浅谈用极大化思想解决最大子矩形问题

福州第三中学　王知昆

问题：奶牛浴场

- 题意简述：

  John 要在牛场中建造一个大型浴场，但是这个大型浴场不能覆盖任何一个奶牛的产奶点。 John 的牛场和规划的浴场都是矩形，浴场要完全位于牛场之内，并且浴场的轮廓要与牛场的轮廓平行或者重合。要求所求浴场的面积尽可能大。

- 参数约定：产奶点的个数 S 不超过 5000，牛场的范围 N×M 不超过 30000×30000 。

问题的模型

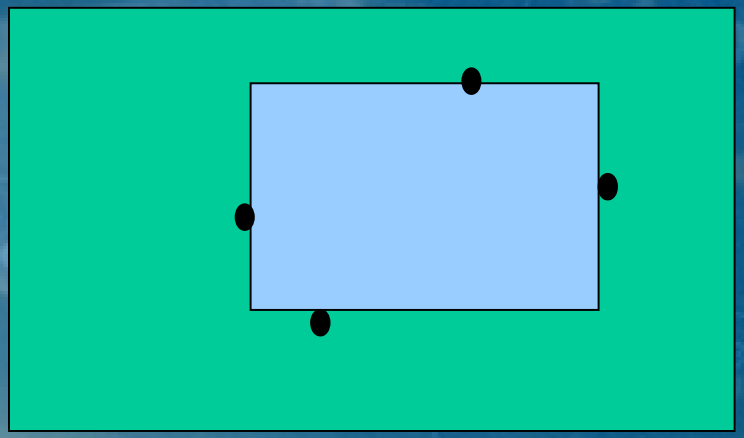
- 最大子矩形问题：

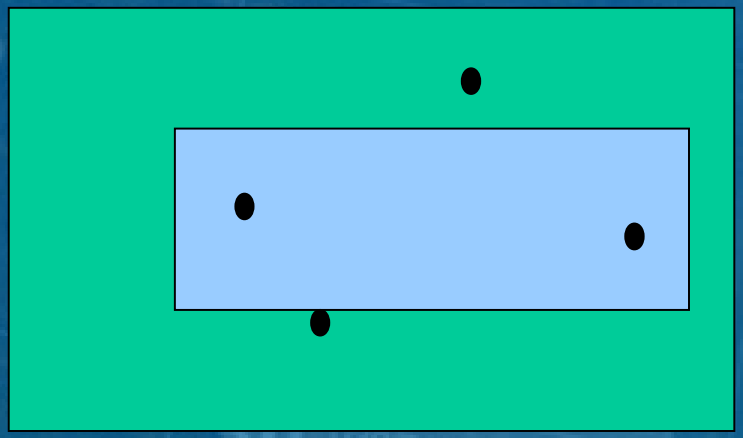
  在一个给定的矩形中有一些障碍点，要找出内部不包含任何障碍点的，轮廓与整个矩形平行或重合的最大子矩形。

