Finding the Shortest Path

A* Algorithm

Anton Gerdelan <gerdela@scss.tcd.ie>

Motivation

- Greedy Best-First may not be suitable
  - graph has lots of local maxima traps
  - want to guaranty the shortest route
  - our heuristic didn't take actual cost into account
- let's upgrade Dijkstra's algorithm instead
  - add a heuristic

A*

- pronounced " 'A' star "
- 1968 extension to Dijkstra's algorithm
  - PE Hart, NJ Nilsson, B Raphael, "*A Formal Basis for the Heuristic Determination of Minimum Cost Paths*", IEEE Trans. Systems Science and Cybernetics, 1968
- commonly used in **path finding** / path planning
  - video games, robot motion
- good performance + optimal path

Reference Material

- Amit Patel's website is amazing (animated)

  http://www.redblobgames.com/pathfinding/a-star/introduction.html

A*

- actual cost of path so far + estimated cost to goal
    - $f(n) = g(n) + h(n)$
- This helps avoid local maxima traps
- Investigates fewer vertices than Dijkstra
- May be slower than Greedy Best-First
- But guarantees shortest path

heuristics

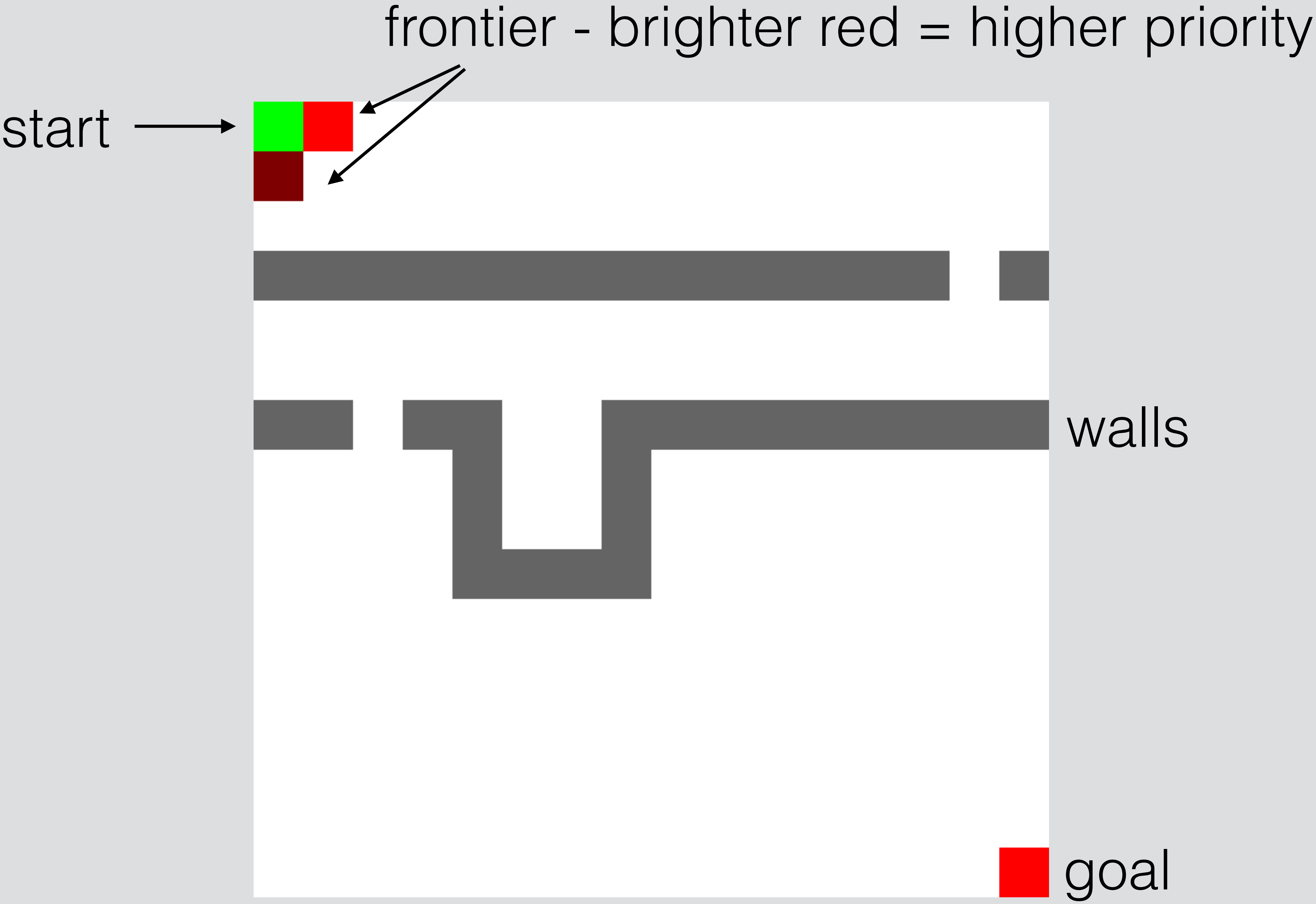
- create function ***h(n)*** which gives a numerical guess to rate choices (which node to try next)
    - Manhattan Distance - count city blocks and and across
    - **Q.** why is this commonly used again?
- make sure heuristic function always returns a value larger than the actual cost or distance - why?

