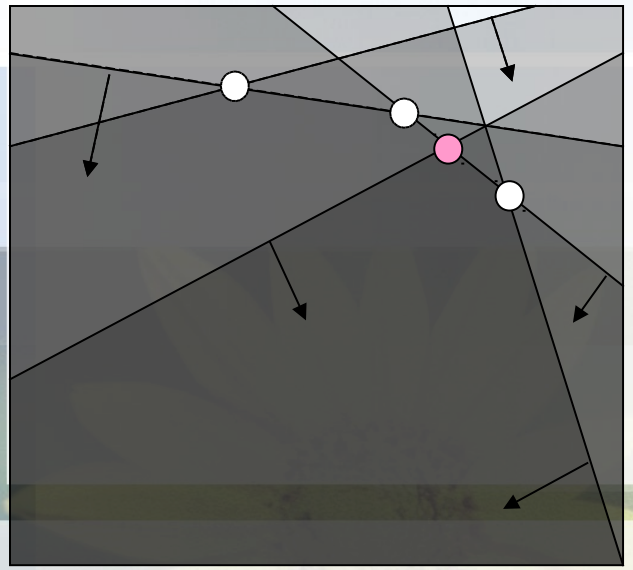
4. Sort-and-Incremental Algorithm

- **Step 3:**



4. Sort-and-Incremental Algorithm

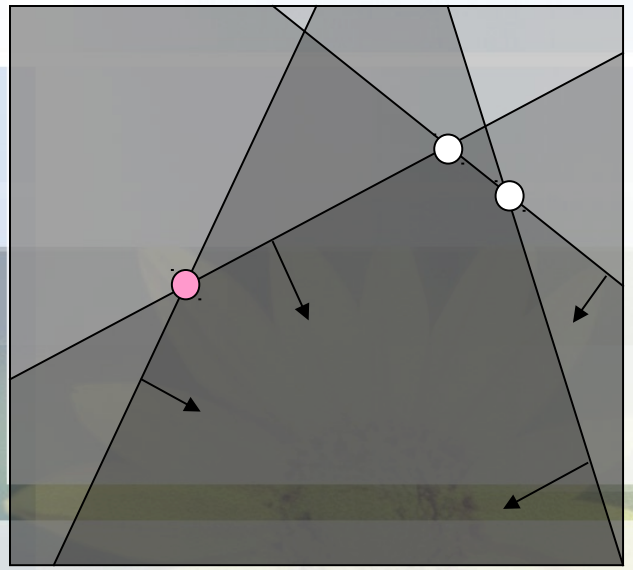
- **Step 3: …we pop the stack once. Once? Is it enough? Nie! Do this repeatedly until it is to the left of the top intersection.**



•前问我们说到出栈，出栈只需要一次么？ **Nie!** 我们要继续交点检查，如果还在右边我们要继续出栈，直到当前交点在栈顶交点的左边。

4. Sort-and-Incremental Algorithm

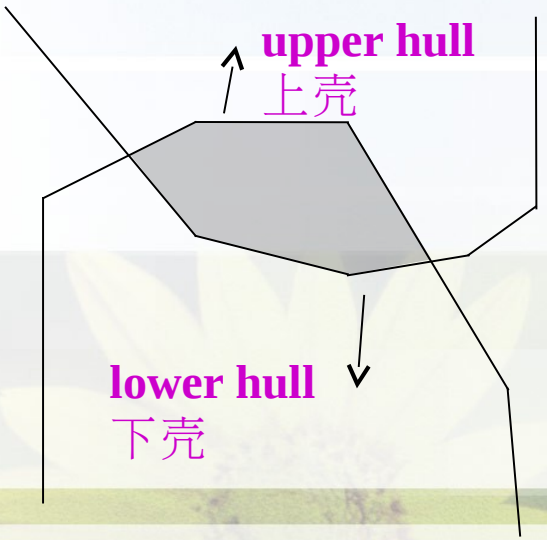
- **Step 4: Intersections of adjacent h-plane pairs in stack form half a convex polygon. For the two sets, we have two halves – $(-\frac{1}{2}\pi, \frac{1}{2}\pi]$ gives an upper hull and $(-\pi, -\frac{1}{2}\pi] \cup (\frac{1}{2}\pi, \pi]$ gives a lower hull.**



