

Labyrint-Lab

Laging og løsning av labyrinter i Kotlin

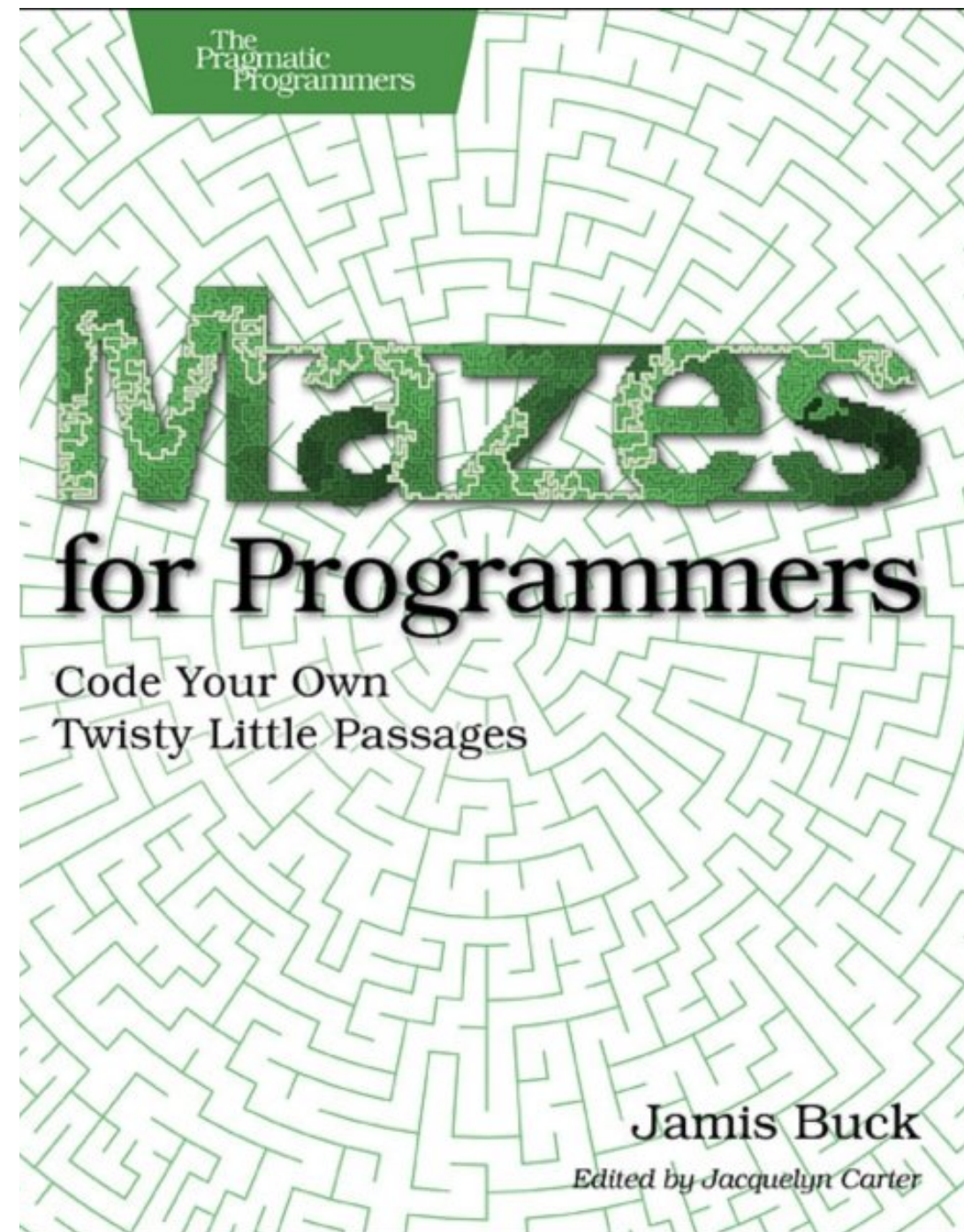
Agenda

🗣️ Sondre yap sesh (still spørsmål underveis)