定义和说明

- 定义有效子矩形为内部不包含任何障碍点的，边界与坐标轴平行的子矩形。

- 如下图所示，第一个是有效子矩形，第二个不是。





定义和说明

- 定义极大子矩形为每条边都不能向外扩展的有效子矩形。
- 定义最大子矩形为所有有效子矩形中最大的一个（或多个）。

极大化思想

- 在一个有障碍点的矩形中的最大子矩形一定是一个极大子矩形。

- 设计算法的思路：通过枚举所有的极大子矩形，找出最大子矩形。

两个不同的算法

- 针对问题的性质，可以设计出两个不同的算法。他们分别适用于不同的情况。
- 约定：为了叙述方便，设整个矩形的大小为 N×M，其中障碍点个数为 S 。

- 算法 1
- 时间复杂度： $O(S^2)$
- 空间复杂度： $O(S)$

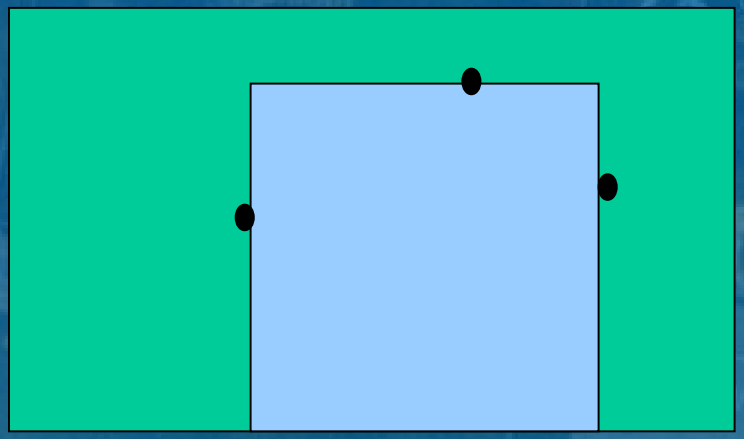
- 算法 2
- 时间复杂度： $O(NM)$
- 空间复杂度： $O(S)$

算法 1　思路

- 从极大子矩形的性质入手。

- 极大子矩形的性质：

  一个极大子矩形的每条边一定都不能向外扩展。更进一步地说，一个有效子矩形是极大子矩形的条件是这个子矩形的每条边要么覆盖了障碍点，要么与整个矩形的边界重合。



算法设计

基本算法

