Computation of the MLE for bivariate interval censored data

Marloes H. Maathuis

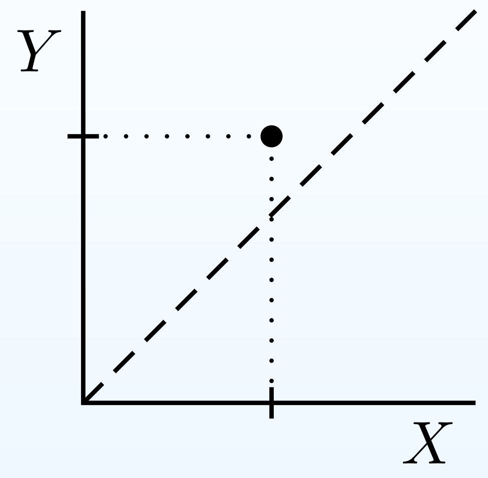
ETH Zürich, Seminar für Statistik

maathuis@stat.math.ethz.ch

http://stat.ethz.ch/∼maathuis

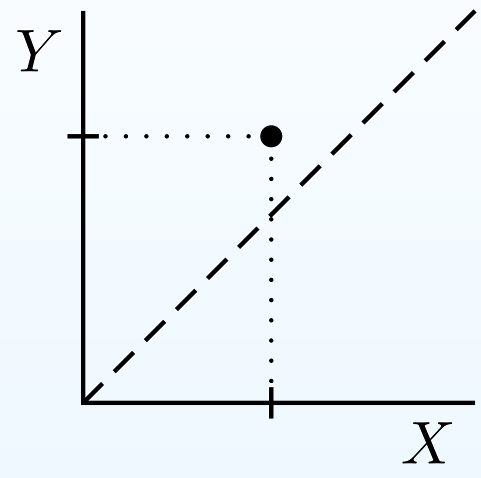
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS



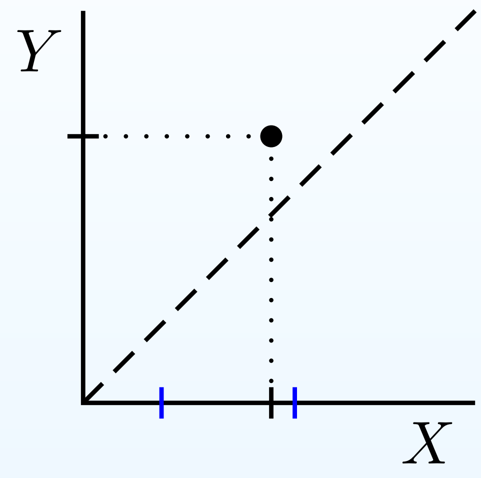
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS
- X and Y can be interval censored.



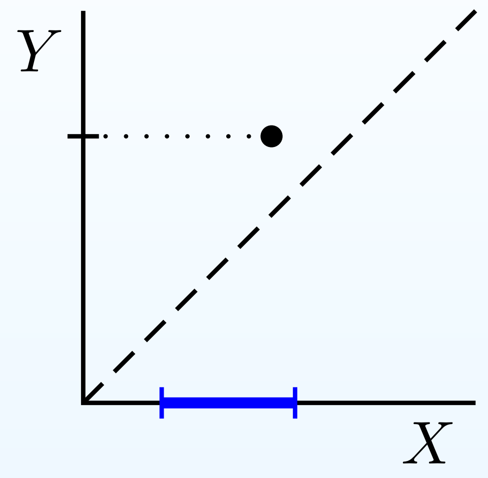
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS
- X and Y can be interval censored.



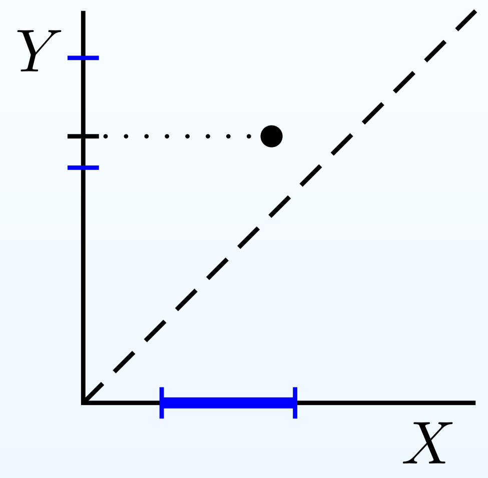
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS
- X and Y can be interval censored.



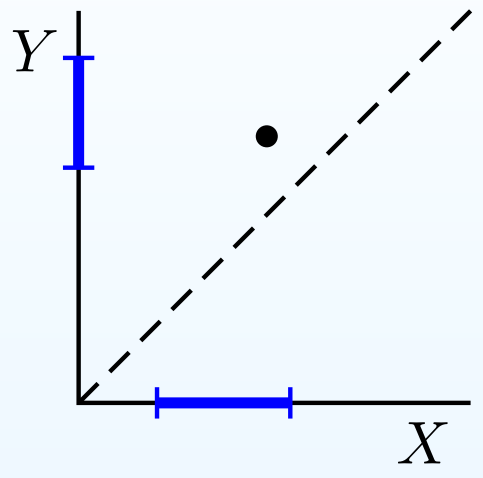
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS
- X and Y can be interval censored.



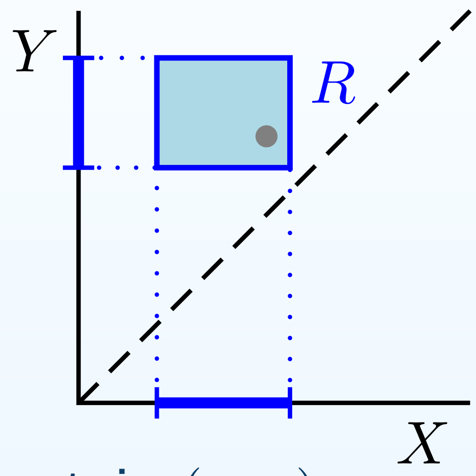
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS
- X and Y can be interval censored.



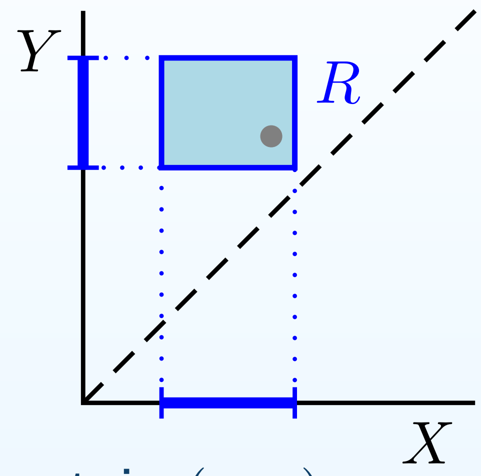
Bivariate interval censored data: an example

- We want to estimate the joint distribution function of (X, Y), where:
    - X: time of HIV infection
    - Y: time of onset of AIDS
- X and Y can be interval censored. Instead of a realization (x, y), we observe an *observation rectangle* R that is known to contain (x, y).



Bivariate interval censored data: an example

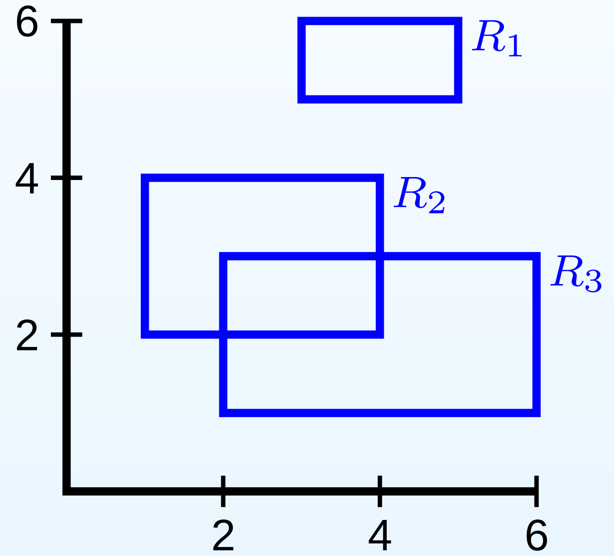
- We want to estimate the joint distribution function of (X, Y), where:
  - X: time of HIV infection
  - Y: time of onset of AIDS



- X and Y can be interval censored. Instead of a realization (x, y), we observe an *observation rectangle* R that is known to contain (x, y).
- Goal: based on n i.i.d observation rectangles $R_1, \ldots, R_n$ we want to compute the MLE for the joint distribution function of (X, Y).

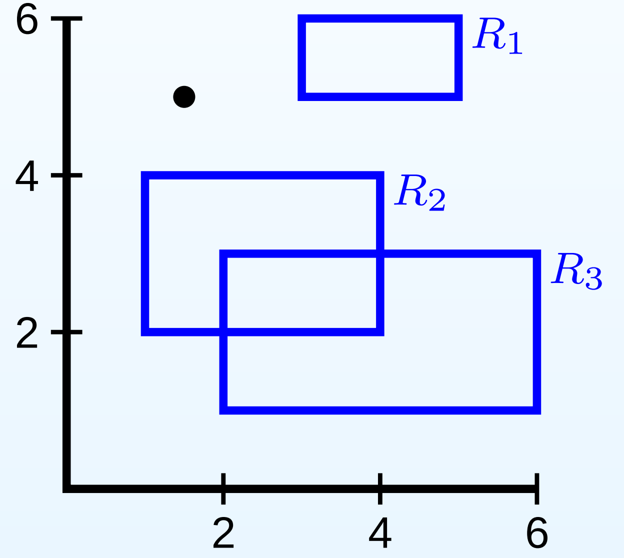
The nonparametric maximum likelihood estimator

$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$



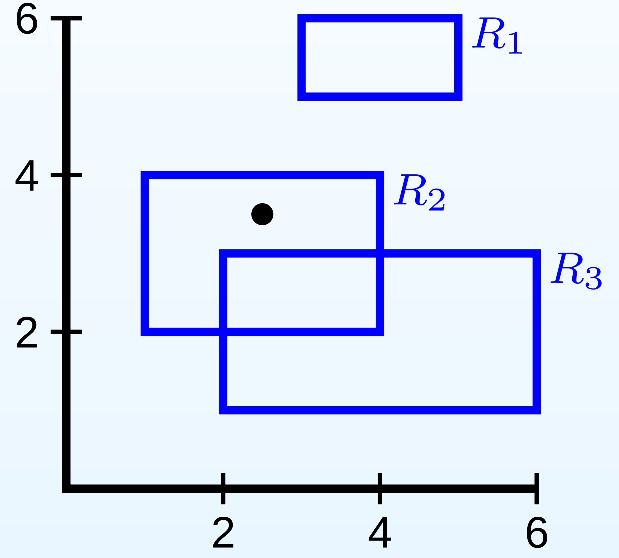
The nonparametric maximum likelihood estimator

$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$



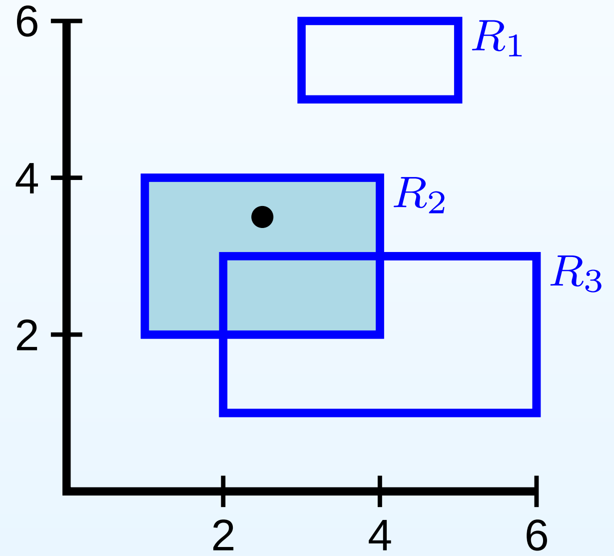
The nonparametric maximum likelihood estimator

$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$



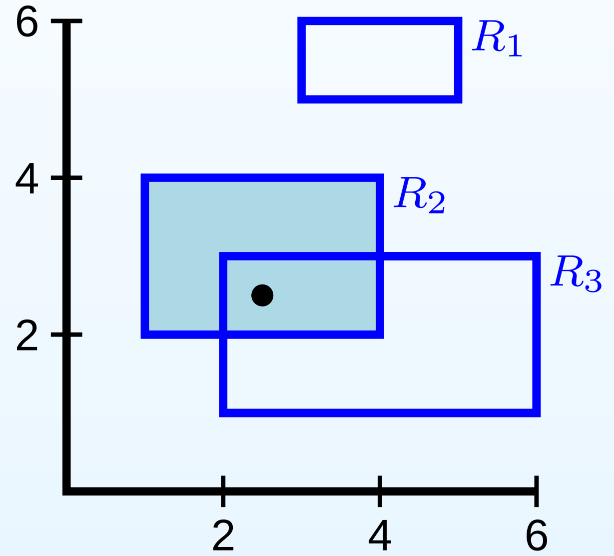
The nonparametric maximum likelihood estimator

$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$



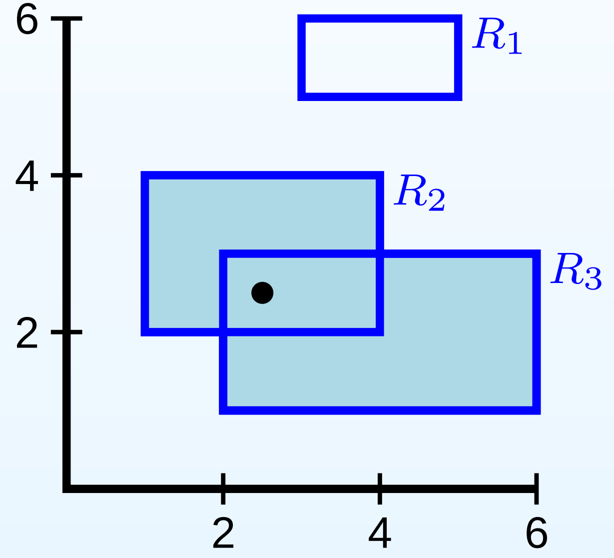
The nonparametric maximum likelihood estimator

$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$



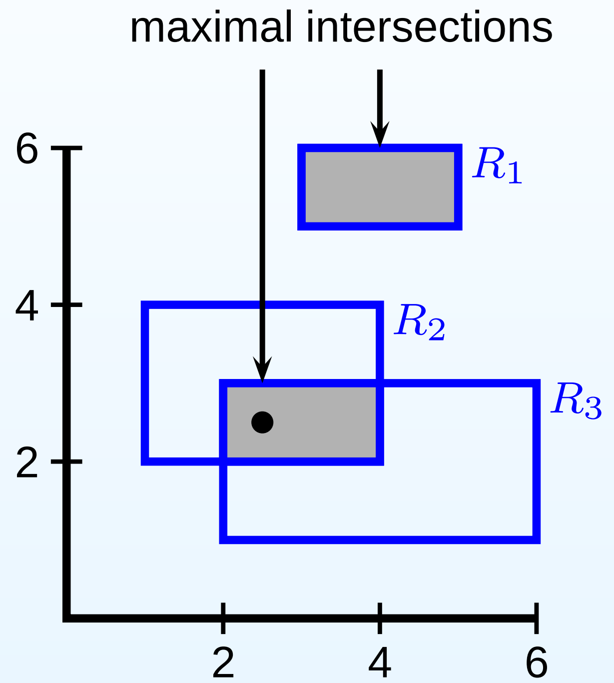
The nonparametric maximum likelihood estimator

$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$



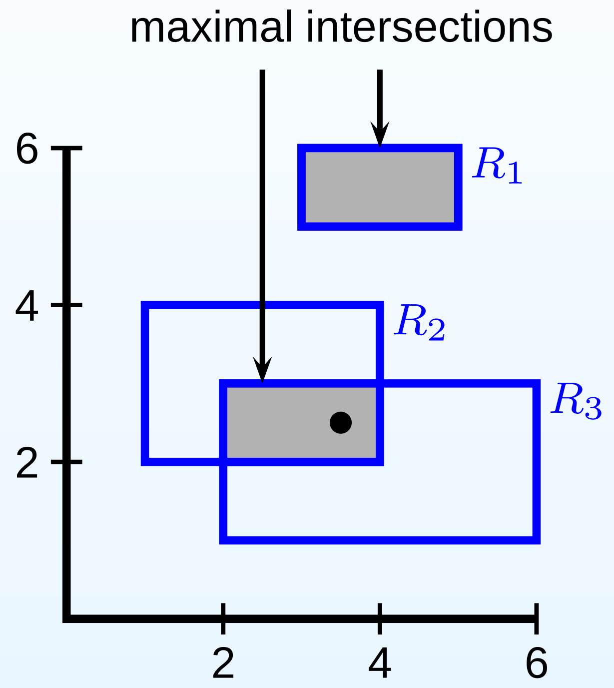
The nonparametric maximum likelihood estimator

$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$



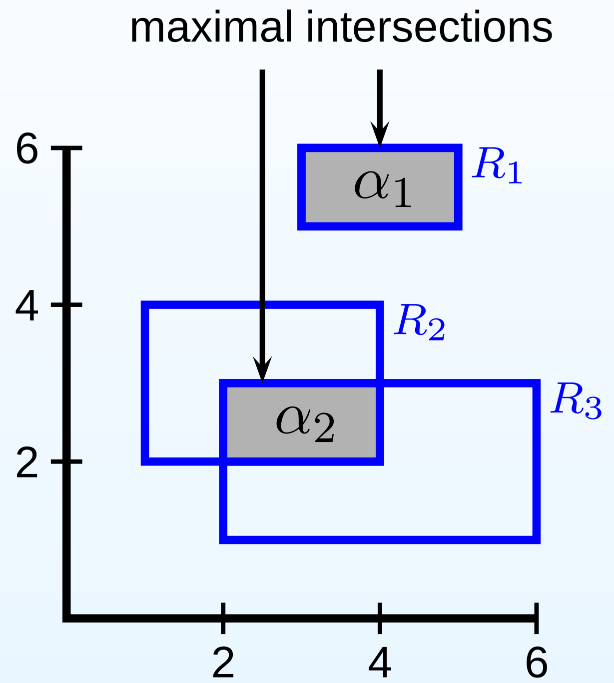
The nonparametric maximum likelihood estimator

$\max_{F\in\mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$



The nonparametric maximum likelihood estimator

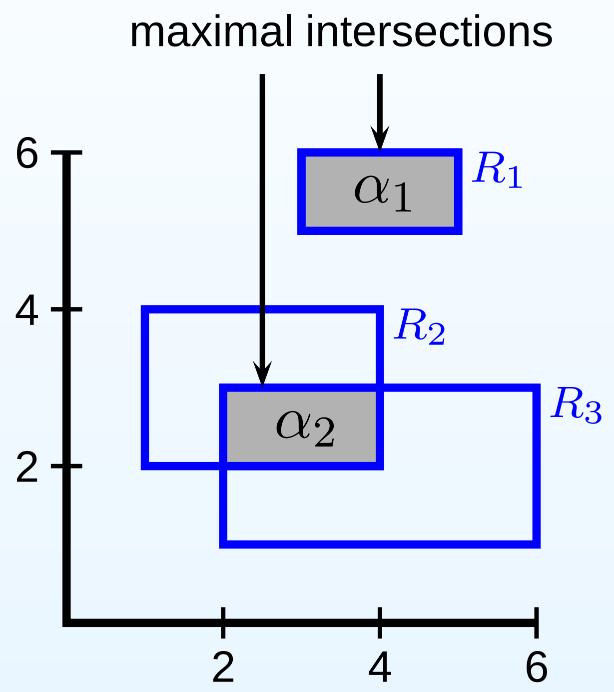
$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$



The nonparametric maximum likelihood estimator

$$\max_{F\in\mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$

$$= \max_{\alpha_1,\alpha_2} \log(\alpha_1) + 2\log(\alpha_2)$$

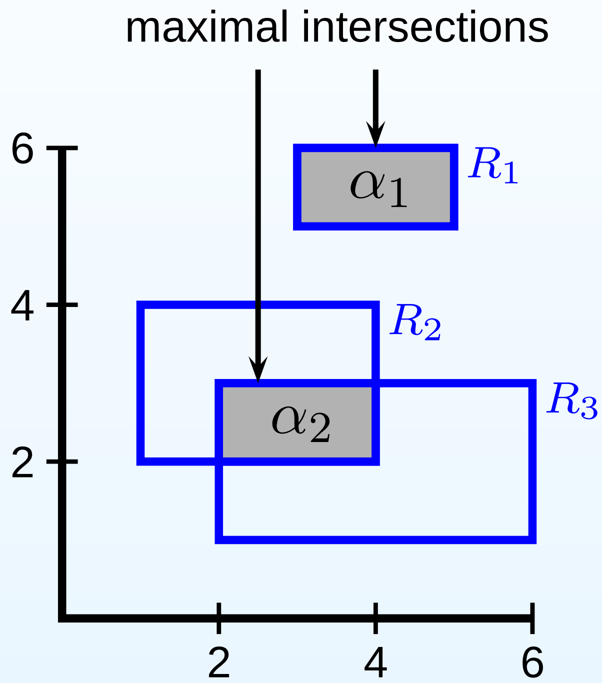


The nonparametric maximum likelihood estimator

$$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$$

$$= \max_{\alpha_1, \alpha_2} \log(\alpha_1) + 2\log(\alpha_2)$$

$$\Rightarrow \alpha_1 = 1/3, \quad \alpha_2 = 2/3$$

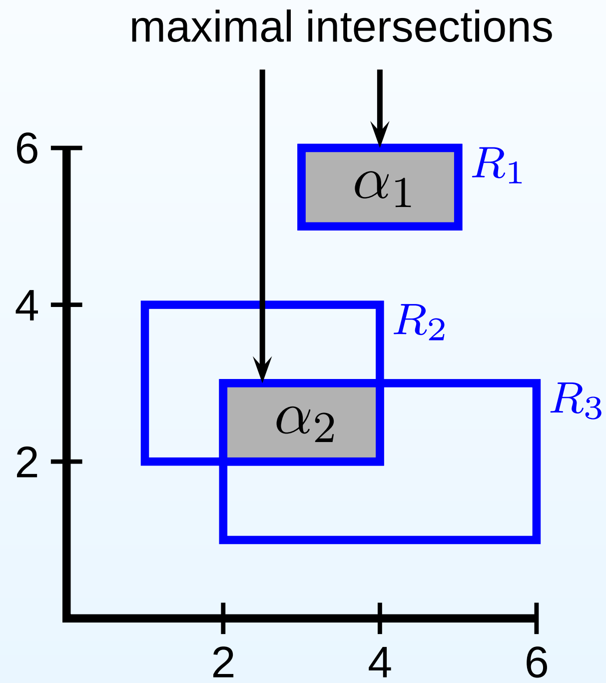


The nonparametric maximum likelihood estimator

$\max_{F\in\mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$

$= \max_{\alpha_1,\alpha_2} \log(\alpha_1) + 2\log(\alpha_2)$

$\Rightarrow \alpha_1 = 1/3, \quad \alpha_2 = 2/3$

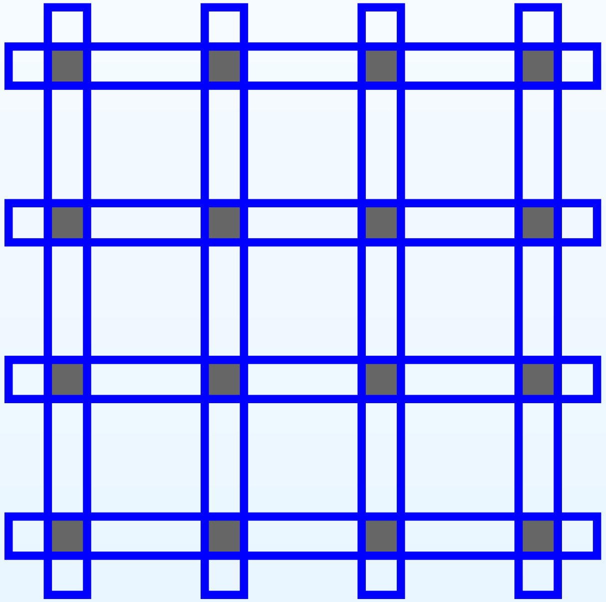


Computation of the MLE:

- Reduction step: find maximal intersections
- Optimization step: solve optimization problem in α

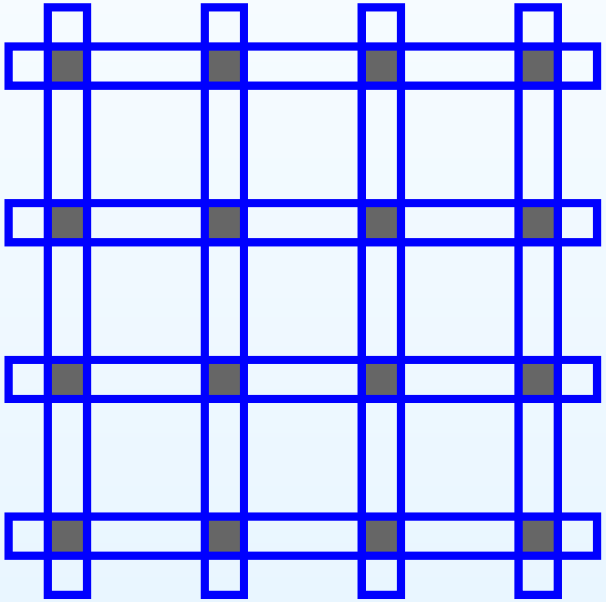
Difficulty in the computation of the MLE

- Number of maximal intersections can be very large:
  - For bivariate censored data: $O(n^2)$



Difficulty in the computation of the MLE

- Number of maximal intersections can be very large:
  - For bivariate censored data: $O(n^2)$



  - For d-variate censored data: $O(n^d)$

Outline

- Reduction step
- Optimization step
- R-package ‘MLEcens’

Reduction step

Previous work:

- Betensky and Finkelstein (1999)
- Song (2001)
- Gentleman and Vandal (2001), time complexity $O(n^5)$
- Bogaerts and Lesaffre (2004), time complexity $O(n^3)$

Reduction step

Previous work:

- Betensky and Finkelstein (1999)
- Song (2001)
- Gentleman and Vandal (2001), time complexity $O(n^5)$
- Bogaerts and Lesaffre (2004), time complexity $O(n^3)$

Related algorithm: finding the maximum number of rectangles having a non-empty intersection:

- Lee (1983), time complexity $O(n \log n)$

Reduction step

Previous work:

- Betensky and Finkelstein (1999)
- Song (2001)
- Gentleman and Vandal (2001), time complexity $O(n^5)$
- Bogaerts and Lesaffre (2004), time complexity $O(n^3)$

Related algorithm: finding the maximum number of rectangles having a non-empty intersection:

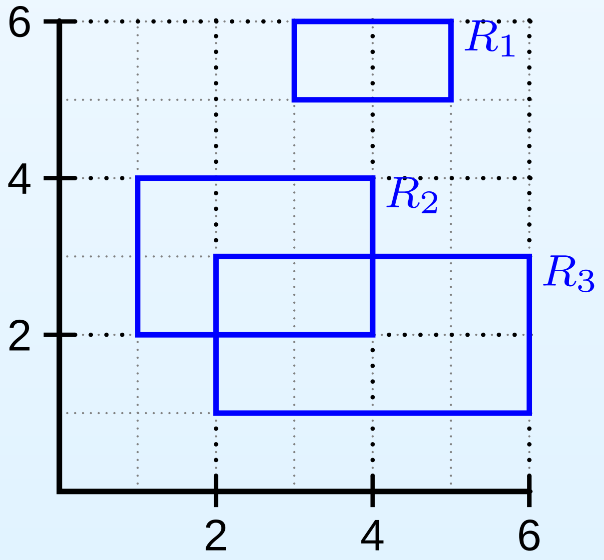
- Lee (1983), time complexity $O(n \log n)$

Our reduction algorithm, motivated by Lee (1983):

- Height Map Algorithm, time complexity $O(n^2)$

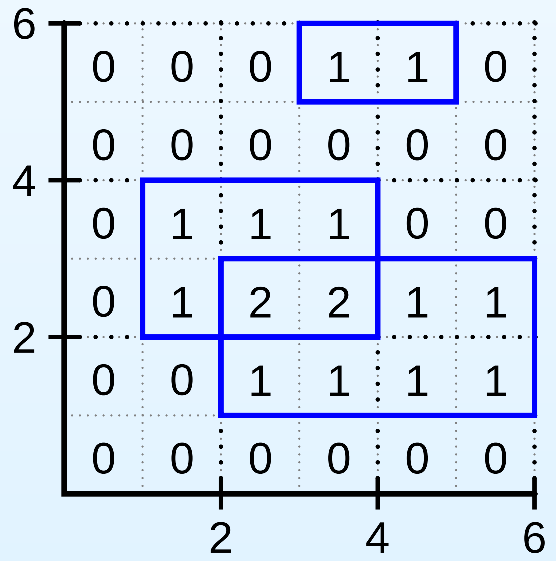
Height Map Algorithm

- Definition: $A_j \neq \emptyset$ is a *maximal intersection* if and only if $A_j = \cap_{i \in \beta_j} R_i$ for some set $\beta_j \subset \{1, \ldots, n\}$ and there is no strict superset $\beta_j^* \subset \{1, \ldots, n\}$ of β_j for which $\cap_{i \in \beta_j^*} R_i \neq \emptyset$.



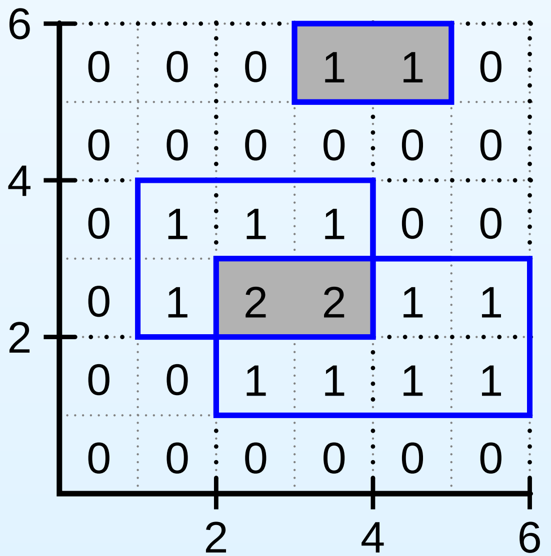
Height Map Algorithm

- Definition: $A_j \neq \emptyset$ is a *maximal intersection* if and only if $A_j = \cap_{i \in \beta_j} R_i$ for some set $\beta_j \subset \{1, \ldots, n\}$ and there is no strict superset $\beta_j^* \subset \{1, \ldots, n\}$ of β_j for which $\cap_{i \in \beta_j^*} R_i \neq \emptyset$.
- Basic idea of the Height Map Algorithm:
  - Define a height map of the observed sets:



Height Map Algorithm

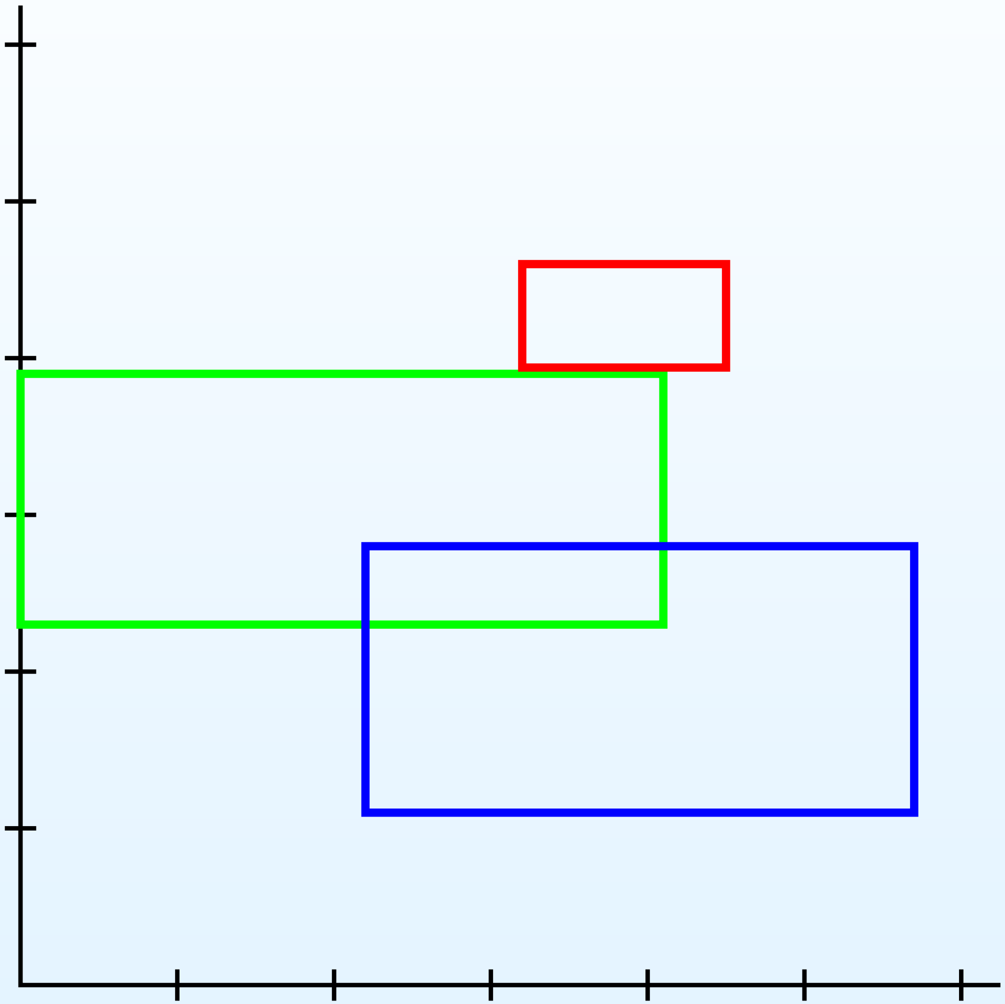
- Definition: $A_j \neq \emptyset$ is a *maximal intersection* if and only if $A_j = \cap_{i \in \beta_j} R_i$ for some set $\beta_j \subset \{1, \ldots, n\}$ and there is no strict superset $\beta_j^* \subset \{1, \ldots, n\}$ of β_j for which $\cap_{i \in \beta_j^*} R_i \neq \emptyset$.
- Basic idea of the Height Map Algorithm:
  - Define a height map of the observed sets:
  - The maximal intersections are exactly the local maximum regions of the height map



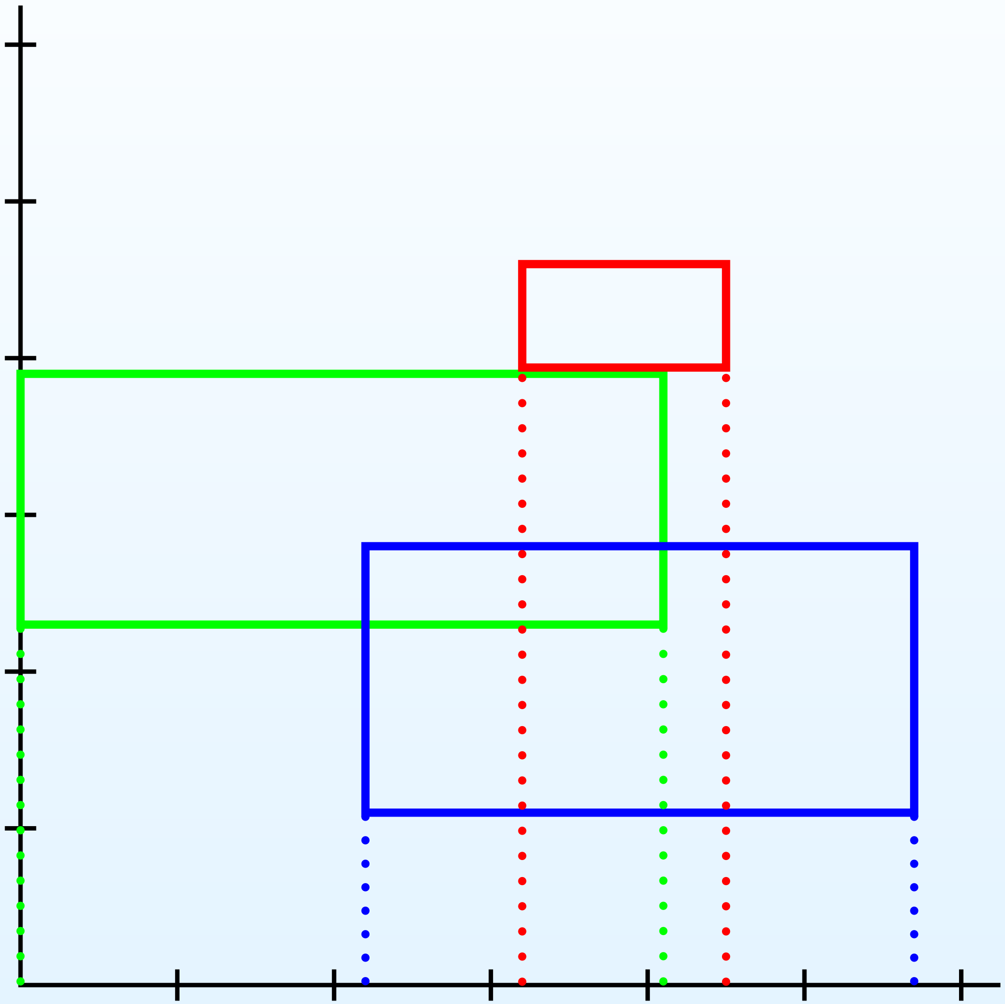
The algorithm

- Transform the observation rectangles into “canonical rectangles"
- Find local maximum regions of the height map of the canonical rectangles (by sweeping)
- Transform local maxima back to original coordinates

Transform rectangles into canonical rectangles

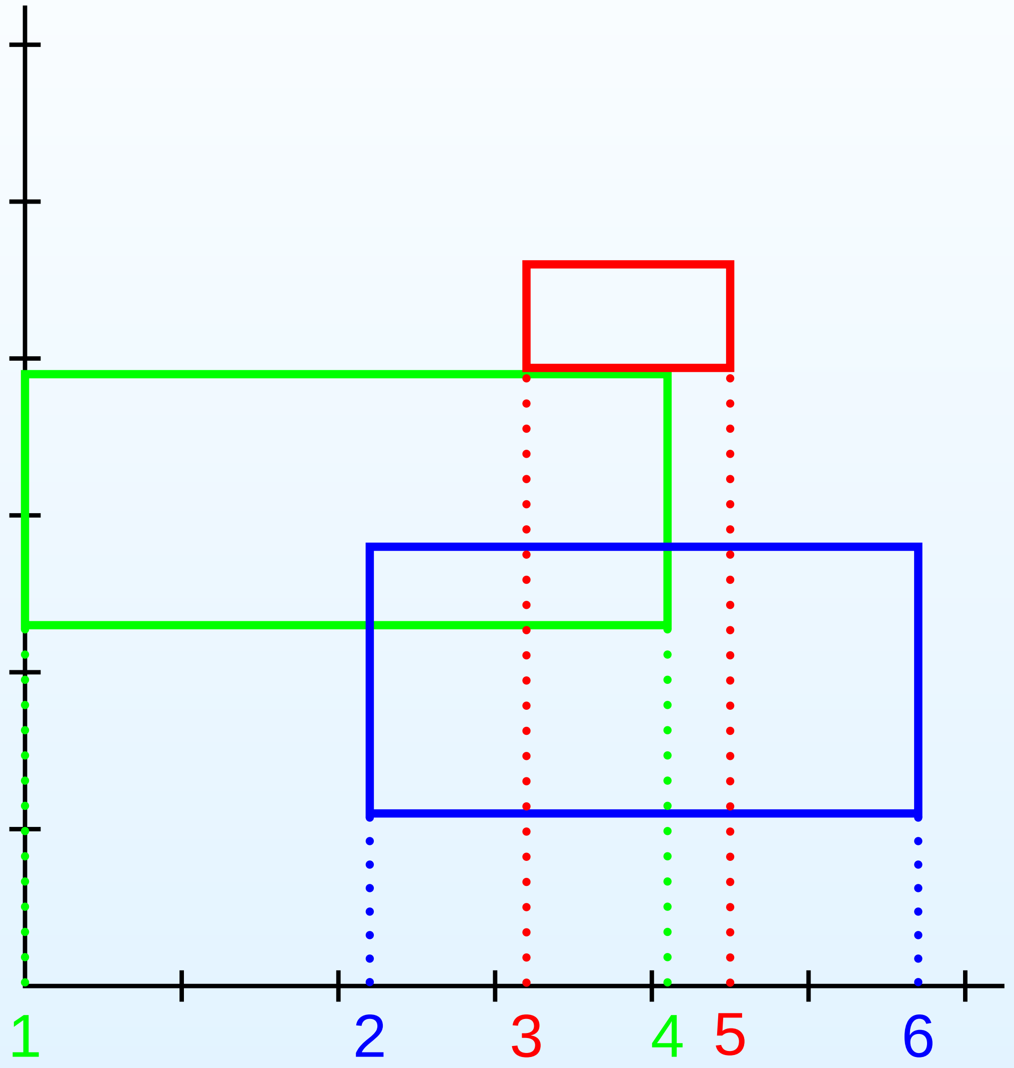


Transform rectangles into canonical rectangles



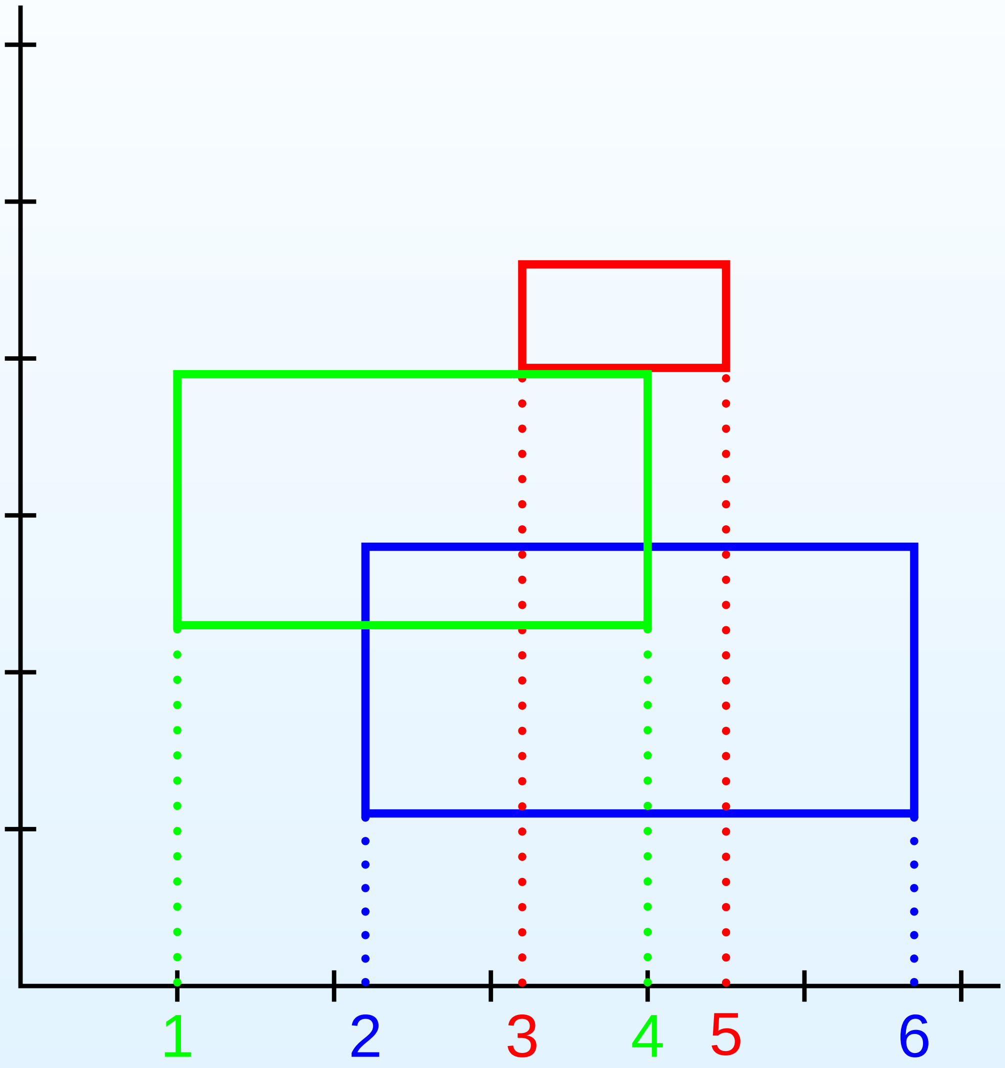
- Replace x-coordinates by their order statistics.

Transform rectangles into canonical rectangles



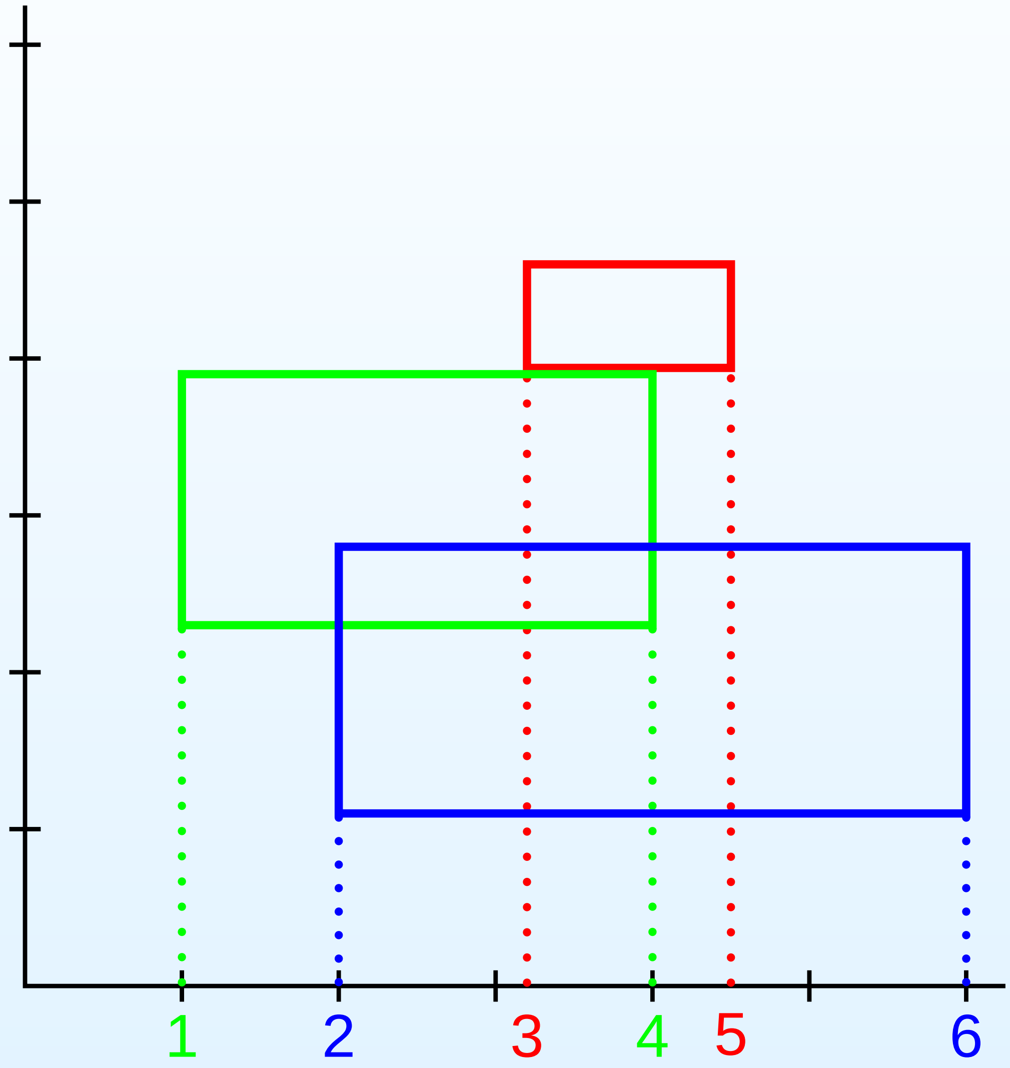
- Replace x-coordinates by their order statistics.