💻 Det progges

🎄 16:30 - julelandsby utenfor Skur 39

Basert på boka



Teori om labyrinter

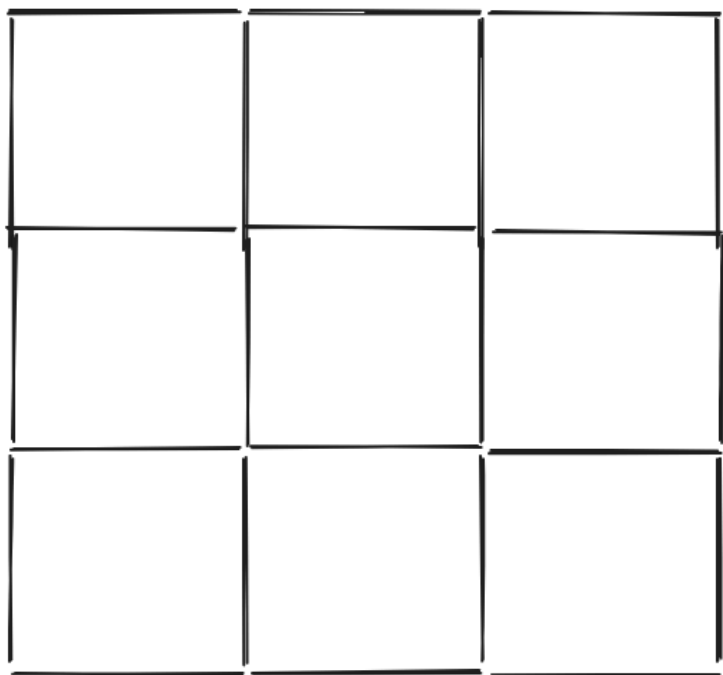
Vi skal lage perfekte labyrinter

Altså, kun én sti mellom to celler

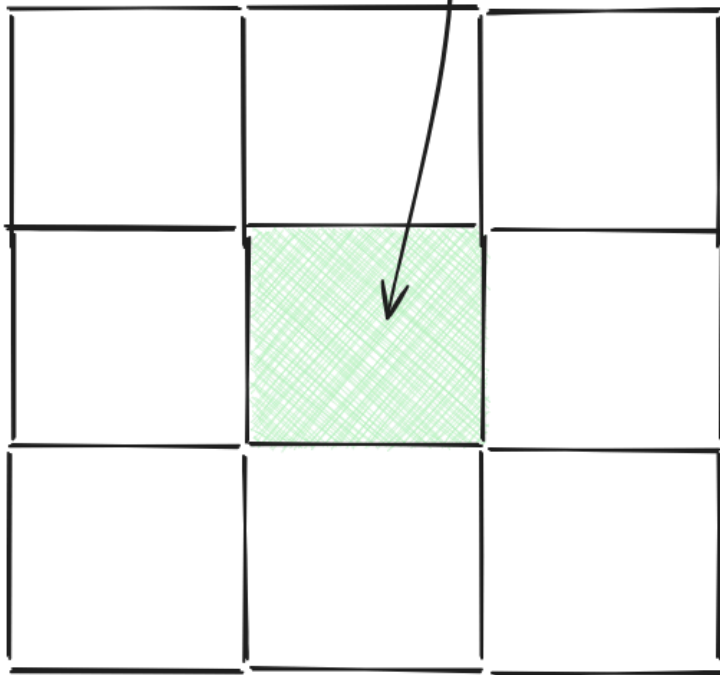
Grafteori-messig betyr dette at labyrintene er trær 🧐