Cost

- g(n) is the actual cost e.g.
  - distance to the next node
  - + the distance on the path so far
- so for each frontier node
  - f(n) = path cost + Manhattan Distance to goal
- keep frontier choices in a priority queue
  - insertion sort?
  - frontier can get quite big

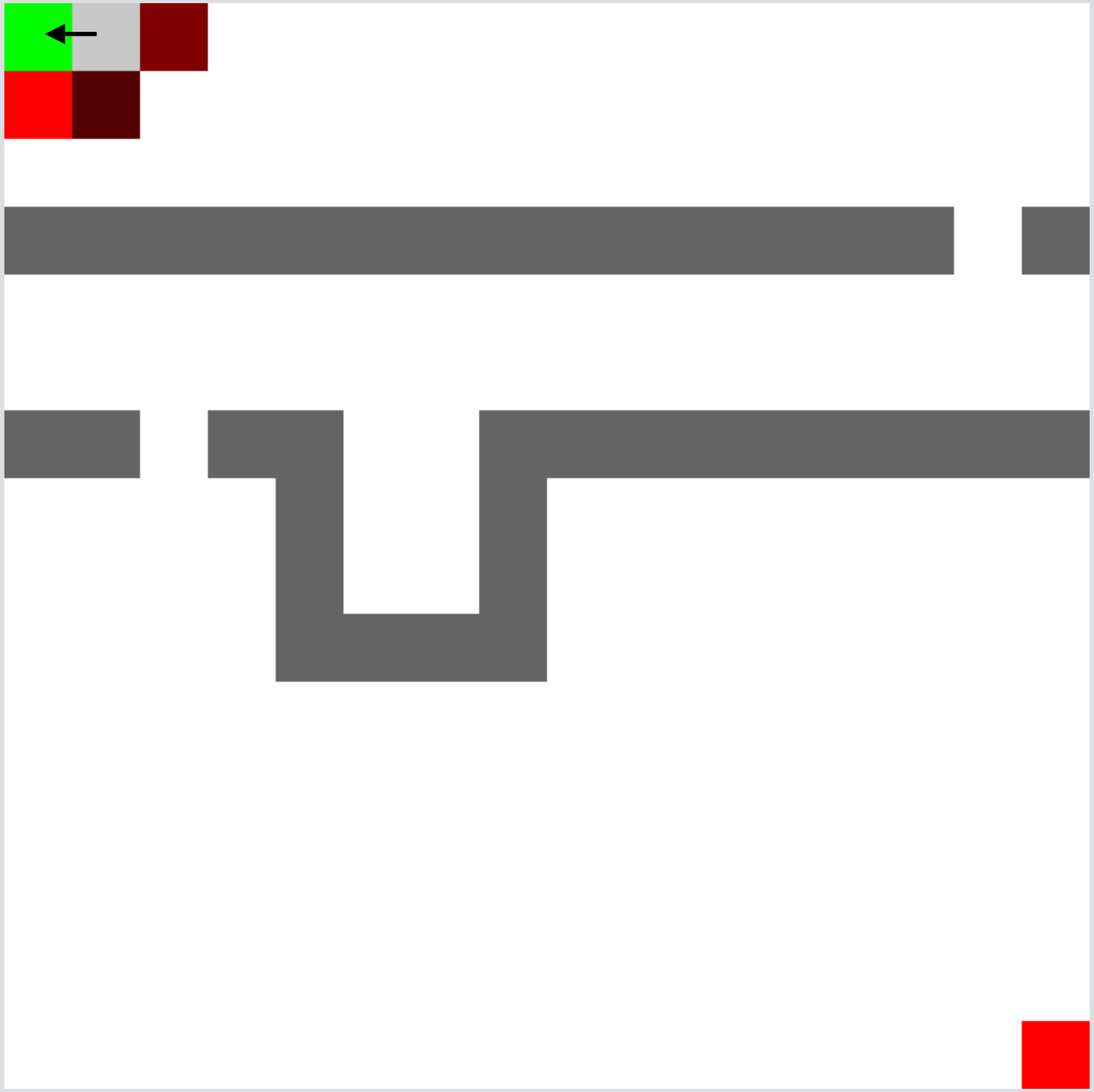
Demo

- Grid/tile environment for my graph
- Assumed you can't move diagonally
- Walls (node can't be entered)
- Every move has cost g(n) = 1 + path so far
- Heuristic h(n) = distance to goal across + down
- Used our ppm writing code to output image at each step
- source code: https://github.com/capnramses/data_structures_algorithms