Transform rectangles into canonical rectangles



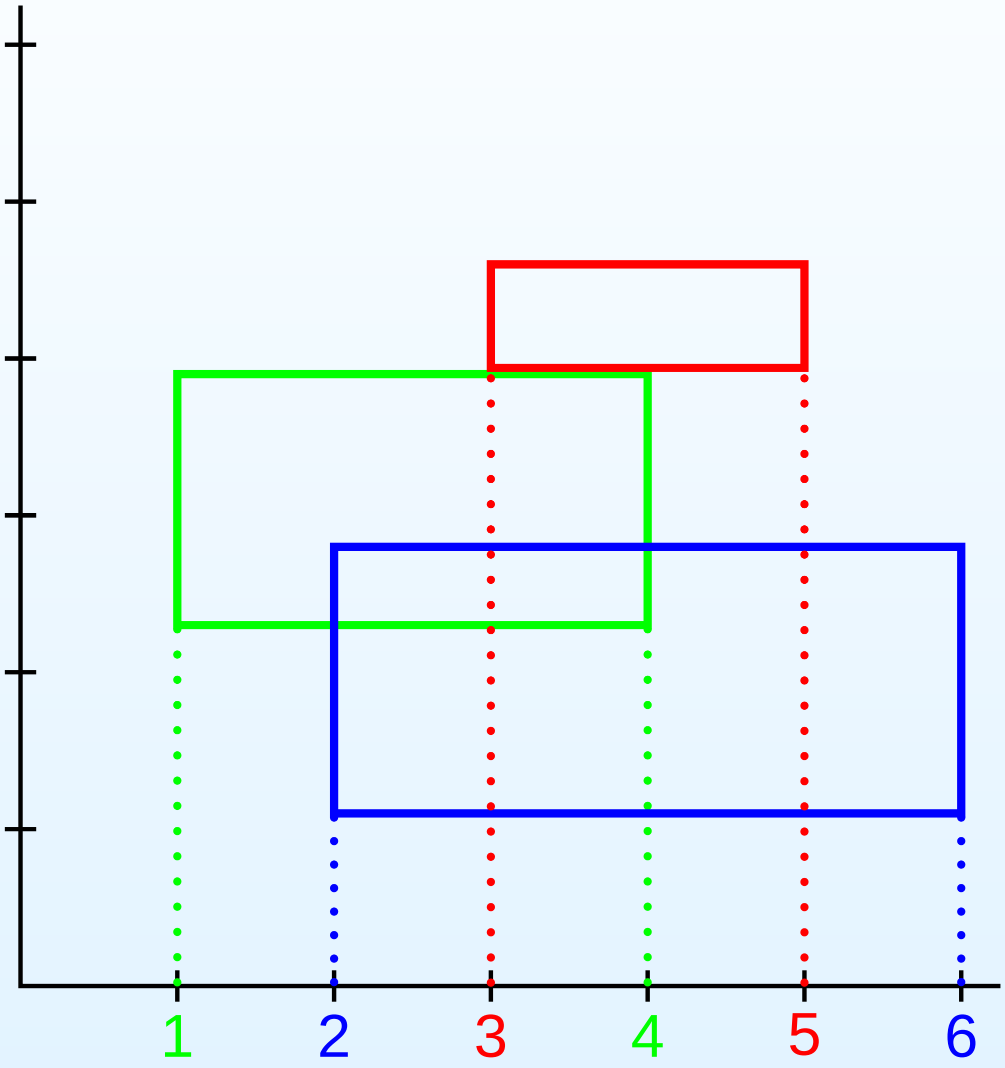
- Replace x-coordinates by their order statistics.

Transform rectangles into canonical rectangles



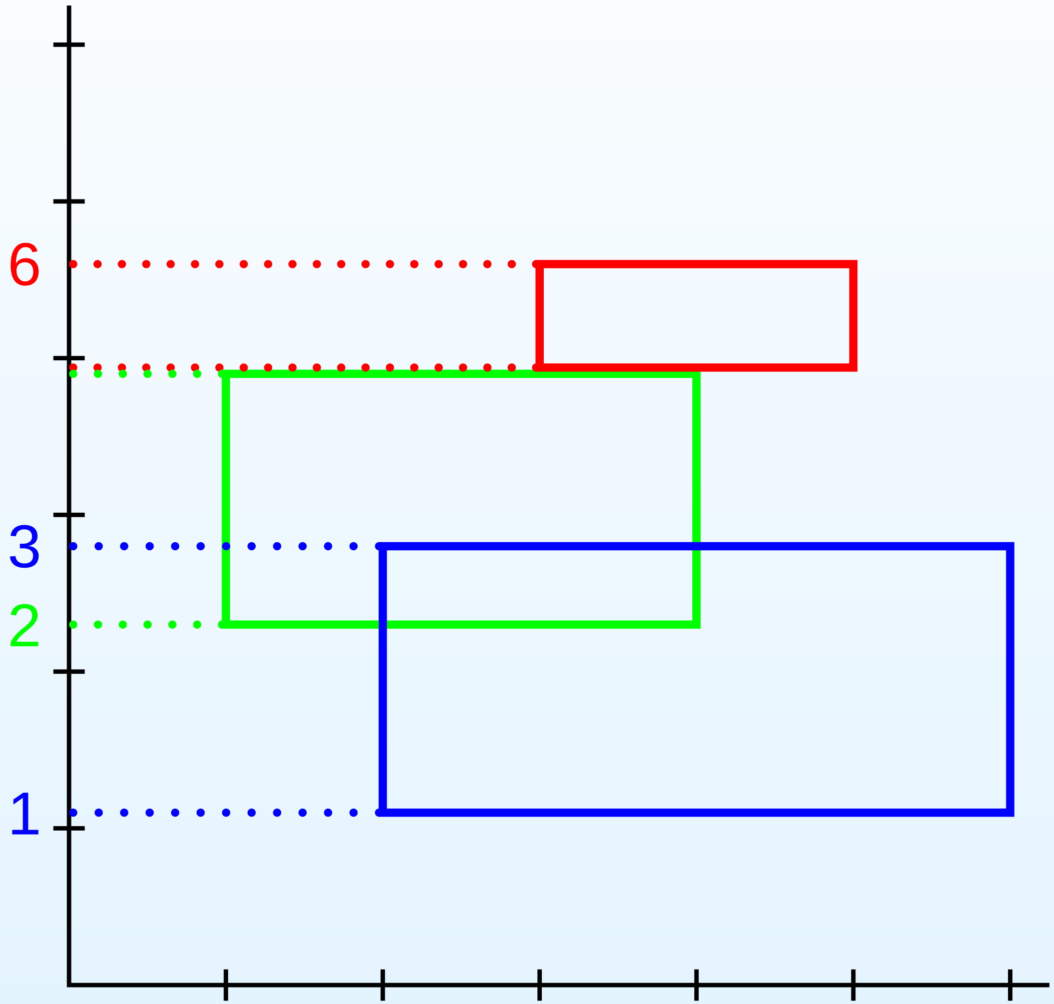
- Replace x-coordinates by their order statistics.

Transform rectangles into canonical rectangles



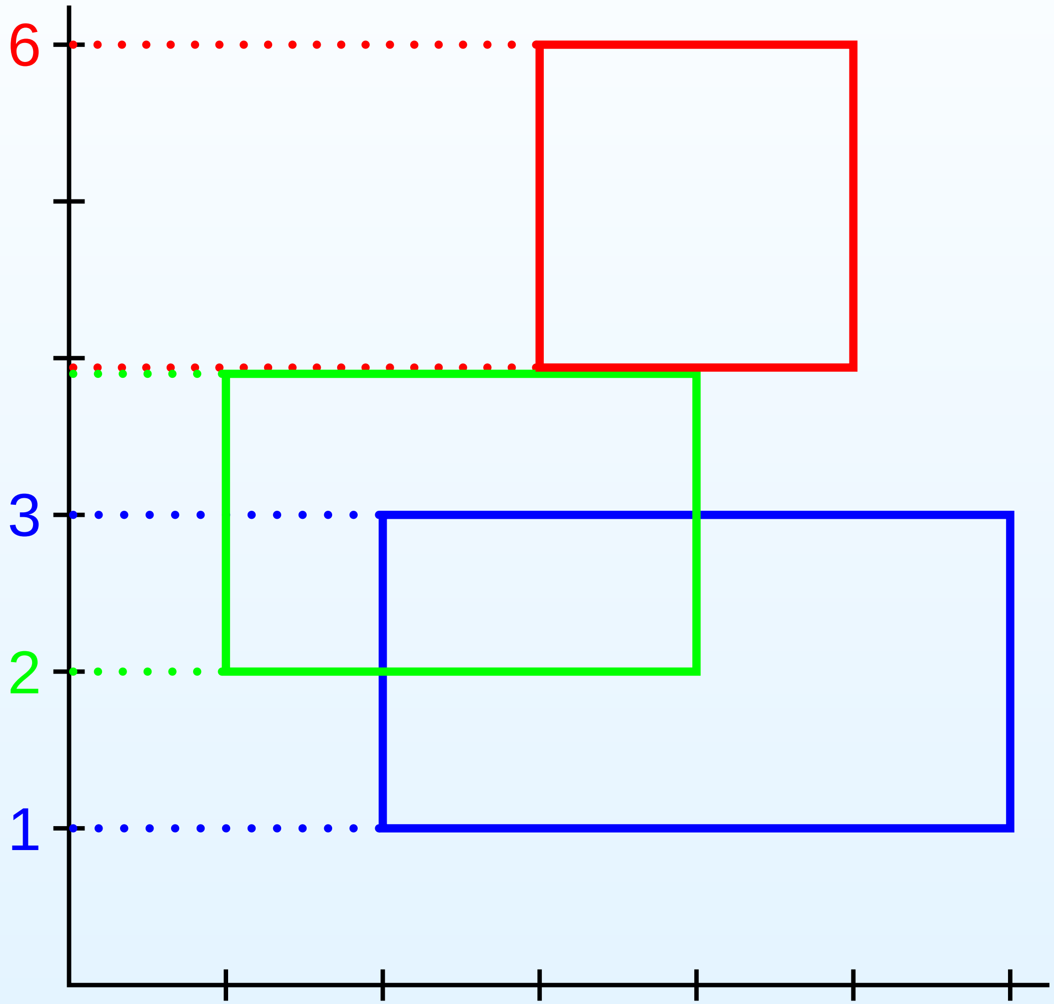
- Replace x-coordinates by their order statistics.

Transform rectangles into canonical rectangles



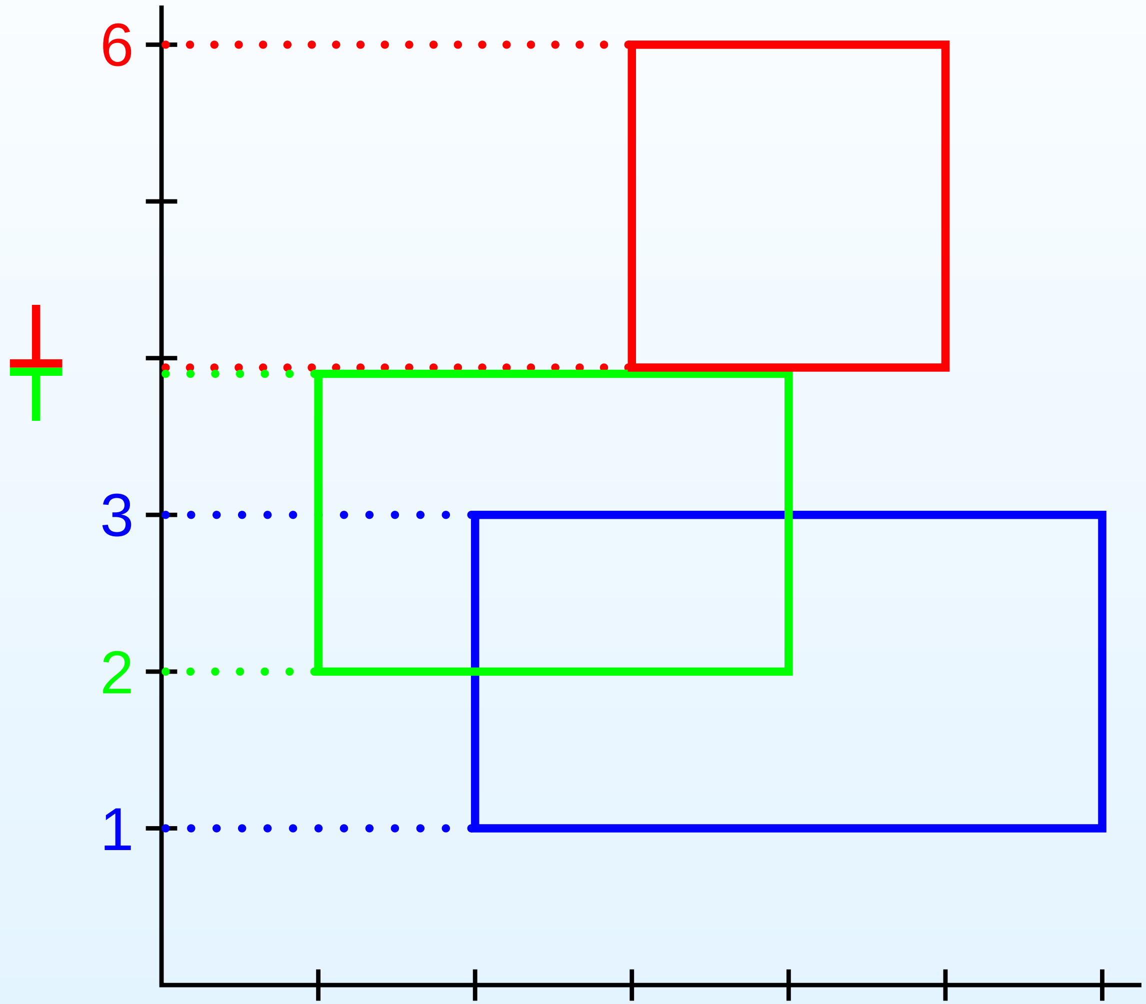
- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles



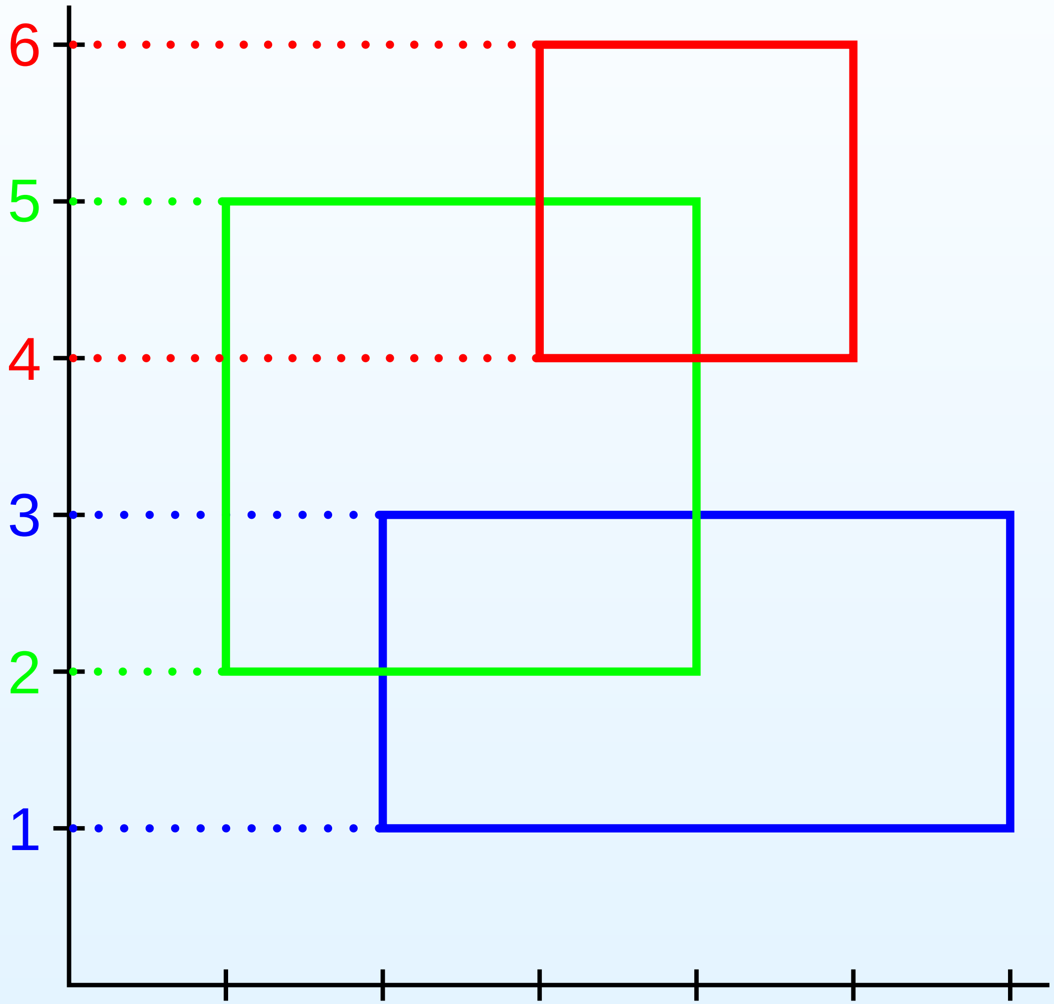
- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles



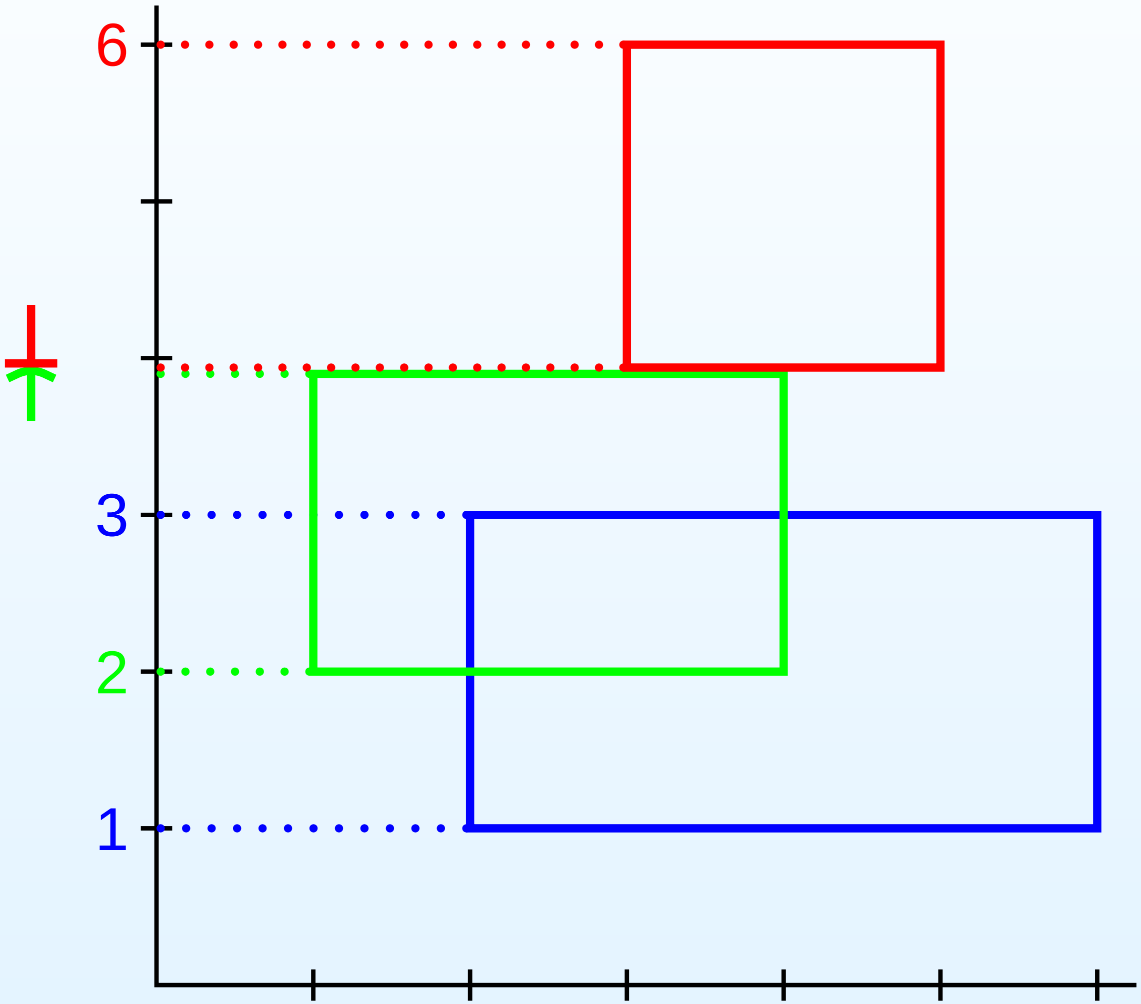
- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles



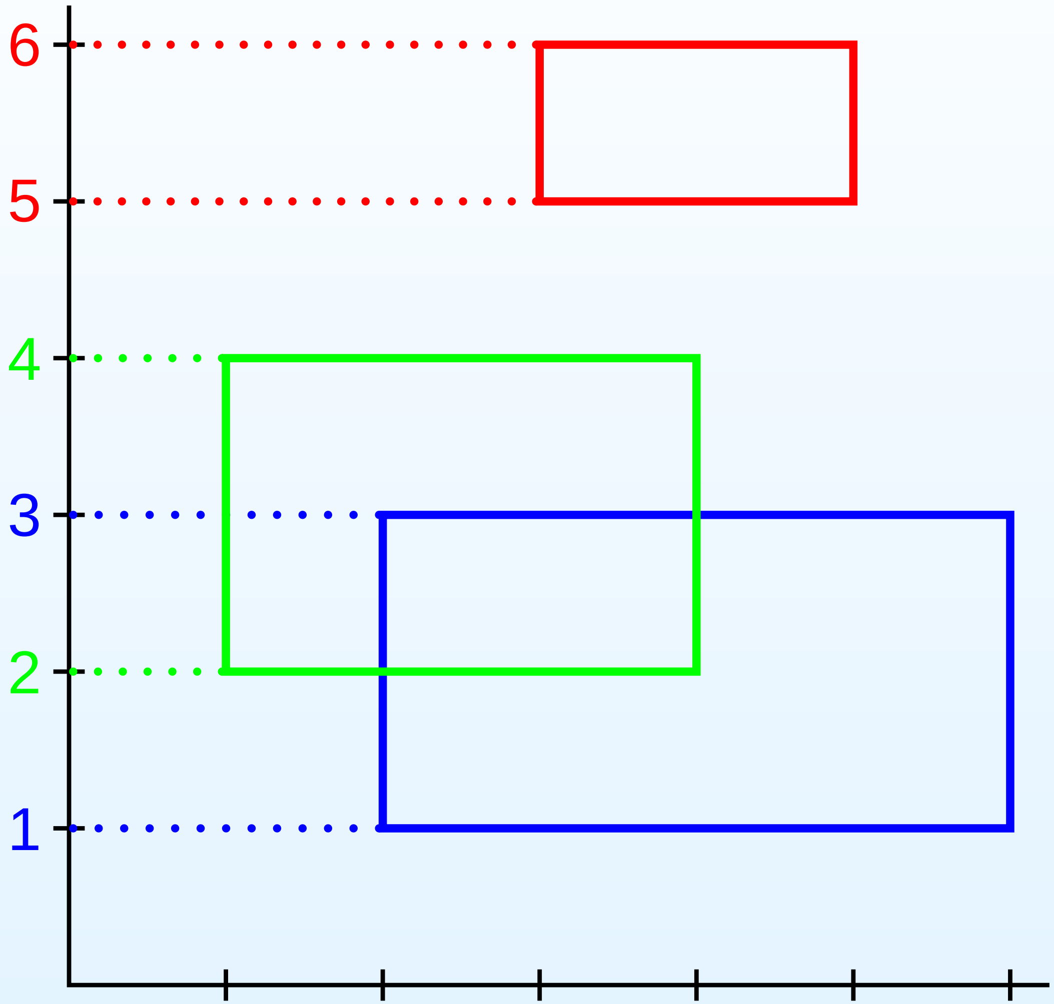
- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles



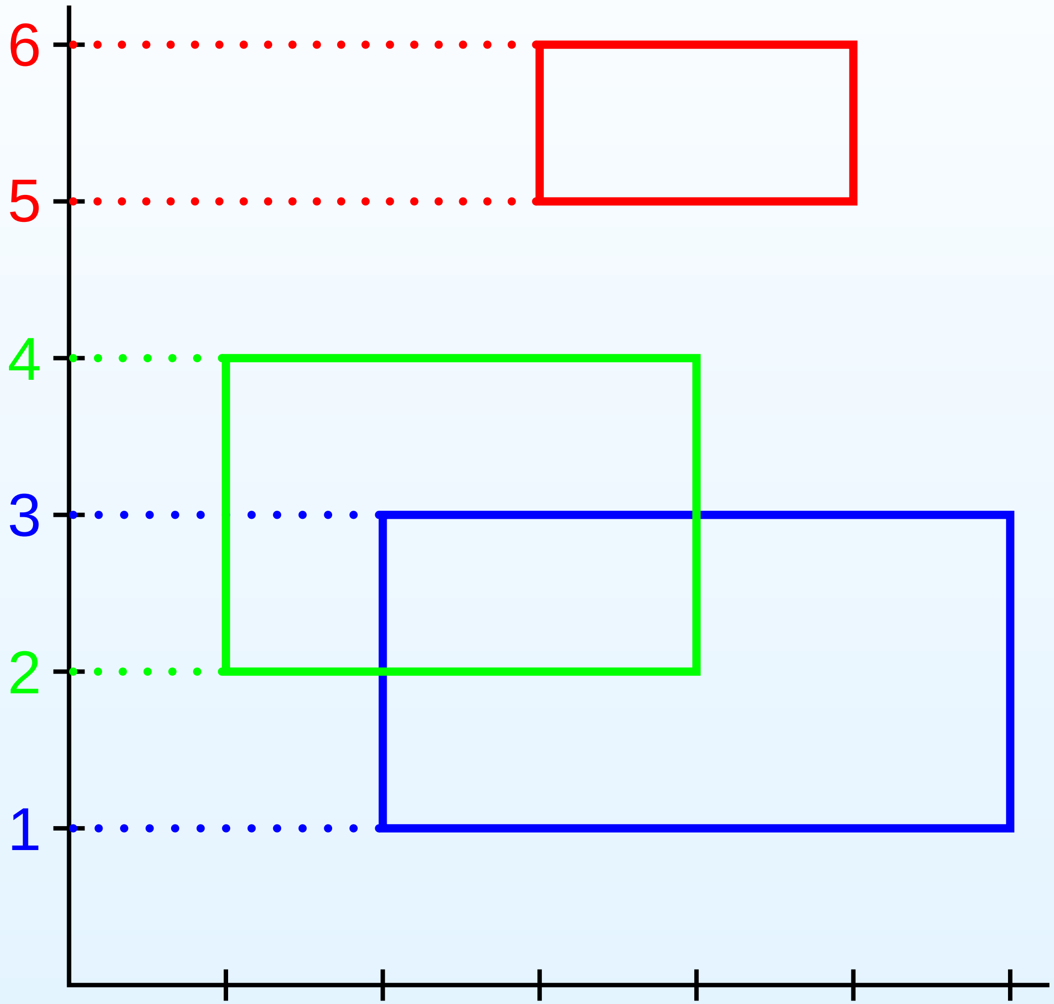
- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles



- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.

Transform rectangles into canonical rectangles

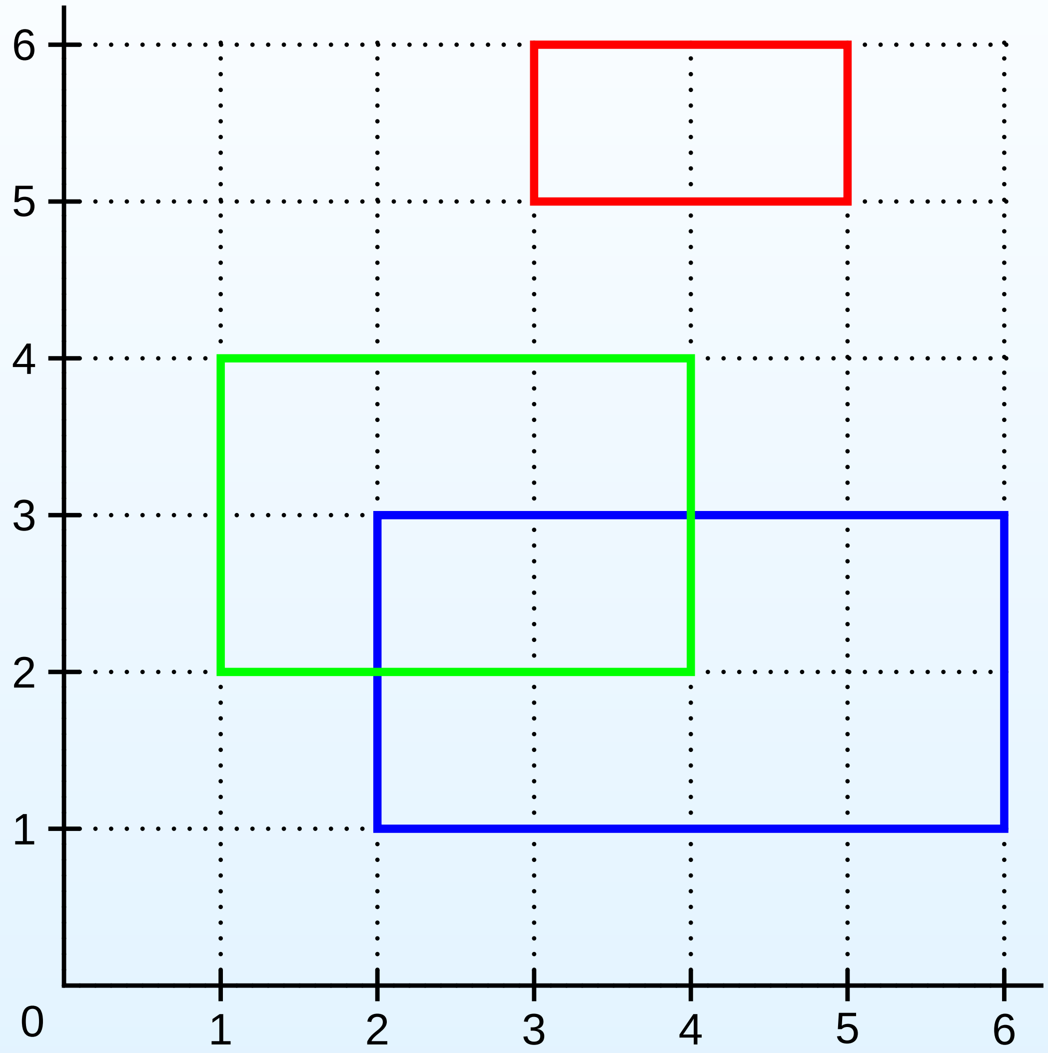


- Replace x-coordinates by their order statistics.
- Replace y-coordinates by their order statistics.
- All x-coodinates are different and take values in $\{1, 2, \ldots, 2n\}$ (same for y-coordinates).
- Intersection structure of the original and canonical rectangles is identical.

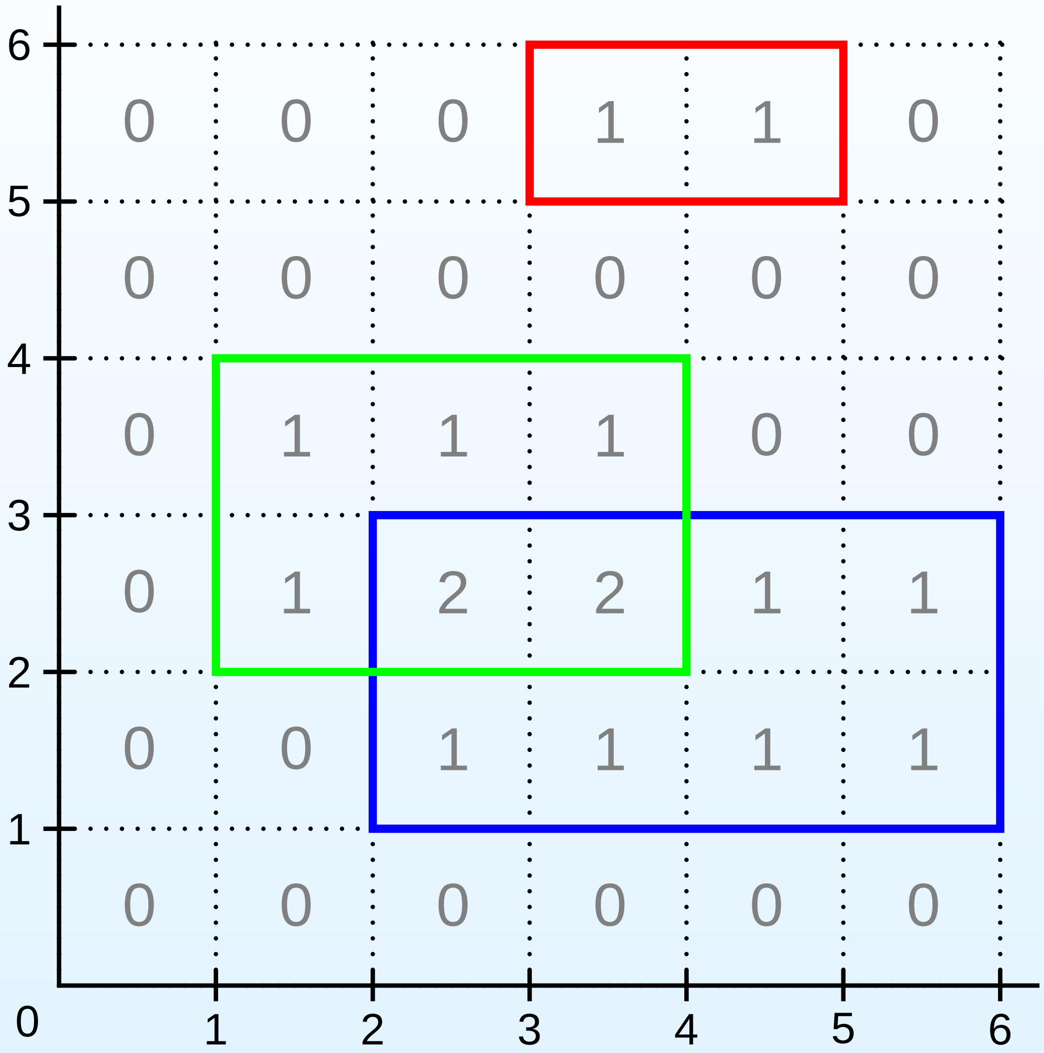
Why use canonical rectangles?

- We break possible ties early, so that we don't have to worry about this anymore
- It is easier and faster to work with integer coordinates

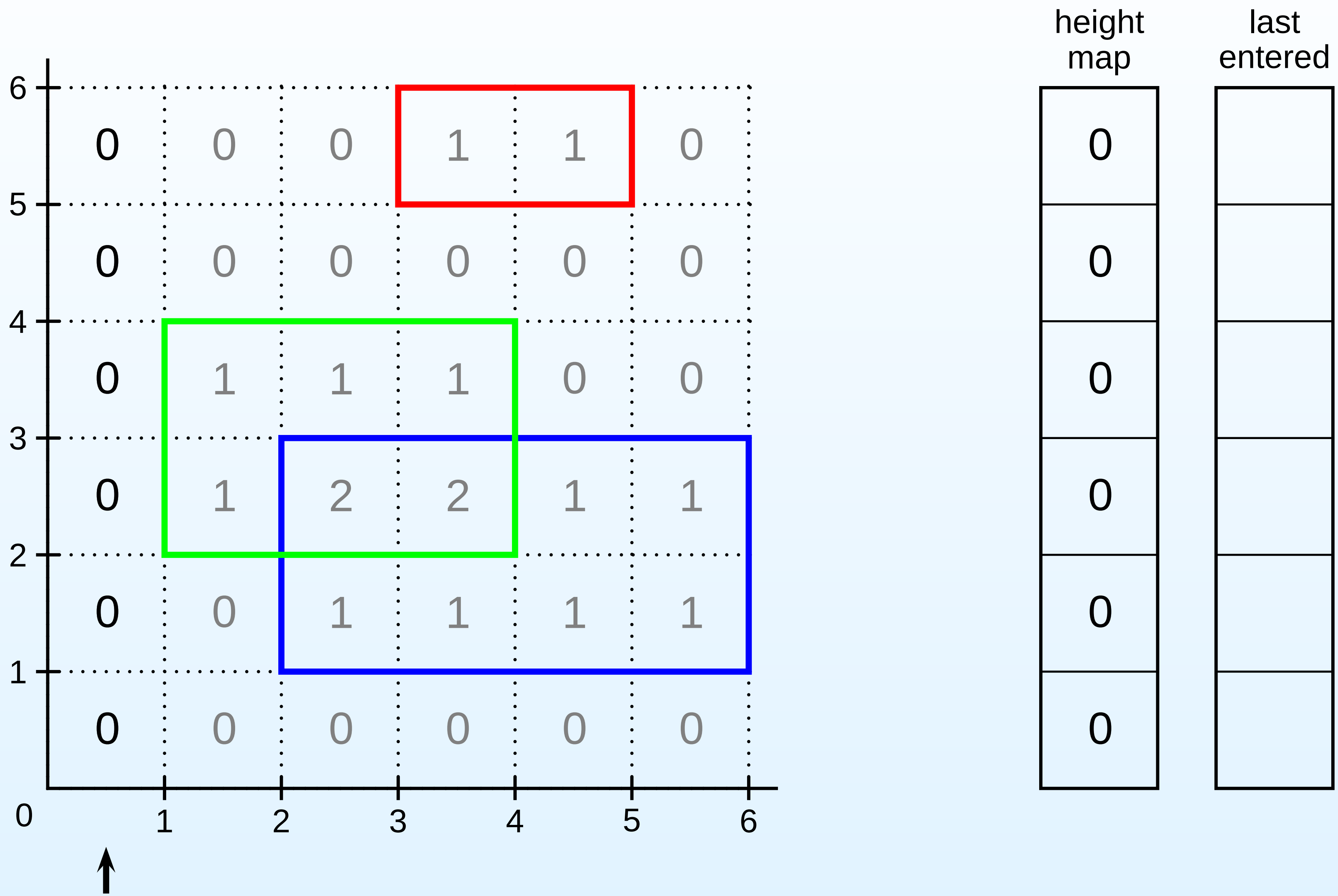
Find local maxima by sweeping through the height map



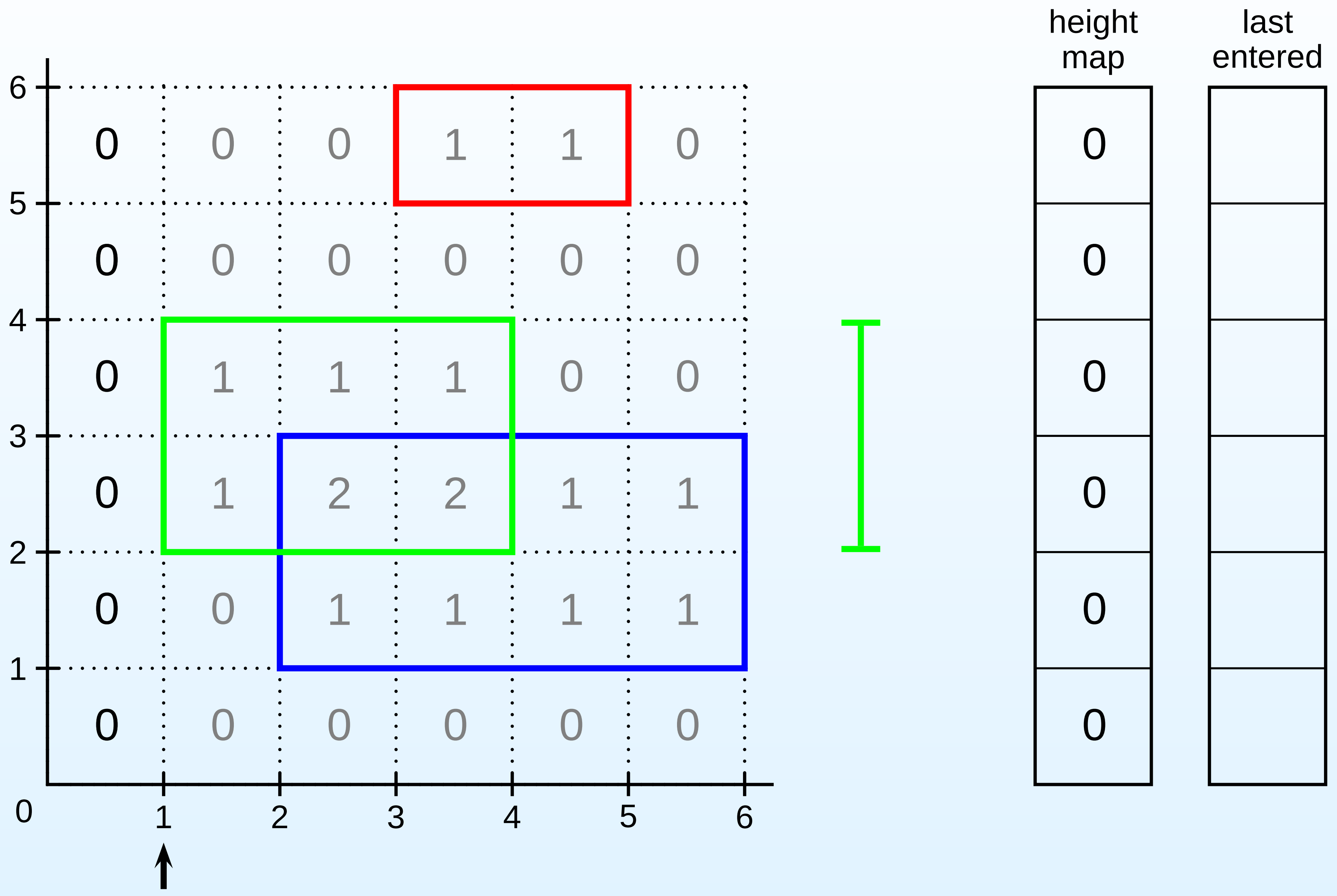
Find local maxima by sweeping through the height map



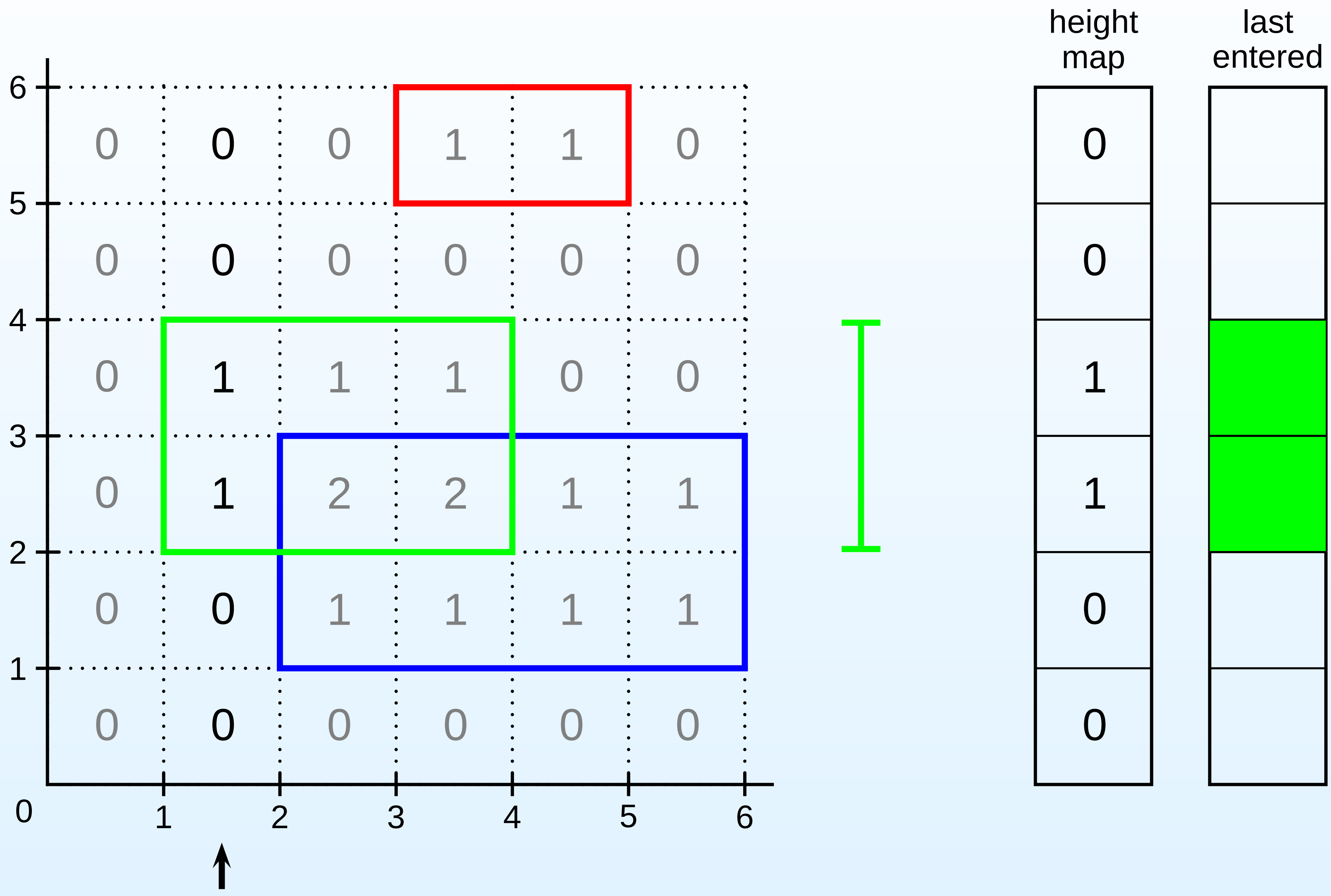
Find local maxima by sweeping through the height map