Generell strategi: lag et grid, koble celler sammen

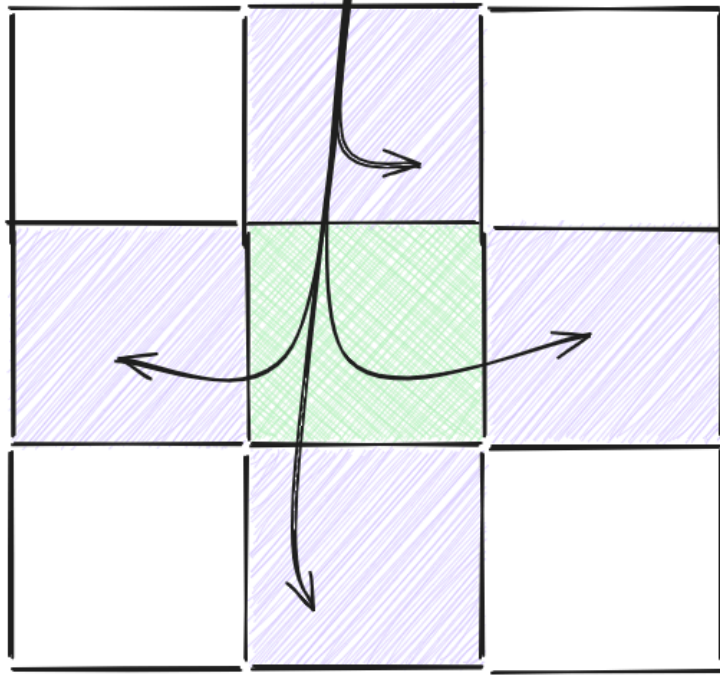
grid

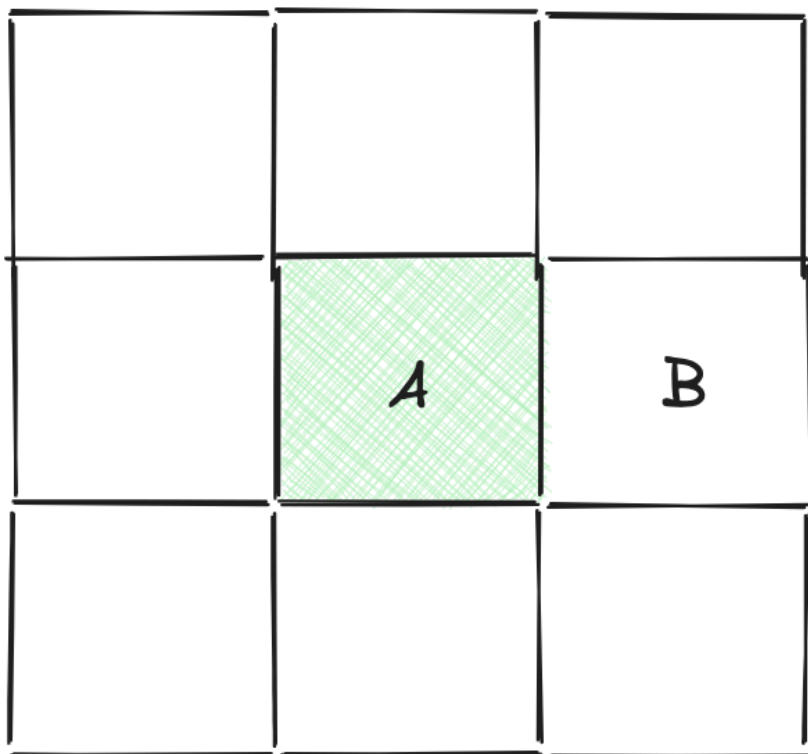


cell



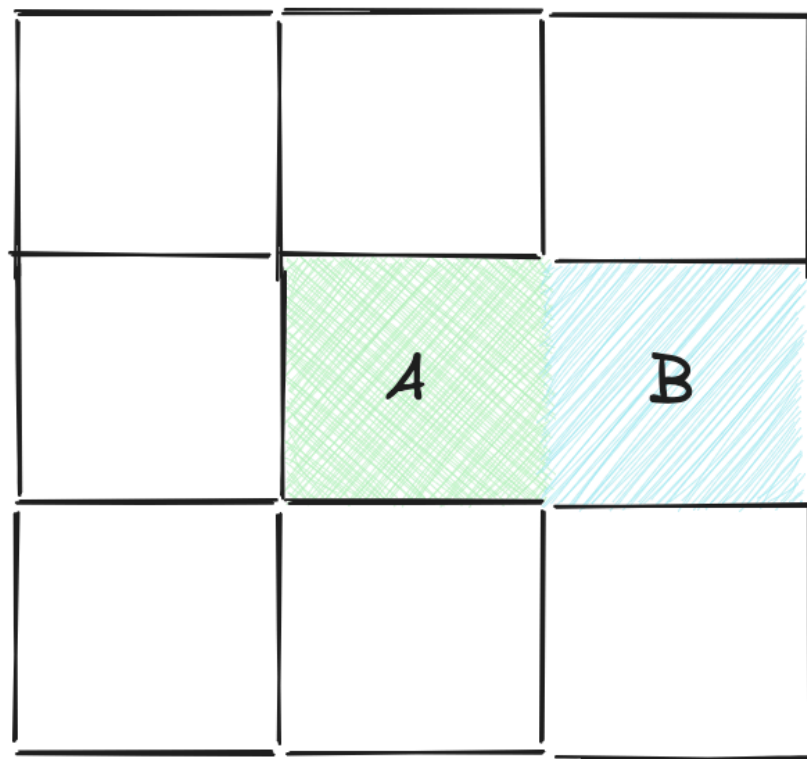
cell.neighbors





`cellA.link(cellB)`

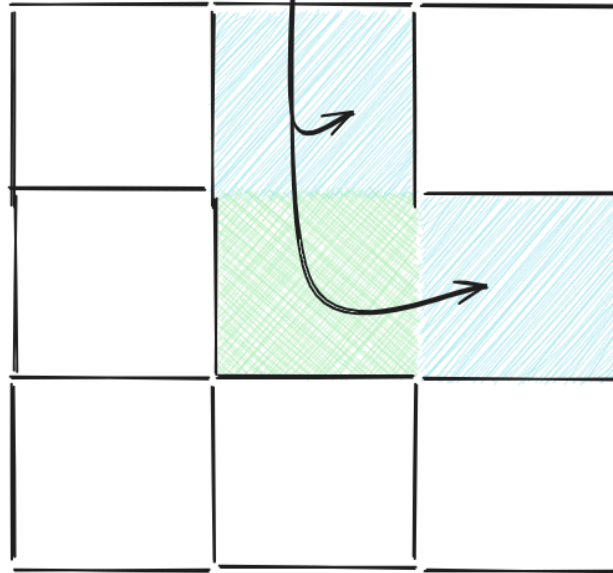
→



`cellA.unlink(cellB)`

←

cell.links



```
abstract class Cell {  
    val links = mutableSetOf<Cell>()  
  
    abstract fun neighbors(): List<Cell>  
  
    fun link(other: Cell, bidirectional: Boolean = true) {  
        links.add(other)  
        if (bidirectional) {  
            other.link(this, false)  
        }  
    }  
  
    fun unlink(other: Cell, bidirectional: Boolean = true) {  
        links.remove(other)  
        if (bidirectional) {  
            other.unlink(this, false)  
        }  
    }  
  
    fun resetLinks() = links.clear()  
  
    fun isLinked(other: Cell?): Boolean {  
        return links.contains(other)  
    }  
}
```

```
abstract class Grid {  
    abstract fun randomCell(): Cell  
  
    abstract val size: Int  
  
    abstract fun cells(): List<Cell>  
  
    fun resetLinks() {  
        cells().forEach { it.resetLinks() }  
    }  
}
```



```
data class SquareCell(val row: Int, val column: Int) : Cell() {  
    var north: SquareCell? = null  
    var east: SquareCell? = null  
    var south: SquareCell? = null  
    var west: SquareCell? = null  
  
    override fun neighbors(): List<SquareCell> {  
        return listOfNotNull(north, east, south, west)  
    }  
}
```