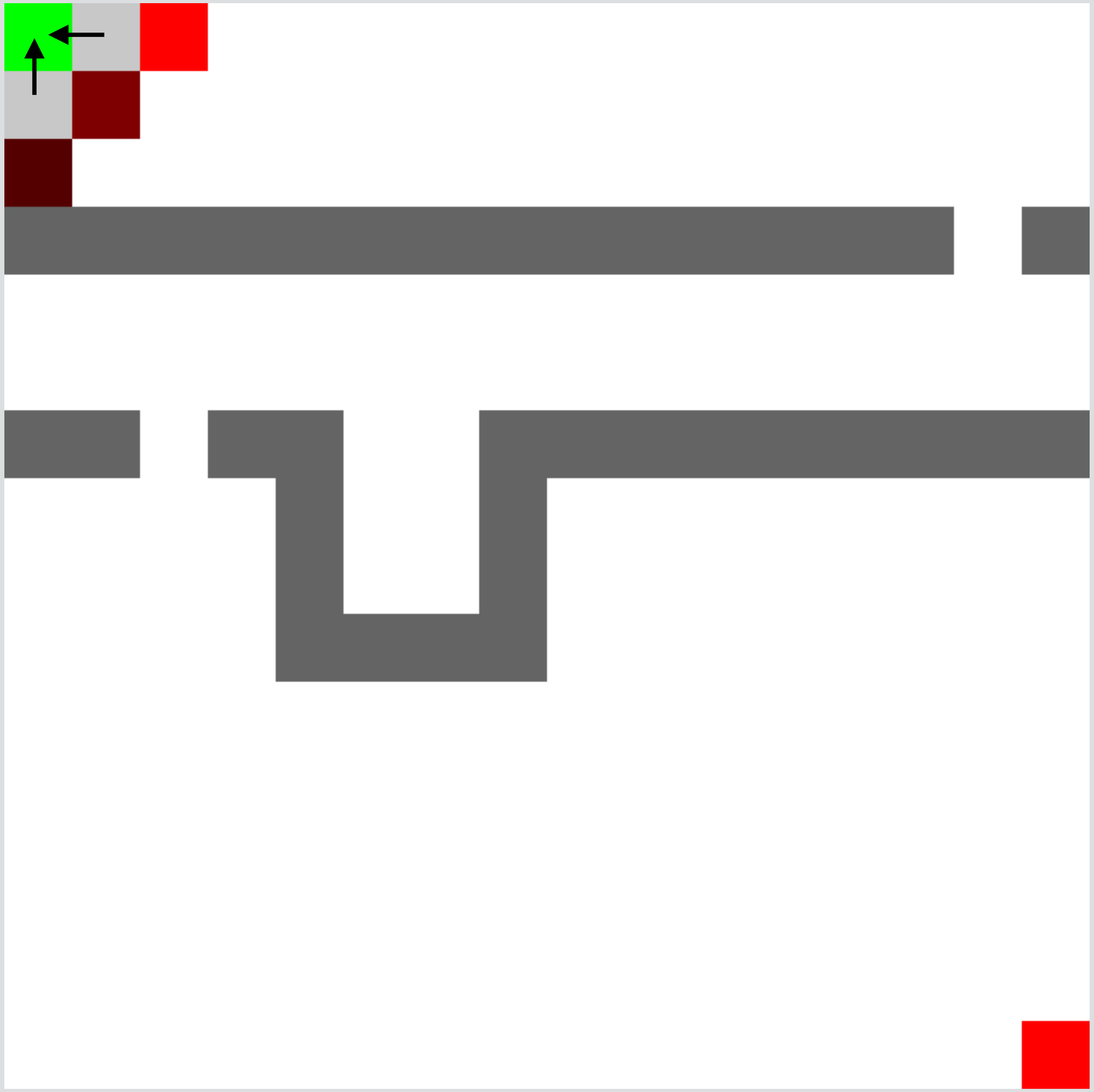


so the next node investigated is…

