

Algorithms for GIS:

Computing flow on terrains

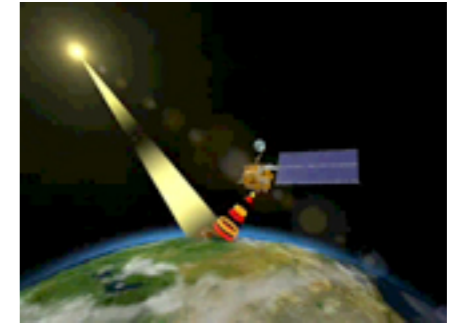
Today

- Modeling and computing flow on terrains
- Announcements:
 - Assignment 2 due today.
 - Assignment 3 posted.

Flow on terrains

- Where does the water go when it rains?
- What will happen when it rains (a lot)?
- What are the areas susceptible to flooding?
- What areas will flood first?
- River data is expensive to collect. Is it possible to model and automatically compute rivers on a terrain?
- Flow used for many other processes, e.g. sediment flow, pollutant movement

Efficiency (really) matters!



- Massive amounts of terrain data available
 - e.g. NASA SRTM, acquired 80% of Earth at 30m resolution. Total 5TB !!
 - USGS: most USA at 10m resolution
 - LIDAR data: 1m resolution

...need efficient algorithms



- Example:
 - Area if approx. 800 km x 800 km
 - Sampled at:
 - 100 resolution: 64 million points **(128MB)**
 - 30m resolution: 640 **(1.2GB)**
 - 10m resolution: 6400 = 6.4 billion **(12GB)**
 - 1m resolution: 600.4 billion **(1.2TB)**

Flow on terrains

Input: terrain model (DEM = digital elevation model)

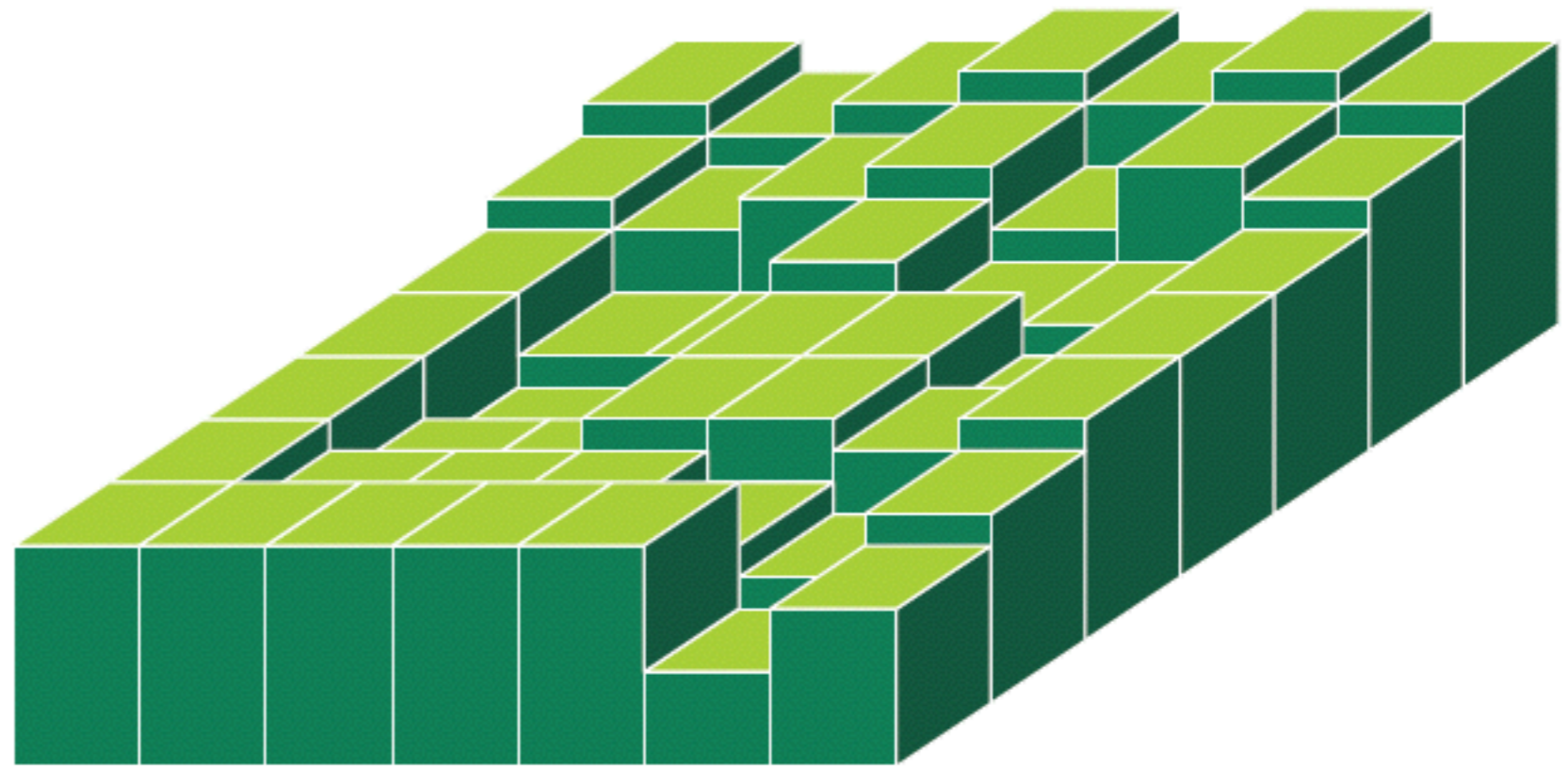
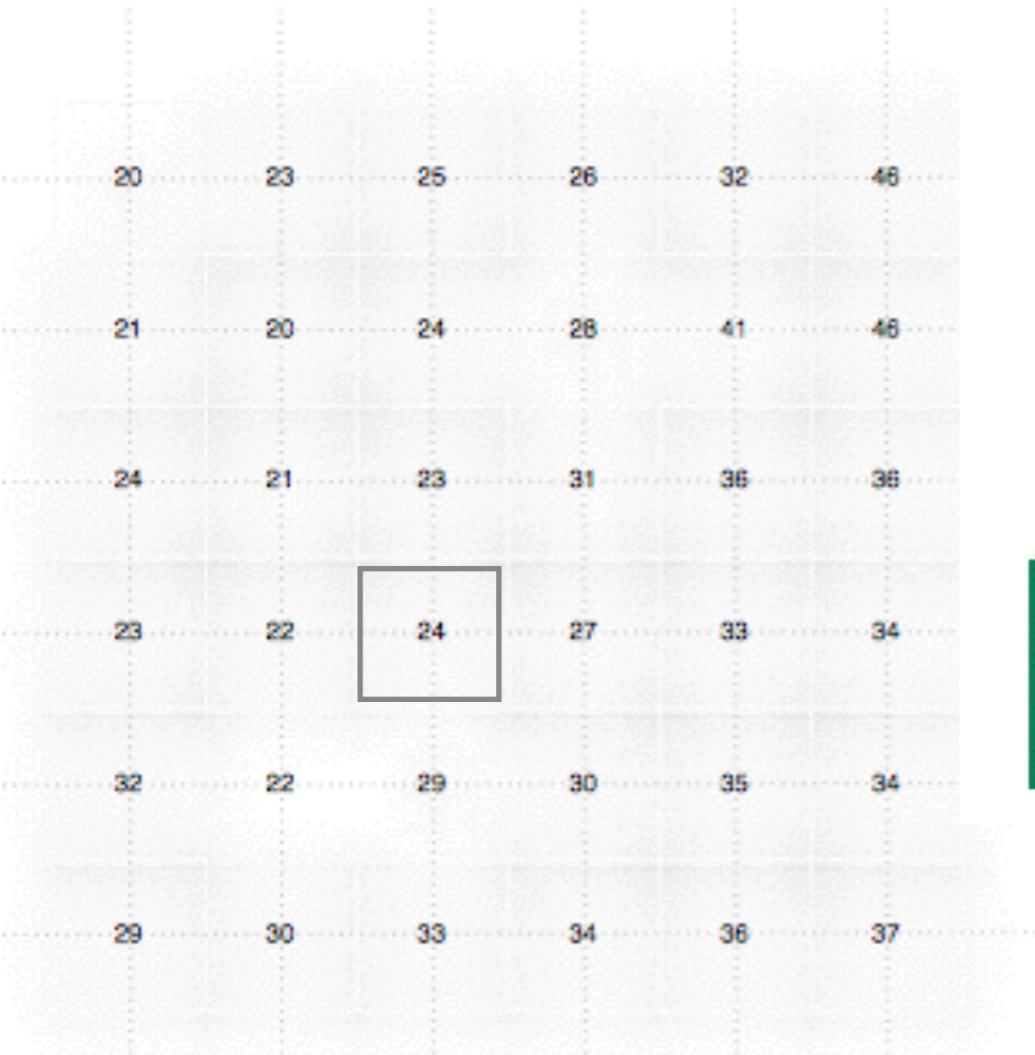
- grid DEM
- TIN (triangulation) DEM

Flow modeled by

- flow direction (FD)
 - the direction water flows at a point
- flow accumulation (FA)
 - total amount of water flowing through a point
- (Pfafsteter) river and watershed hierarchy

Flow on grid terrains

Grid terrain models

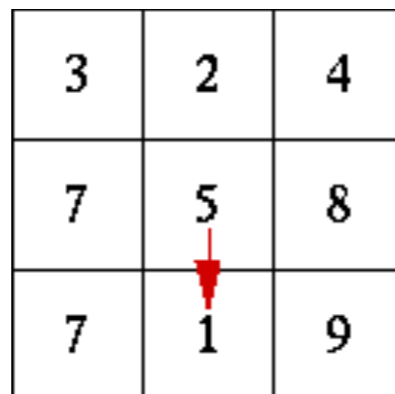


thanks!!! to H. Haverkort

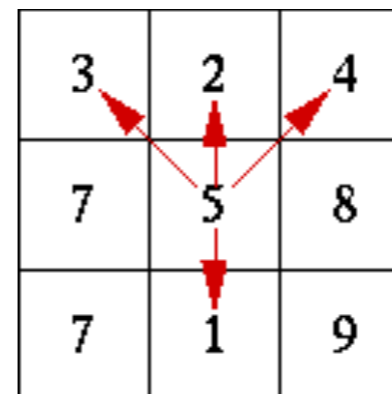
Flow direction (FD)

- $FD(p)$ = the direction water flows at p
- Generally,
 - FD is direction of gradient at p , i.e. direction of greatest decrease
 - FD can be approximated based on a neighborhood of p
- FD on grids:
 - FD = discretized to eight directions
 - SFD: Single flow direction (steepest downslope)
 - MFD: Multiple flow directions (all downslope)