- 算法：枚举上下左右四个边界，然后判断组成的矩形是否是有效子矩形。
- 复杂度： $O(S^5)$
- 可以改进的地方：

  产生了大量的无效子矩形

初步改进算法

- 算法：枚举左右边界，然后对处在边界内的点排序。每两个相邻的点和左右边界一起组成一个矩形。
- 复杂度： $O(S^3)$
- 可以改进的地方：

  枚举了部分不是极大子矩形的情况

算法改进

- 设计算法的方向：

  1、保证每一个枚举的子矩形都是有效的

  2、保证每一个枚举的子矩形都是极大的

- 算法的过程：

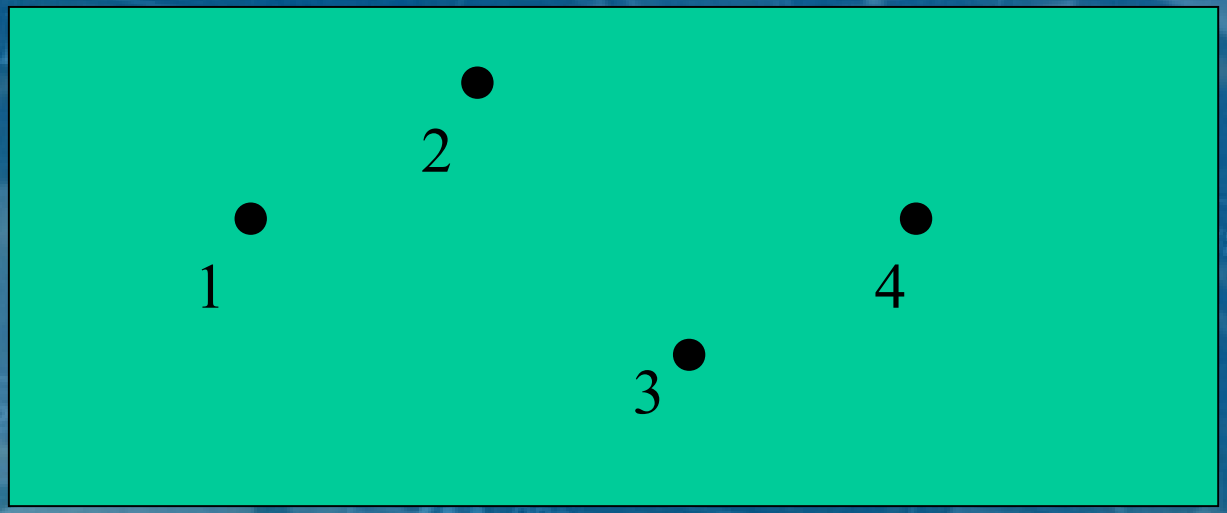
  枚举极大子矩形的左边界

  → 根据确定的左边界，找出相关的极大子矩形

  → 检查和处理遗漏的情况

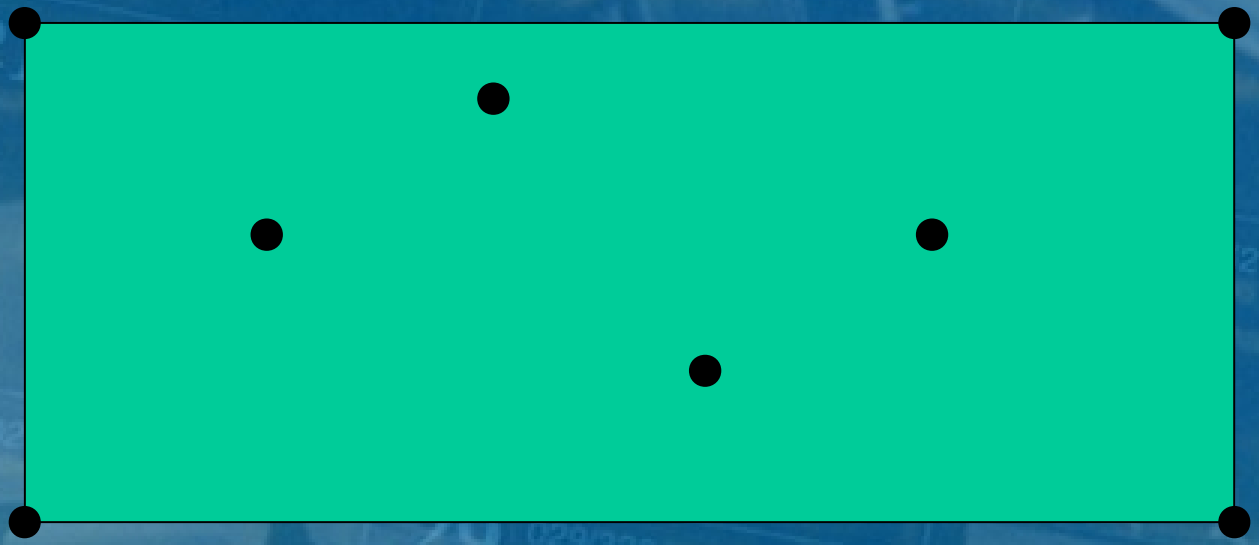
算法 1

- 首先，将所有障碍点按横坐标从小到大的顺序将点标为 1 号点，2 号点……



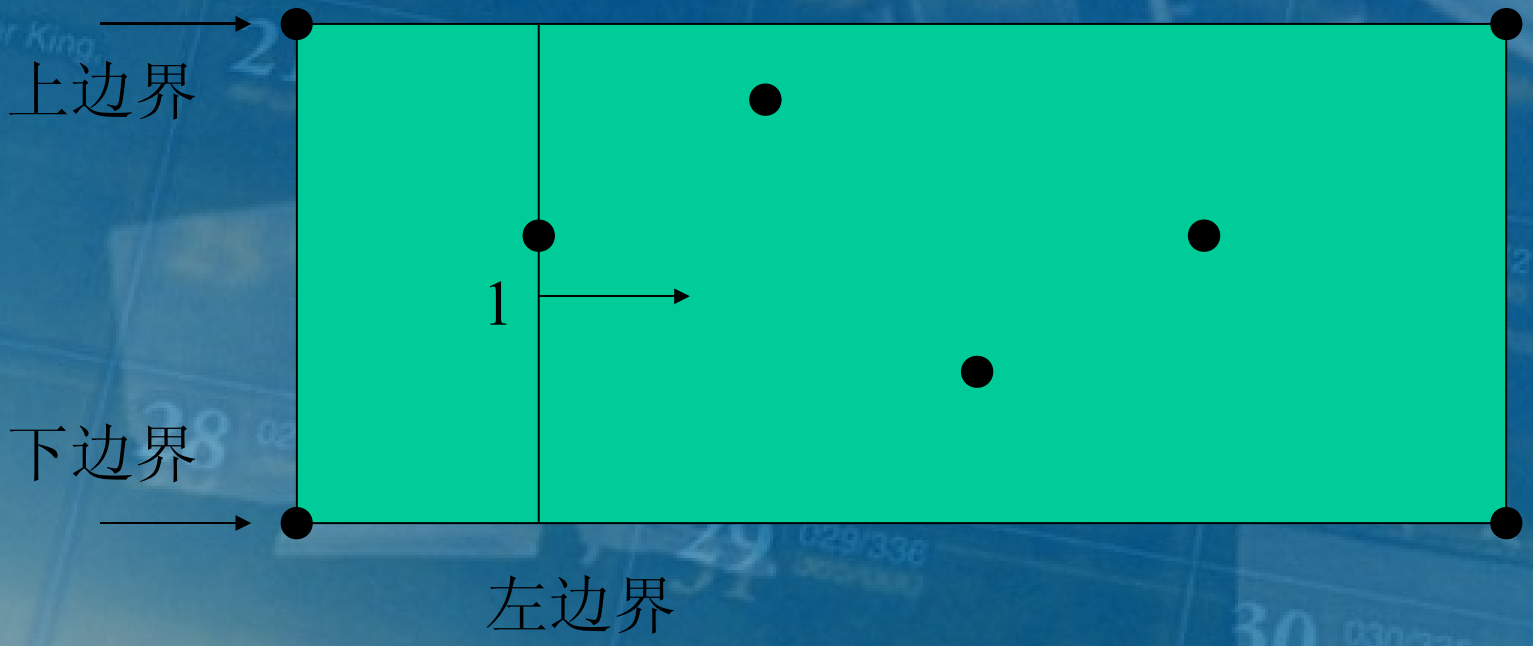
算法 1

- 为了处理方便，在矩形的四个顶点上各增加 1 个障碍点。



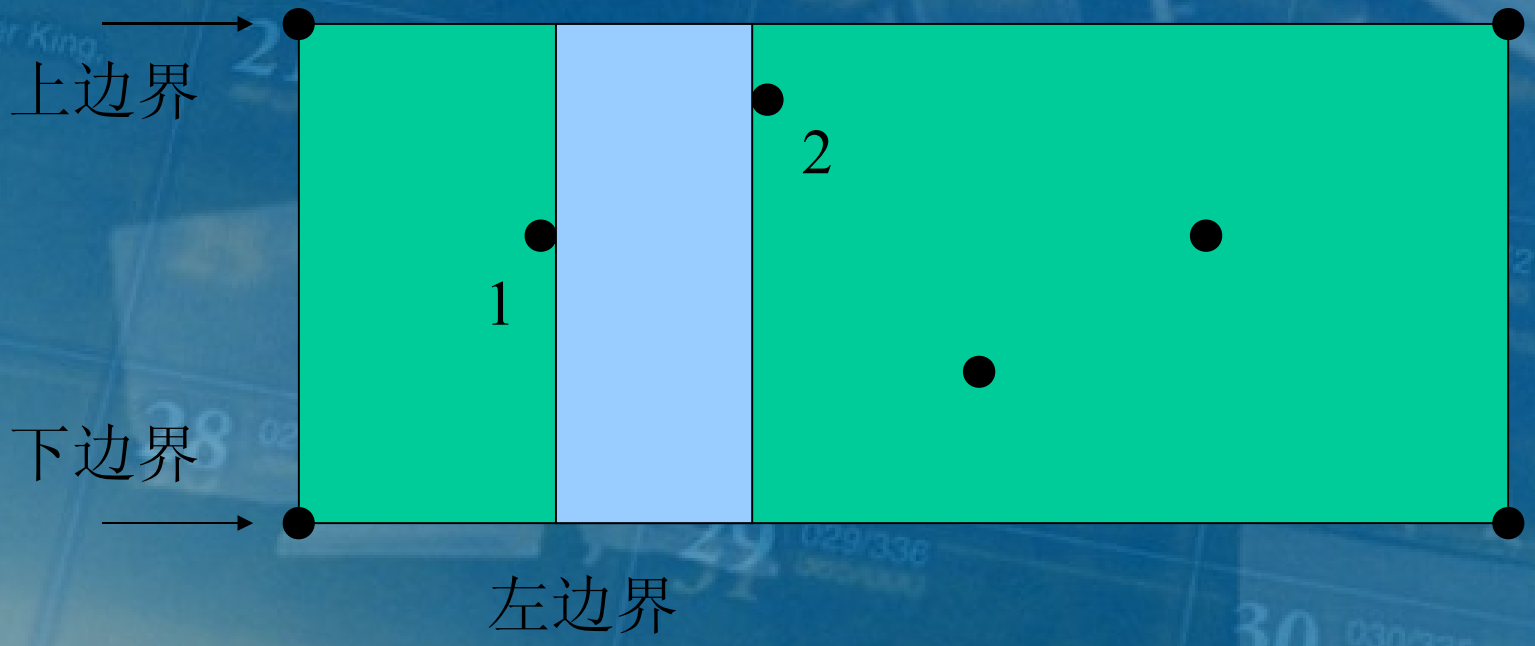
算法 1