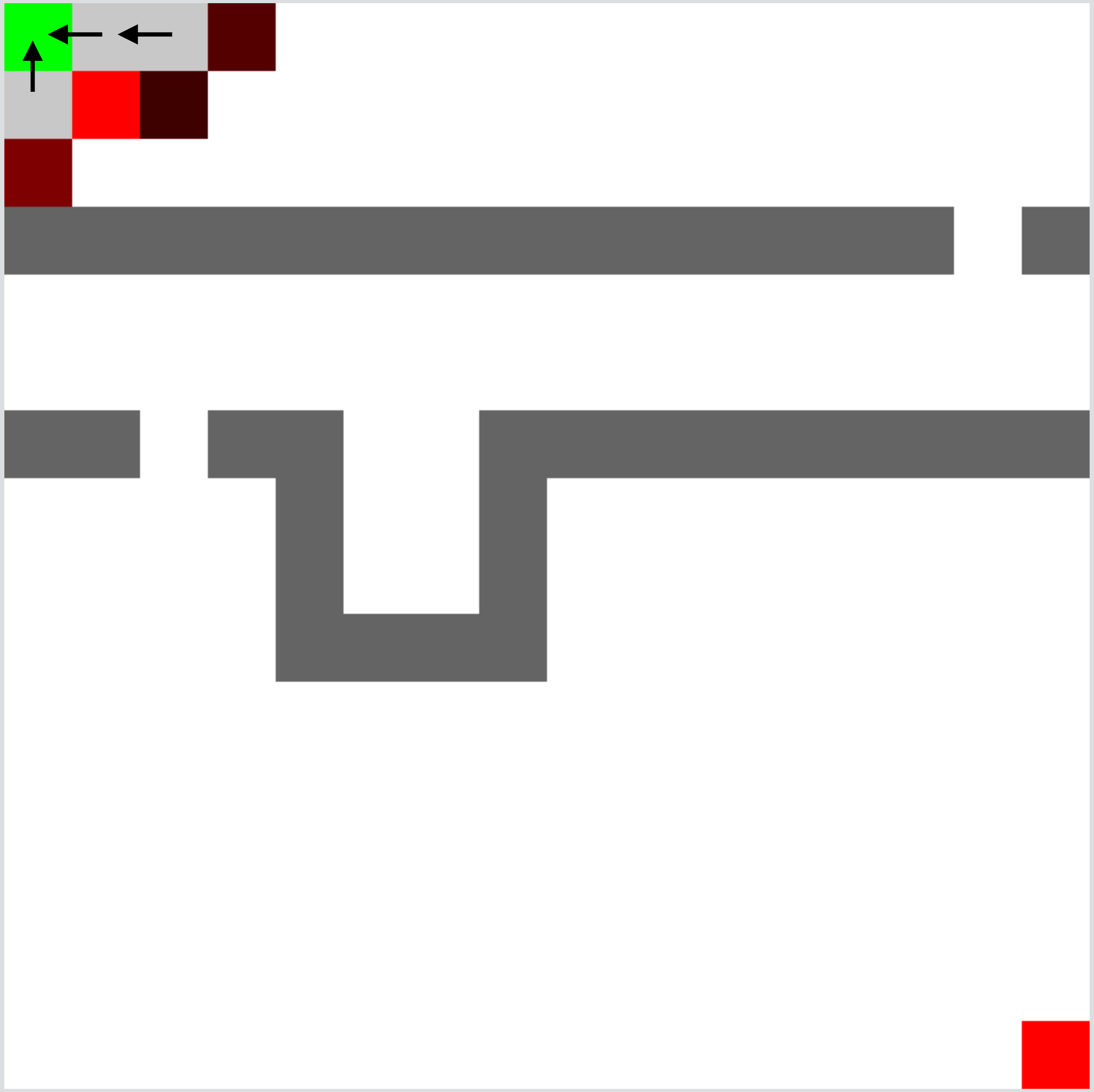
grey is investigated



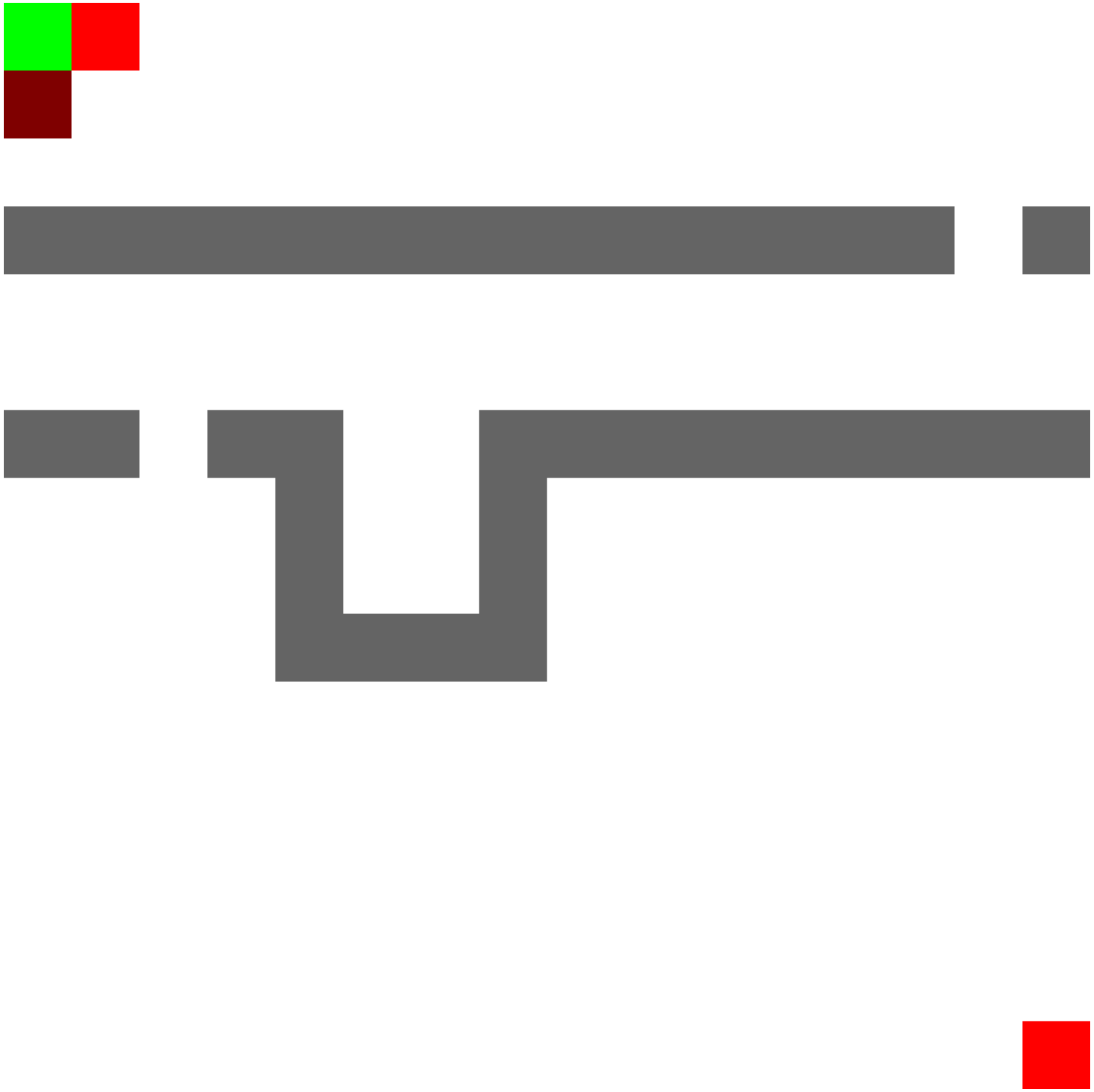
each investigated node also has pointer to its parent
i didn't draw this in the output