3	2	4
7	5	8
7	1	9

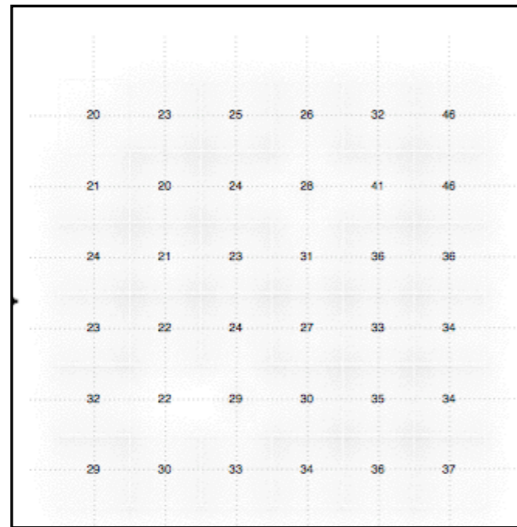


3	2	4
7	5	8
7	1	9

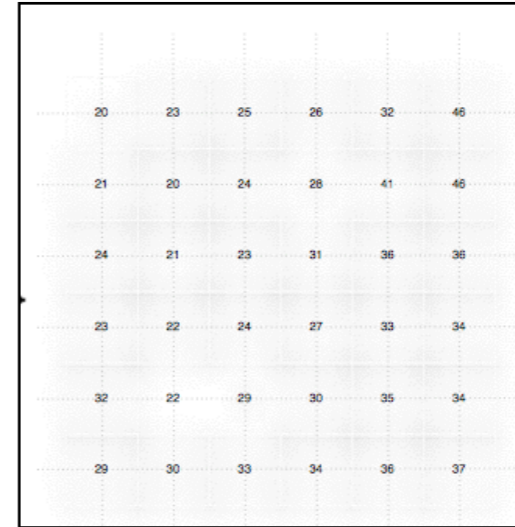


$n = \text{nb. of cells in the grid}$

Flow direction



elevation grid

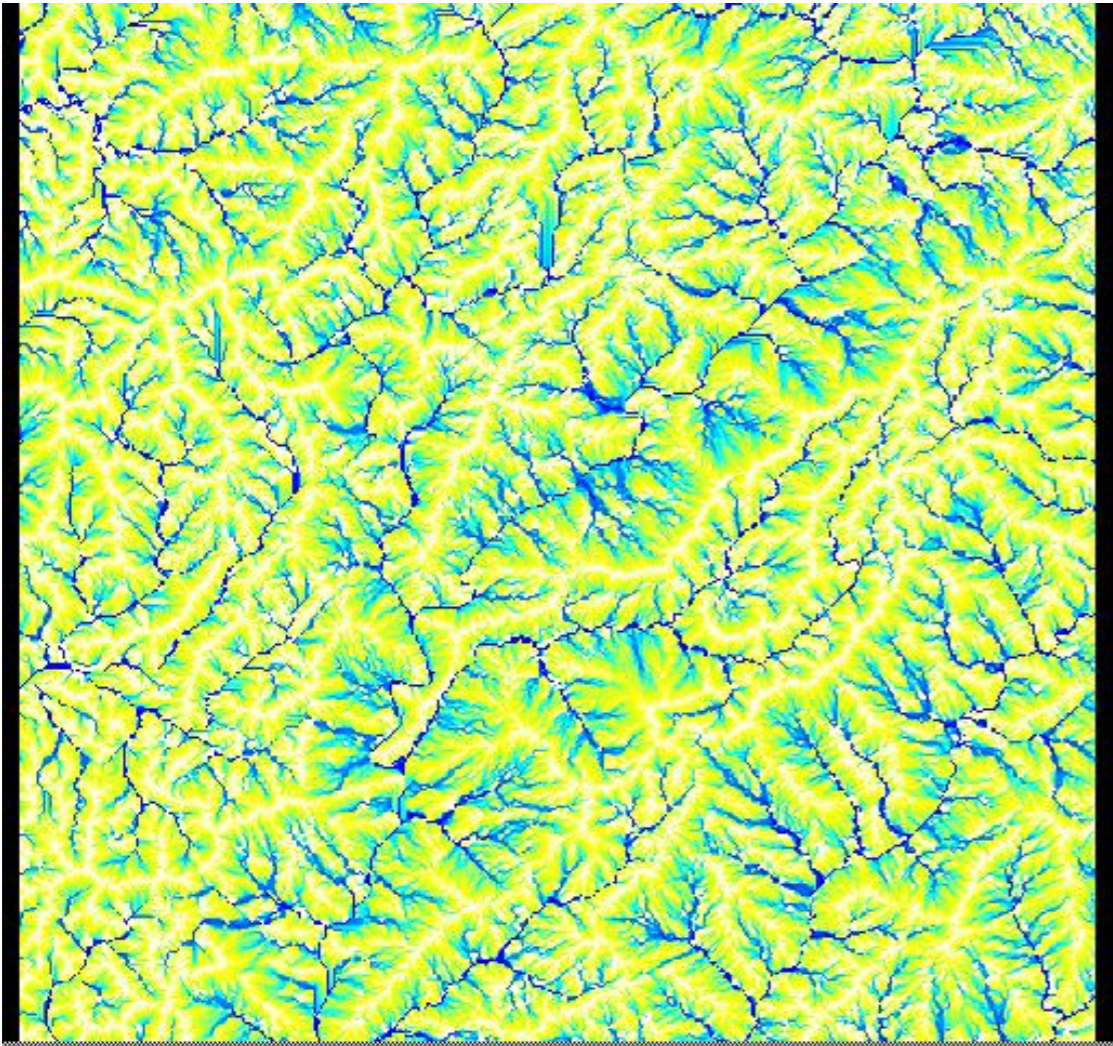


FD grid

- FD can be computed in $O(n)$ time
- Issue: flat areas... later

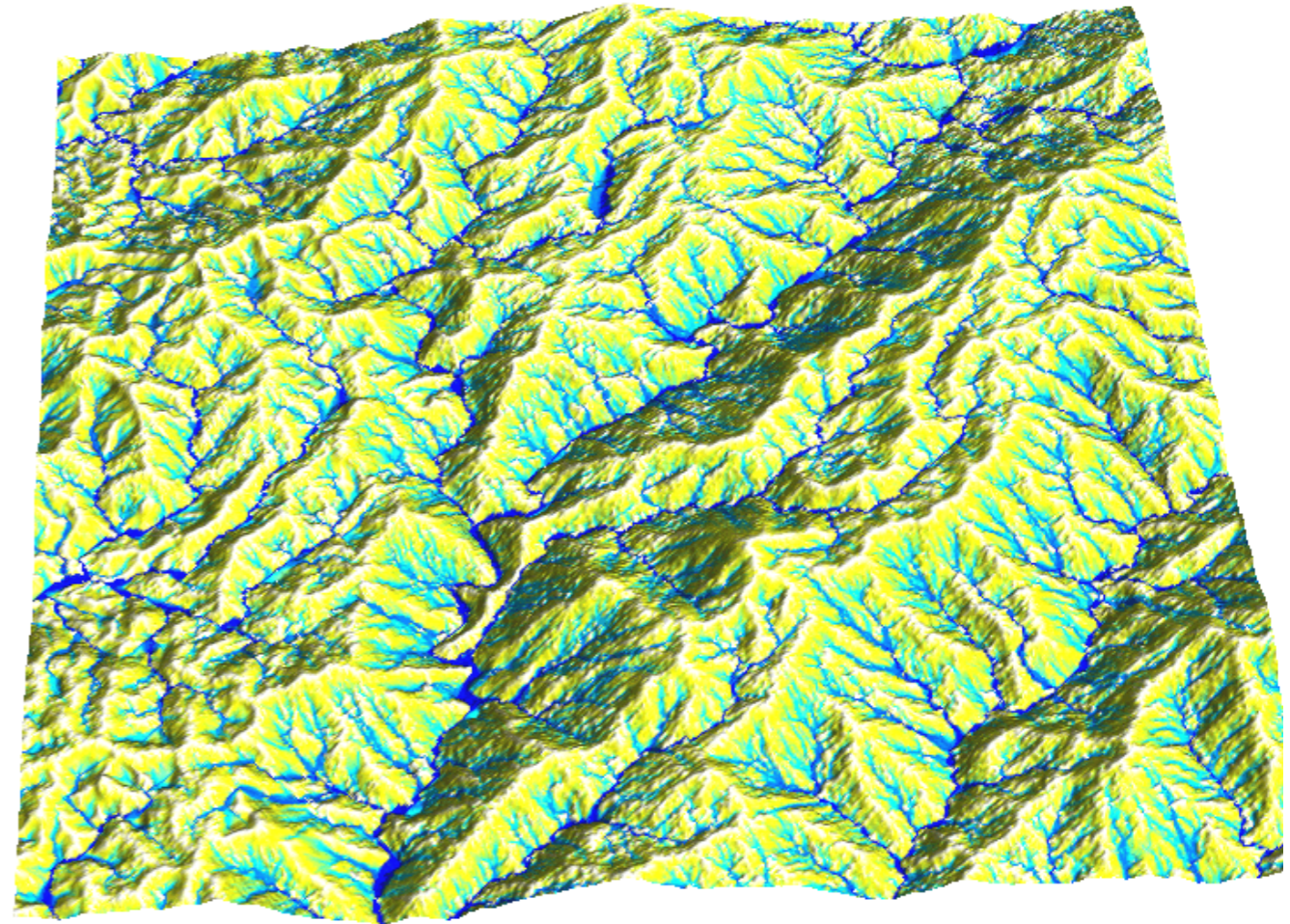
Flow accumulation

- FA models rivers



FA 2D view

- high values: blue
- medium values: light blue
- low values: yellow