- 第一次取 1 号点作为所要枚举的极大子矩形的左边界
- 设定上下边界为矩形的上下边界



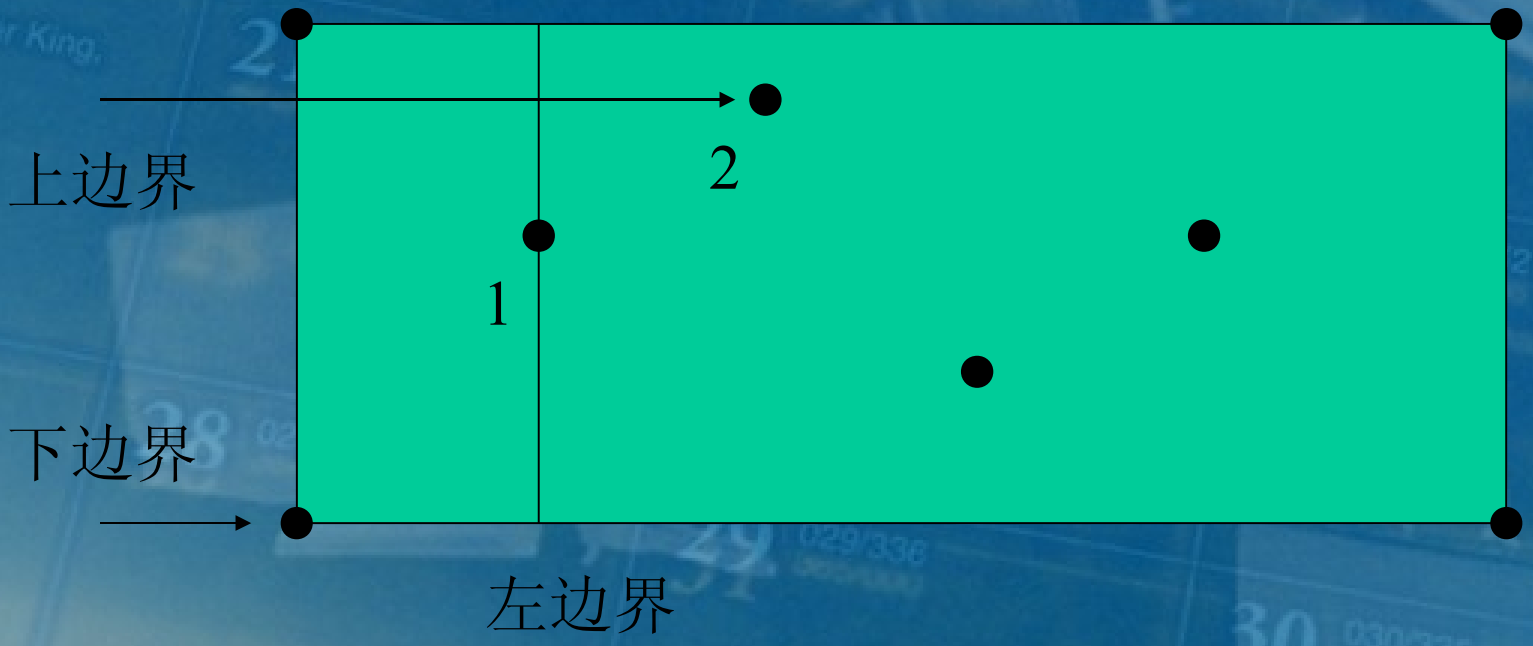
算法 1

- 从左向右扫描，第一次遇到 2 号点，可以得到一个有效的极大子矩形，如图所示



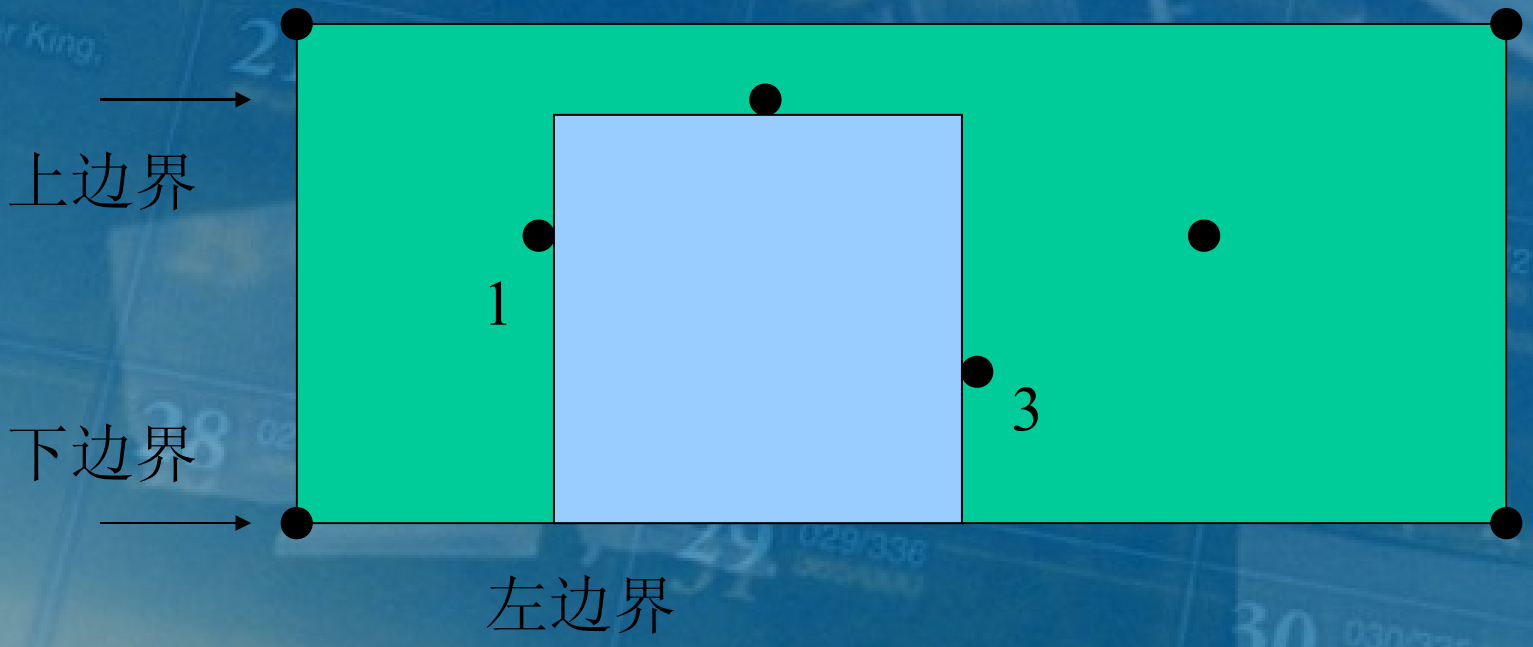
算法 1

- 因为左边界覆盖 1 号点且右边界在 2 号点右边的有效子矩形都不能包含 2 号点，所以需要修改上下边界
- 2 号点在 1 号点上方，因此要修改上边界



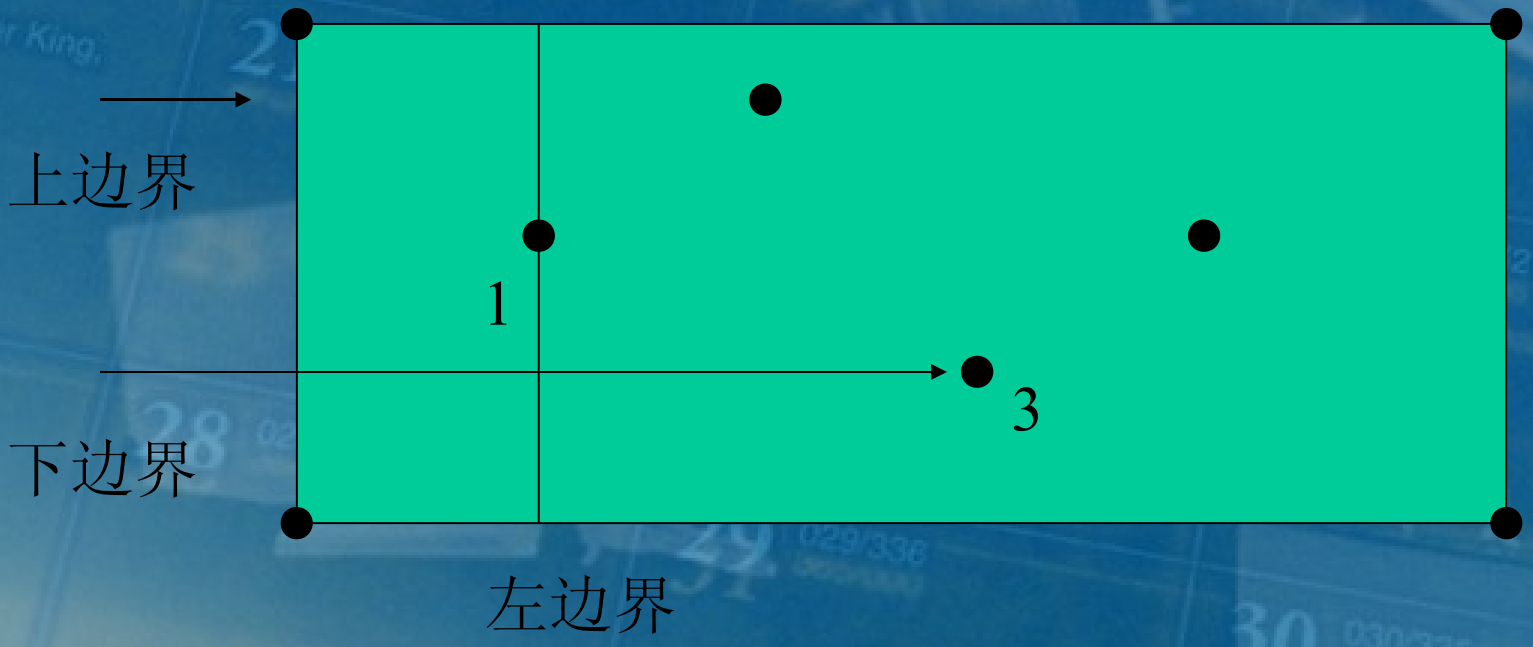
算法 1

- 继续扫描到 3 号点，又得到一个极大有效子矩形，如图所示



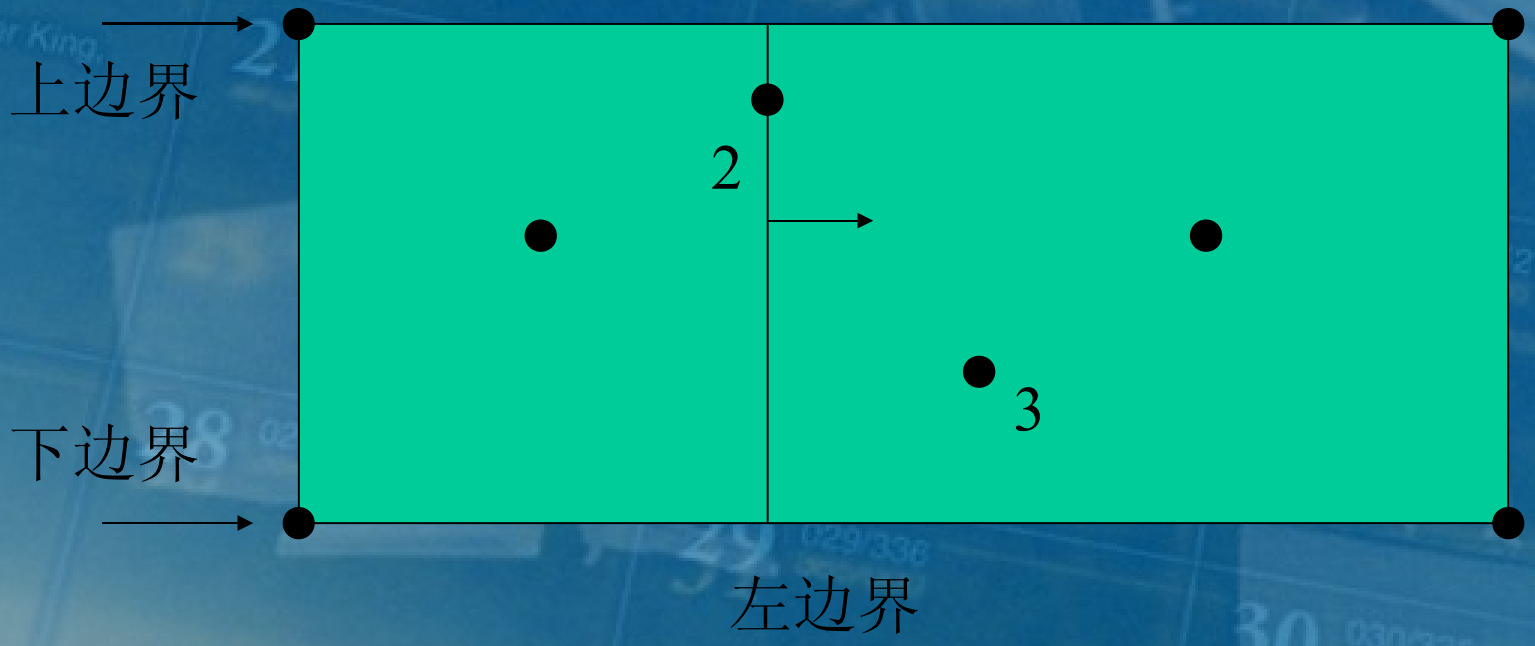
算法 1

- 因为 3 号点在 1 号点下方，所以要修改下边界。



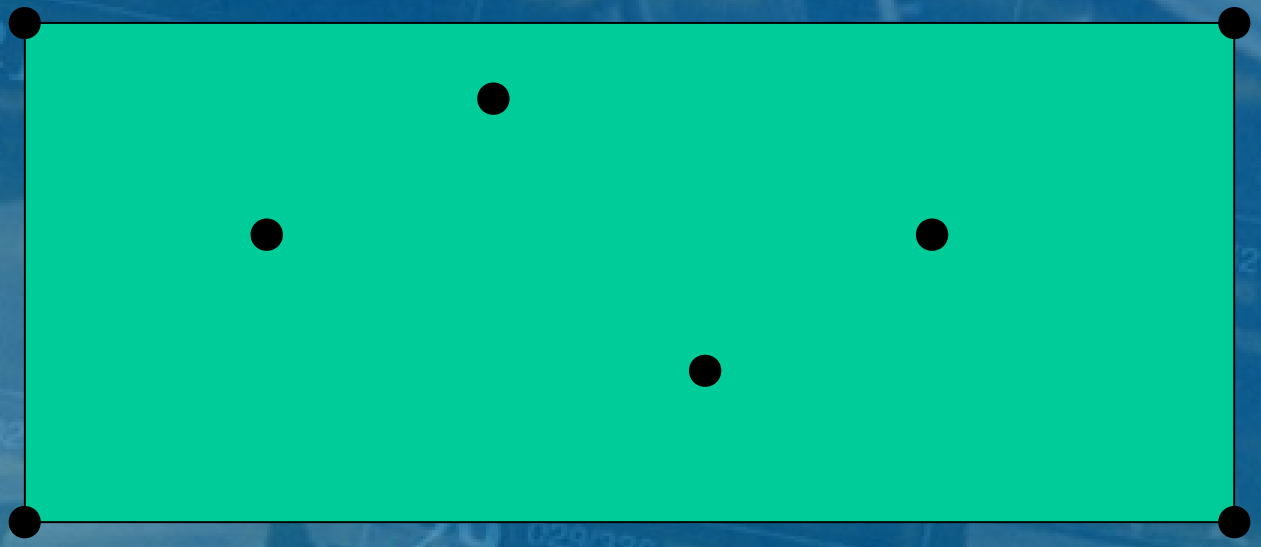
算法 1

- 以此类推，可以得到所有以 1 号点为左边界的极大有效子矩形。
- 然后将左边界移动到 2 号点、3 号点……横坐标的位置。开始扫描以 2 号点、3 号点……为左边界的极大子矩形。