Bekk
B k
B

```
class SquareGrid(val height: Int, val width: Int) : Grid() {

    val grid: List<List<SquareCell>> = List(height) { row ->
        List(width) { column -> SquareCell(row, column) }
    }

    init {
        grid.forEach { row ->
            row.forEach { cell ->
                val r = cell.row
                val c = cell.column

                cell.north = get(r + 1, c)
                cell.east = get(r, c + 1)
                cell.south = get(r - 1, c)
                cell.west = get(r, c - 1)
            }
        }
    }

    override fun cells() = grid.flatten()

    override fun randomCell() = grid.random().random()

    override val size get() = height * width

    fun get(row: Int, column: Int): SquareCell? {
        return grid.getOrNull(row)?.getOrNull(column)
    }
}
```

```
abstract class GridDrawer<T>(protected val shapeRenderer: ShapeRenderer) {  
    abstract fun fill(cell: T)  
    fun fill(cell: T, color: Color) {  
        shapeRenderer.withColor(color) { fill(cell) }  
    }  
  
    abstract fun drawUnlinkedBorders(cell: T)  
    fun drawUnlinkedBorders(cell: T, color: Color) {  
        shapeRenderer.withColor(color) { drawUnlinkedBorders(cell) }  
    }  
  
    abstract fun drawAllBorders(cell: T)  
    fun drawAllBorders(cell: T, color: Color) {  
        shapeRenderer.withColor(color) { drawAllBorders(cell) }  
    }  
}
```



OPPGAVE-IDÉ:

Lag et nytt grid (det ligger forslag med mer info i en README)

```
fun binaryTree(grid: SquareGrid) {  
    grid.cells().forEach {  
        val neighbors = listOfNotNull(it.north, it.east)  
        if (neighbors.isNotEmpty()) {  
            it.link(neighbors.random())  
        }  
    }  
}  
  
class BinaryTree(val grid: SquareGrid) : Iterator<Unit> {  
    private val cellIterator = grid.cells().iterator()  
  
    override fun hasNext(): Boolean {  
        return cellIterator.hasNext()  
    }  
  
    override fun next() {  
        val cell = cellIterator.next()  
        val neighbors = listOfNotNull(cell.north, cell.east)  
        if (neighbors.isNotEmpty()) {  
            cell.link(neighbors.random())  
        }  
    }  
}
```



OPPGAVE-IDÉ:

Implementer en ny algoritme (det ligger forslag med pseudokode i en README)

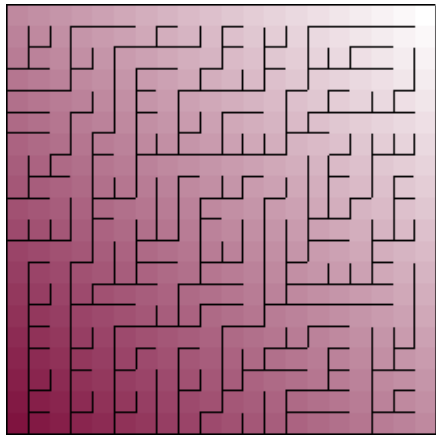
Gjør en eksisterende algoritme animert

Legg på mer state i animert algoritme, og tegn staten!

f.eks. spesiell farge på cellen som nettop ble sjekket

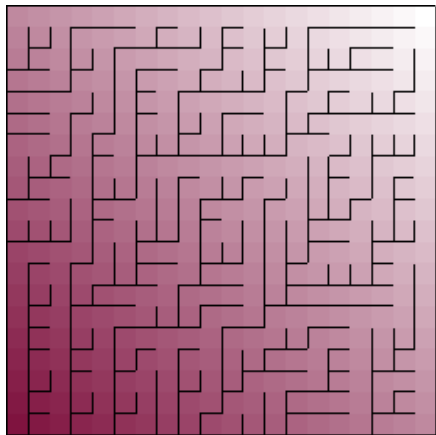
Hvorfor lage flere algoritmer?