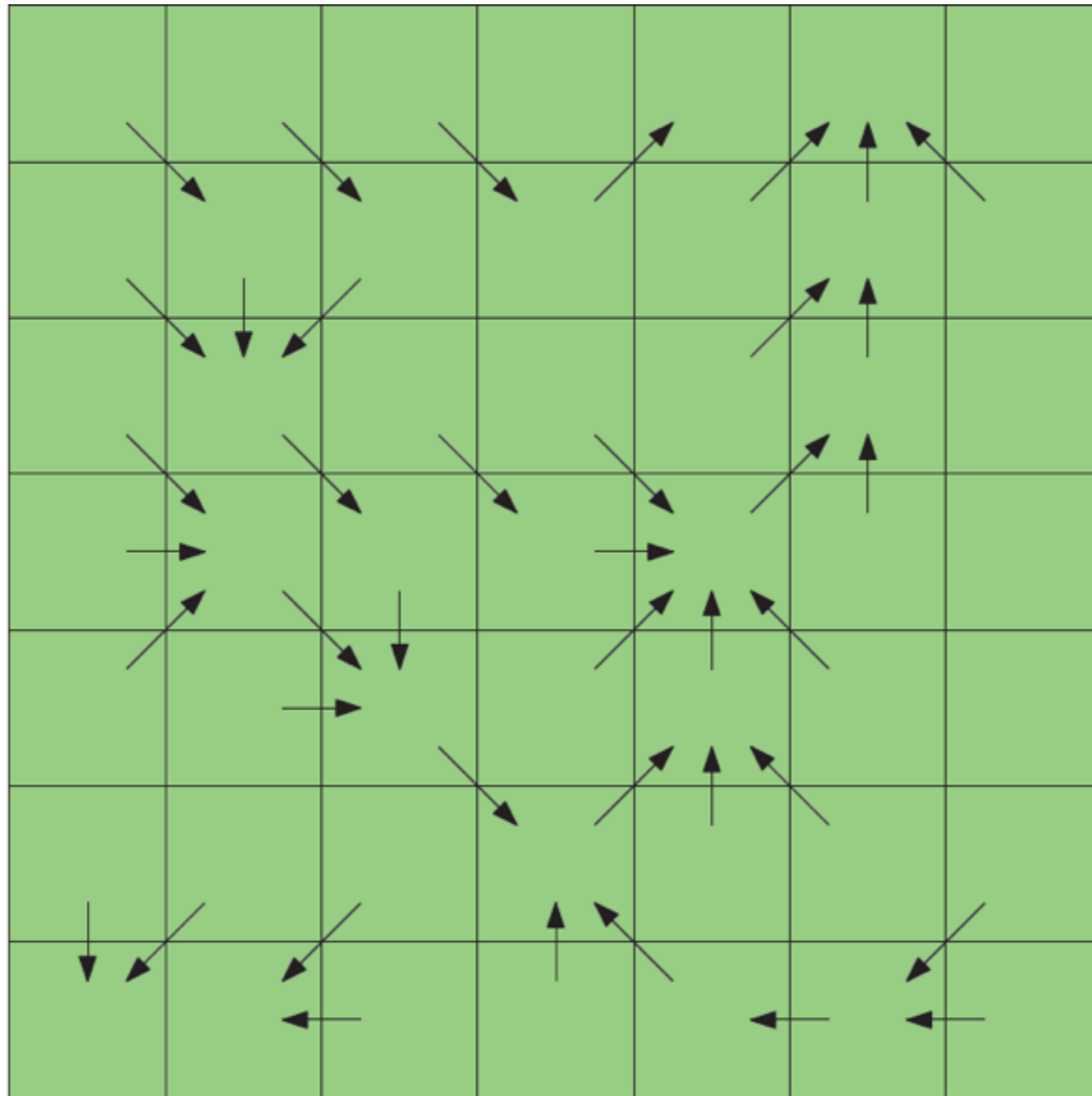


FA grid draped over elevation grid

Computing FA: naive algorithms

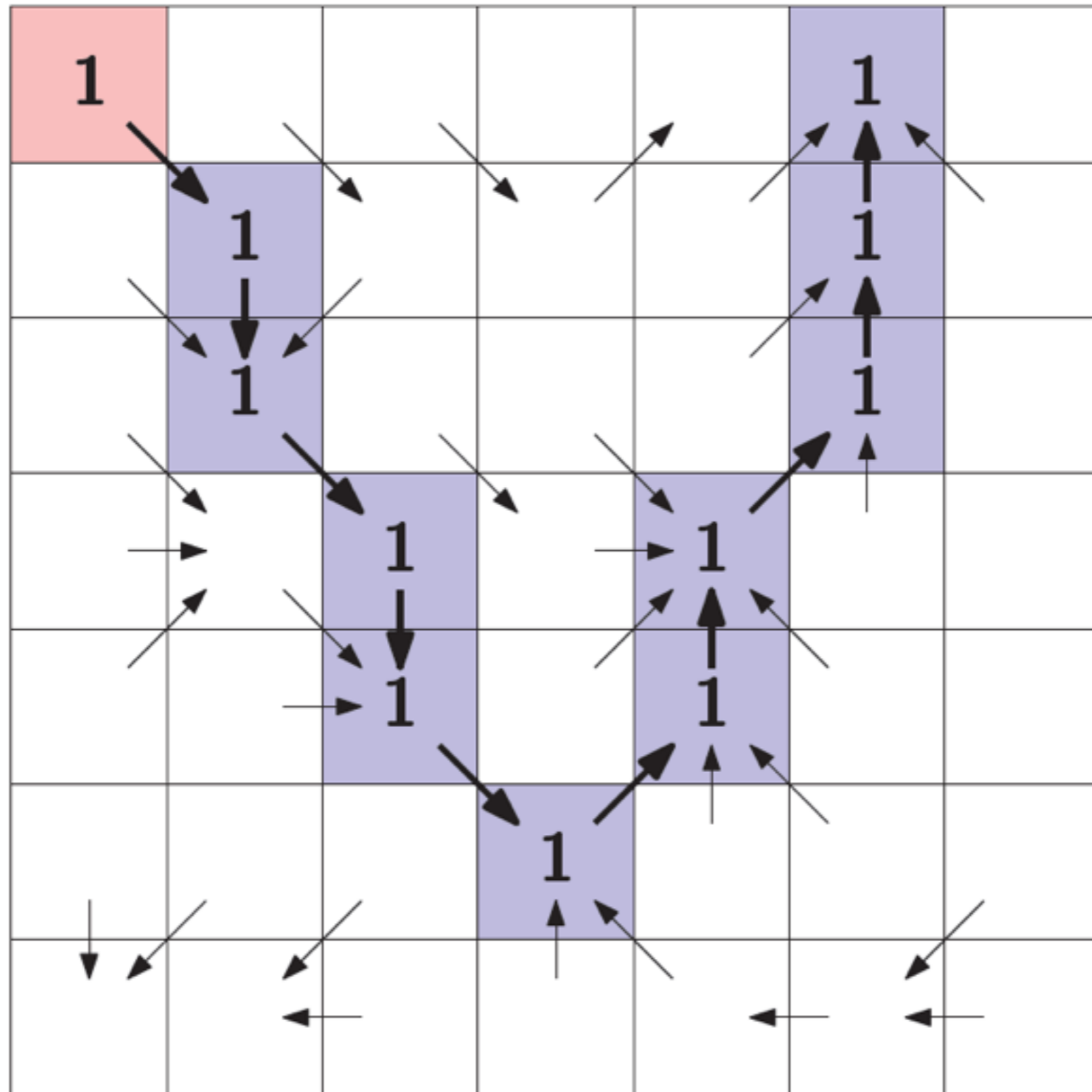
- Idea 1:
 - Scan row-by-row: for each cell add +1 to flow of all cells along its downstream path
- Idea 2:
 - flow at cell c is the sum of the flows of the neighbors that flow into c
 - use recursion
 - do this for every cell

Computing FA: naive algorithm (1)



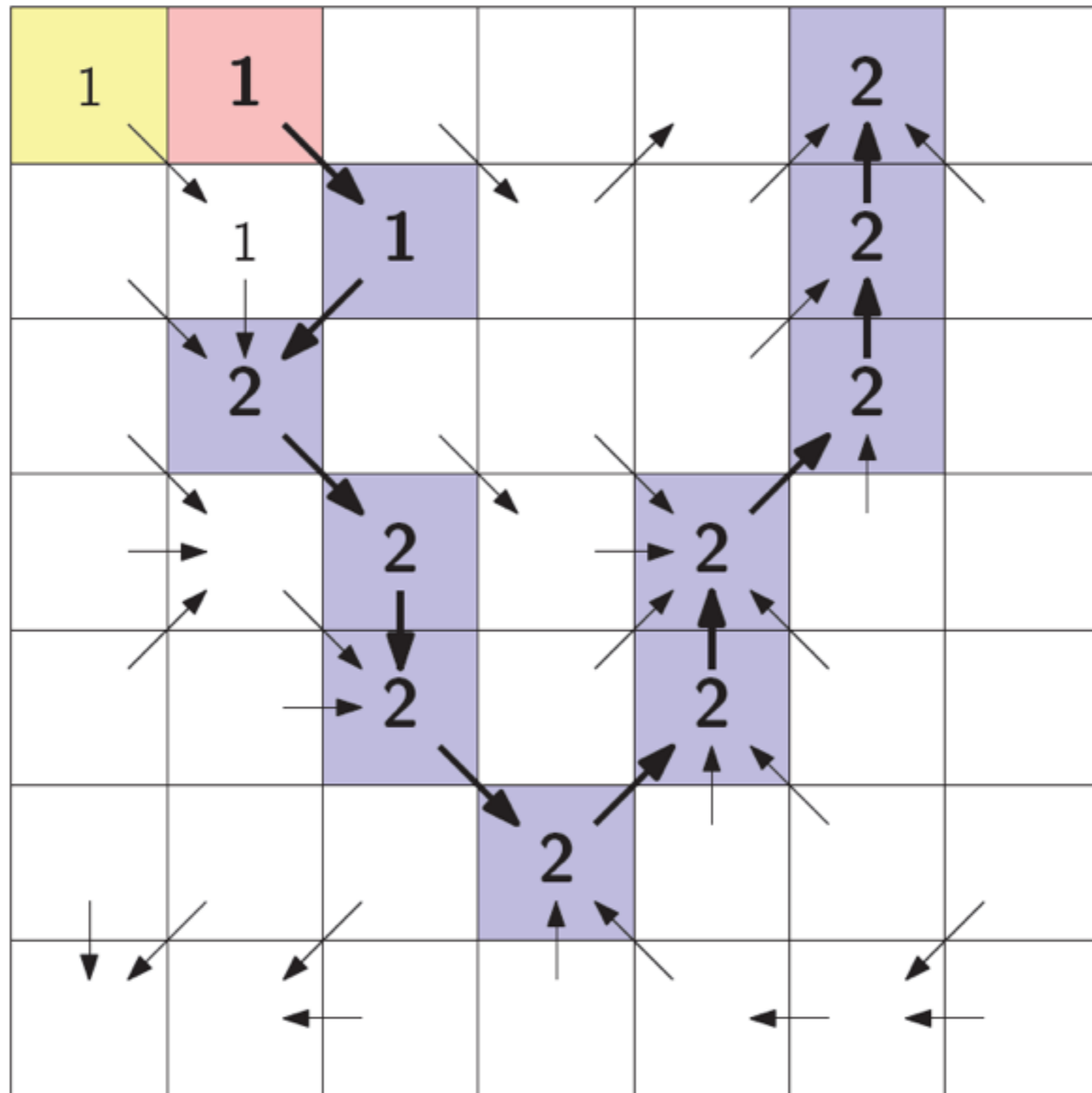
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