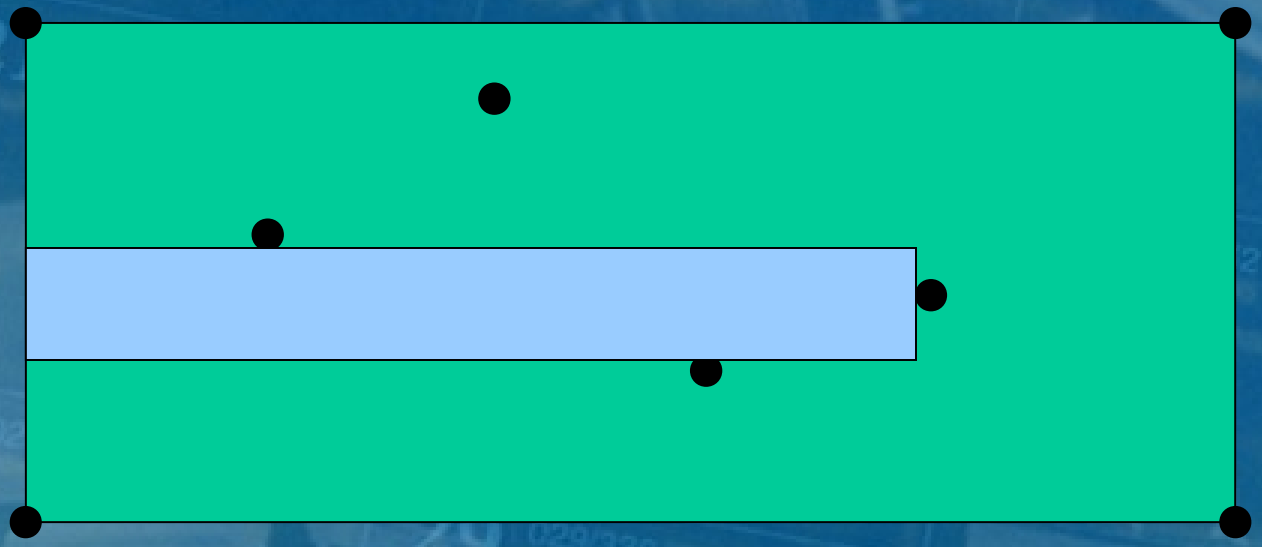
算法1　遗漏的情况

- 前面的做法可以找出所有左边界覆盖了一个障碍点的极大子矩形，此外，还有两类遗漏的情况。



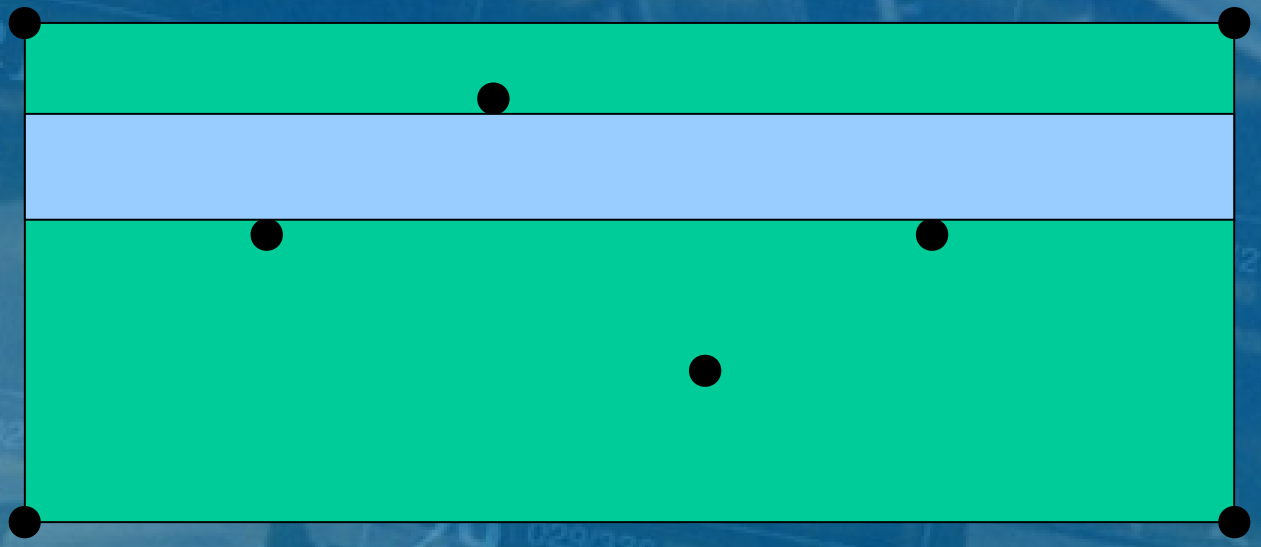
算法1　遗漏的情况

- 一类是左边界与整个矩形的左边界重合，右边界覆盖一个障碍点的情况。
- 解决方法：用类似的方法从右向左扫描一次。



算法1　遗漏的情况

- 另一类是左边界与整个矩形的左边界重合，且右边界也与整个矩形的右边界重合的情况。
- 解决方法：预处理时增加特殊判断。



算法 1　优劣分析

- 算法 1 的时间复杂度为 $O(S^2)$，空间复杂度为 $O(S)$。
- 优点：利用了极大化思想，复杂度可以接受，编程实现简单。
- 缺点：使用有一定的局限性，不适合障碍点较密集的情况。

算法2 设计的目的和思路

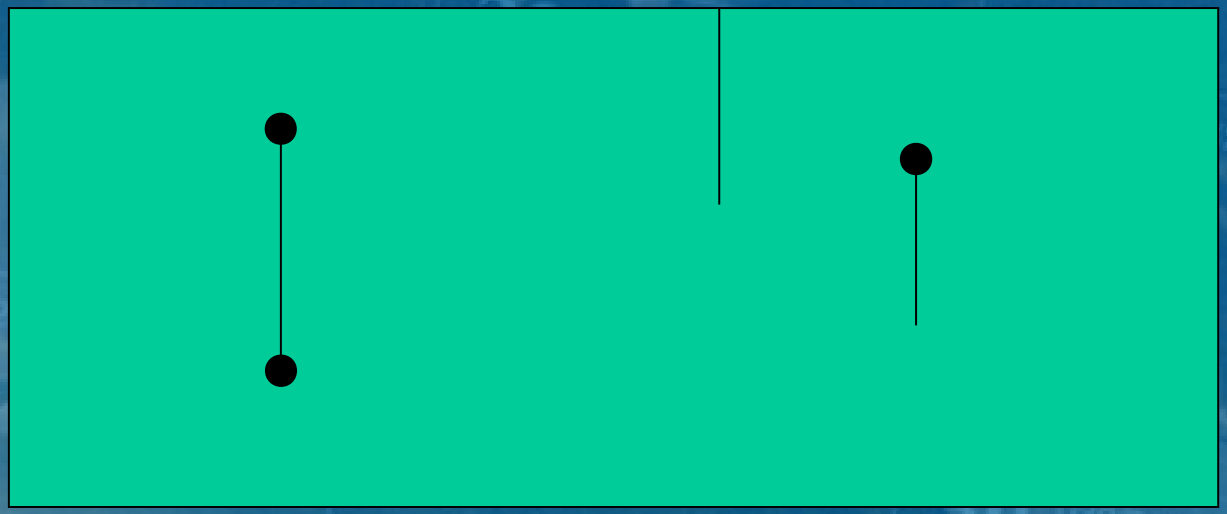
- 因为算法1有使用的局限性，所以我们需要一种在障碍点很密集的时候仍能奏效的算法。

- 设计一种复杂度依赖于整个矩形面积的算法

  说明：如果整个矩形面积很大，可以通过离散化处理来优化。

算法 2　悬线

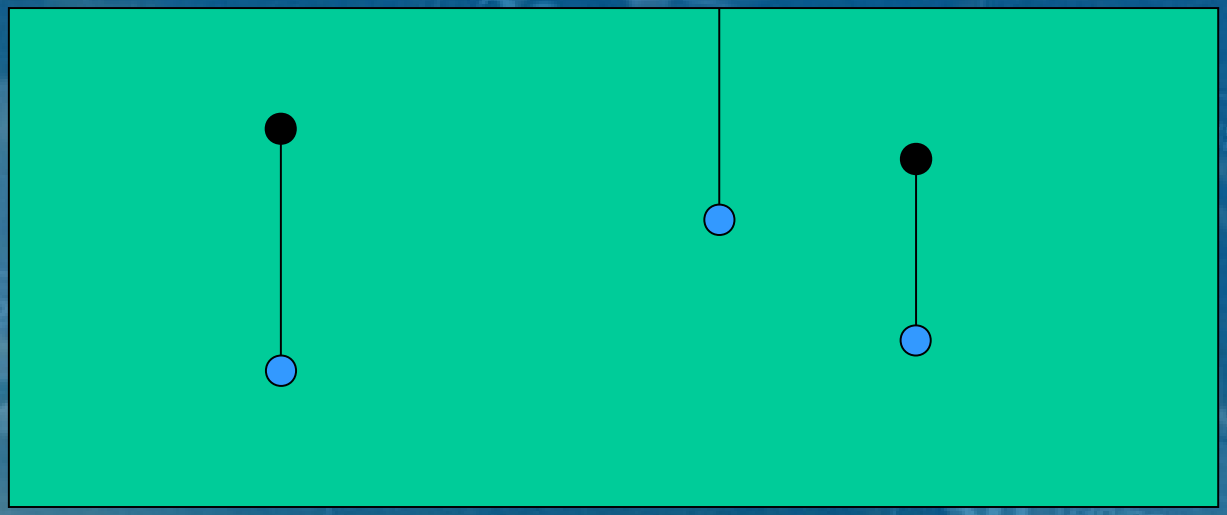
- 有效竖线：除了两个端点外，不覆盖任何障碍点的竖直线段。
- 悬线：上端点覆盖了一个障碍点或达到整个矩形上端的有效竖线。
- 图中所示的线段均为悬线。