Bias og texture

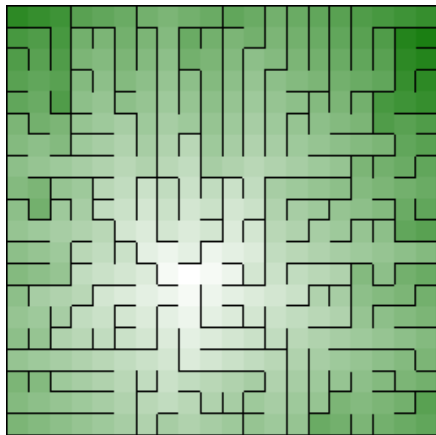


Binary Tree

Bias og texture

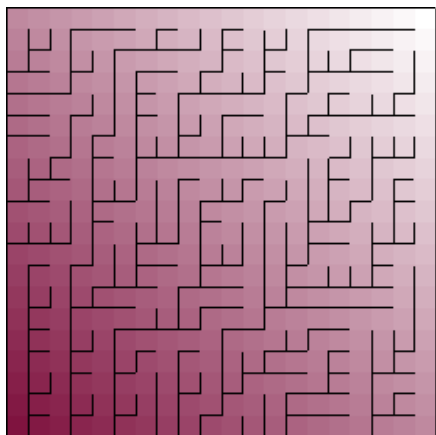


Binary Tree

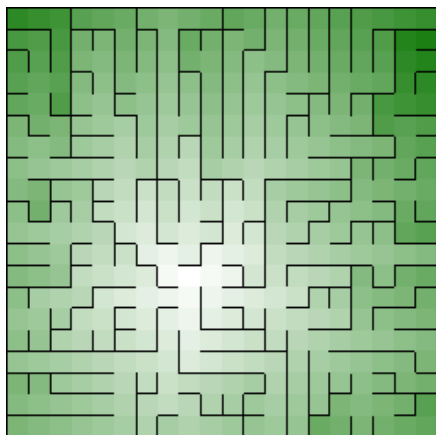


Prim's (forenklet)

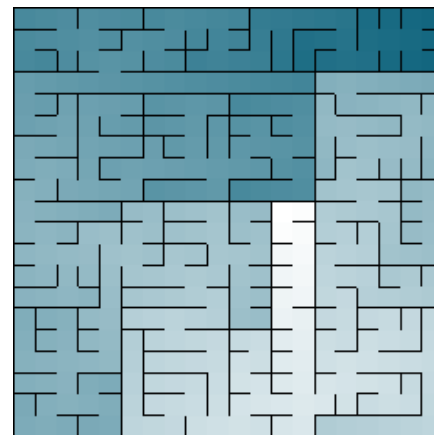
Bias og texture



Binary Tree

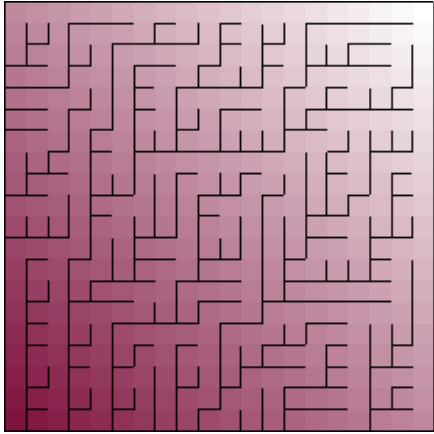


Prim's (forenklet)