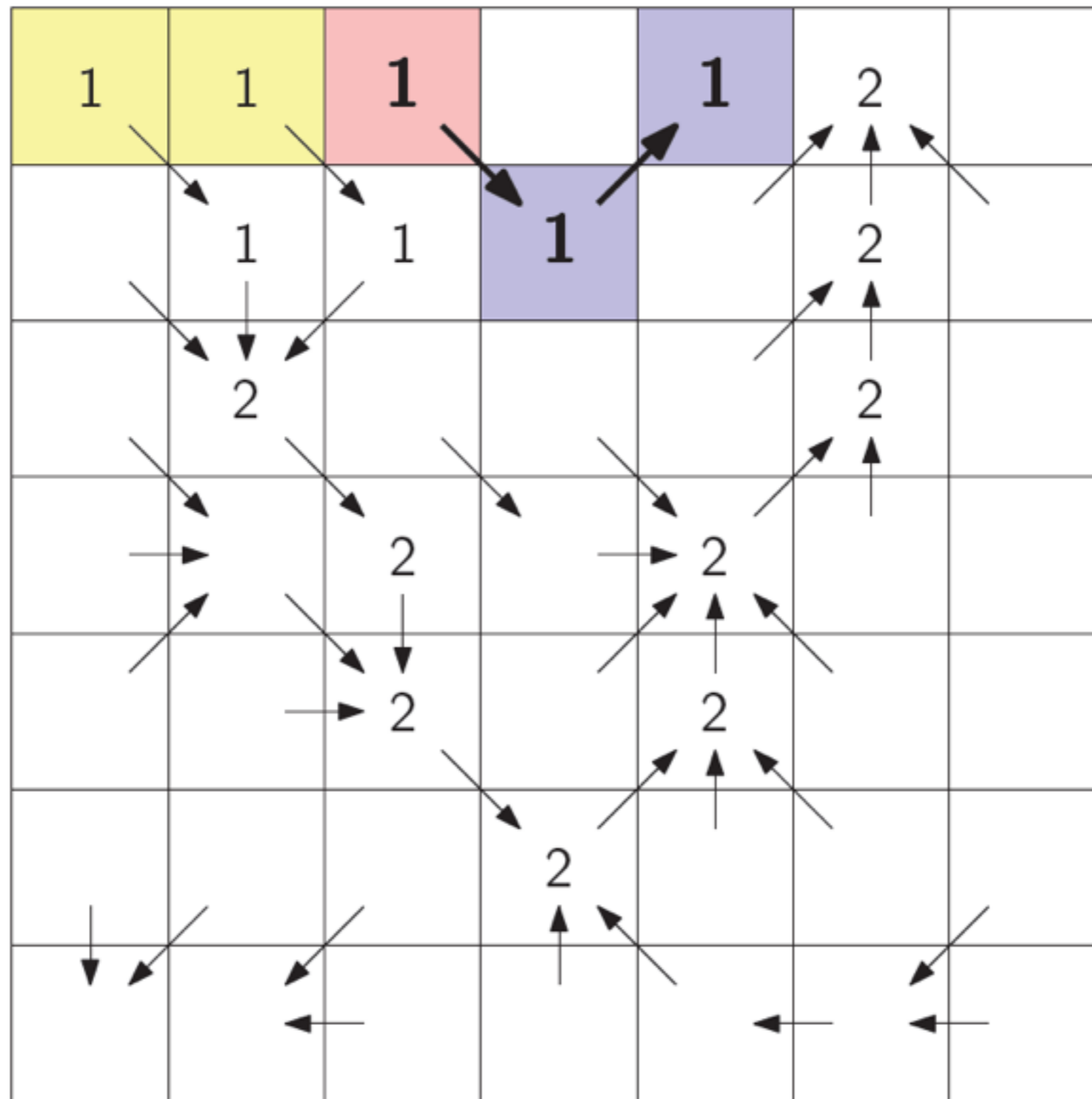
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



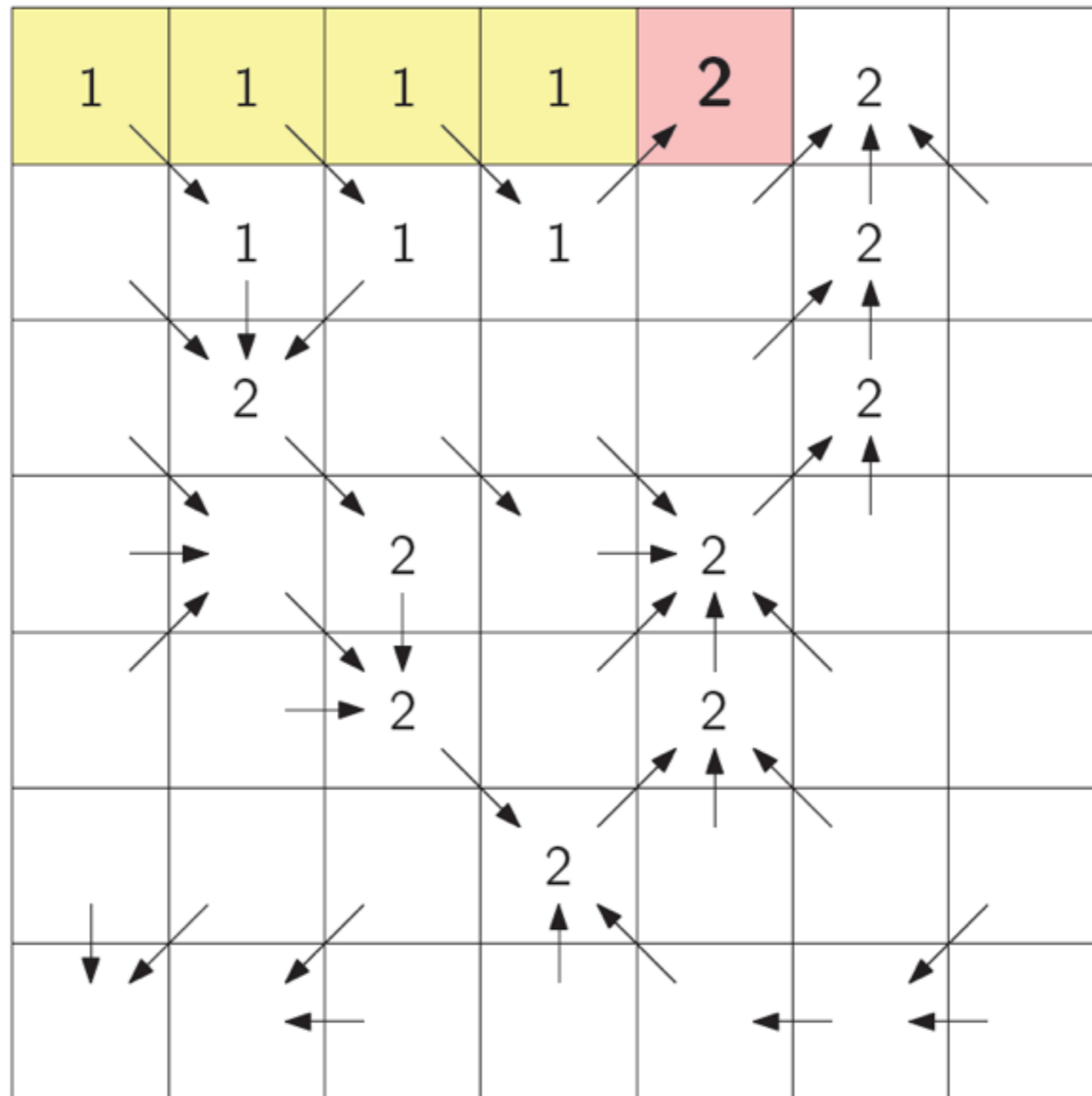
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



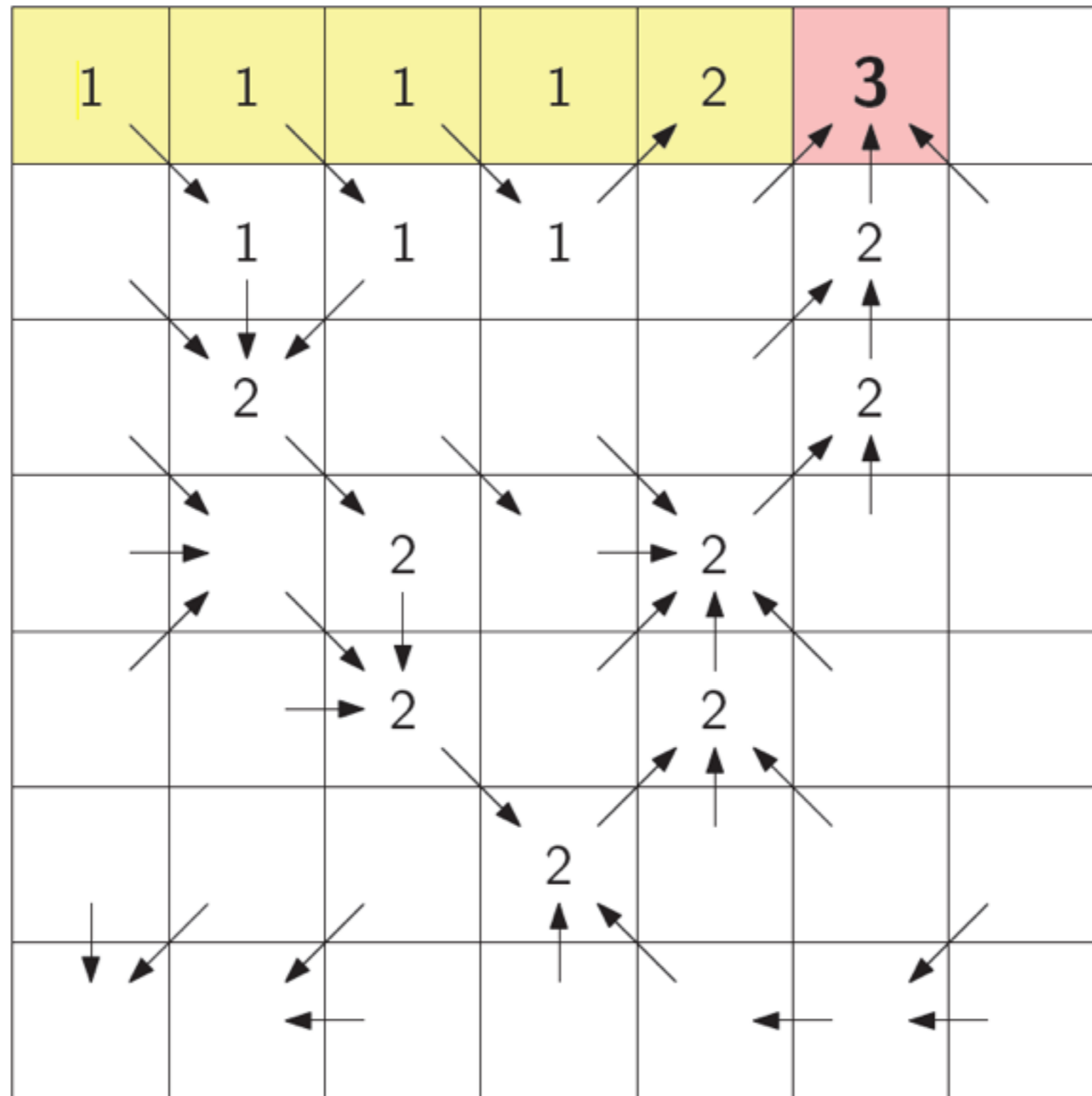
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