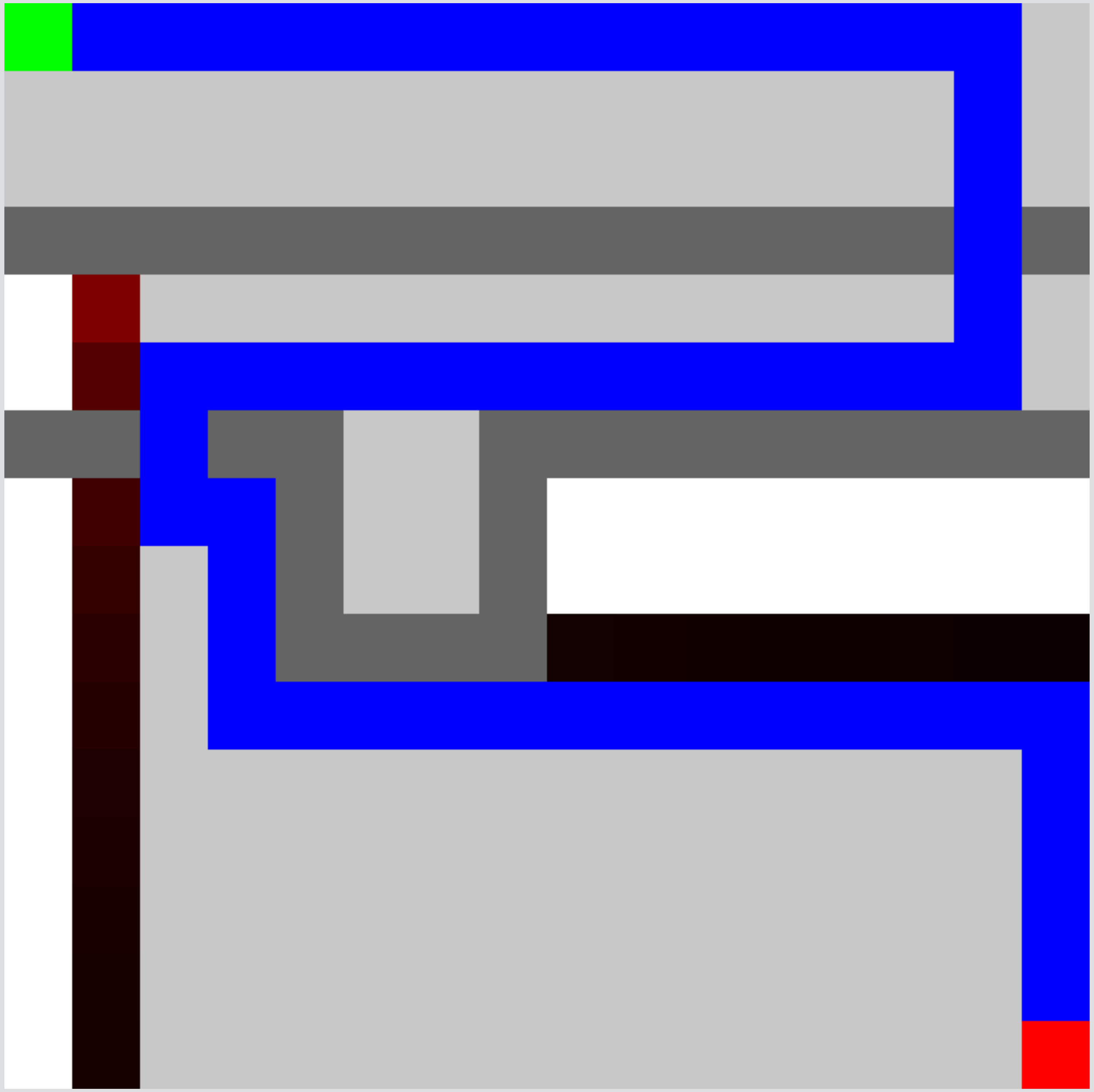


like Dijkstra and Greedy BFS - when goal found follow arrows back

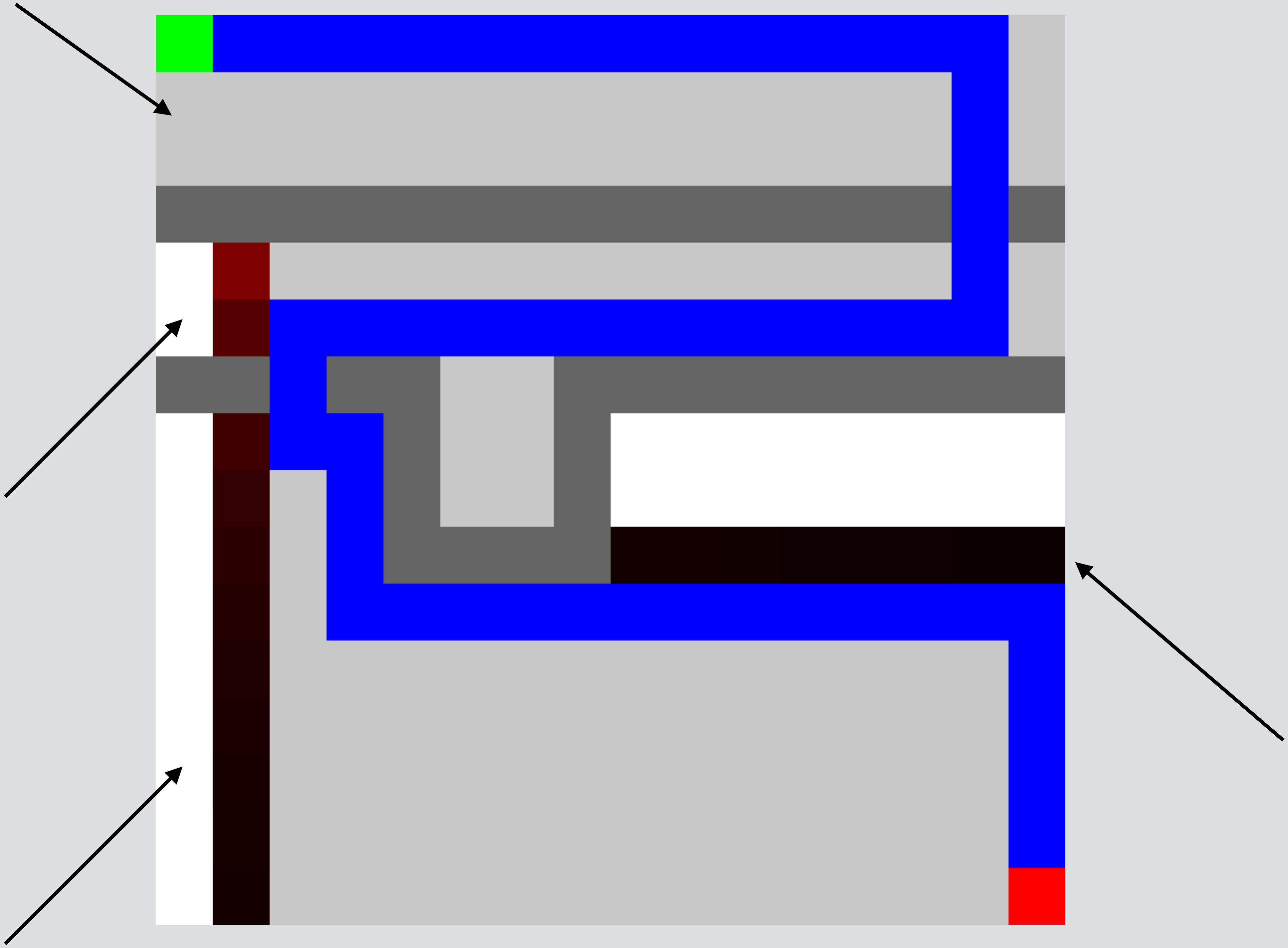


Animation Time





work from goal along parents: optimal path is shown in blue moving diagonally would have reduced grey visited area



frontier gets quite large compared to Greedy Best-First

Variations

- Commonly used for video games and robotics
  - may not consider fine detail
  - can't handle dynamic obstacles
    - doors opening
    - people running into the road
  - motion path is very robotic-looking
    - perception of virtual characters
    - may not obey physical constraints like momentum of a car
  - if path to goal is long - may need to limit depth

Making a .gif / video

- Output image for each step as numbered filename
  - i used 50x50 pixels for each graph node (bigger for-loops)
  - 00001.ppm, 00002.ppm, 00003.ppm
- PPM big in MB - PNG would be better
  - consider Sean Barrett's stb_image_write.h
- Can use The GIMP to open the series as layers
  - export as .gif
  - lots of big files = runs out of memory
- I used **Image Magick** tools to convert my numbered sequence to a .gif
  - command-line **convert** tool
  - Mac can install with HomeBrew

Variations

- D* - dynamic (nodes can represent on/off state for doors etc)
- Multiple levels of detail grid
    - plan route across country
    - high-resolution path around local obstacles
- Pair with other reactive system for precision steering
    - i used fuzzy controllers
    - interpolate between points / splines
        - less robotic / smoother animations