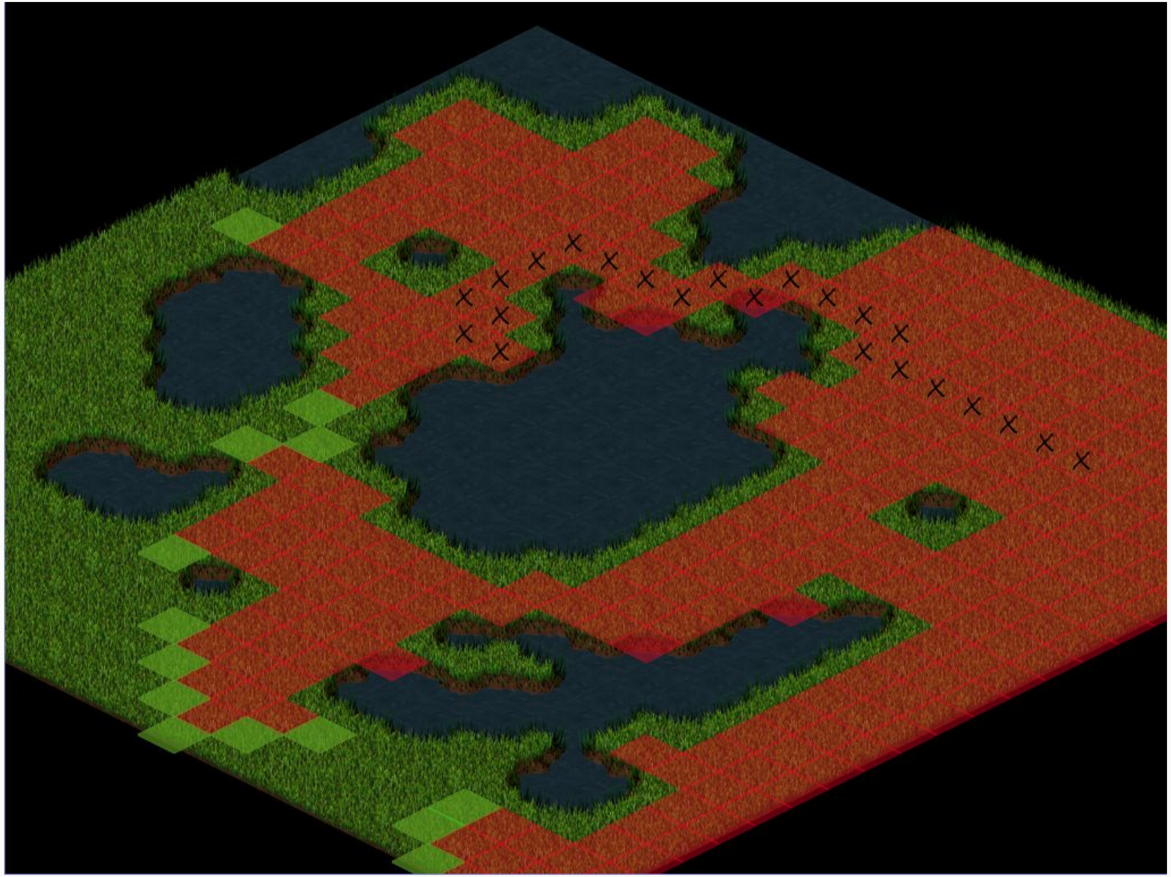
Game dev: BFS to Dijkstra

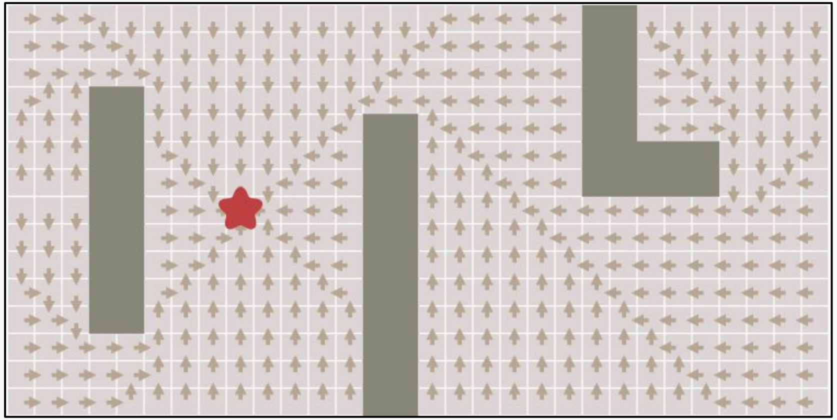
Ricard Pillosu - UPC

Solution BFS



Creating a path out of BFS

- BFS only navigates the whole map
- It’s actually calculating the path to **all** other nodes
- Let’s keep on “the node I come from”
- It should give us a map like