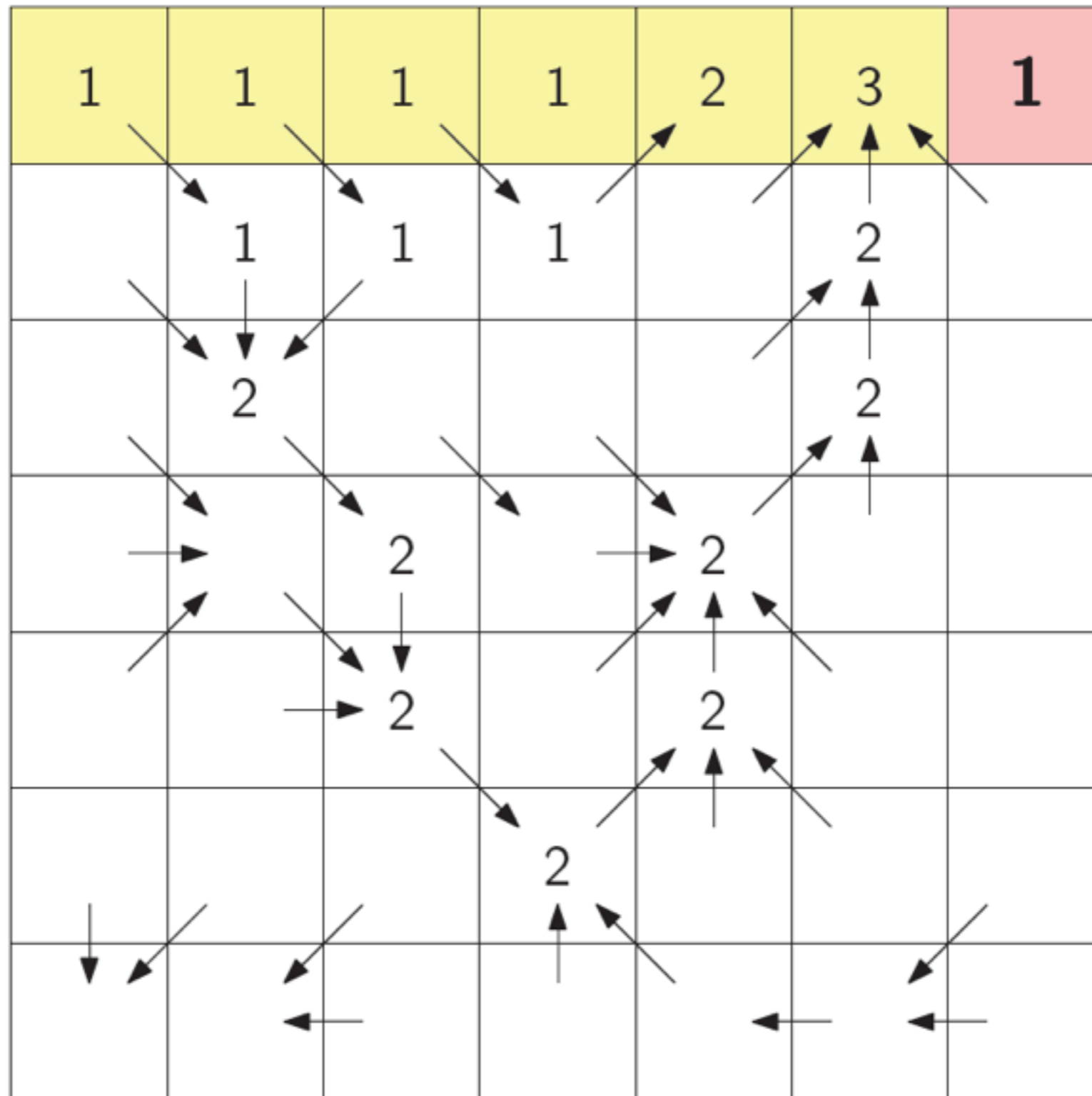
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



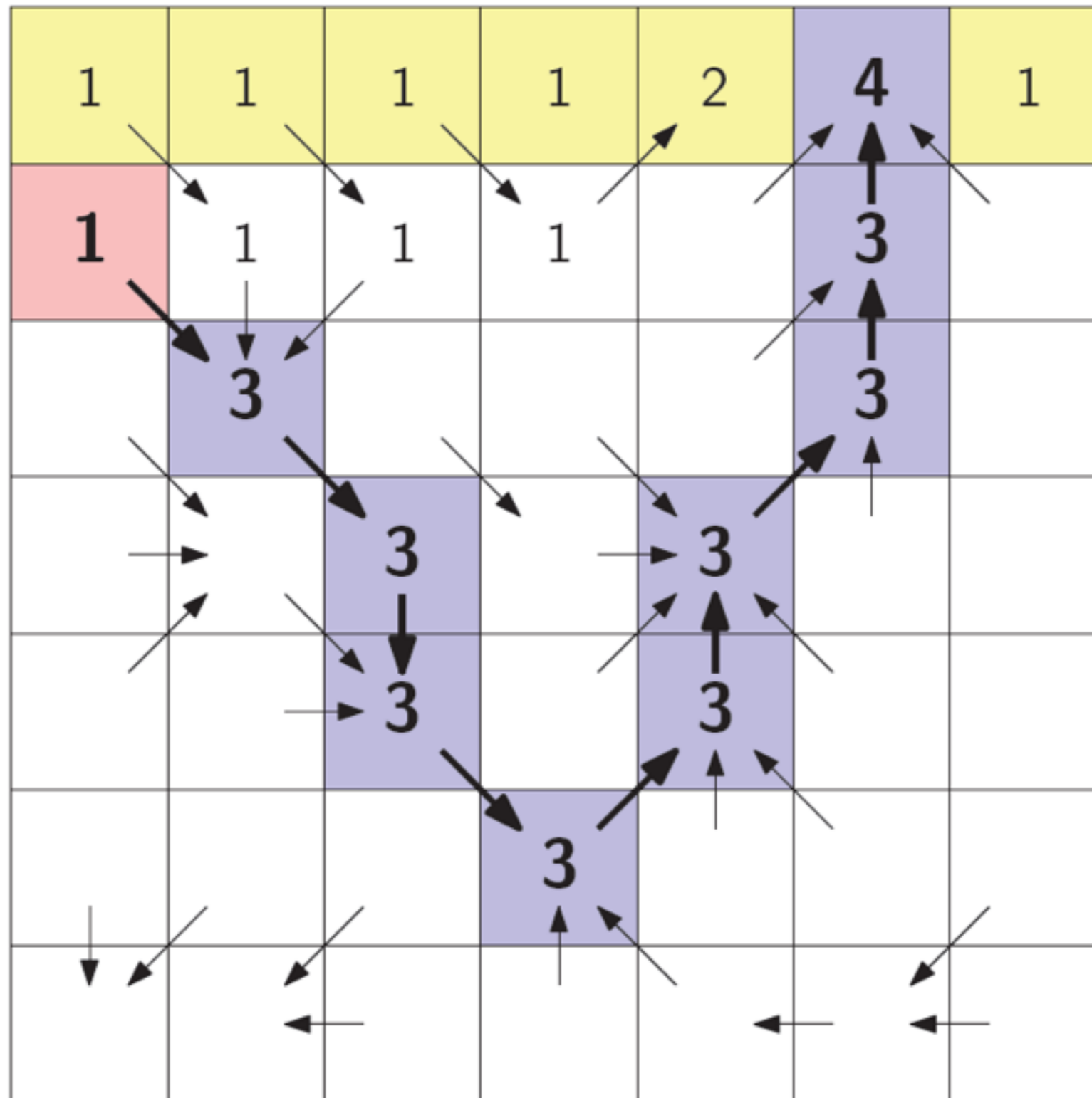
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



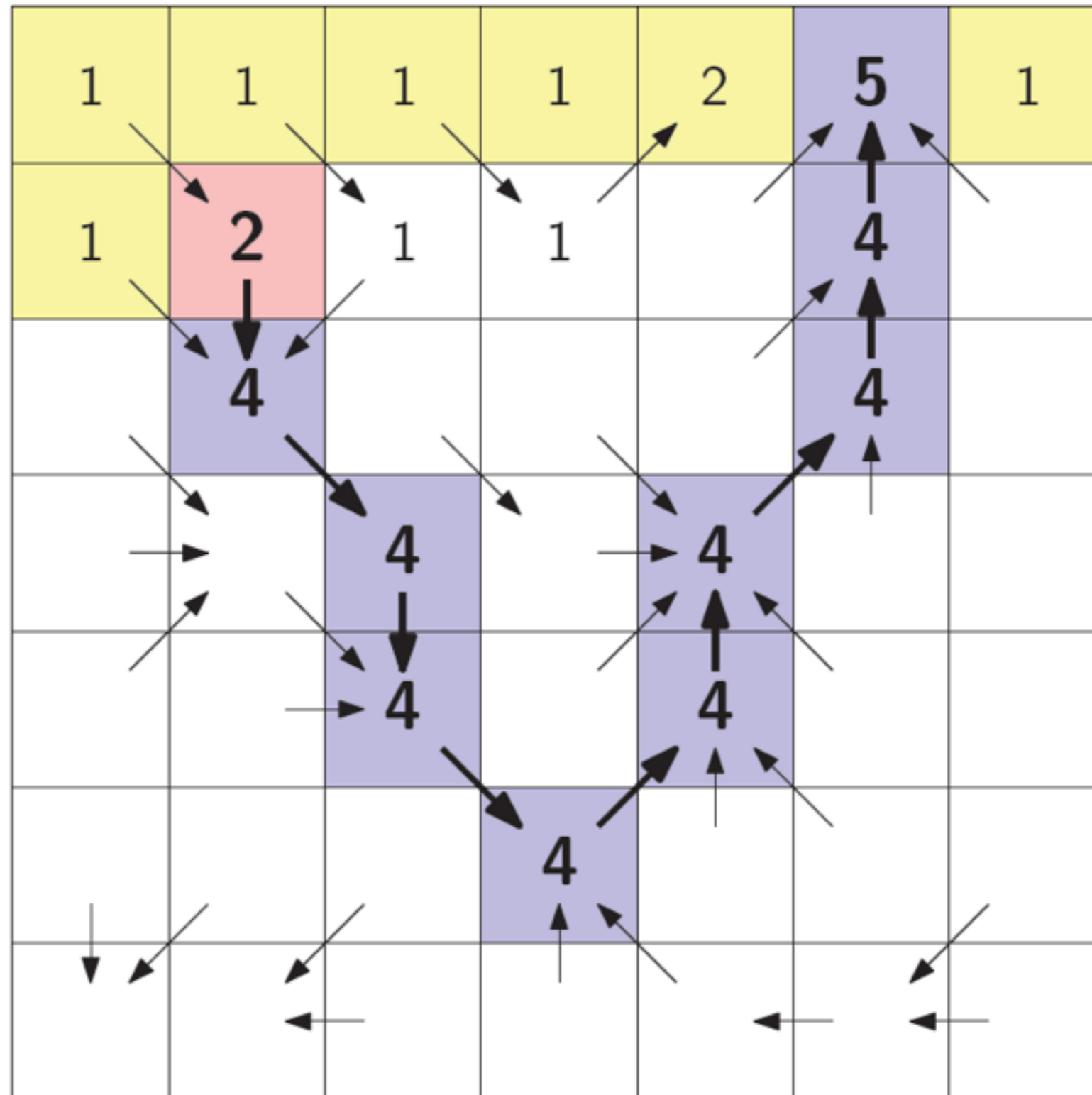
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