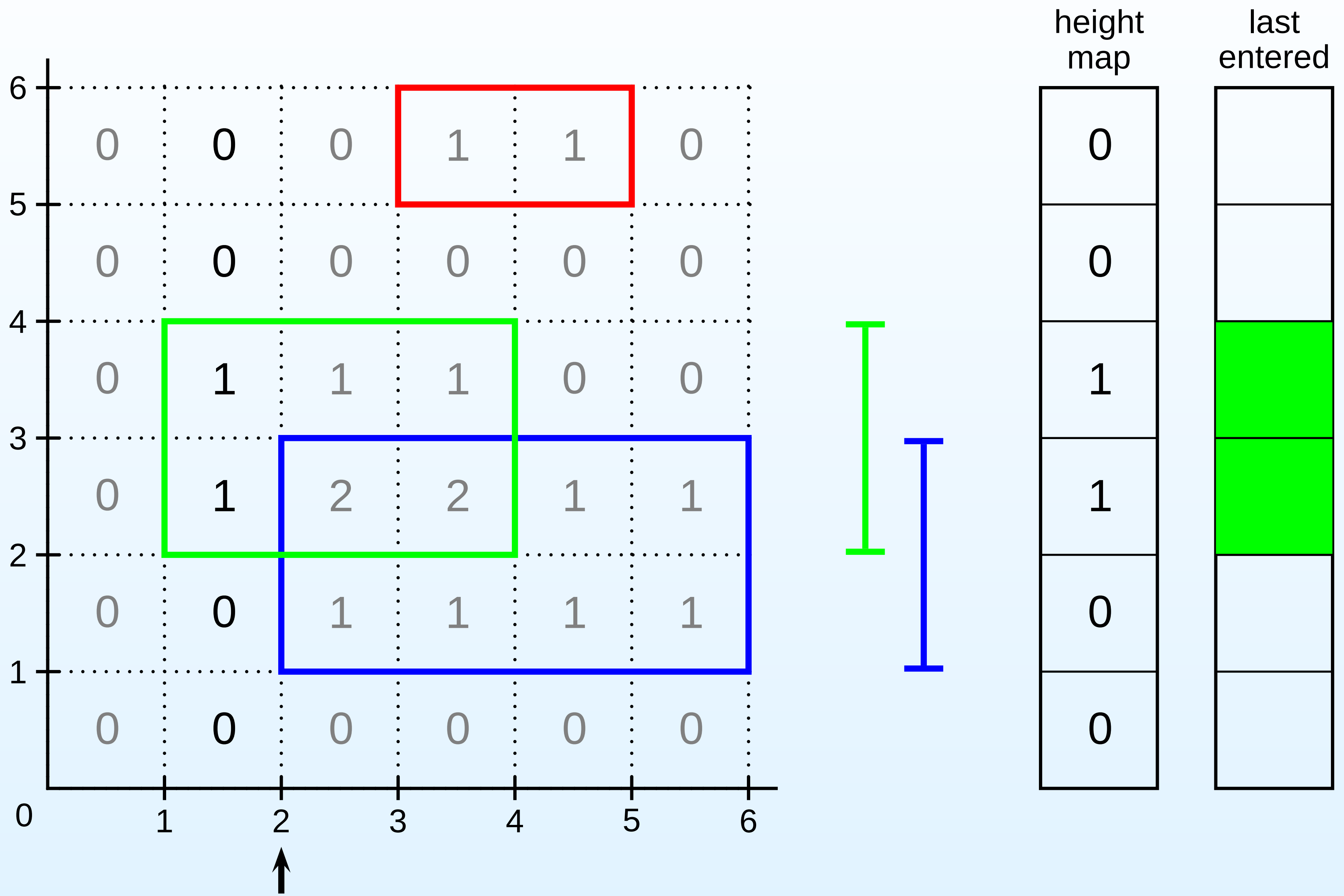
Find local maxima by sweeping through the height map



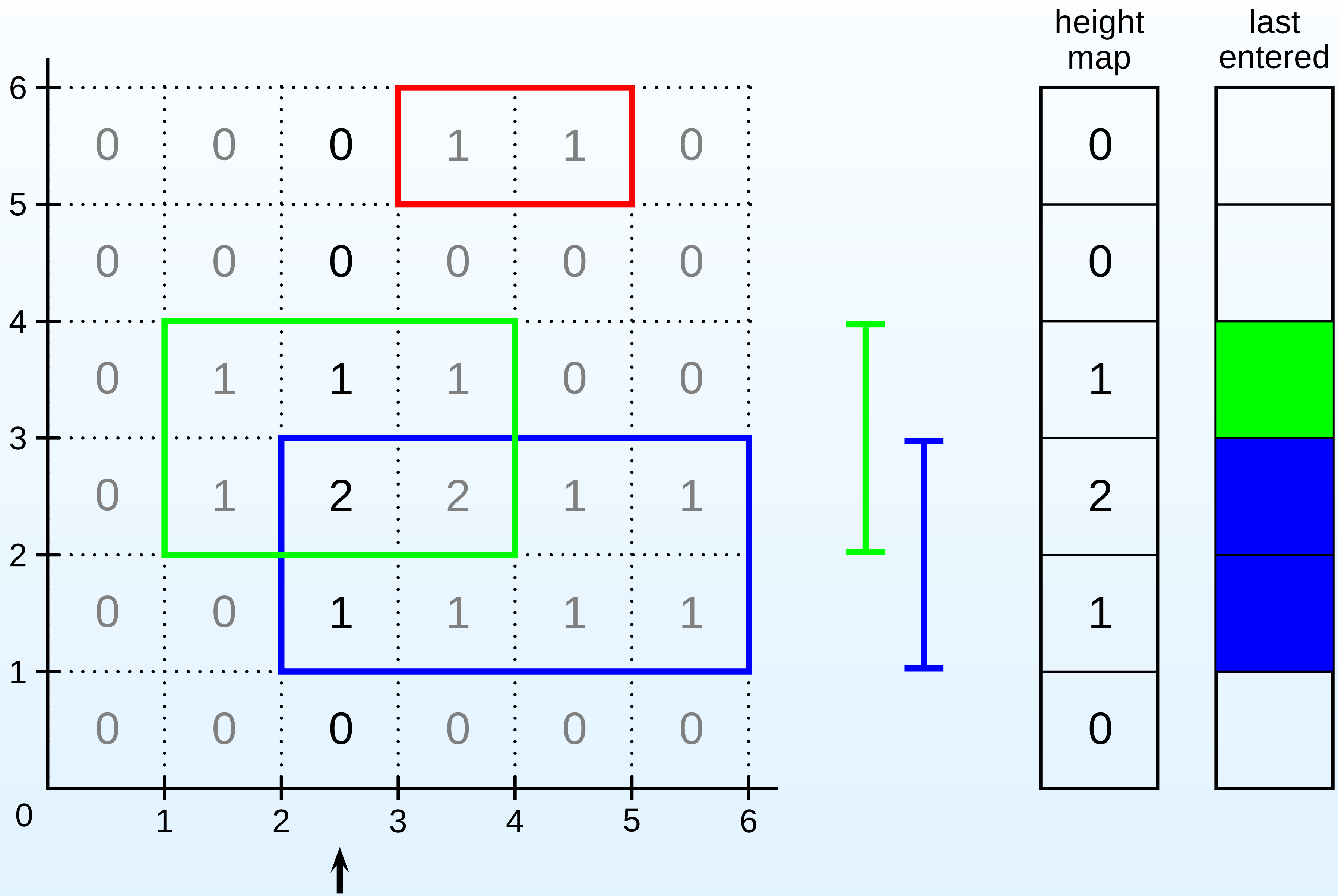
Find local maxima by sweeping through the height map



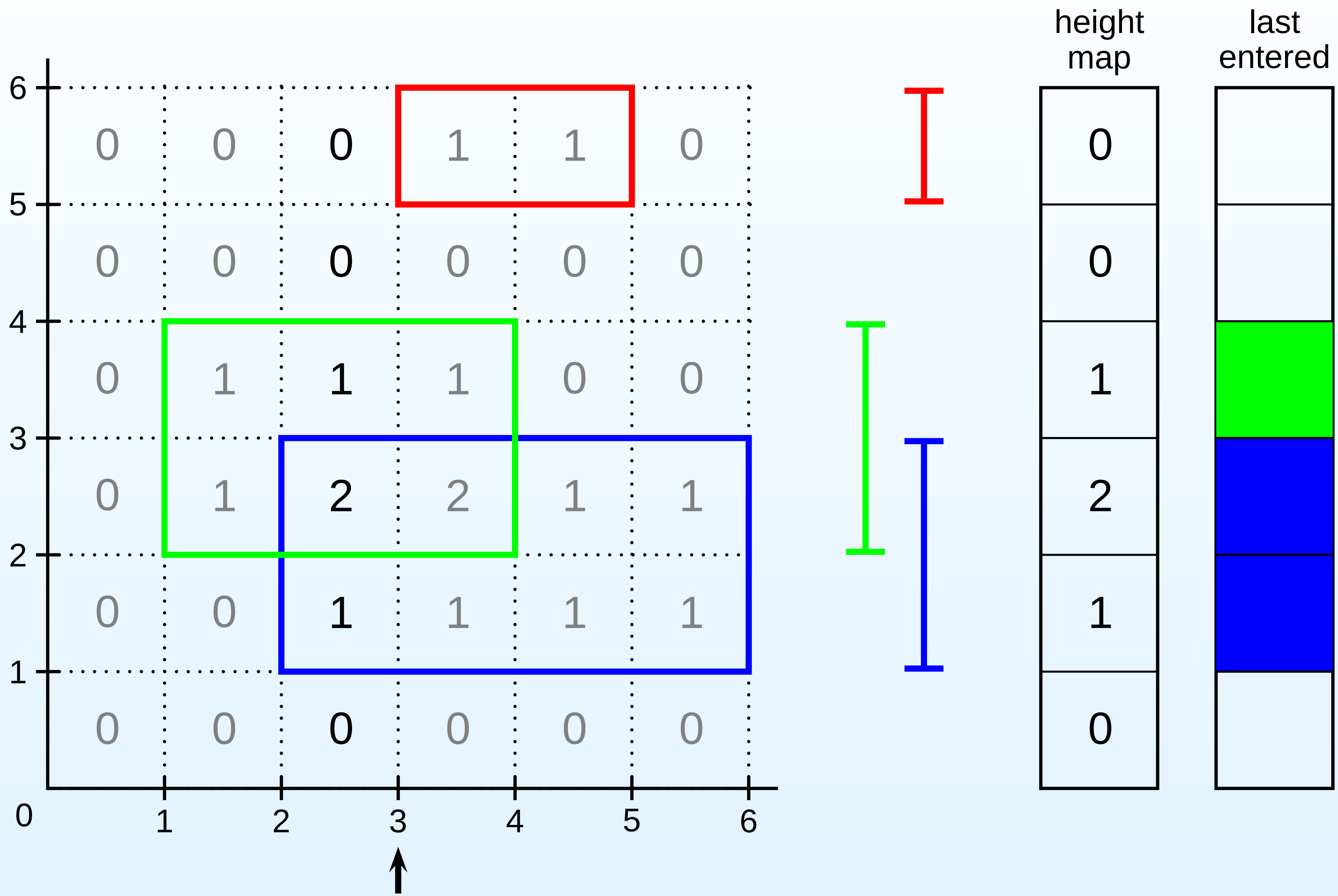
Find local maxima by sweeping through the height map



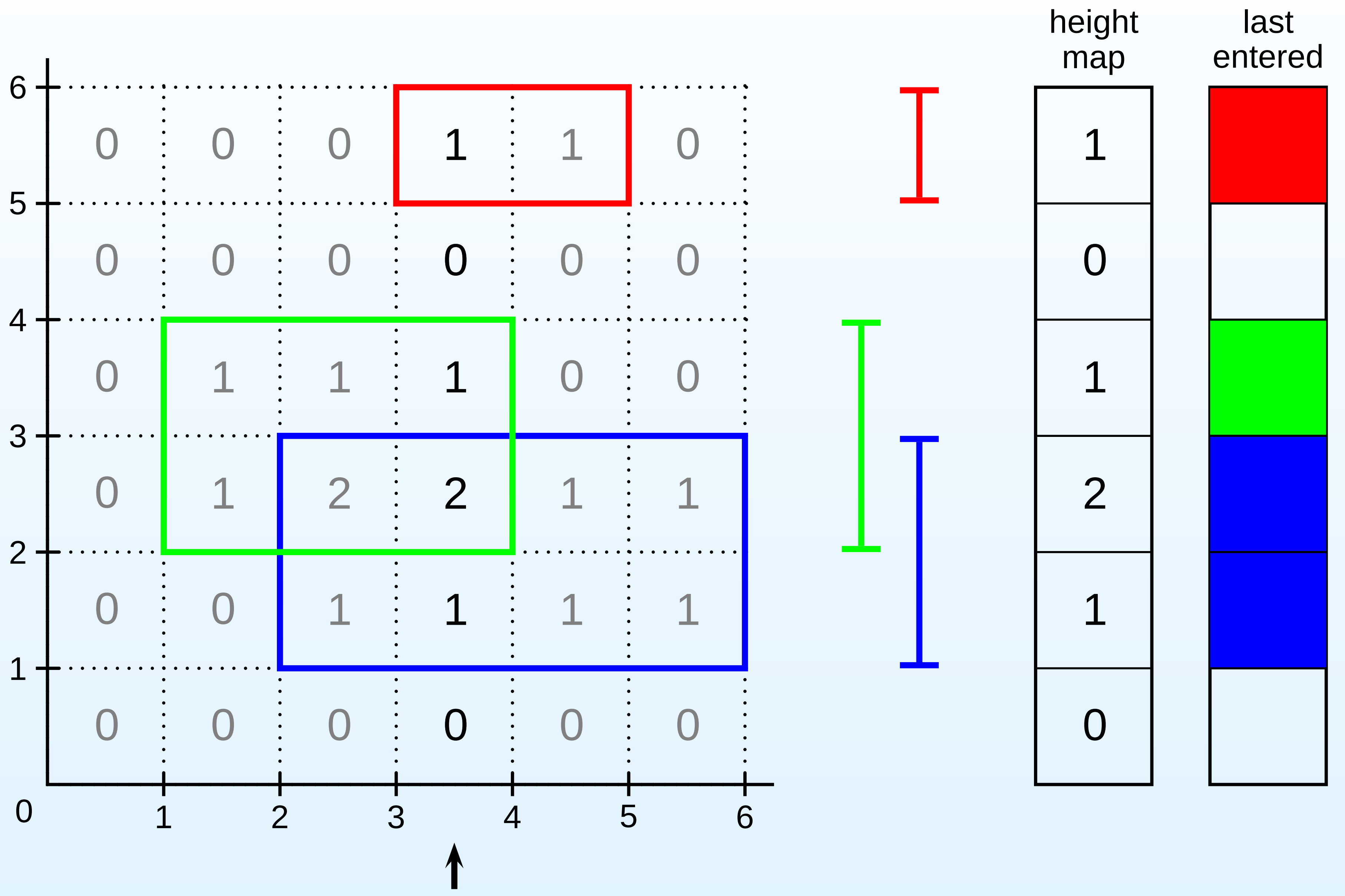
Find local maxima by sweeping through the height map



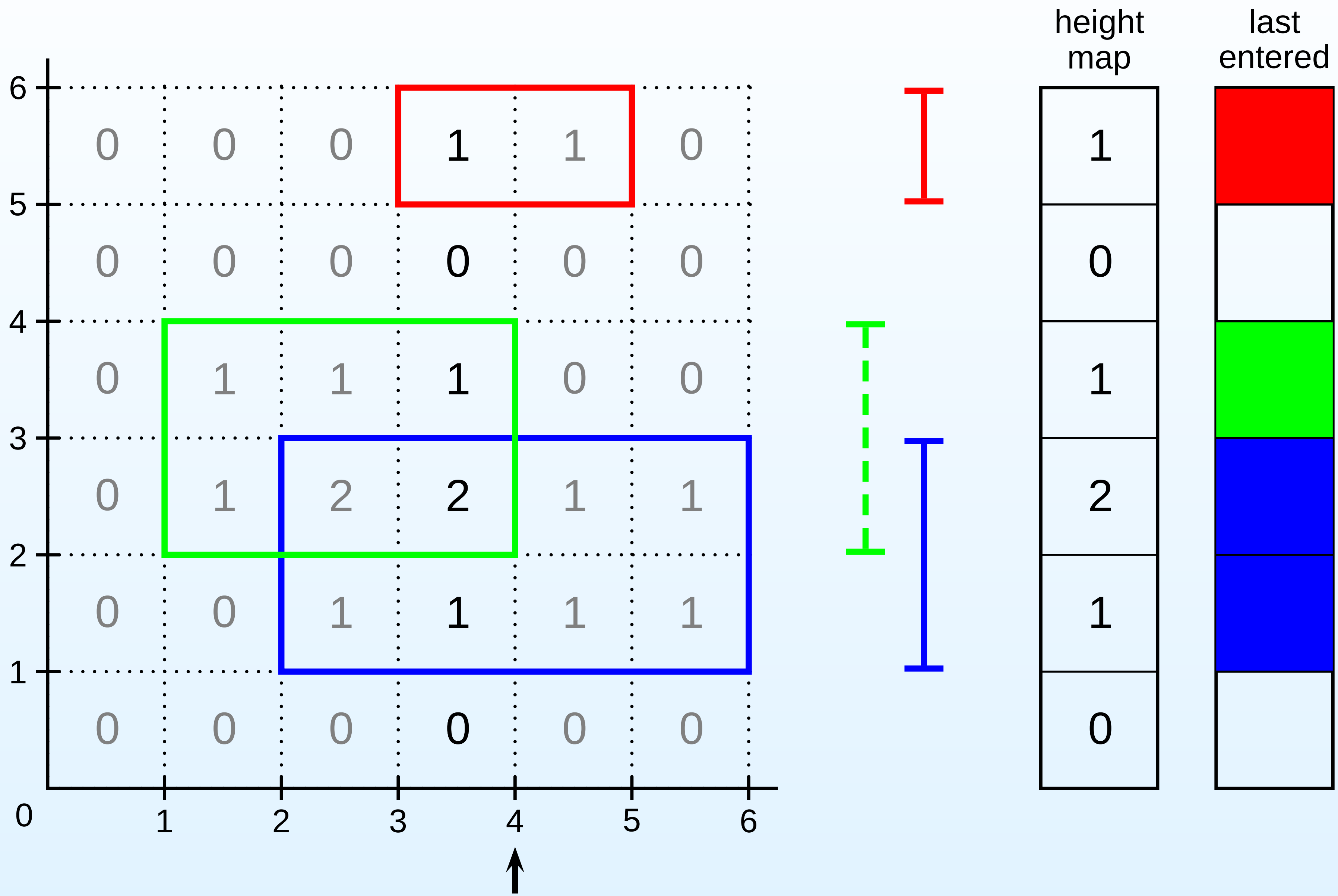
Find local maxima by sweeping through the height map



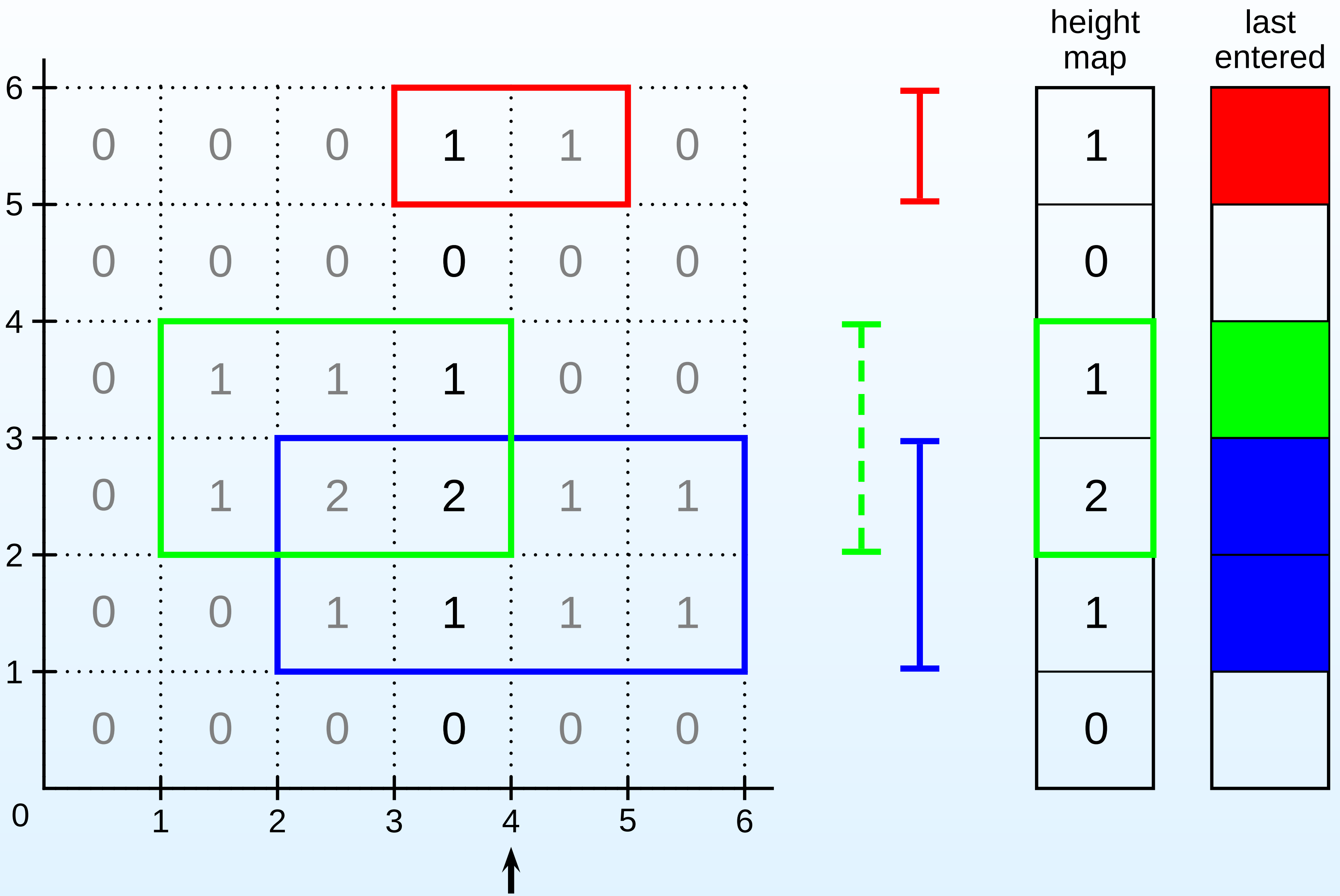
Find local maxima by sweeping through the height map



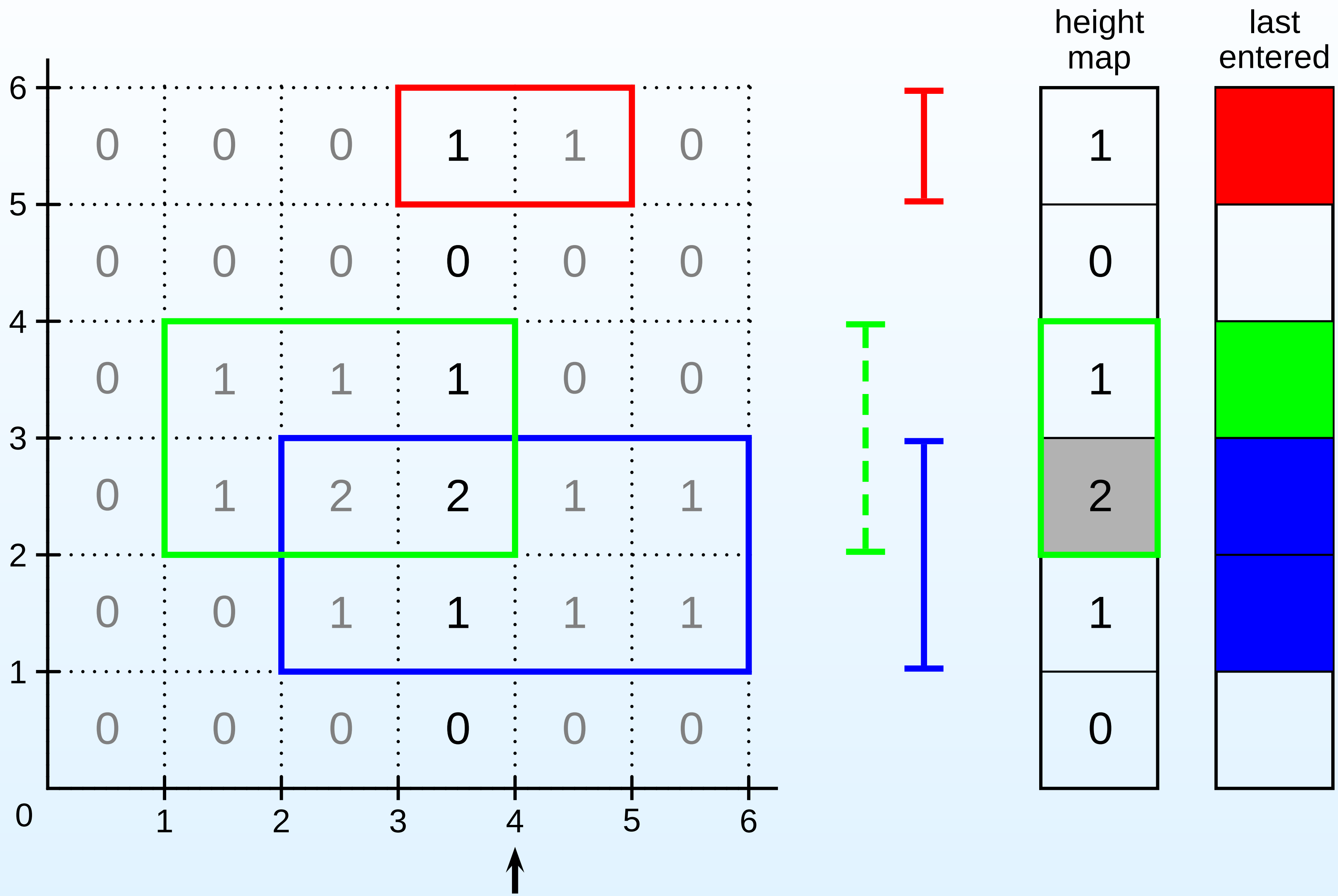
Find local maxima by sweeping through the height map