算法2 悬线

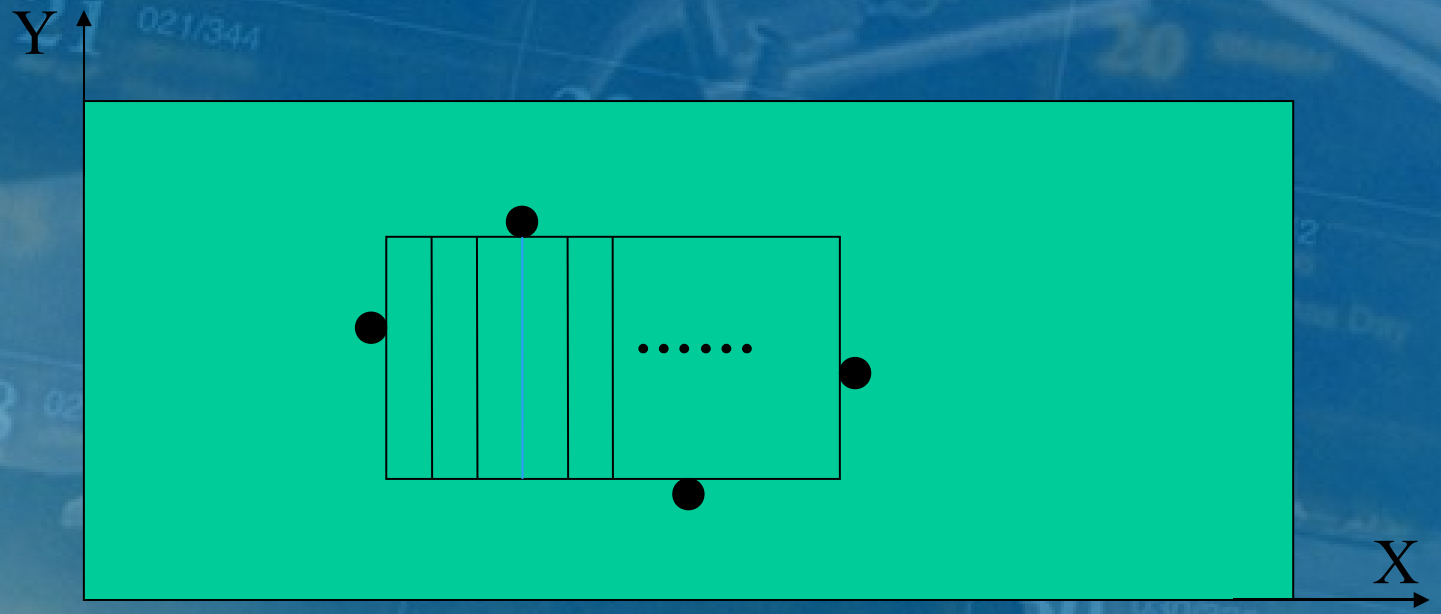
- 每个悬线都与它底部的点一一对应。
  矩形中的每一个点（矩形顶部的点除外）都对应了一个悬线。