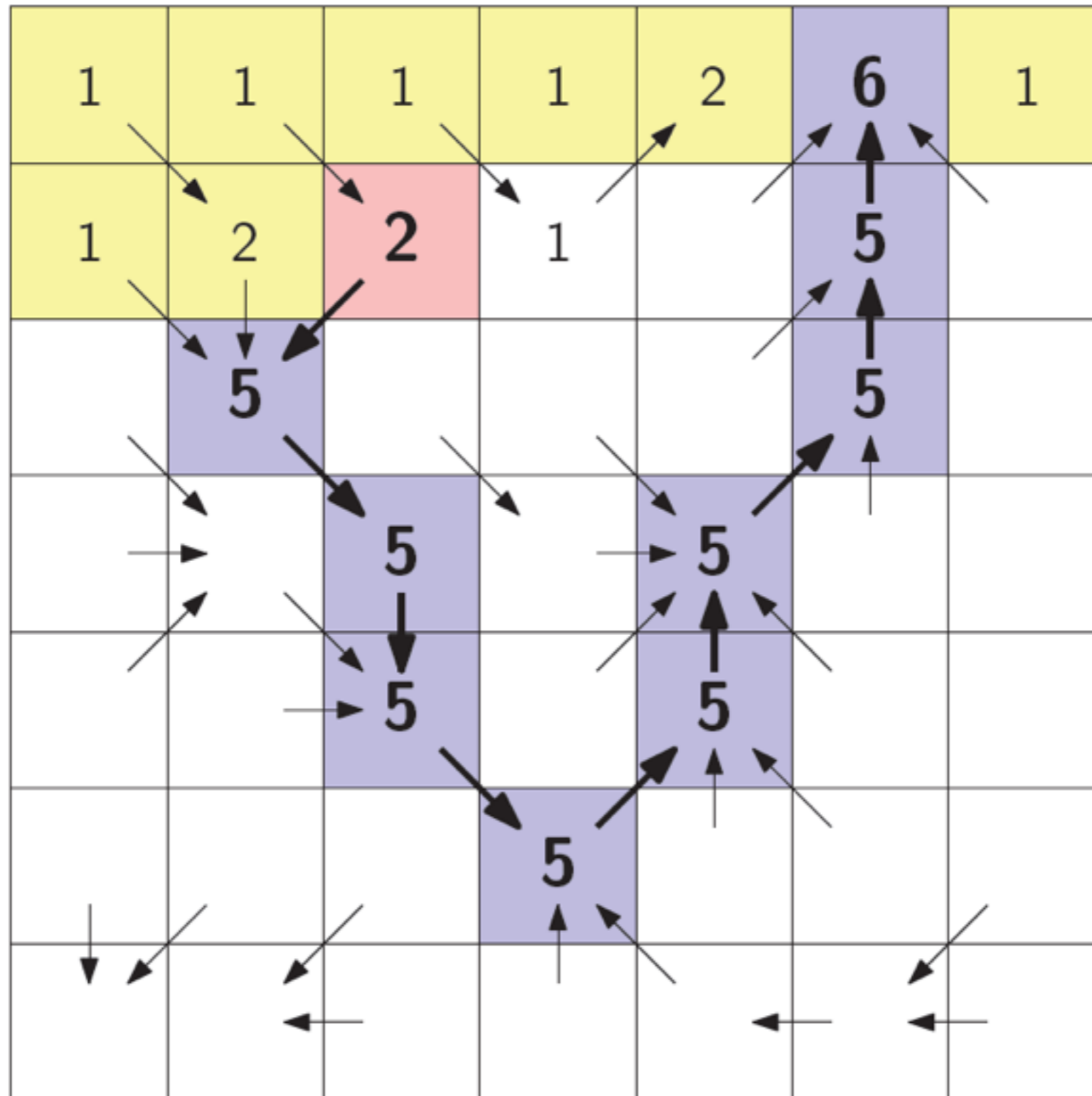
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



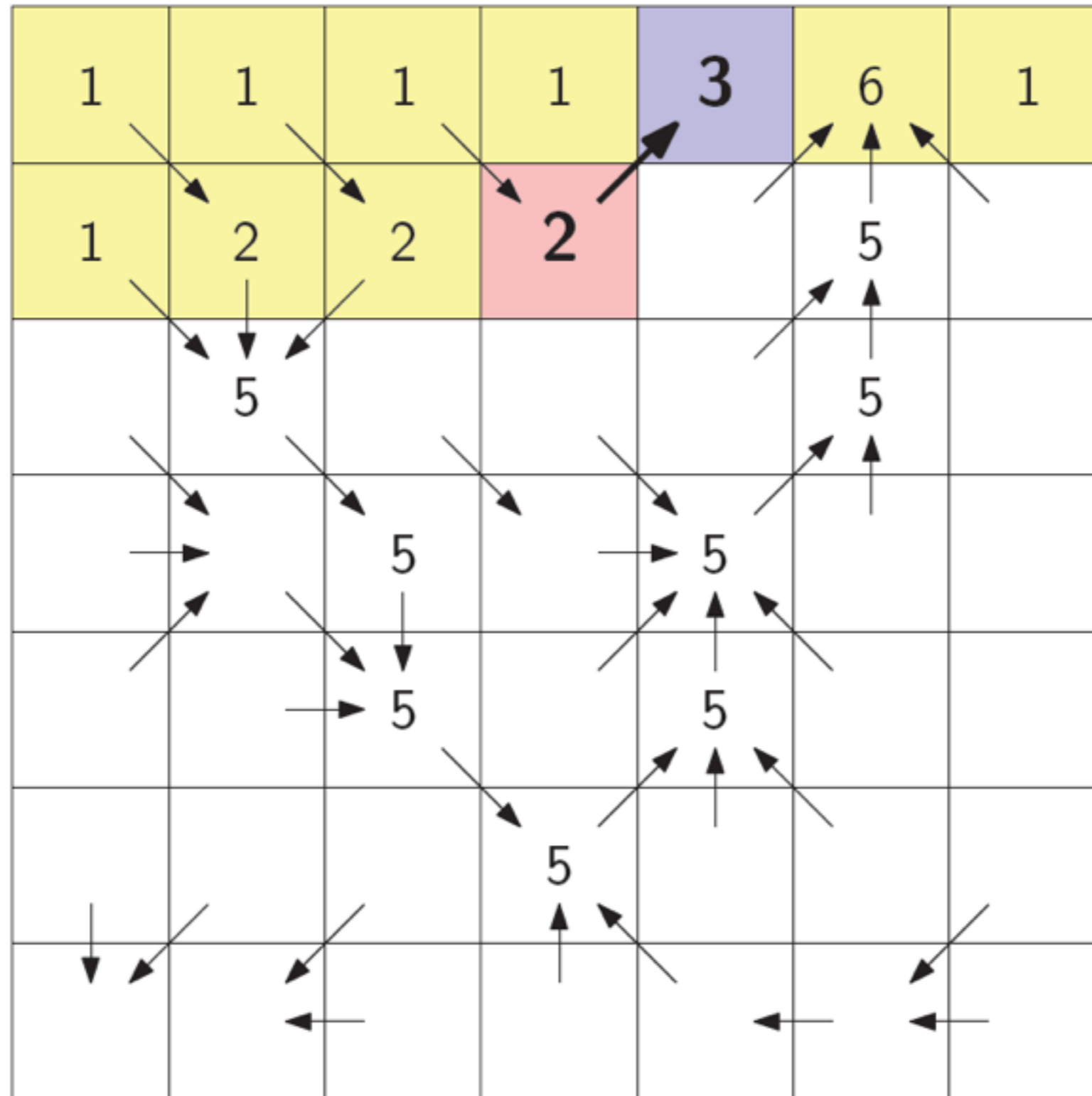
thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)



thanks!!! to H. Haverkort

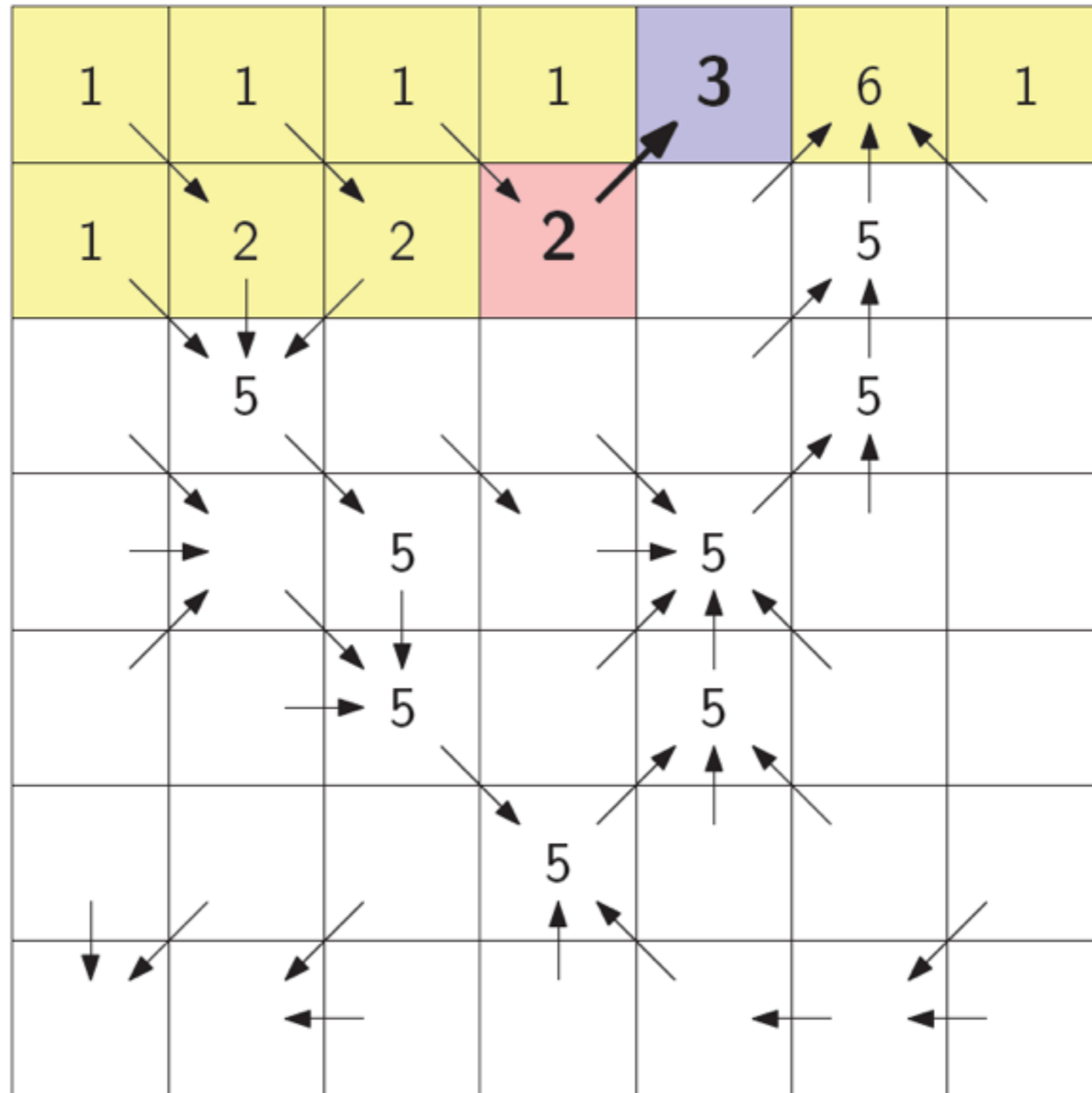
Computing FA: naive algorithm (1)



thanks!!! to H. Haverkort

Computing FA: naive algorithm (1)

n = nb. of cells in the grid



worst-case running time $\Theta(n^2)$

thanks!!! to H. Haverkort

Computing FA: naive algorithm (2)

```
//do it for all
for (i=0; i<nrows; i++)
    for (j=0; j<ncols; j++)
        flow[i][j] =compute_flow(i,j);
```

```
//return 1 if cell (a,b) flows into cell (x,y)
// that is, if (a,b)'s FD points towards (x,y)
int flows_into(a,b, x,y) {
    if (!inside_grid(a,b)) return 0;
    ...
}
```

```
//return the flow of cell (i,j)
void compute_flow(i,j) {
    assert(inside_grid(i,j));
    int f = 0; //initial flow at (i,j)
    for (k=-1; k<= 1; k++) {
        for (l=-1; l<= 1; l++) {
            if flows_into(i+k, j+l, i,j)
                f += compute_flow(i+k, j+l);
        }//for k
    }//for l
    return f;
}
```