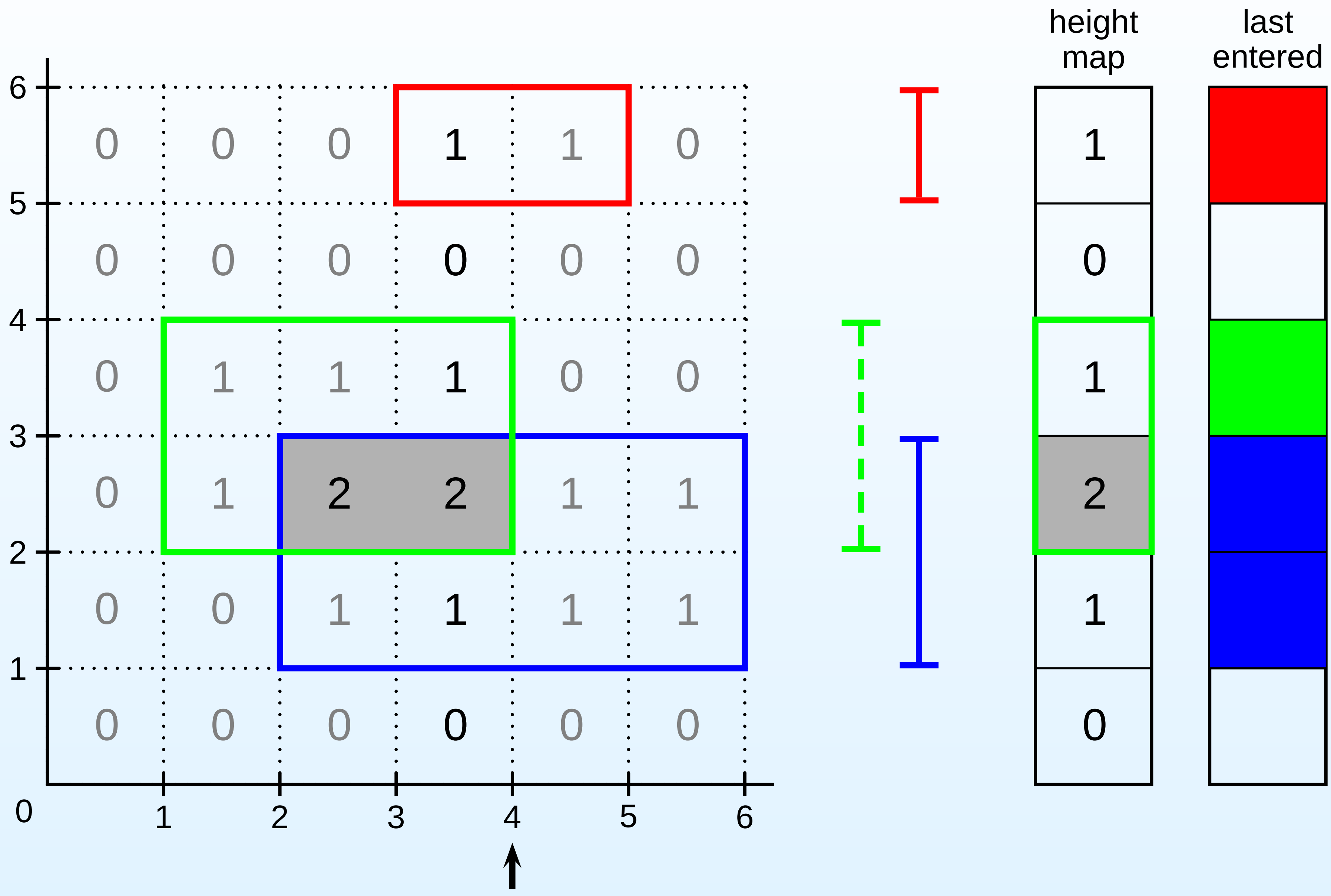
Find local maxima by sweeping through the height map



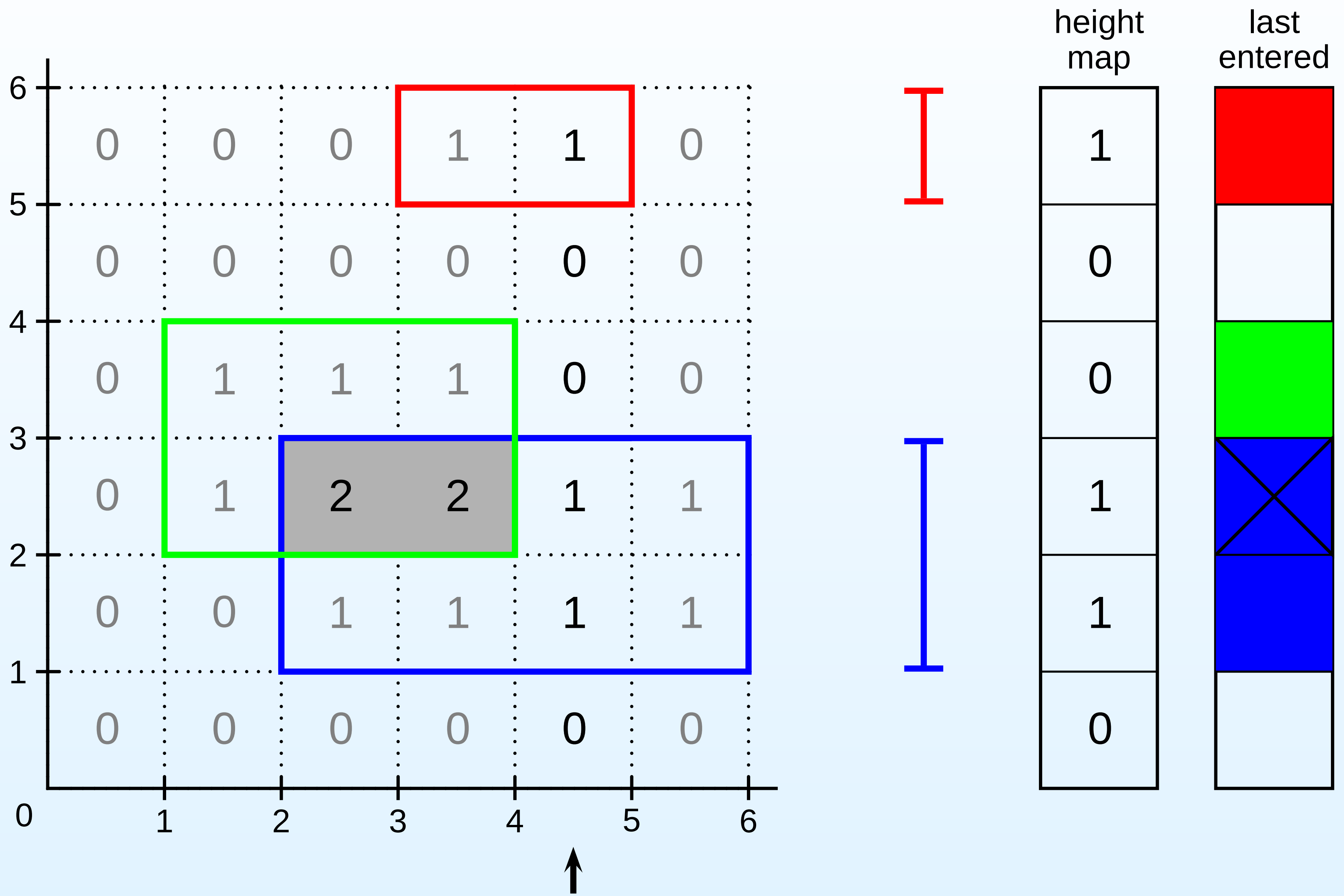
Find local maxima by sweeping through the height map



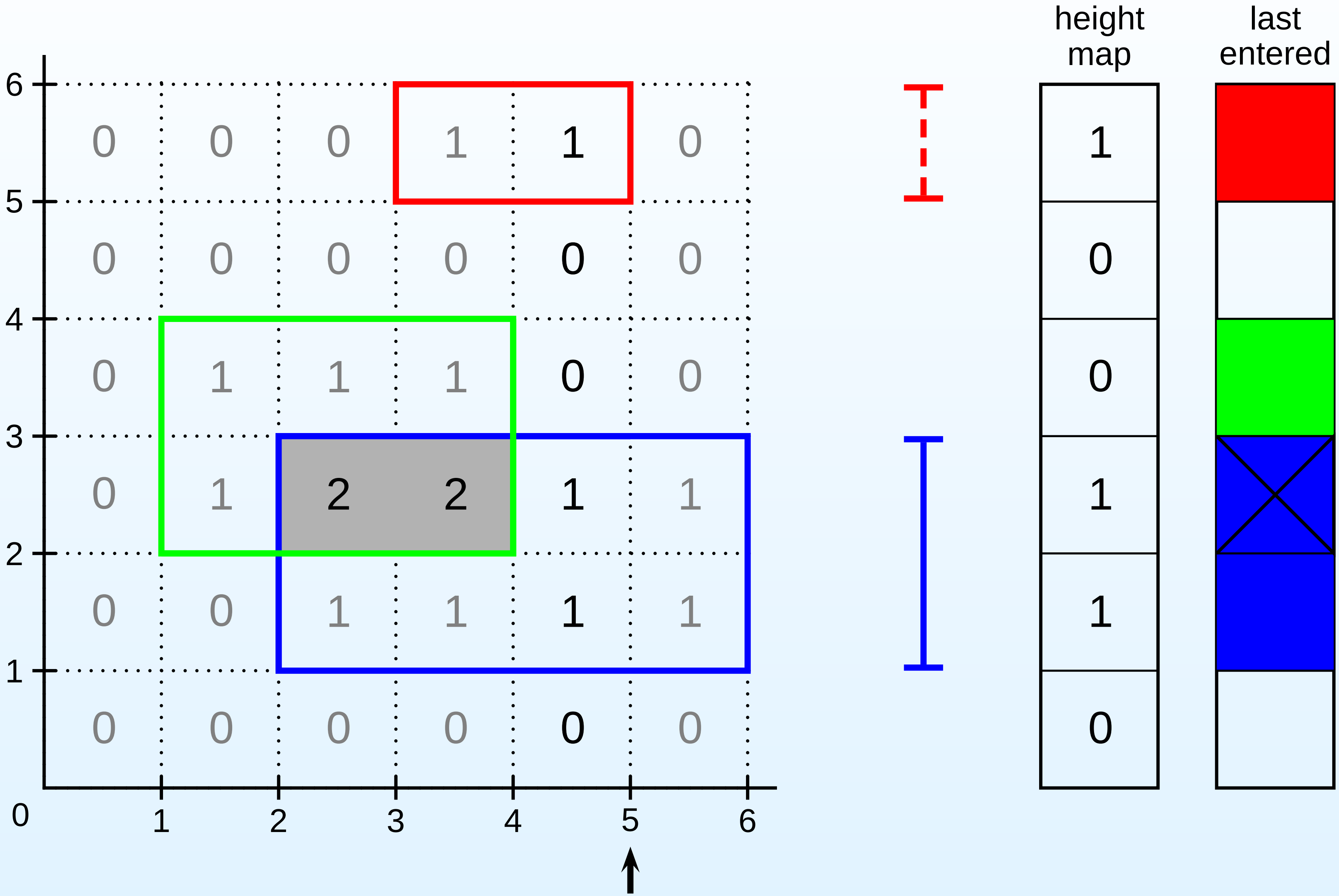
Find local maxima by sweeping through the height map



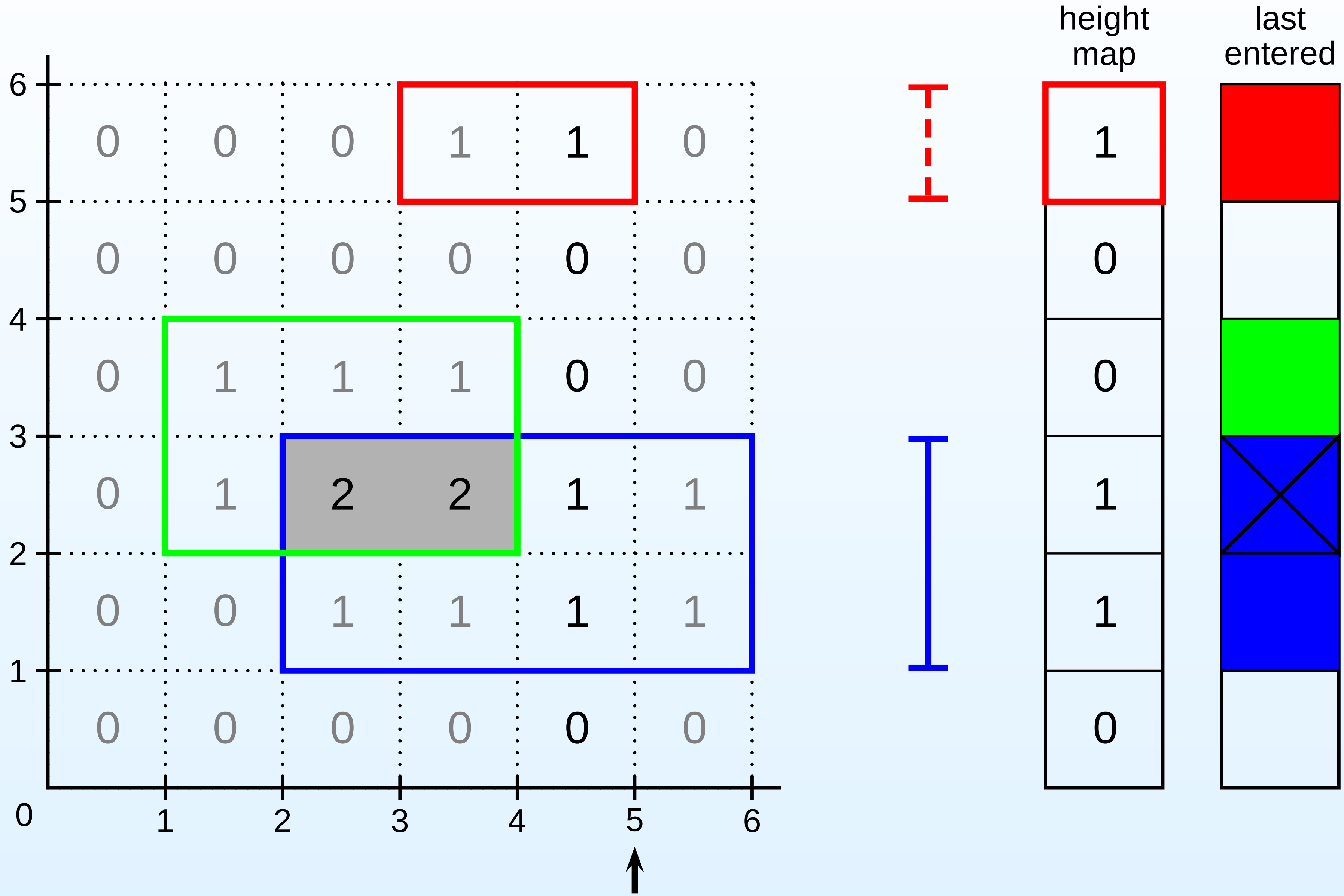
Find local maxima by sweeping through the height map



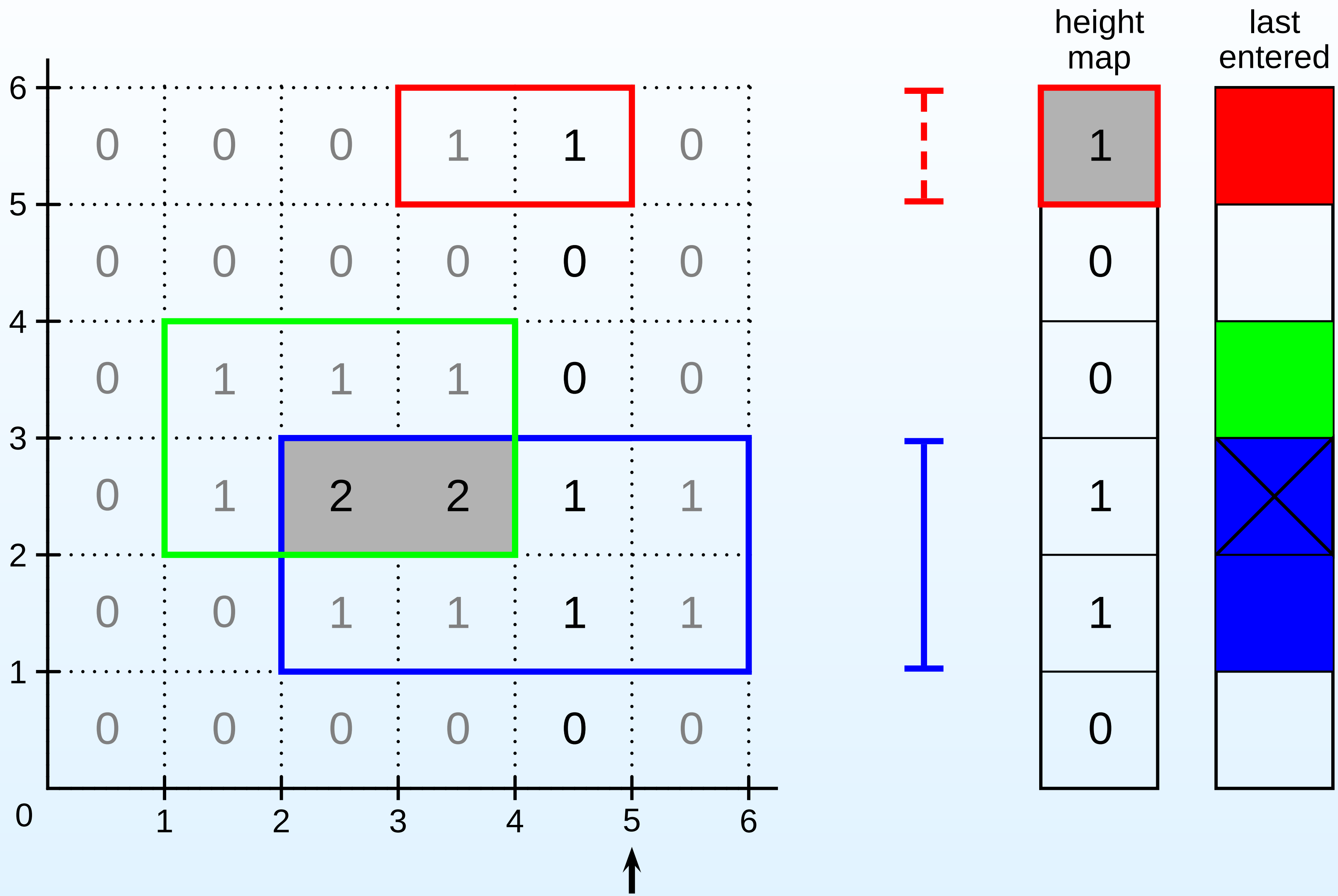
Find local maxima by sweeping through the height map



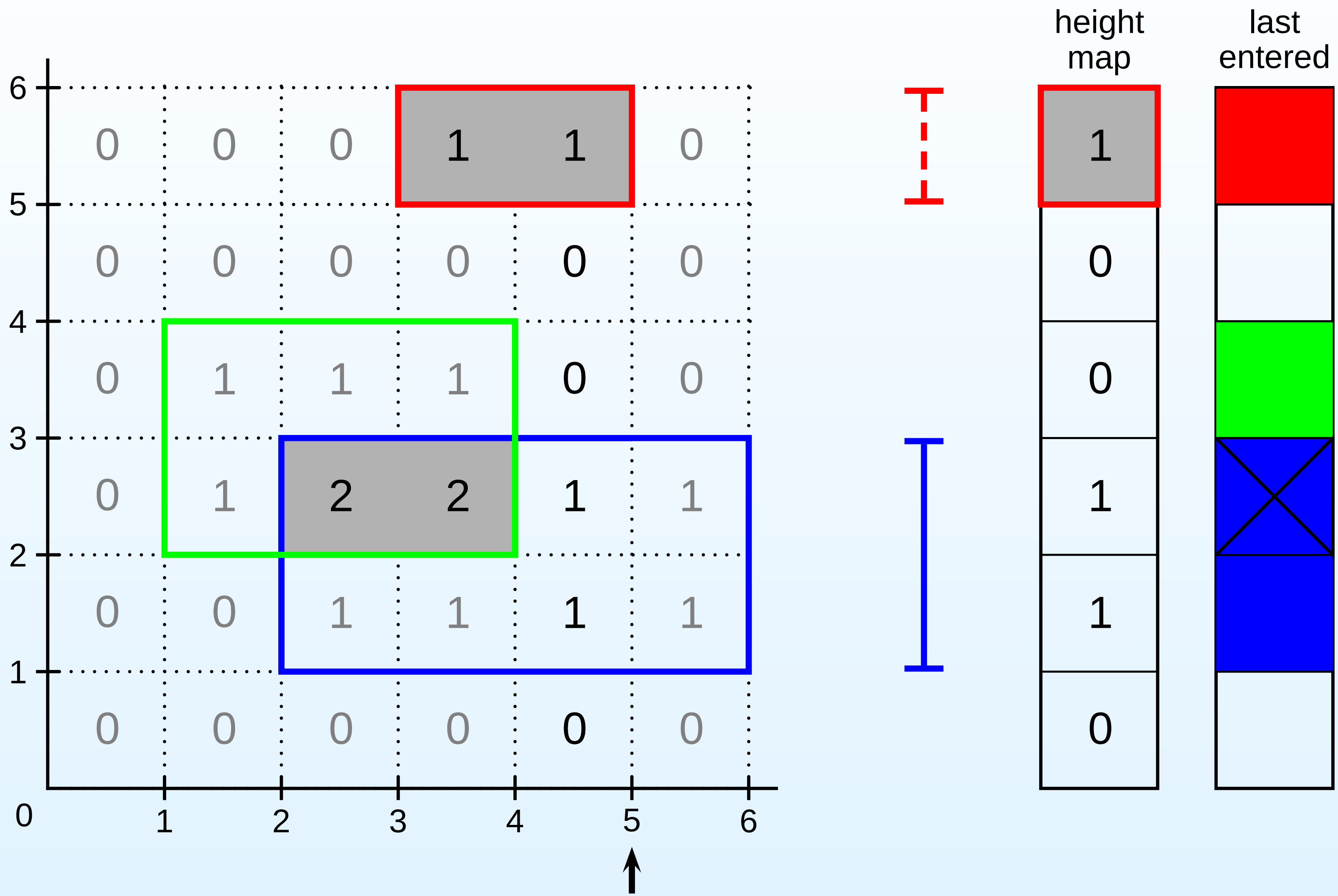
Find local maxima by sweeping through the height map