•相邻半平面的交点组成半个凸多边形。我们有两个点集，$(-\frac{1}{2}\pi, \frac{1}{2}\pi]$ 给出上半个，$(-\pi, -\frac{1}{2}\pi] \cup (\frac{1}{2}\pi, \pi]$ 给出下半个

4. Sort-and-Incremental Algorithm

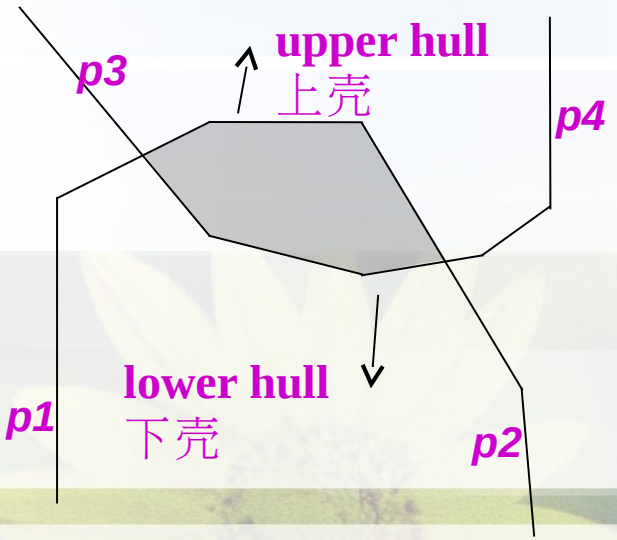
- **Step 4: Intersections of adjacent h-plane pairs in stack form half a convex polygon. For the two sets, we have two halves – $(-\frac{1}{2}\pi, \frac{1}{2}\pi]$ gives an upper hull and $(-\pi, -\frac{1}{2}\pi] \cup (\frac{1}{2}\pi, \pi]$ gives a lower hull.**



- 相邻半平面的交点组成半个凸多边形。我们有两个点集，$(-\frac{1}{2}\pi, \frac{1}{2}\pi]$ 给出上半个，$(-\pi, -\frac{1}{2}\pi] \cup (\frac{1}{2}\pi, \pi]$ 给出下半个

4. Sort-and-Incremental Algorithm

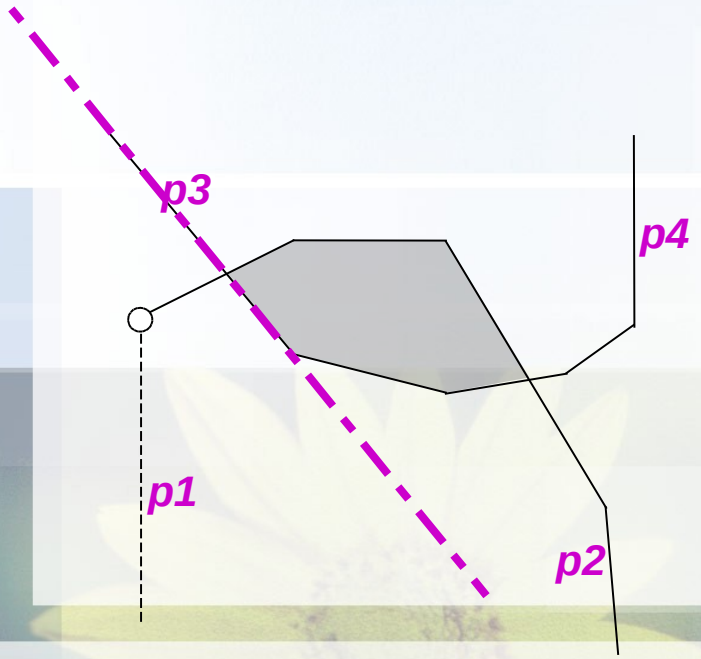
- **Step 4: At the beginning, four pointers *p1, p2, p3* and *p4* indicate leftmost/rightmost edges of both upper and lower hulls.**
- ***p1* and *p3* move rightwards, while *p2* and *p4* walks leftwards.**