$n = \text{nb. of}$
cells in the grid

Computing FA: naive algorithm (2)

- Questions:
 - What is the worst case running time?
 - Sketch a grid direction graph that triggers worst-case

n = nb. of
cells in the grid

Computing FA: naive algorithms

- **Idea 1:**
 - Scan row-by-row: for each cell add +1 to flow of all cells along its downstream path
- **Idea 2:**
 - flow at cell c is the sum of the flows of the neighbors that flow into c
 - use recursion
 - do this for every cell

worst-case
running time
 $\Theta(n^2)$

Flow accumulation: smarter algorithms?

- Ideas?

n = nb. of
cells in the grid

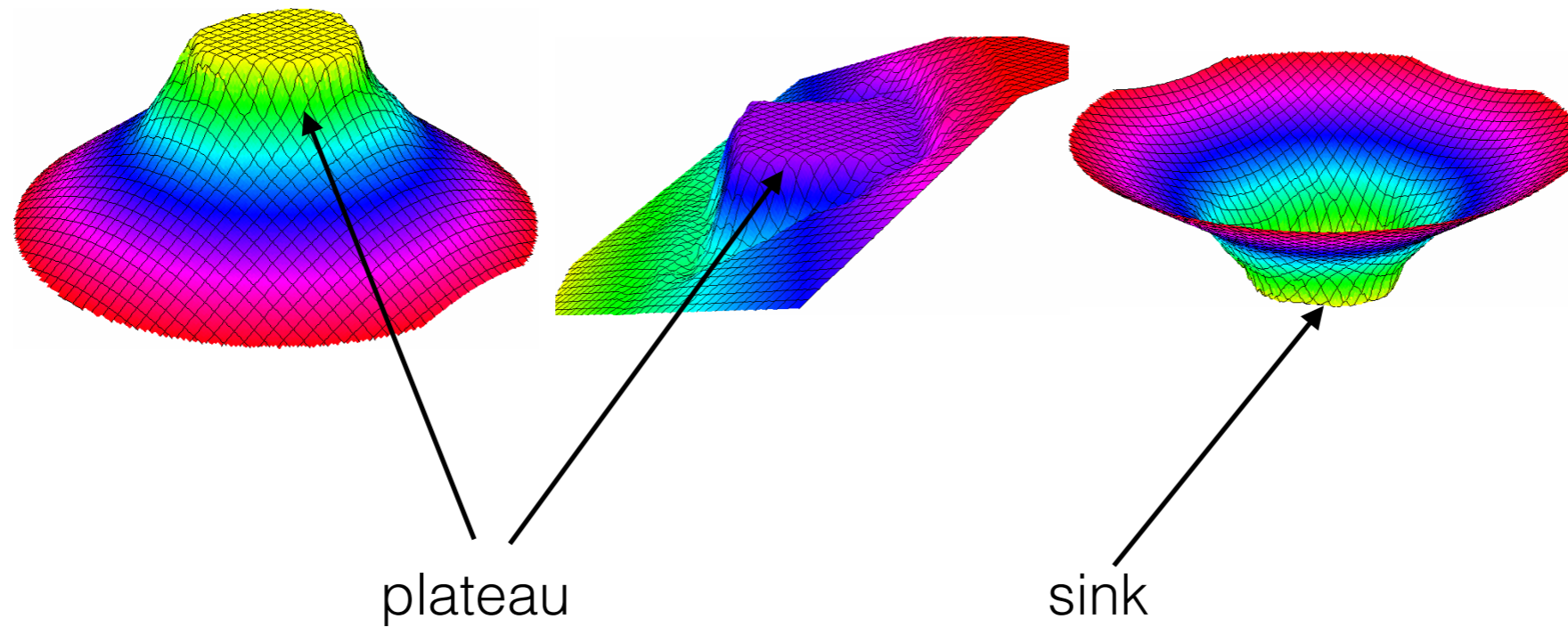
Flow accumulation: smarter algorithms?

- Use recursion, but once a value $\text{flow}(i,j)$ is computed, store it in a table. This avoids recomputation.
 - i.e. dynamic programming
- To completely avoid recursion, compute $\text{flow}(i,j)$ in topological order of FD graph
 - topological order can be computed in linear time
 - basically elevation order; could sort by height, but that's $O(n \lg n)$

worst-case
running time
 $\Theta(n)$

Question: Which algorithm would you chose in practice?

FD: Dealing with flat areas



- Plateau: flat area has at least one spill point from which water flows downhill
- Sink: flat area that's not a plateau

FD: Dealing with flat areas

- Want to label each flat area (maximally connected blob) with its own label.
- How?... and how long?

FD on plateaus

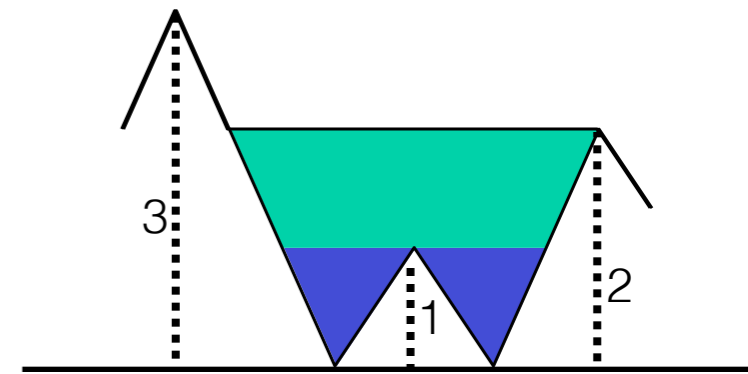
- Water comes in on the plateau from higher grounds and leaves through the spill points ==> direct FD towards the spill points
- HOW?
 - One way is to do a BFS from all the spill points in parallel ... when seeing a new (white) cell, set its FD towards the cell that discovered it.
 - This way, water finds its shortest way to the spill point (recall BFS computes shortest paths) — we are ignoring that diagonal edges are longer.

FD on sinks

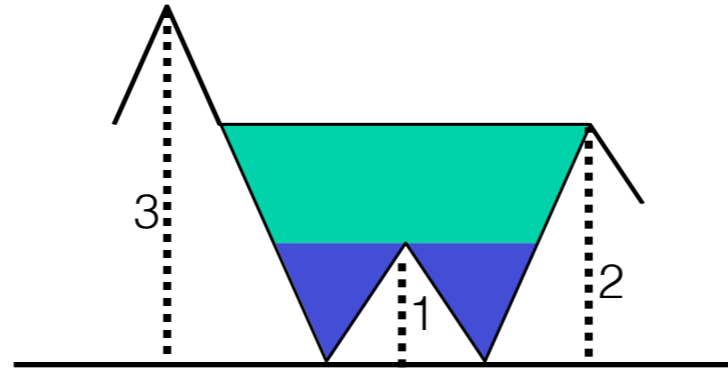
- Water comes in the sink from the surrounding area and cannot go out following downslope paths. The only way to route water out of the sink is to let water go uphill.
- Options for setting FD on sinks:
 1. Do not route water out of the sinks, i.e. let FD be undefined on sinks. Doing this essentially interrupts the flow of water at the sink.
 2. Let the water escape the sink by assigning upslope paths in the sink and possibly around it. This is done by simulating a process of **flooding**.
- Flooding simulates steady state. Imagine the terrain is surrounded by a giant ocean and an infinite amount of rain falling down. At steady state all sinks are filled and, at every point in the terrain, water has found a path towards the outside.

Flooding

- Goal: compute FD such that every point in the terrain has a flow path to the outside ocean. Want path of lowest max height.
 - Put differently: Want to compute for every point in the terrain its “shortest” path to the boundary, where we define the length of a path = max height of a point on the path.
- General idea:
 - For each sink, compute its basin (all grid points that flow to the sink). This induces a partition of the terrain into (sink) basins.
 - Find adjacencies between the sink basins. Mark each edge (s,t) between two sinks s and t with the elevation of the lowest edge along the s-t boundary. This represents the height at which the basins of s and t will merge.
 - Merge sinks in order, until all sinks are DONE. Initially, the ocean is DONE. When a sink merges with a DONE sink it becomes DONE.