- 悬线的个数＝（N－1）×M



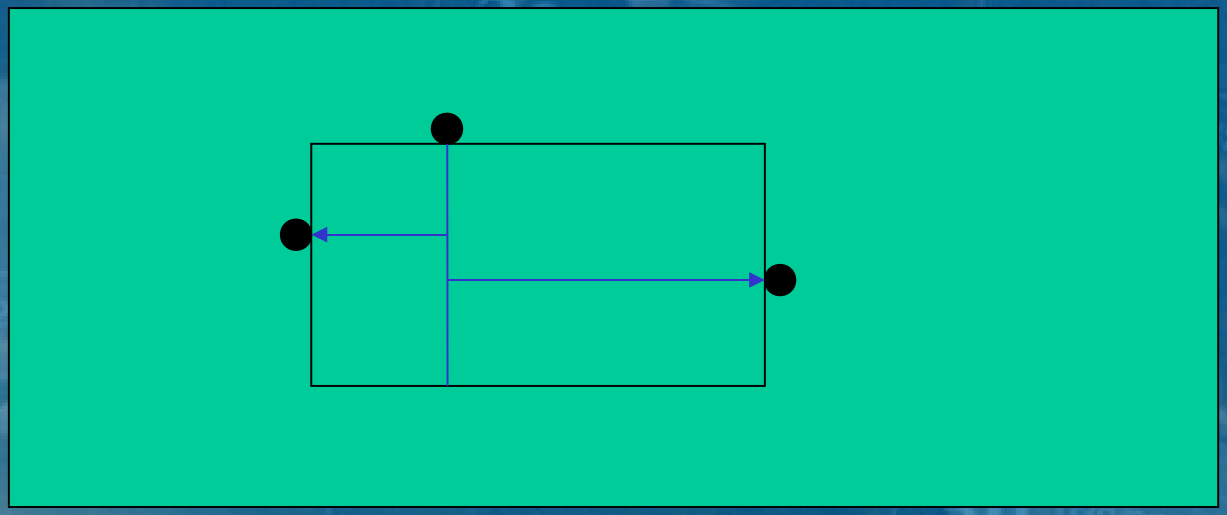
算法 2　悬线与极大子矩形

- 如果把一个极大子矩形按 x 坐标不同切割成多个与 y 轴平行的线段，则其中至少存在一个悬线。



算法 2　悬线与极大子矩形

- 如果把一个悬线向左右两个方向尽可能移动，就能得到一个矩形，不妨称为这个悬线对应的矩形。
- 悬线对应的矩形不一定是极大子矩形，因为下边界可能还可以向下扩展。



设计算法

- 原理：所有悬线对应矩形的集合一定包含了极大子矩形的集合。
- 通过枚举所有的悬线，找出所有的极大子矩形。
- 算法规模：

  悬线个数＝(N － 1)×M

  极大子矩形个数≤悬线个数

算法2 关键点

- 解决问题的关键：

  对每个悬线的处理时间。

- 解决方法：

  充分利用前面得到的信息。

算法2 处理方法

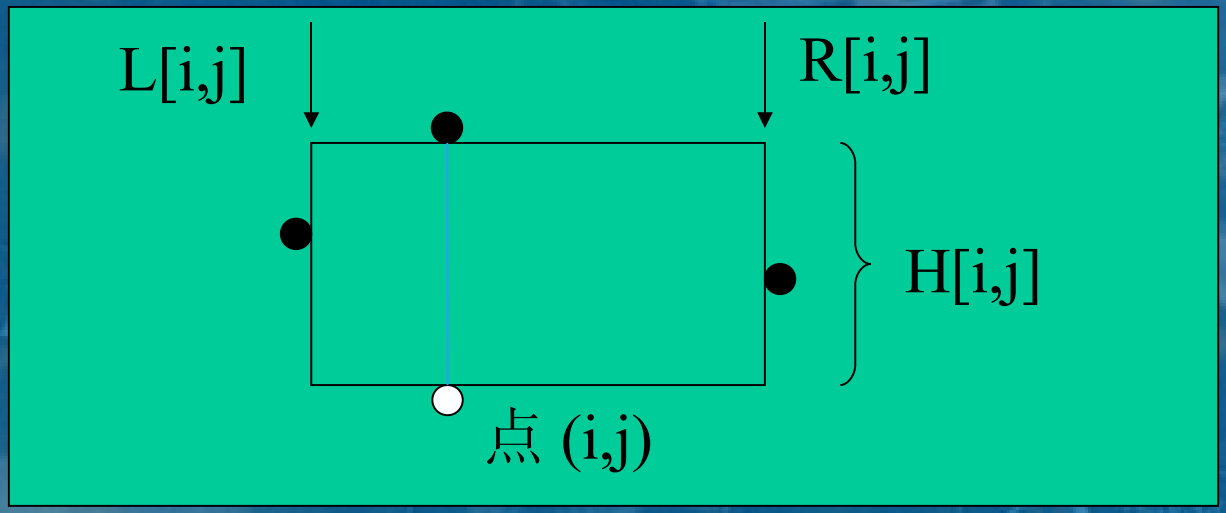
- 具体方法：

设 H[i,j]为点(i,j)对应的悬线的长度。

L[i,j]为点(i,j)对应的悬线向左最多能够移动到的位置。

R[i,j]为点(i,j)对应的悬线向右最多能够移动到的位置。

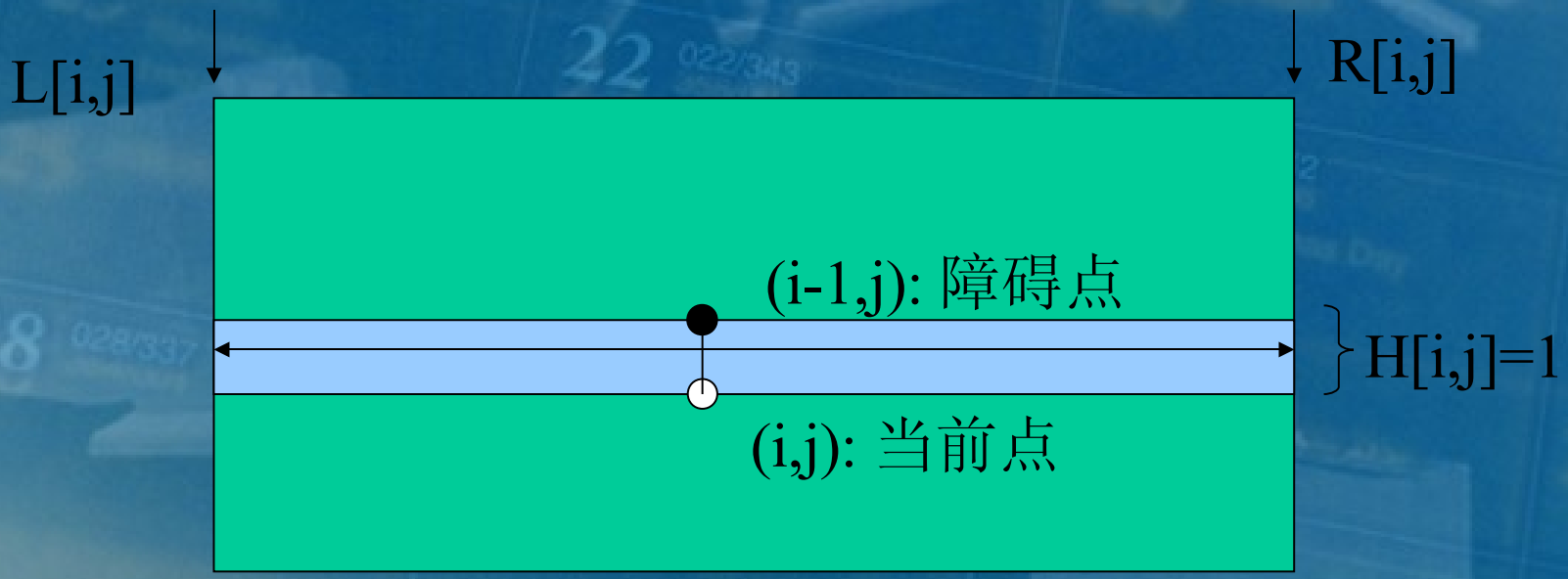
图示



- 考虑点 (i,j) 对应的悬线与点 (i-1,j) 对应的悬线的关系。

算法 2　递推关系

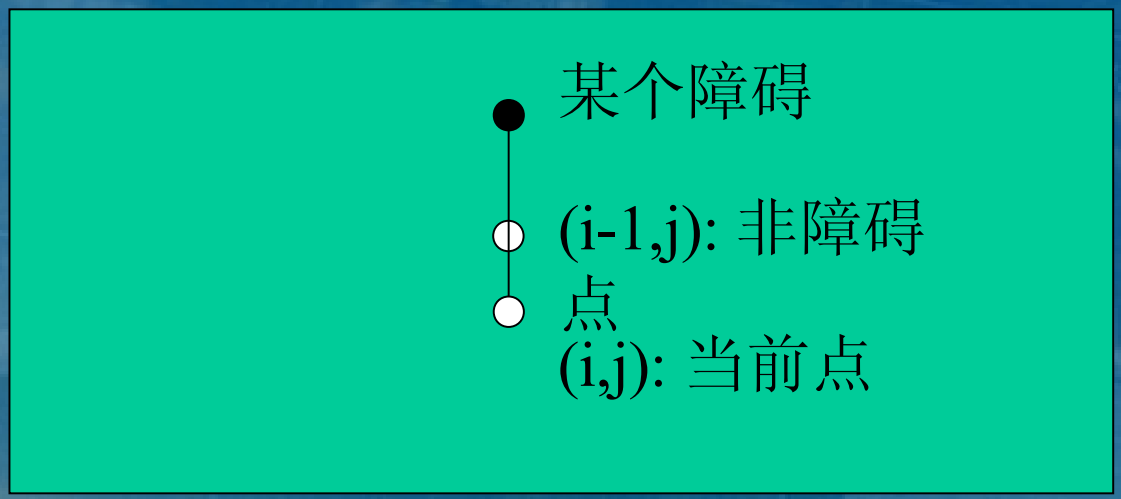
- 如果 (i － 1,j) 为障碍点，那么，如图所示，(i,j) 对应的悬线长度 1，左右能移动到的位置是整个矩形的左右边界。
- 即　H[i,j] ＝ 1,
  L[i,j]=0,R[i,j]=m



算法 2　递推关系

- 如果 (i － 1,j) 不是障碍点，那么，如图所示，(i,j) 对应的悬线长度为 (i-1,j) 对应的悬线长度＋ 1 。

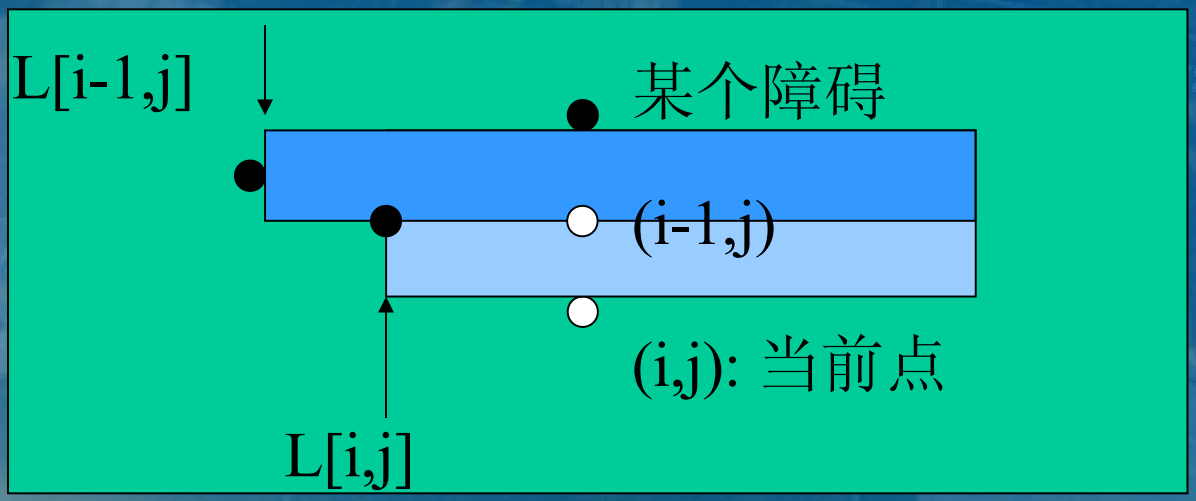
- 即　H[i,j] ＝ H[i-1,j]+1



算法 2　递推关系

- 如果 (i － 1,j) 不是障碍点，那么，如图所示，(i,j) 对应的悬线左右能移动的位置要在 (i-1,j) 的基础上变化。

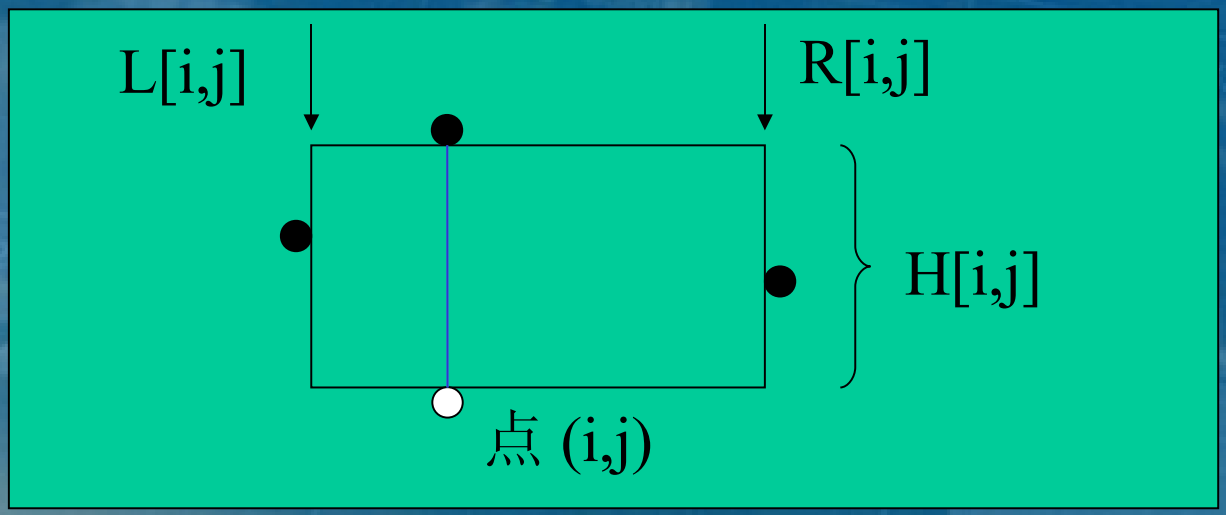
$$L[i,j]=\max\begin{cases}L[i-1,j]\\ (i-1,j)\text{ 左边第一个障碍点的位置}\end{cases}$$