TODO 1

“Record the direction to the previous node with the new list "breadcrumps"

- The list **breadcrumps** is already created
- Note the change of name of few functions
- For each neighbor, remember that you come from “current” cell
- Just one line of code somewhere in the method

Reconstructing the path

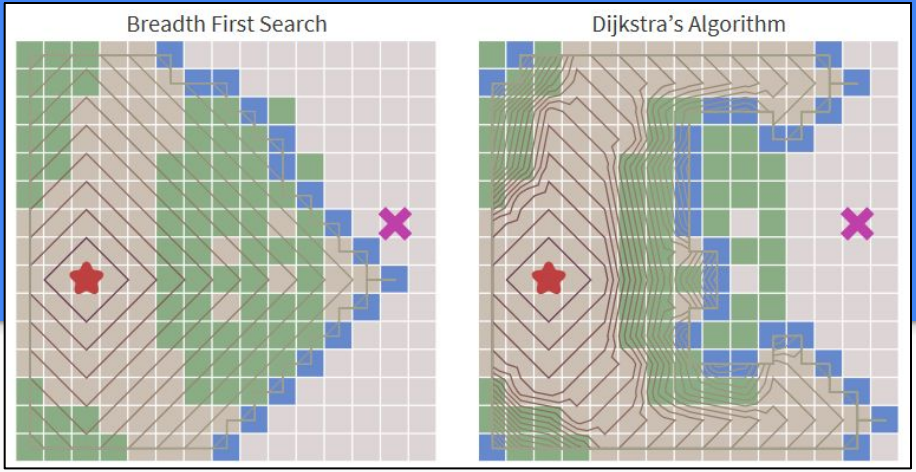
```
current = goal
path = [current]

while current != start:
    current = came_from[current]
    path.append(current)

path.append(start)
```

TODO 2

“Follow the breadcrumps to goal back to the origin add each step into "path" dyn array (it will then draw automatically)"

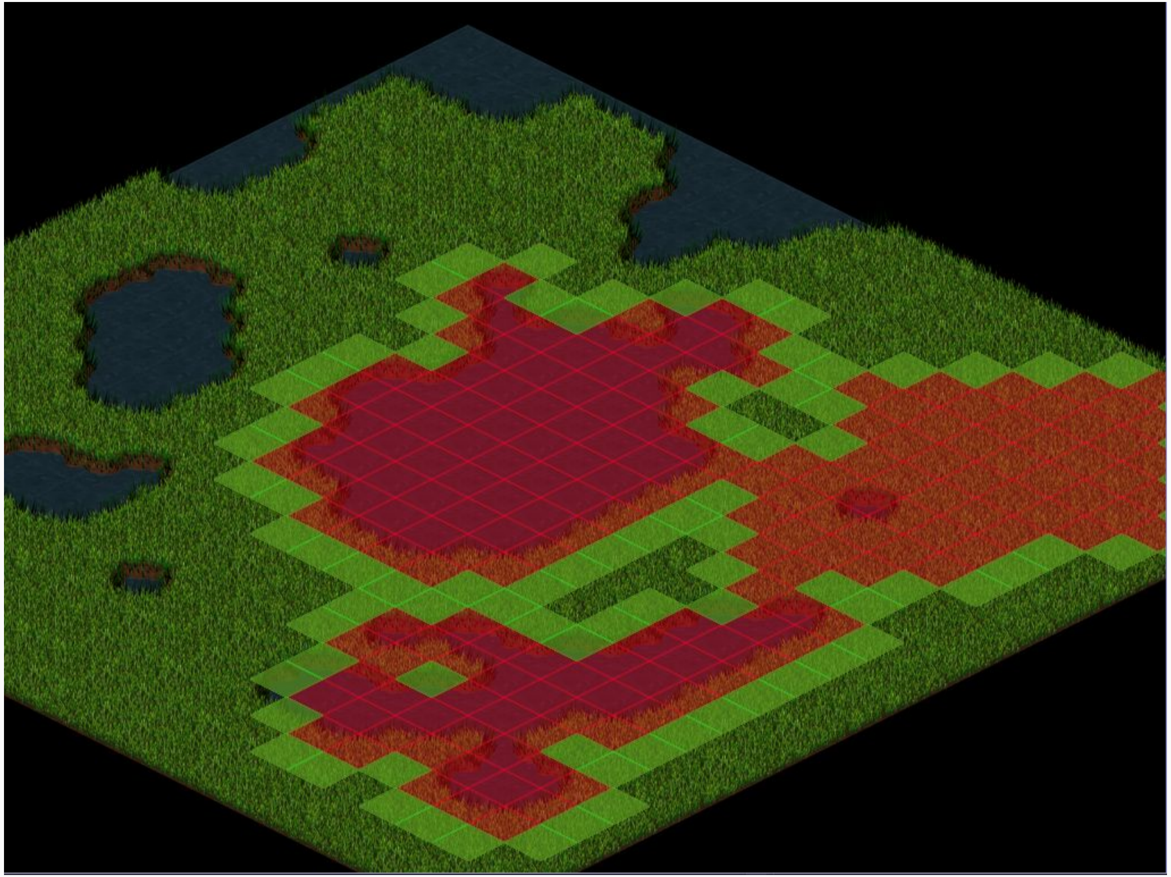
- The dyn array for path already exists
- If filled, it will draw “X” on each tile
- The mouse position when clicked is already calculated for you