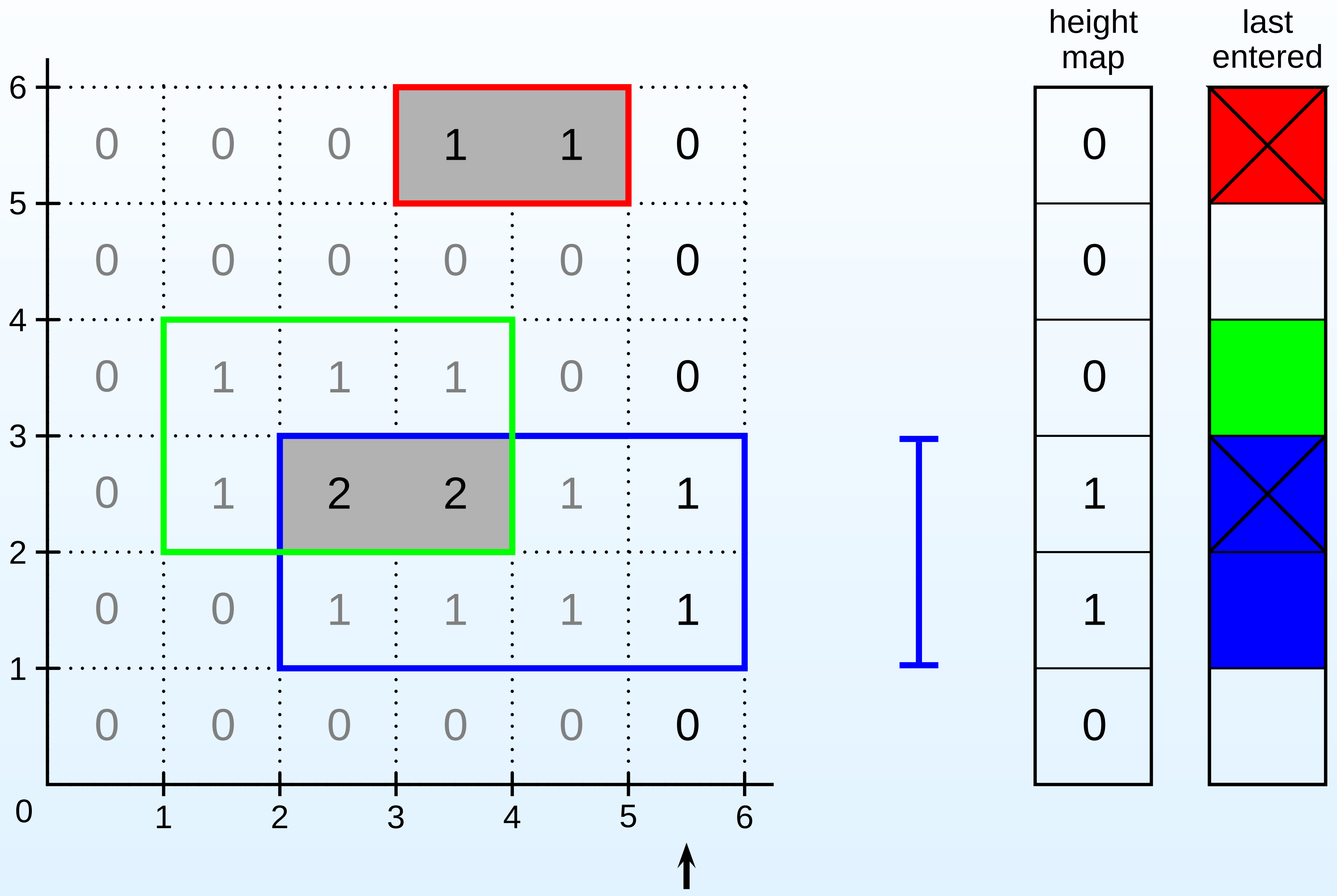
Find local maxima by sweeping through the height map



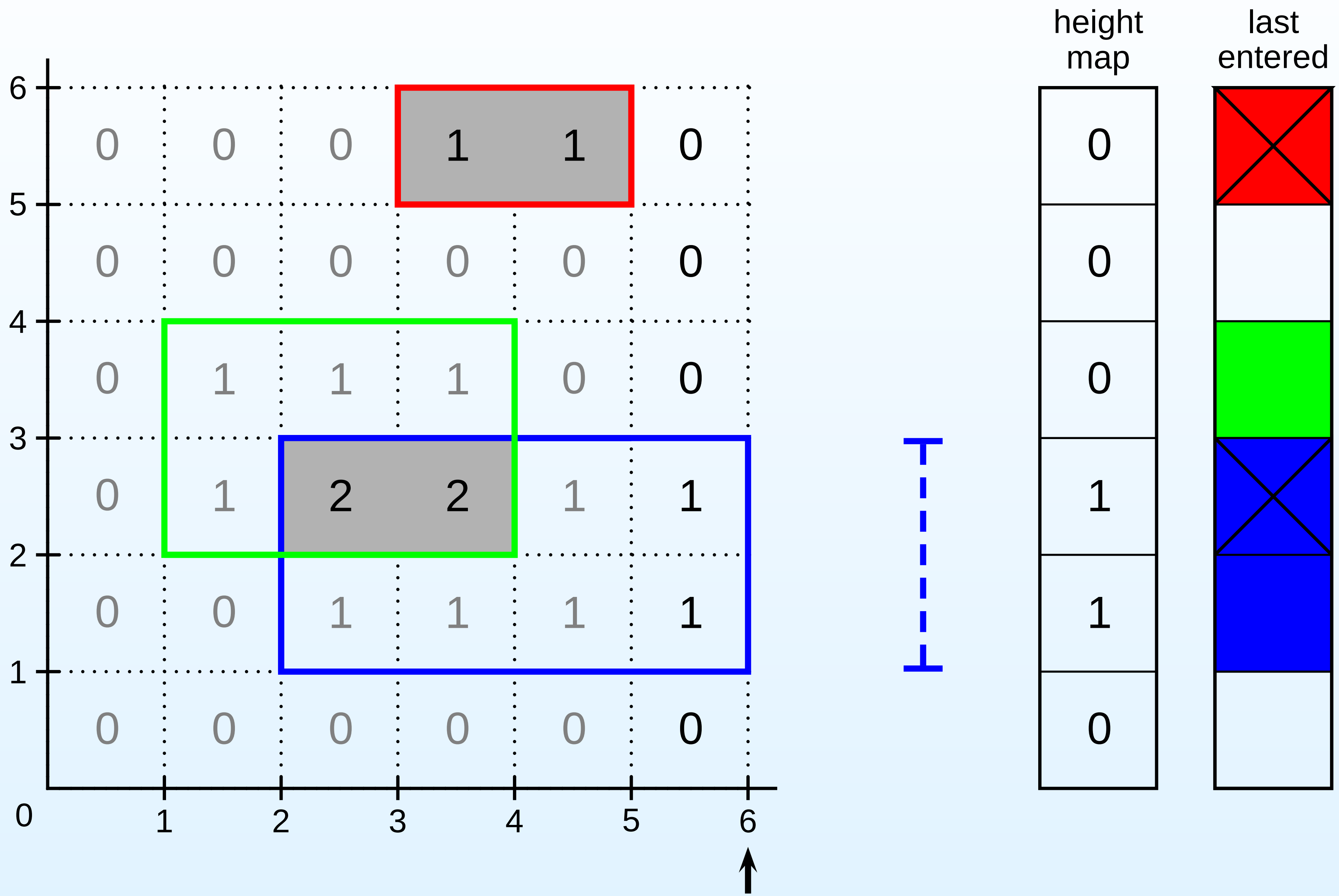
Find local maxima by sweeping through the height map



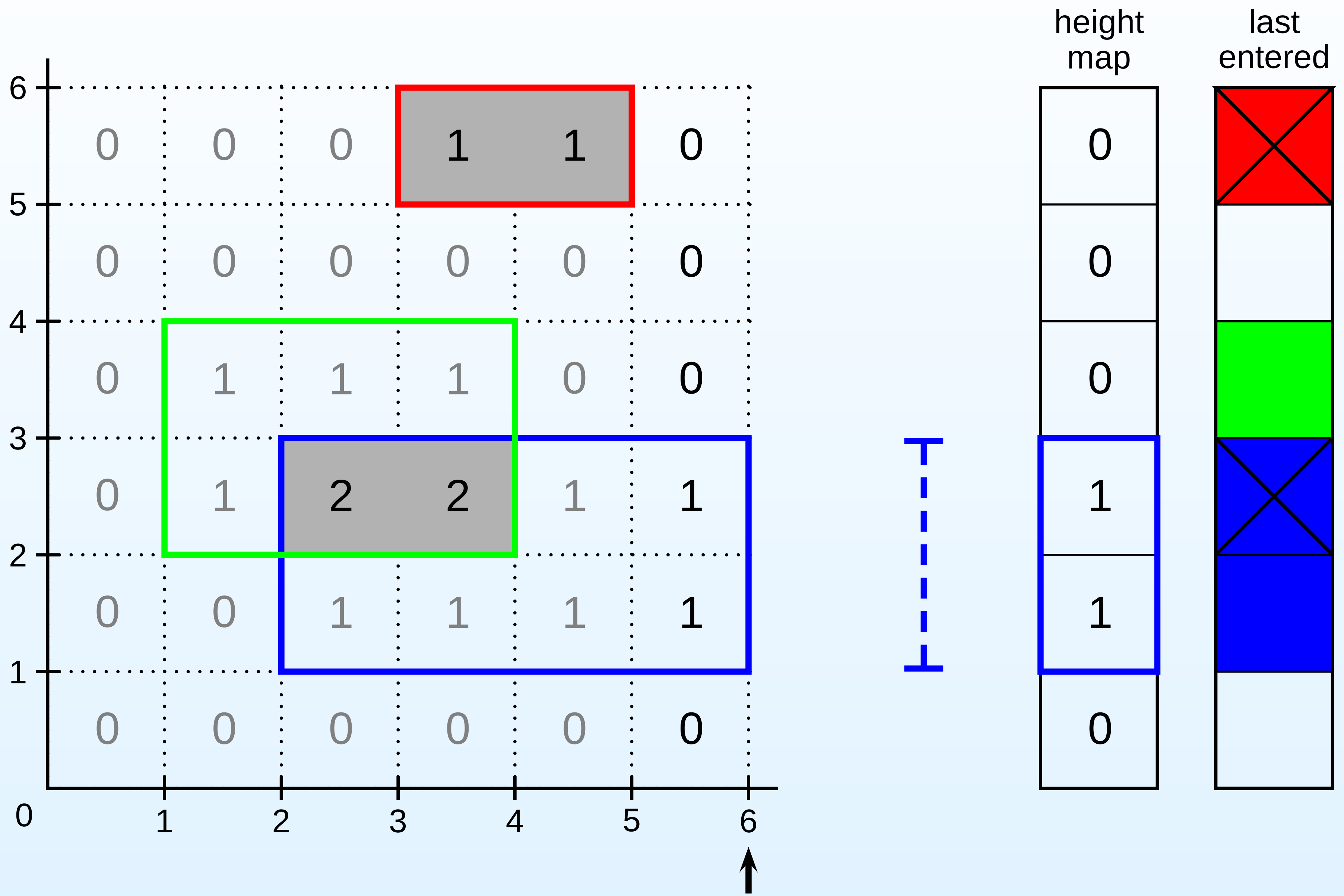
Find local maxima by sweeping through the height map



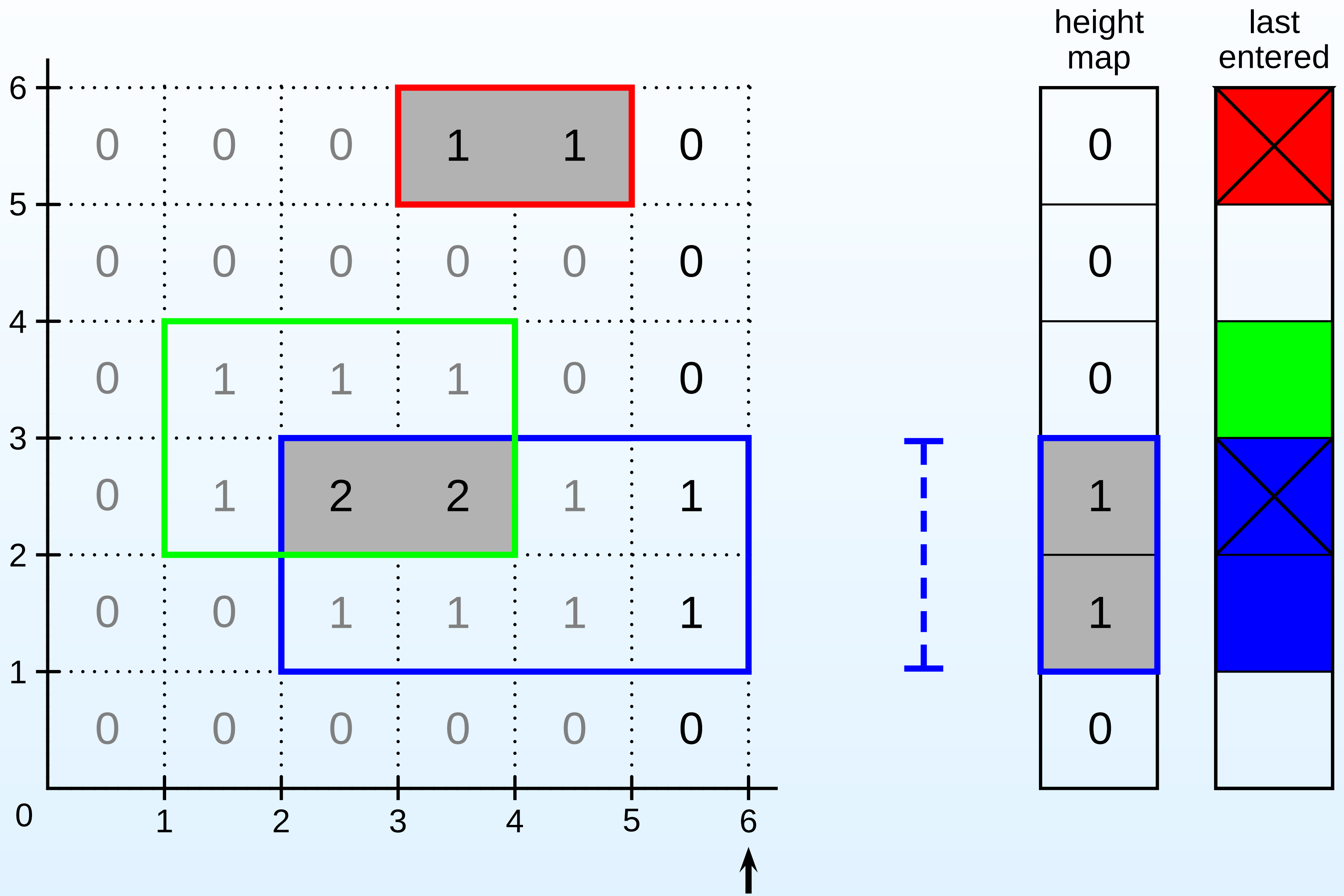
Find local maxima by sweeping through the height map



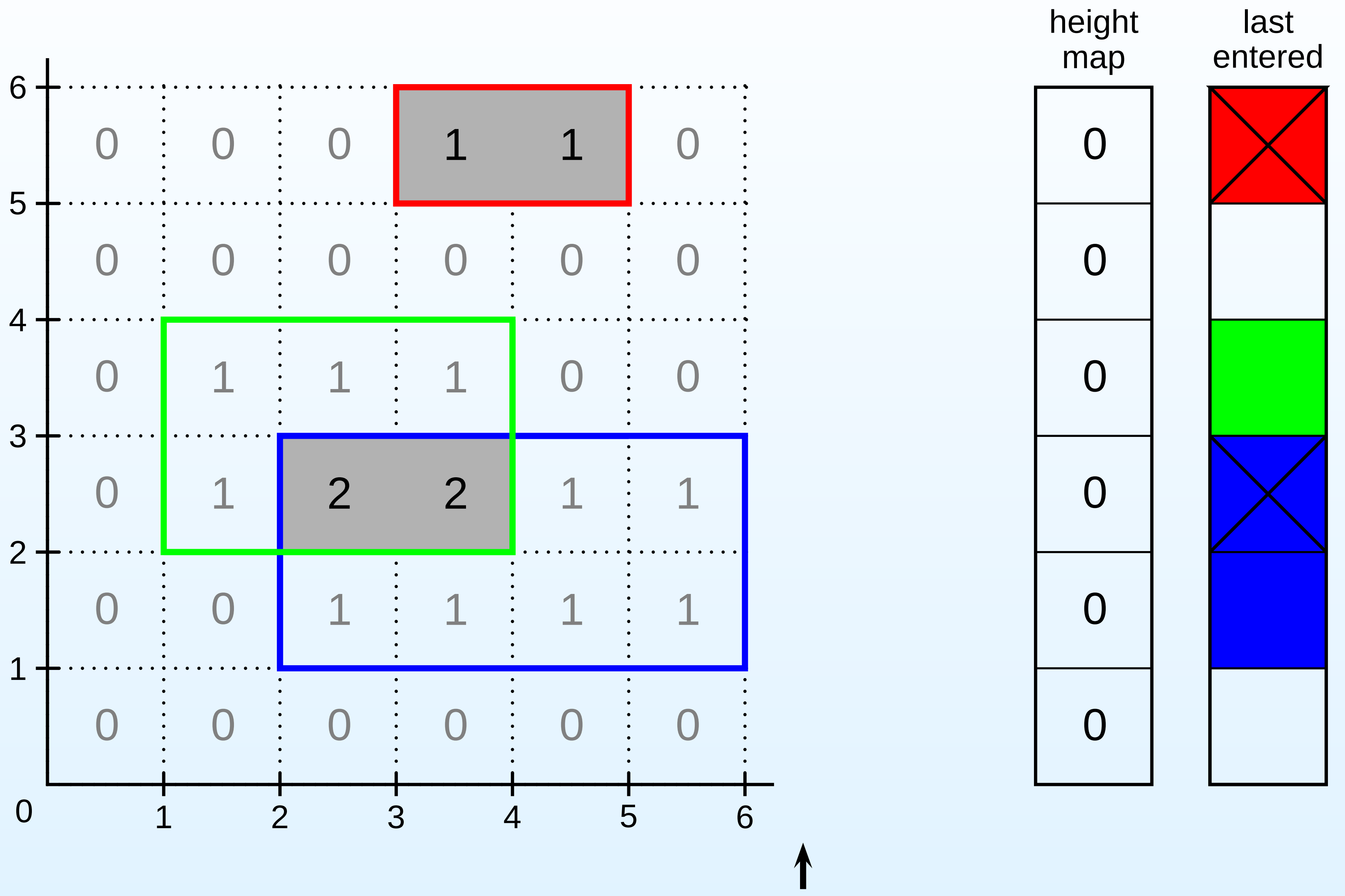
Find local maxima by sweeping through the height map



Find local maxima by sweeping through the height map



Find local maxima by sweeping through the height map



Time and space complexity of the algorithm

For bivariate interval censored data:

- time complexity: $O(n^2)$
- space complexity:
  - computation: $O(n)$
  - output: $O(n^2)$

Time and space complexity of the algorithm

For bivariate interval censored data:

- time complexity: $O(n^2)$
- space complexity:
  - computation: $O(n)$
  - output: $O(n^2)$

For d-dimensional interval censored data:

- time complexity: $O(n^d)$
- space complexity:
  - computation: $O(n^{d-1})$
  - output: $O(n^d)$

Simulation study

Bivariate current status data from a simple exponential model:

- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

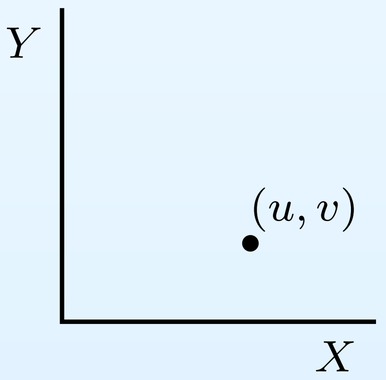
Simulation study

Bivariate current status data from a simple exponential model:

- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

Observation rectangles:



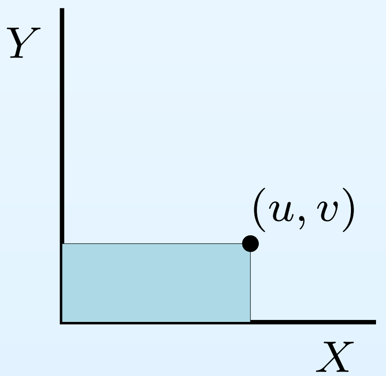
Simulation study

Bivariate current status data from a simple exponential model:

- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

Observation rectangles:



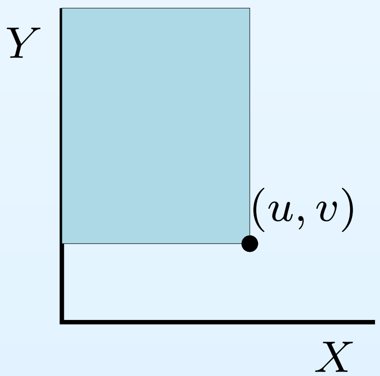
Simulation study

Bivariate current status data from a simple exponential model:

- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

Observation rectangles:



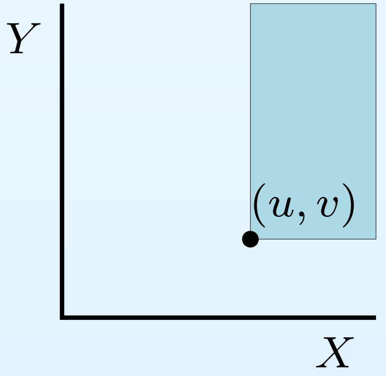
Simulation study

Bivariate current status data from a simple exponential model:

- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

Observation rectangles:



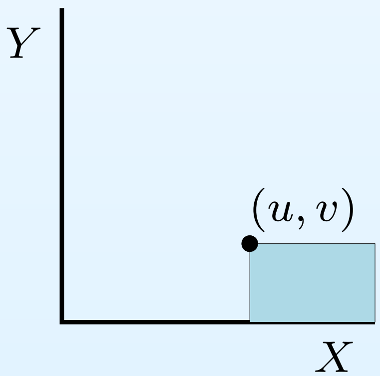
Simulation study

Bivariate current status data from a simple exponential model:

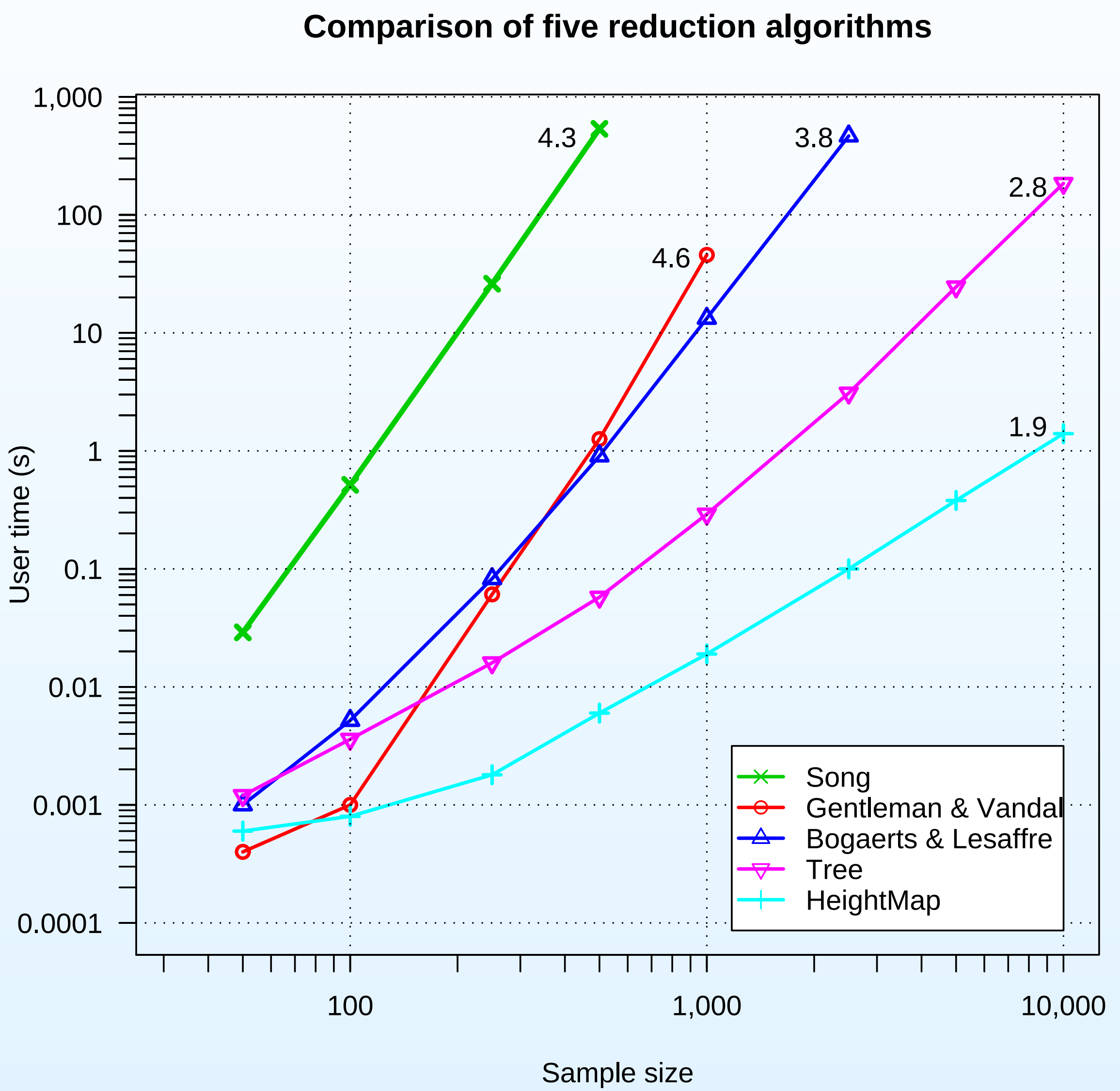
- Variables of interest: $X, Y \sim \text{Exp}(1)$
- Observation times: $U, V \sim \text{Exp}(1)$
- X, Y, U, V mutually independent
- 50 simulations for sample sizes

$$50 \quad 100 \quad 250 \quad 500 \quad 1{,}000 \quad 2{,}500 \quad 5{,}000 \quad 10{,}000$$

Observation rectangles:

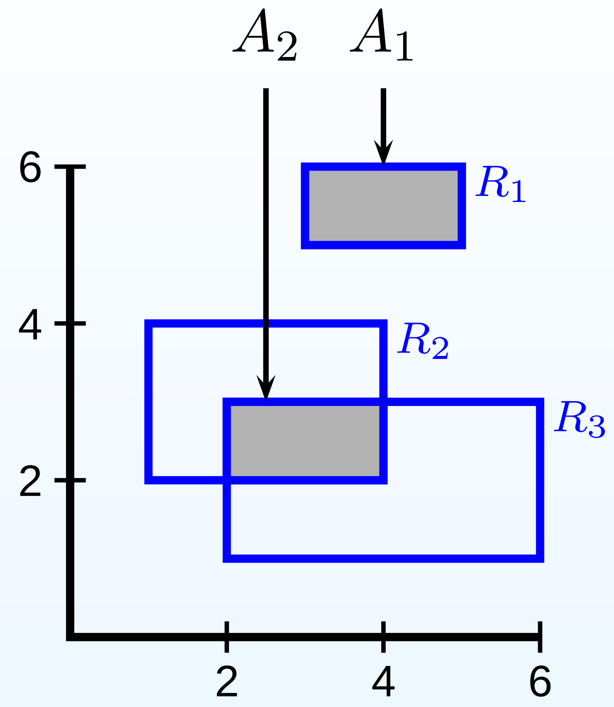


Simulation study



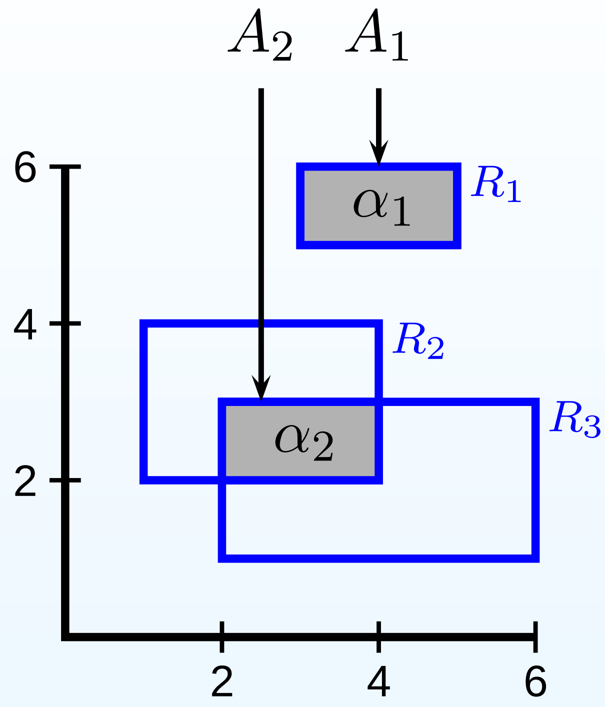
Computing the MLE: optimization step

$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$



Computing the MLE: optimization step

$\max_{F\in\mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$

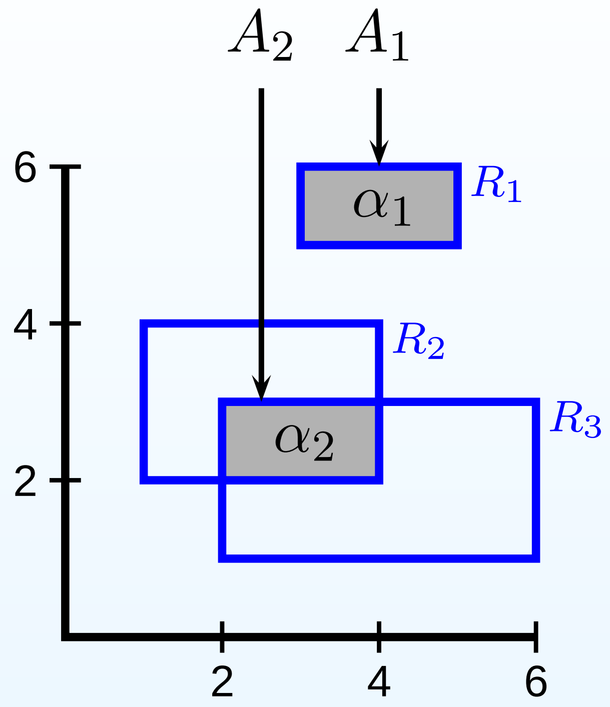


Computing the MLE: optimization step

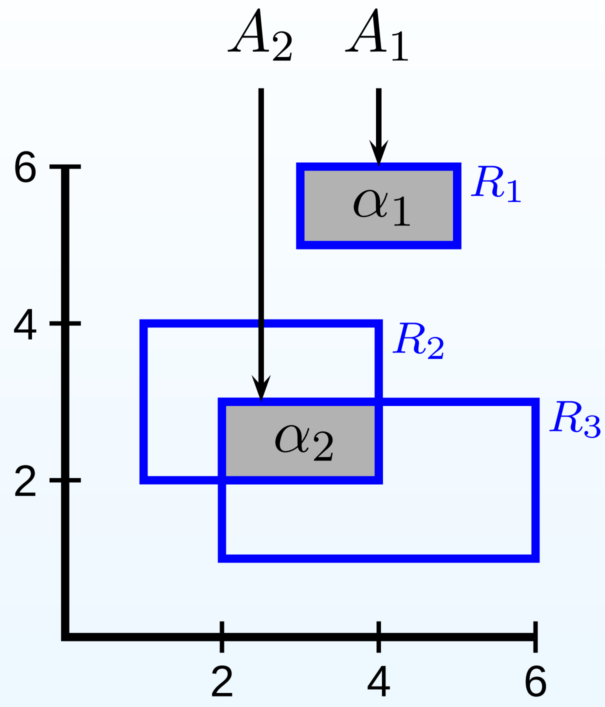
$\max_{F \in \mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$

$\max_{\alpha \in \mathcal{A}} \sum_{i=1}^{n} \log(\sum_{j=1}^{m} \alpha_j 1\{A_j \subseteq R_i\})$

where $\mathcal{A} = \{\alpha \in \mathbb{R}^m : \alpha_j \geq 0$ and $\sum_{j=1}^{m} \alpha_j = 1\}$



Computing the MLE: optimization step



$\max_{F\in\mathcal{F}} \sum_{i=1}^n \log(P_F(R_i))$

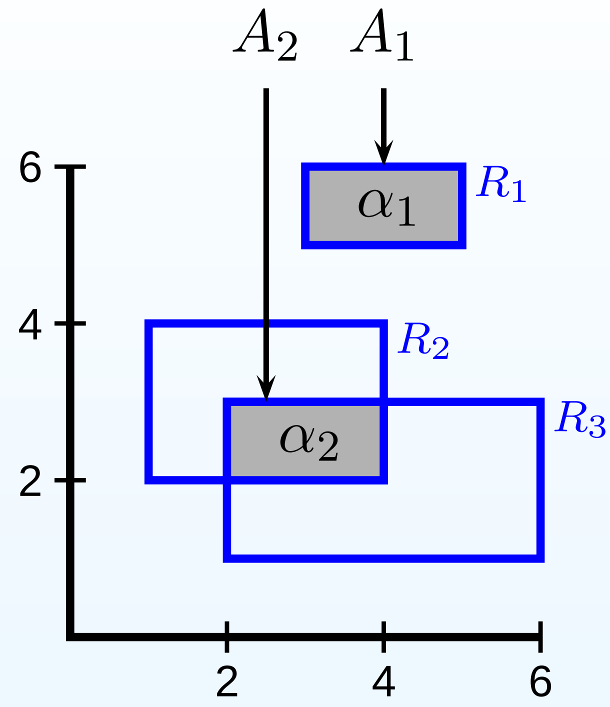
$\max_{\alpha\in\mathcal{A}} \sum_{i=1}^n \log(\sum_{j=1}^m \alpha_j 1\{A_j \subseteq R_i\})$

where $\mathcal{A} = \{\alpha \in \mathbb{R}^m : \alpha_j \geq 0$ and $\sum_{j=1}^m \alpha_j = 1\}$

$\max_{\alpha\in\mathcal{A}} \sum_{i=1}^n \log(C^T\alpha)_i$

where C is the $m \times n$ clique matrix, $C_{ji} = 1\{A_j \subseteq R_i\}$

Computing the MLE: optimization step



$\max_{F\in\mathcal{F}} \sum_{i=1}^{n} \log(P_F(R_i))$

$\max_{\alpha\in\mathcal{A}} \sum_{i=1}^{n} \log(\sum_{j=1}^{m} \alpha_j 1\{A_j \subseteq R_i\})$

where $\mathcal{A} = \{\alpha \in \mathbb{R}^m : \alpha_j \geq 0$ and $\sum_{j=1}^{m} \alpha_j = 1\}$

$\max_{\alpha\in\mathcal{A}} \sum_{i=1}^{n} \log(C^T\alpha)_i$

where C is the $m \times n$ clique matrix, $C_{ji} = 1\{A_j \subseteq R_i\}$

$\min_{\alpha\in\mathcal{A}^*} \left[-\frac{1}{n}\sum_{i=1}^{n} \log(C^T\alpha)_i + \sum_{j=1}^{m} \alpha_j\right] = \min_{\alpha\in\mathcal{A}^*} \phi(\alpha)$

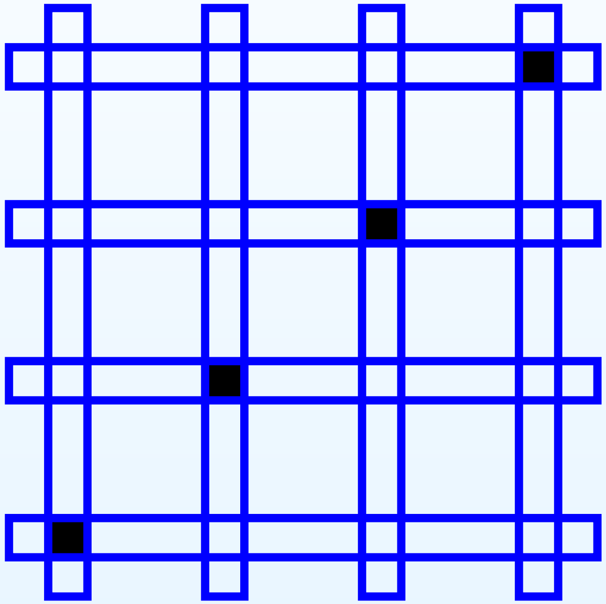
where $\mathcal{A}^* = \{\alpha \in \mathbb{R}^m : \alpha_j \geq 0\}$

Necessary and sufficient conditions for the MLE

- There is not always a unique solution $\hat{\alpha}$ to this optimization problem

Necessary and sufficient conditions for the MLE

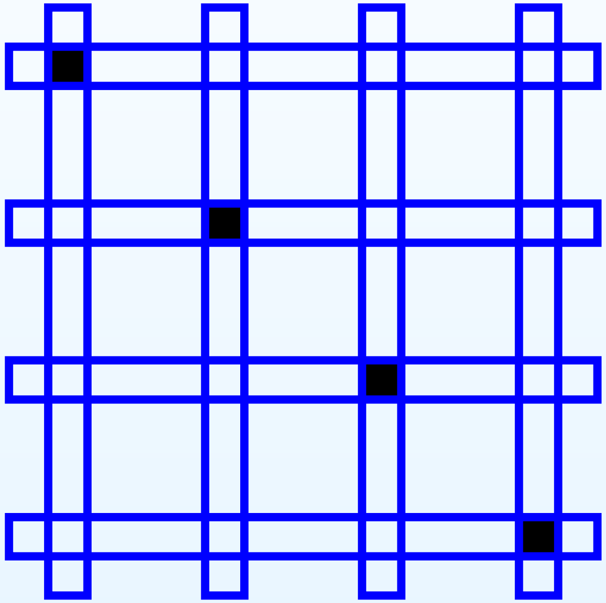
- There is not always a unique solution $\hat{\alpha}$ to this optimization problem



mass $1/4$

Necessary and sufficient conditions for the MLE

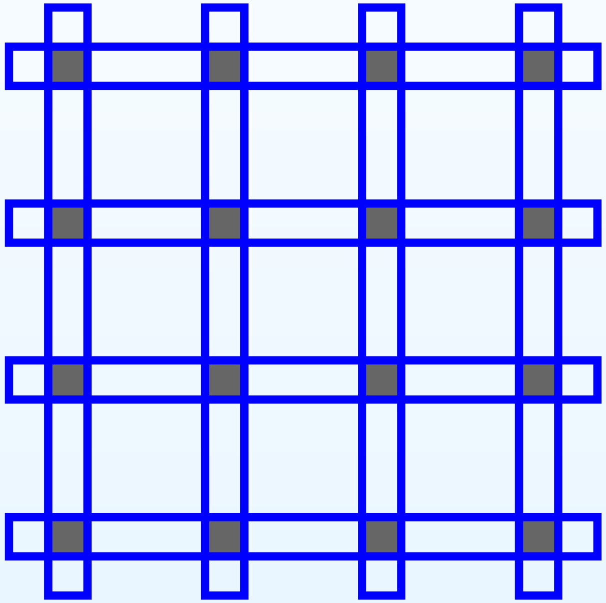
- There is not always a unique solution $\hat{\alpha}$ to this optimization problem



mass $1/4$

Necessary and sufficient conditions for the MLE

- There is not always a unique solution $\hat{\alpha}$ to this optimization problem



mass $1/16$

Necessary and sufficient conditions for the MLE

- There is not always a unique solution $\hat{\alpha}$ to this optimization problem
- There is no explicit formula available for $\hat{\alpha}$

Necessary and sufficient conditions for the MLE

- There is not always a unique solution $\hat{\alpha}$ to this optimization problem
- There is no explicit formula available for $\hat{\alpha}$
- $\hat{\alpha} \in \mathcal{A}^*$ is an MLE iff the following conditions are satisfied:

$$\frac{\partial \phi(\hat{\alpha})}{\partial \alpha_j} \begin{cases} \geq 0 & \text{for all } j = 1, \ldots, m \\ = 0 & \text{if } \hat{\alpha}_j > 0 \end{cases} \qquad (*)$$

Necessary and sufficient conditions for the MLE

- There is not always a unique solution $\hat{\alpha}$ to this optimization problem
- There is no explicit formula available for $\hat{\alpha}$
- $\hat{\alpha} \in \mathcal{A}^*$ is an MLE iff the following conditions are satisfied:

$$\frac{\partial \phi(\hat{\alpha})}{\partial \alpha_j} \begin{cases} \geq 0 & \text{for all } j = 1, \ldots, m \\ = 0 & \text{if } \hat{\alpha}_j > 0 \end{cases} \qquad (*)$$

- We compute $\hat{\alpha}$ with an iterative algorithm, and stop if $(*)$ is satisfied within some tolerance ϵ, e.g. $\epsilon = 10^{-10}$

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$
- While the necessary and sufficient conditions $(*)$ are not satisfied, do:

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$
- While the necessary and sufficient conditions $(*)$ are not satisfied, do:
  - Let $\tilde{\phi}^{(k)}$ be the quadratic approximation of ϕ around $\alpha^{(k)}$

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$
- While the necessary and sufficient conditions $(*)$ are not satisfied, do:
  - Let $\tilde{\phi}^{(k)}$ be the quadratic approximation of ϕ around $\alpha^{(k)}$
  - Compute $\tilde{\alpha}^{(k)} \approx \text{argmin}\, \tilde{\phi}^{(k)}$, using the support reduction algorithm

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$
- While the necessary and sufficient conditions $(*)$ are not satisfied, do:
  - Let $\tilde{\phi}^{(k)}$ be the quadratic approximation of ϕ around $\alpha^{(k)}$
  - Compute $\tilde{\alpha}^{(k)} \approx \text{argmin}\, \tilde{\phi}^{(k)}$, using the support reduction algorithm
  - Let $\alpha^{(k+1)} = \lambda\alpha^{(k)} + (1-\lambda)\tilde{\alpha}^{(k)}$, where λ is determined by Armijo's rule

Iterative algorithm for optimization step

We use a combination of Sequential Quadratic Programming, and the Support Reduction Algorithm of Groeneboom, Jongbloed and Wellner (2007):

- Let $k = 1$ and take a starting value $\alpha^{(1)}$
- While the necessary and sufficient conditions $(*)$ are not satisfied, do:
  - Let $\tilde{\phi}^{(k)}$ be the quadratic approximation of ϕ around $\alpha^{(k)}$
  - Compute $\tilde{\alpha}^{(k)} \approx \text{argmin}\, \tilde{\phi}^{(k)}$, using the support reduction algorithm
  - Let $\alpha^{(k+1)} = \lambda\alpha^{(k)} + (1-\lambda)\tilde{\alpha}^{(k)}$, where λ is determined by Armijo's rule
  - Let $k = k + 1$

R-packages for interval censored data

- Existing packages:
  - `Icens` (Gentleman and Vandal):
    several functions for univariate/bivariate interval censored data
  - `bicreduc` (Maathuis):
    height map algorithm for the reduction step
  - `intcox` (Henschel, Heiss and Mansmann)
    fitting Cox model for univariate interval censored data

R-packages for interval censored data

- Existing packages:
  - `Icens` (Gentleman and Vandal):
    several functions for univariate/bivariate interval censored data
  - `bicreduc` (Maathuis):
    height map algorithm for the reduction step
  - `intcox` (Henschel, Heiss and Mansmann)
    fitting Cox model for univariate interval censored data
- New package `MLEcens`
  - Includes package `bicreduc` (which is no longer maintained)
  - Some overlap with `Icens`, but `MLEcens` has:
    - Very fast reduction step
    - Fast and stable optimization step
    - Various new plotting functions

Structure of R-package

- Algorithm written in C, with wrappers in R

Structure of R-package

- Algorithm written in C, with wrappers in R
- Optimization step requires some linear algebra routines (e.g. solve linear system)

Structure of R-package

- Algorithm written in C, with wrappers in R
- Optimization step requires some linear algebra routines (e.g. solve linear system)
  - We first used Numerical Recipes to do this, but this is not open source

Structure of R-package

- Algorithm written in C, with wrappers in R
- Optimization step requires some linear algebra routines (e.g. solve linear system)
  - We first used Numerical Recipes to do this, but this is not open source
  - We considered using GNU Scientific Library. This is open source, but not everybody has it

Structure of R-package

- Algorithm written in C, with wrappers in R
- Optimization step requires some linear algebra routines (e.g. solve linear system)
  - We first used Numerical Recipes to do this, but this is not open source
  - We considered using GNU Scientific Library. This is open source, but not everybody has it
  - We finally chose to use the open source BLAS (Basic Linear Algebra Subprograms) and LAPACK (Linear Algebra PACKage) libraries that come with R

Overview of `MLEcens`

- Available functions:
  - Basic plot function: `plotRects`
  - Canonical rectangles: `real2canon, canon2real`
  - Reduction step: `reduc, plotHM, plotCM`
  - Computation MLE: `computeMLE`
  - Plot functions to display the MLE: `plotDens1, plotDens2, plotCDF1, plotCDF2, plotCDF3`

Overview of `MLEcens`

- Available functions:
  - Basic plot function: `plotRects`
  - Canonical rectangles: `real2canon, canon2real`
  - Reduction step: `reduc, plotHM, plotCM`
  - Computation MLE: `computeMLE`
  - Plot functions to display the MLE: `plotDens1, plotDens2, plotCDF1, plotCDF2, plotCDF3`
- Available data sets:
  - bivariate interval censored data: `actg181`
  - univariate interval censored data: `cosmesis`
  - interval censored data with competing risks: `menopause`

Overview of `MLEcens`

- Available functions:
  - Basic plot function: `plotRects`
  - Canonical rectangles: `real2canon, canon2real`
  - Reduction step: `reduc, plotHM, plotCM`
  - Computation MLE: `computeMLE`
  - Plot functions to display the MLE: `plotDens1, plotDens2, plotCDF1, plotCDF2, plotCDF3`
- Available data sets:
  - bivariate interval censored data: `actg181`
  - univariate interval censored data: `cosmesis`
  - interval censored data with competing risks: `menopause`
- Documentation and examples for all functions

Overview of `MLEcens`

- Available functions:
  - Basic plot function: `plotRects`
  - Canonical rectangles: `real2canon, canon2real`
  - Reduction step: `reduc, plotHM, plotCM`
  - Computation MLE: `computeMLE`
  - Plot functions to display the MLE: `plotDens1, plotDens2, plotCDF1, plotCDF2, plotCDF3`
- Available data sets:
  - bivariate interval censored data: `actg181`
  - univariate interval censored data: `cosmesis`
  - interval censored data with competing risks: `menopause`
- Documentation and examples for all functions
- Demonstration...

Possible extensions of R-package

- 3d-plotting functions
- Specialized algorithms for univariate interval censored data
- Extension to 3-dimensional interval censored data
- . . .

I am happy to modify the package. So please let me know if you are interested in any extensions, or if you have any other feedback/suggestions.

References

- Reduction step:
  - Maathuis (2005). “Reduction algorithm for the NPMLE for distribution function of bivariate interval censored data", *JCGS* **14** 352–362.
- Optimization step:
  - Groeneboom, Jongbloed and Wellner (2007). “The support reduction algorithm for computing nonparametric function estimates in mixture models", *Submitted*.
  - Maathuis (2003). “Nonparametric maximum likelihood estimation for bivariate censored data", Master’s Thesis, Delft University of Technology, The Netherlands.
- R-package:
  - Maathuis (2007), “R-package `MLEcens`", CRAN.

Thanks!

Presentation (including R-code), papers, and R-package
are posted on my website:

http://stat.ethz.ch/∼maathuis