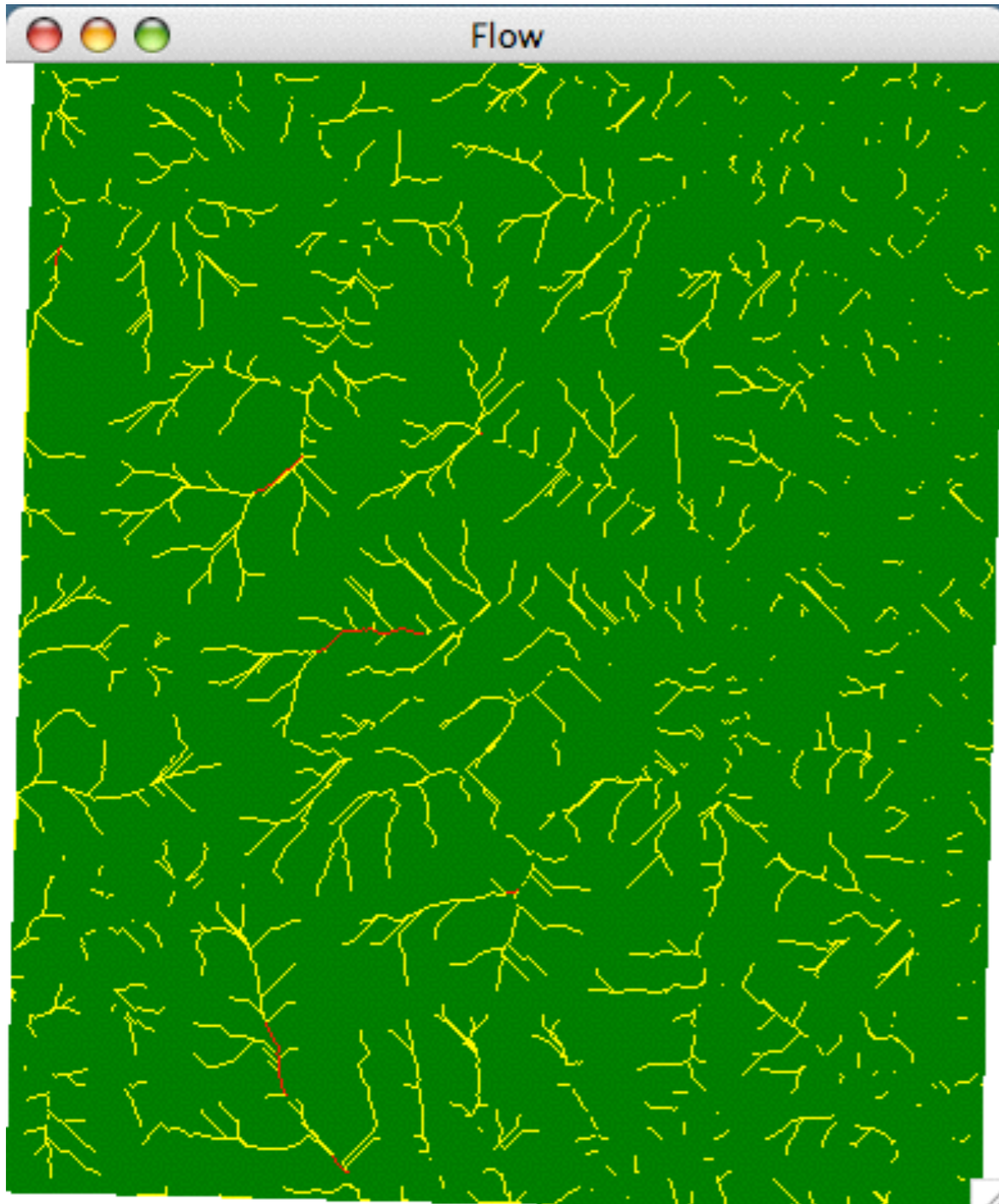
Flooding



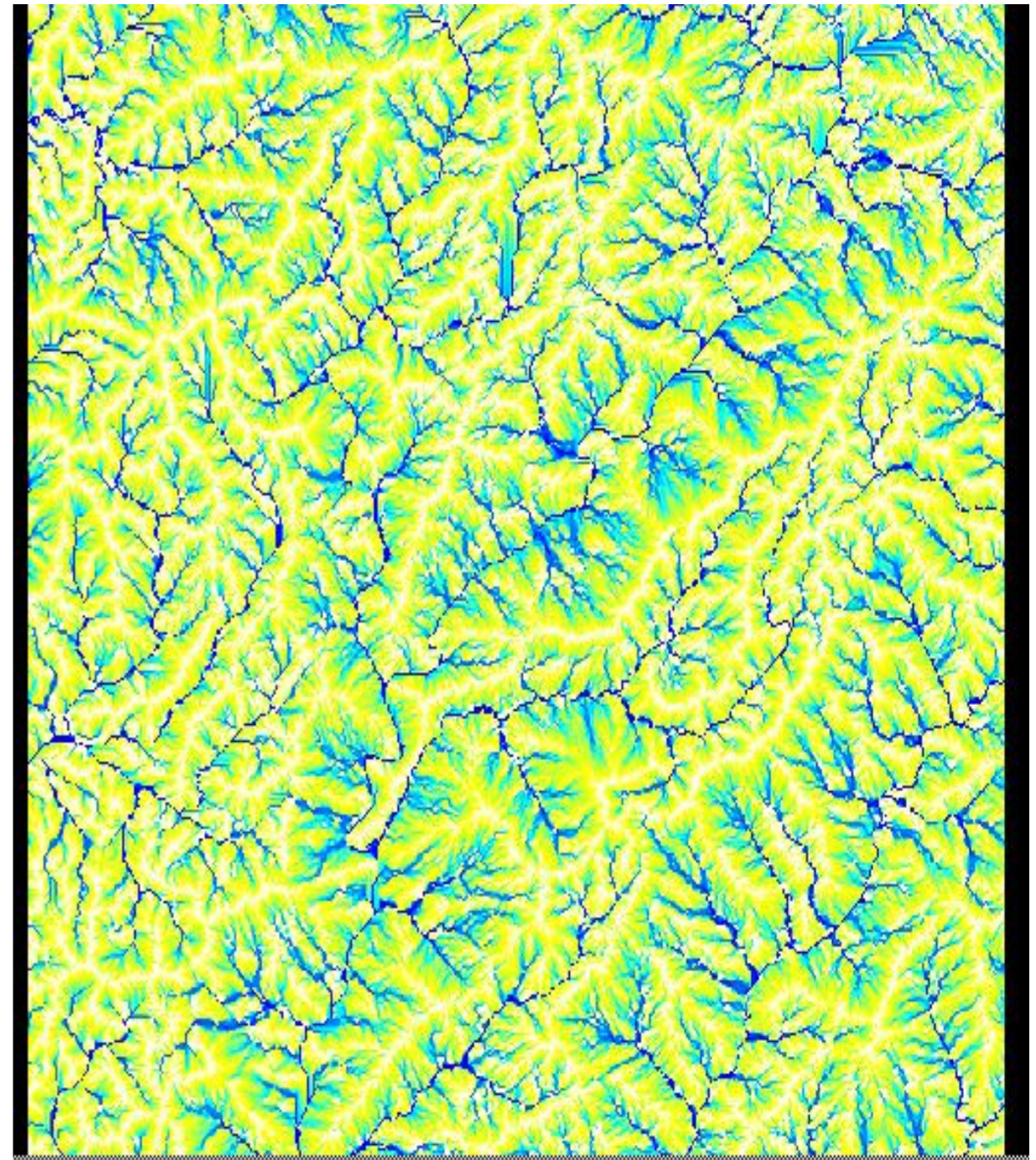
- For each sink, record the lowest height at which it merges with the ocean.
- Raise each basin to its height. This fills the sinks and creates a terrain with no sinks.
- Compute FD on this filled terrain (when projected on the original terrain these FD route water upslope out of sinks).

Flooding

- Flooding creates more realistic, connected river networks.
 - If there is a lot of water, it eventually erodes a canyon ...
- Partial flooding:
 - fill only “small” sinks.
 - fill sinks below given threshold.



FA without flooding



FA with flooding

Computing rivers and watersheds

Rivers: all points with FA above a given threshold

Watershed (s): all cells that have a flow path into s

River hierarchy: Find river backbone and its tributaries. Find their watersheds.

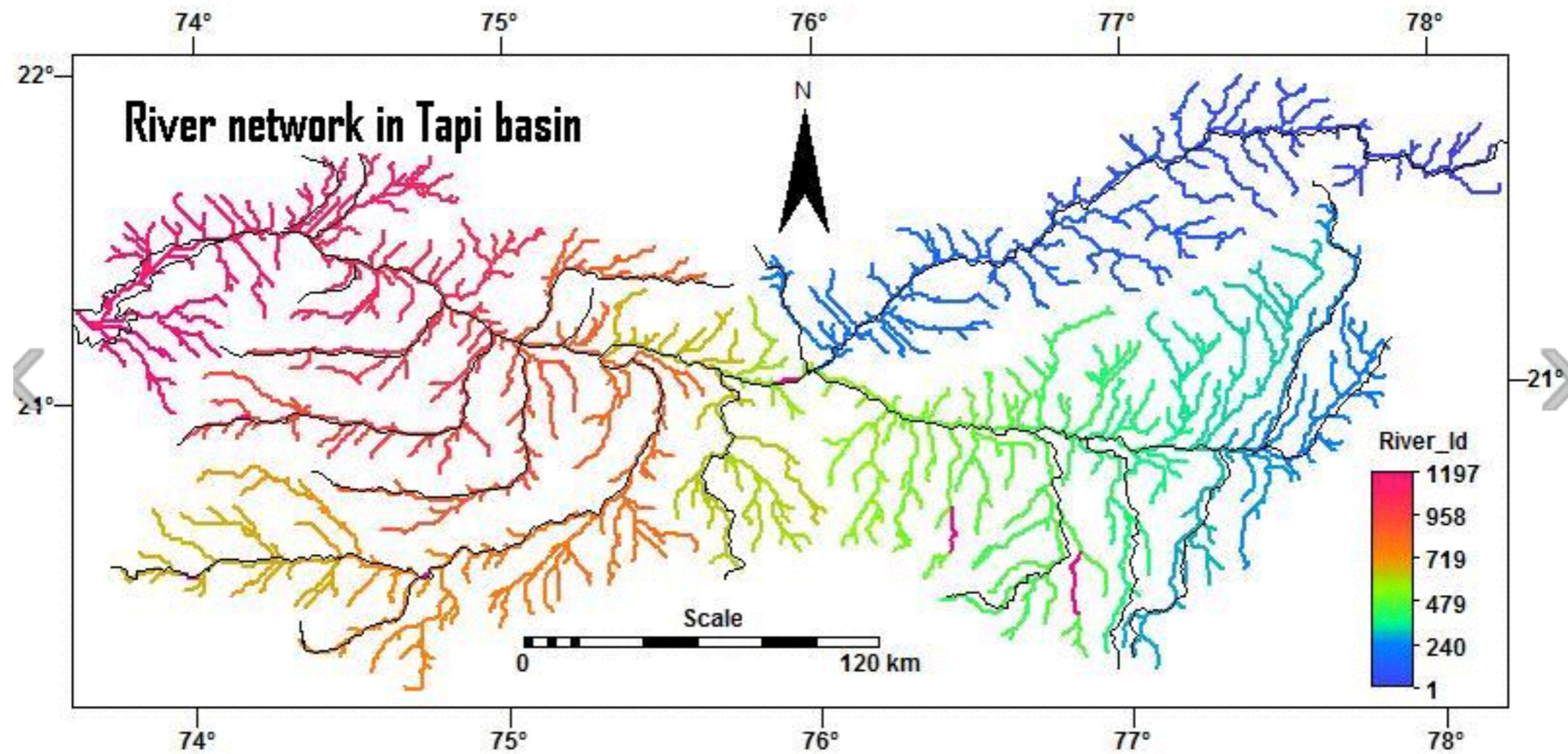
River network

- Rivers: cells with $FA > \text{threshold}$



River network

- Rivers = cells with $FA > \text{threshold}$



Pfafstetter watershed hierarchy

The following animation due to

Herman Haverkort

U. Eindhoven

<http://www.bowdoin.edu/~ltoma/teaching/cs350/fall14/Lectures/pfafstetter-short.pdf>

Practical details

- grid ascii format
- gdal library provides functions to read a grid from any format