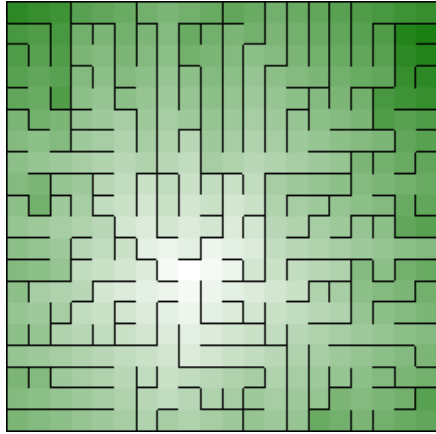


Recursive division

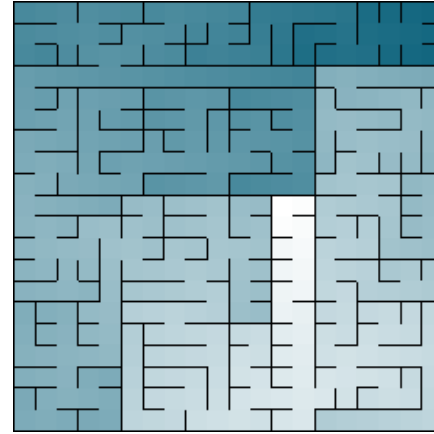
Bias og texture



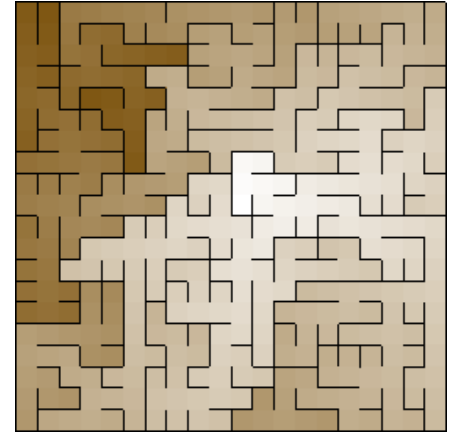
Binary Tree



Prim's (forenklet)



Recursive division



Aldous-Broder

Løsing av labyrinter

DFS, BFS, Dijkstra, A*

Start med en celle, bare følg `cell.links`



OPPGAVE-IDÉ:

Implementer en ny løsningsalgoritme

Gjør den eksisterende algoritmen animert

Scener

Scener

```
class BinaryTree : Scene() {  
    private lateinit var grid: SquareGrid  
    private lateinit var drawer: SquareGridDrawer  
  
    override fun init() {  
        super.init()  
        grid = SquareGrid(20, 20)  
        binaryTree(grid)  
        drawer = SquareGridDrawer(shapeRenderer, grid)  
    }  
  
    override fun draw() {  
        ScreenUtils.clear(Color.WHITE)  
        drawer.drawUnlinkedBorders(Color.GRAY)  
    }  
}
```



OPPGAVE-IDÉ:

Lag nye scener!

Scener

```
class BinaryTree : Scene() {  
    private lateinit var grid: SquareGrid  
    private lateinit var drawer: SquareGridDrawer  
  
    override fun init() {  
        super.init()  
        grid = SquareGrid(20, 20)  
        binaryTree(grid)  
        drawer = SquareGridDrawer(shapeRenderer, grid)  
    }  
  
    override fun draw() {  
        ScreenUtils.clear(Color.WHITE)  
        drawer.drawUnlinkedBorders(Color.GRAY)  
    }  
}
```

```
fun main() {  
    BinaryTree().play()  
}
```



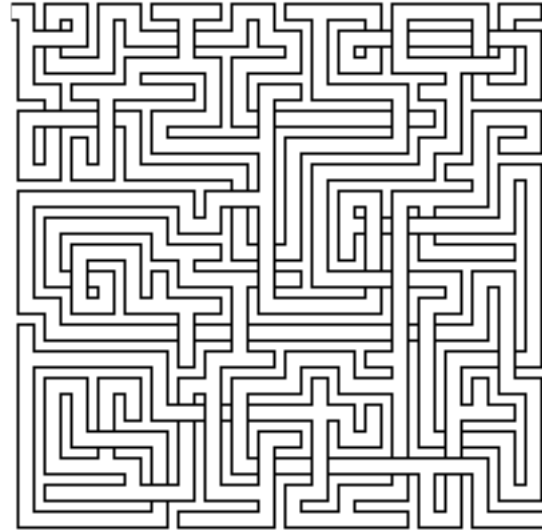
OPPGAVE-IDÉ:
Lag nye scener!

Ting i boka som ikke er i repoet

Weaving

3D grids

Flere algoritmer og grids



Om Copilot

Kjente algoritmer + kode stålet rett fra boka

= Copilot er kjempesterk på dette!

(men kan fortsatt ta feil)

Hvis du vil utforske: skru av Copilot

Hvis du vil komme raskt i mål: skru på Copilot

(og vær forberedt på store completions)



Oppfordring

Det er kjipt å sitte fast!

Spør om hjelp

Hjelp hverandre



Happy hacking!

<https://github.com/sondremb/kotlin-mazes>