- 初始时候，四个指针 ***p1, p2, p3* and *p4*** 指向上 *l* 下凸壳的最左最右边
- ***p1, p3*** 向右走， ***p2, p4*** 向左走

4. Sort-and-Incremental Algorithm

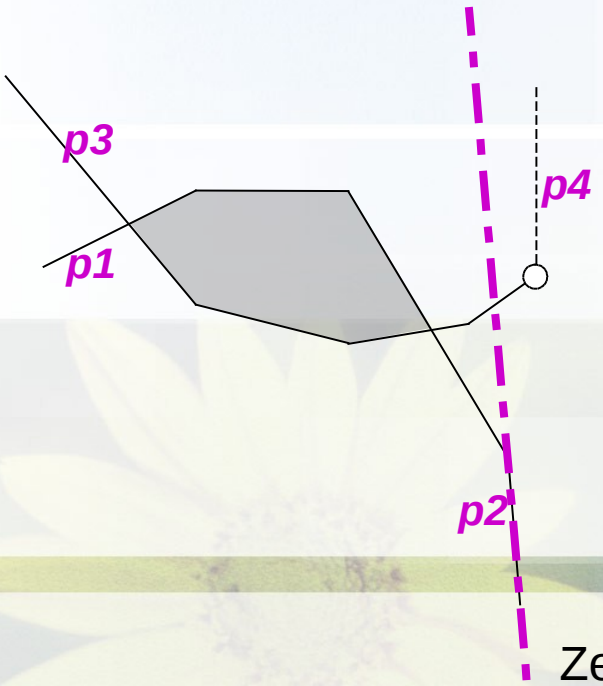
- **Step 4: At anytime, if the leftmost intersection is against the feasible region provided by *p1* or *p3*, we are sure the leftmost one is to be removed.**
- **Naturally, *p1* or *p3* walks rightwards to its adjacent edge.**



- 任意时刻，如果最左边的交点不满足 **p1/p3** 所在半平面的限制，我们相信这个交点需要删除
- **p1** 或 **p3** 走向它右边的相邻边

4. Sort-and-Incremental Algorithm

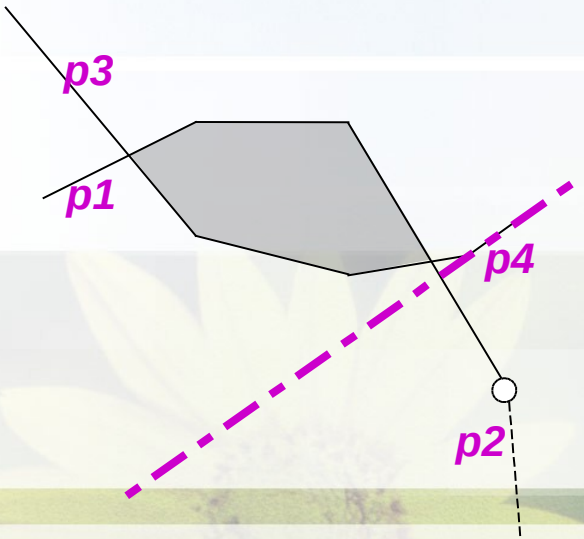
- **Step 4: The judgment holds analogously for rightmost intersection with *p2* and *p4*.**



- 类似地我们处理最右边的交点

4. Sort-and-Incremental Algorithm

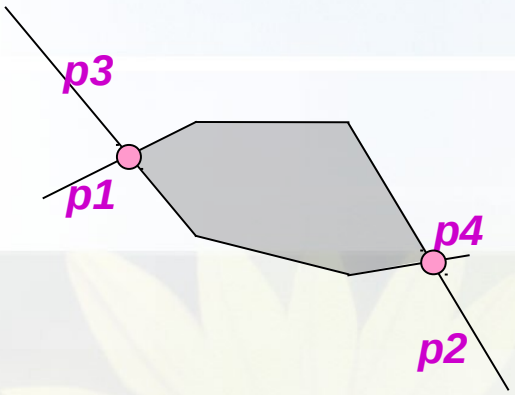
- **Step 4: The judgment holds analogously for rightmost intersection with *p2* and *p4*.**