算法 2　递推关系

- 同理，也可以得到 R[i,j] 的递推式

$$R[i,j]=\min\begin{cases} R[i-1,j] \\ (i-1,j)\text{ 右边第一个障碍点的位置} \end{cases}$$



算法 2 优劣分析

- 算法 2 的时间复杂度为 O(NM)，空间复杂度为 O(S)。
- 优点： 复杂度与障碍点个数没有直接关系。
- 缺点：障碍点少时处理较复杂，不如算法 1

两个不同的算法

- 算法 1
- 时间复杂度： $O(S^2)$
- 空间复杂度： $O(S)$

- 优点：复杂度可以接受，编程实现简单

- 缺点：使用有一定的局限性，不适合障碍点较密集的情况。

- 算法 2
- 时间复杂度： $O(NM)$
- 空间复杂度： $O(S)$

- 优点：复杂度与障碍点个数没有直接关系。

- 缺点：障碍点少时因为要离散化处理，实际复杂度较高。

推广 1　最大权值子矩形问题

- 最大权值子矩形问题

模型：在一个带权（正权）矩形中有一些障碍点，找出一个不包含障碍点的最大权值子矩形。

- 分析：在一个正权值的矩形中的最大权值子矩形一定是极大子矩形。所以，问题实际上可以依据极大化的思想，利用前面的方法解决。

推广 2　最大子正方形问题

- 最大子正方形问题

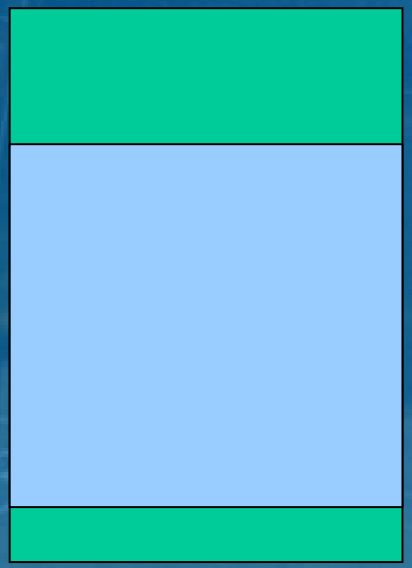
模型：在一个矩形中存在 S 个障碍点，要 求找出最大的不包含障碍点的正方形。

- 分析：在一个有障碍点的矩形中的最大有效子正方形一定是一个极大有效子正方形。

推广 2　最大子正方形问题

- 极大子正方形的性质：

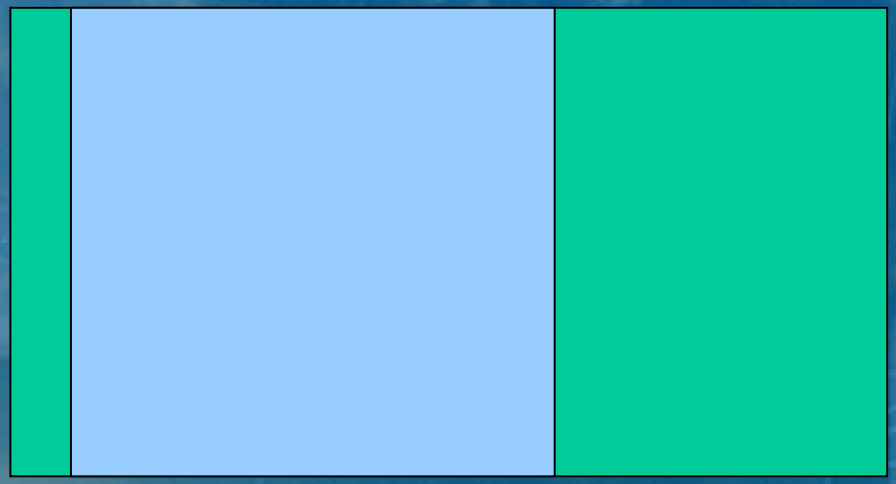
  每一个极大子正方形都至少被一个极大子矩形包含，且这个极大子正方形一定有两条不相邻的边与包含它的极大子矩形的边重合。



推广 2　最大子正方形问题

- 解决方法：通过枚举每一个极大子矩形找出所有的极大子正方形。

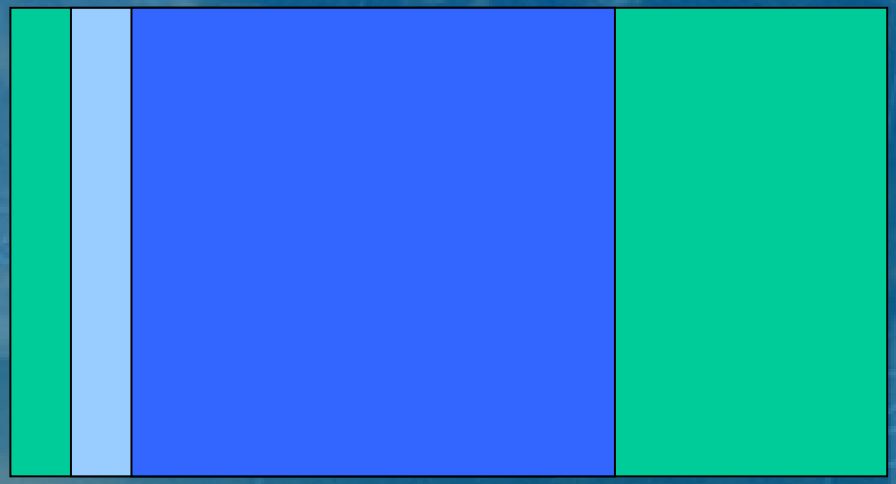
- 每个极大子矩形对应的极大子正方形可能有多个，但大小都一样。



推广 2　最大子正方形问题

- 解决方法：通过枚举每一个极大子矩形找出所有的极大子正方形。

- 每个极大子矩形对应的极大子正方形可能有多个，但大小都一样。



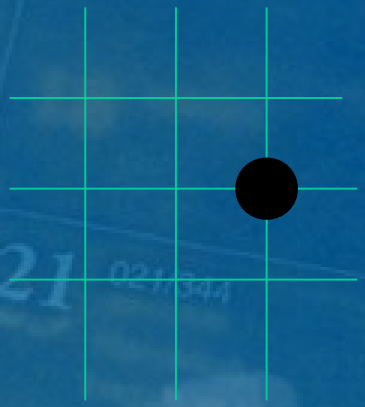
推广 2　最大子正方形问题

- 解决方法：通过枚举每一个极大子矩形找出所有的极大子正方形。

- 每个极大子矩形对应的极大子正方形可能有多个，但大小都一样。

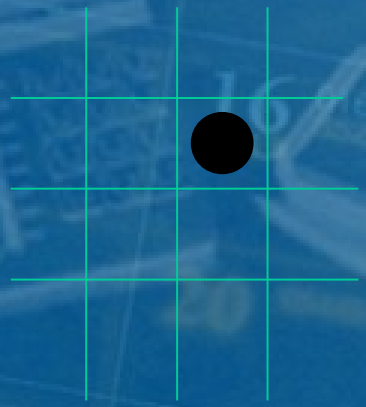
矩形类型变换

- 类型 1



- 矩形中的点都是两条垂直线段的交点，有效子矩形可以在边界包含障碍点。

- 类型 2



- 矩形中的点是单位方格，有效子矩形不能包含任何障碍点。
- 处理方法与类型 1 基本相同

要点回顾