Dijkstra

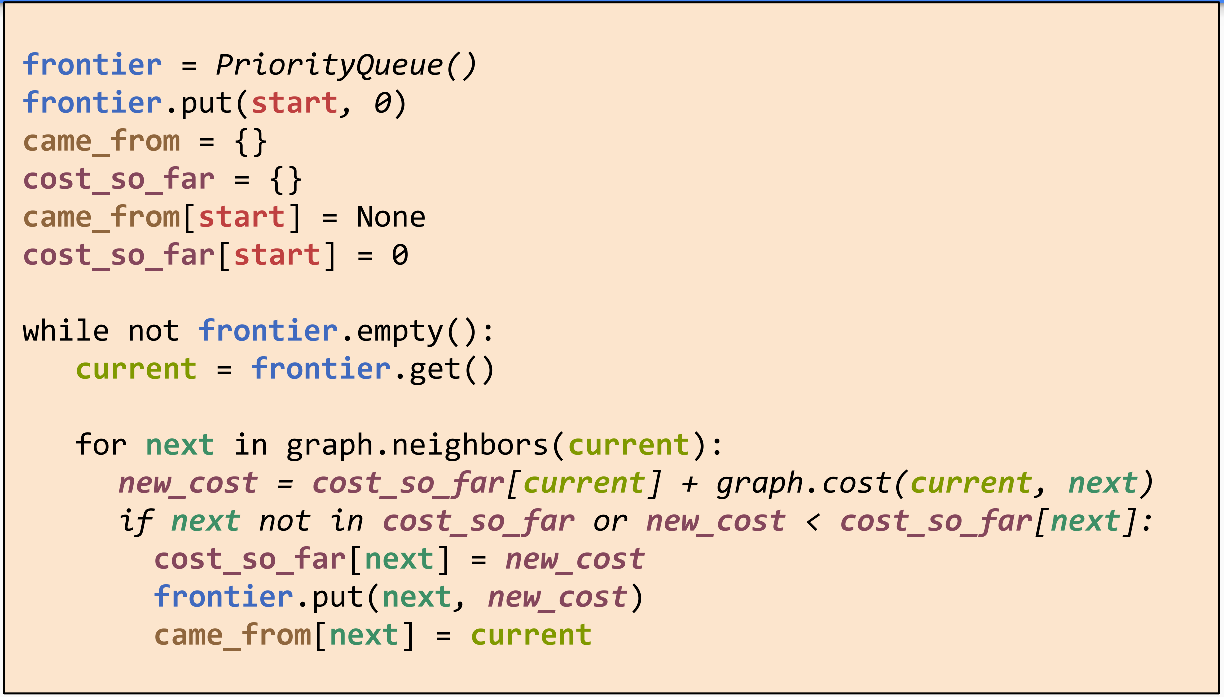


- Expands in all directions like BFS
- But will prefer low cost nodes
- We will simulate that water has a lower cost
- Check solution.exe keys j & k
- We could re-visit a node more than once
- We need to write down in each cell the latest accumulated score

Solution Dijkstra



Dijkstra



```
frontier = PriorityQueue()
frontier.put(start, 0)
came_from = {}
cost_so_far = {}
came_from[start] = None
cost_so_far[start] = 0

while not frontier.empty():
    current = frontier.get()

    for next in graph.neighbors(current):
        new_cost = cost_so_far[current] + graph.cost(current, next)
        if next not in cost_so_far or new_cost < cost_so_far[next]:
            cost_so_far[next] = new_cost
            frontier.put(next, new_cost)
            came_from[next] = current
```

TODO 3

“Taking BFS as a reference, implement the Dijkstra algorithm use the 2 dimensional array "cost_so_far" to track the accumulated costs on each cell (is already reset to 0 automatically)"

- Frontier is already a **priority** queue
- Be sure to understand MovementCost() method
- Cost_so_far is just a big fat array, *just* ok for now :)

Homework

- Try stopping when you reach certain node
- Experiment with an ortographic map with differenttile weigths

Really good article about the three basic navigation methods [here]