•类似地我们处理最右边的交点

4. Sort-and-Incremental Algorithm

- **Step 4: Do the above removing repeatedly until there is no more update.**



•重复运作直到不再有更新出现——迭代

4. Sort-and-Incremental Algorithm

- **Everything except sorting (Step 2) in S&I algorithm remain linear – *O*(*n*) running time. Usually we use quick-sort. The total complexity is *O*(*n*log*n*), with fairly small constant factor hidden.**
- 除了 **Step2** 中的排序以外， **S&I** 算法的每一步都是线性的。通常我们用快速排序实现 **Step2**，总的时间复杂度为 ***O*(*n*log*n*)**，隐蔽其中的常数因子很小

5. Conclusion and Practical Use
总结和实际应用

5. Conclusion and Practical Use

- **Great ideas need landing gear as well as wings.** S&I algorithm seems to work in the same time complexity as D&C algorithm, but some overwhelming advantages of implementing S&I holds.
- **Great ideas need landing gear as well as wings. S&I** 算法似乎和 **D&C** 算法时间复杂度相同，但是它有着压倒性 **(overwhelming)** 的优势。

5. Conclusion and Practical Use

- It is much easier to code S&I program than D&C one. The program in C++ programming language takes less than 3KB.
- 新的 **S&I** 算法代码容易编写，相对于 **D&C** 大大简单化， **C++** 程序语言实现 **S&I** 算法仅需 **3KB** 不到 .

5. Conclusion and Practical Use

- **The coefficient hidden under S&I algorithm's complexity is extraordinarily smaller than D&C, since we no longer need *O*(*n*log*n*) number of intersection calculates. In general speaking, S&I program runs approx five times faster than D&C one.**
- **S&I** 算法复杂度中的系数，远小于 **D&C**，因为我们不再需要 ***O*(*n*log*n*)** 次交点运算．通常意义上来讲，**S&I** 程序比 **D&C** 快五倍。

5. Conclusion and Practical Use

- If the given h-planes are all in (-½π, ½π] (or any span of π), S&I program can be shorten remarkably (to approximately twenty lines in C++), but D&C program may not. An informatics problem appeared in USA Invitational Computing Olympiad contest with such purpose.
- 如果给定半平面均在 **(-½π, ½π]**（或任意一个跨度为 **π** 的区间），**S&I** 算法可被显著缩短，**C++** 程序只需要约二十行。**USAICO** 比赛中就出现了这样一题。

5. Conclusion and Practical Use

- **The bottleneck of this algorithm is sorting. Pay attention the sorting is NOT a *comparison sort* (sorting based on comparison)! Since then, we can replace the *O*(*n*log*n*) quick-sort to *O*(*n*) *radix-sort* to decrease the asymptotical time complexity to *O*(*n*).**
- 本算法瓶颈是排序，这里的排序不是比较排序，因此可以将快速排序替换成基数排序，降低程序渐进时间复杂度到线性。
- **Anyway *O*(*n*) approach usually runs slower than *n*log*n* ones for its additional memory usage!**

5. Conclusion and Practical Use

- < 美丽心灵 > 诺贝尔奖得主 **John Nash**
- **原创的理论——original idea**
- **The principal mark of genius is not perfection but originality, the opening of new frontiers.**
- **创新与技术**
  - **Authur Koestler (1905 – 1983)**
  - **Hungarian-born British writer and jounalist**
- 我心目中的创新——最重要的是思想创新，其次是行为创新，再其次是文章创新，再再其次才是语言创新
  - 思想
  - 实践

5. Conclusion and Practical Use

- 创新如高山
- 山，
  快马加鞭未下鞍。
  惊回首，离天三尺三。
- 山，
  倒海翻江卷巨澜。
  奔腾急，万马战犹酣。
- 山，
  刺破青天锷未残。
  天欲堕，赖以拄其间。

悲寥廓，问苍茫大地，谁主沉浮。

俱往矣，数风流人物，还看今朝。