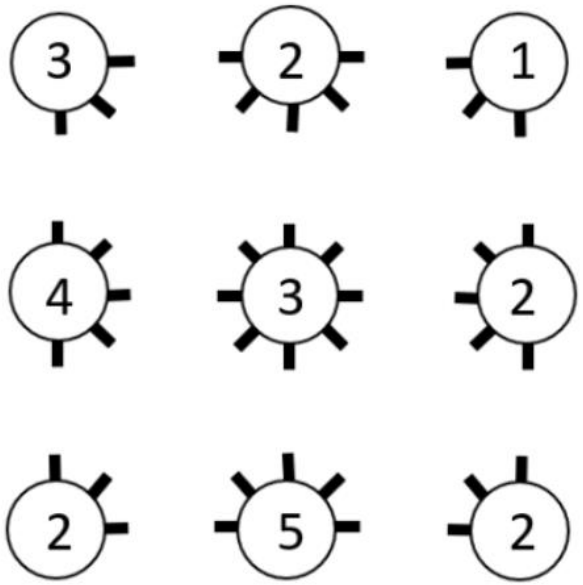
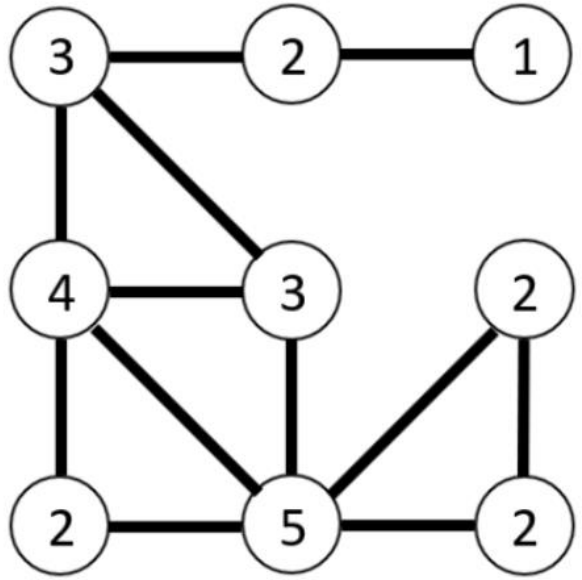
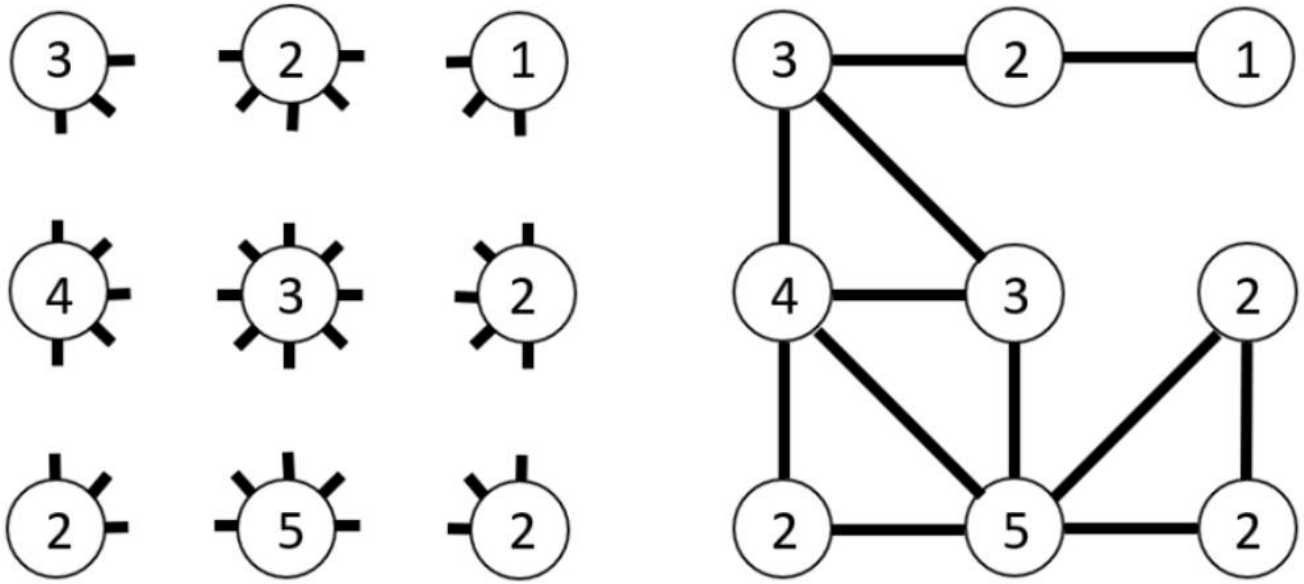


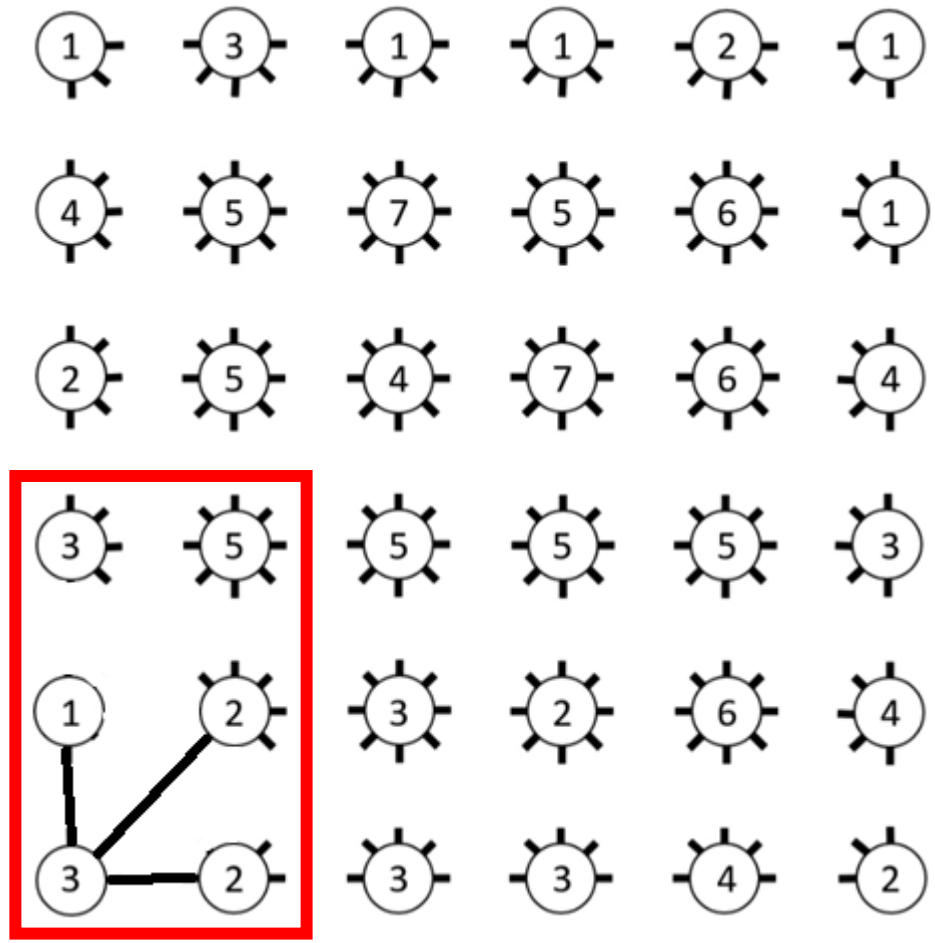


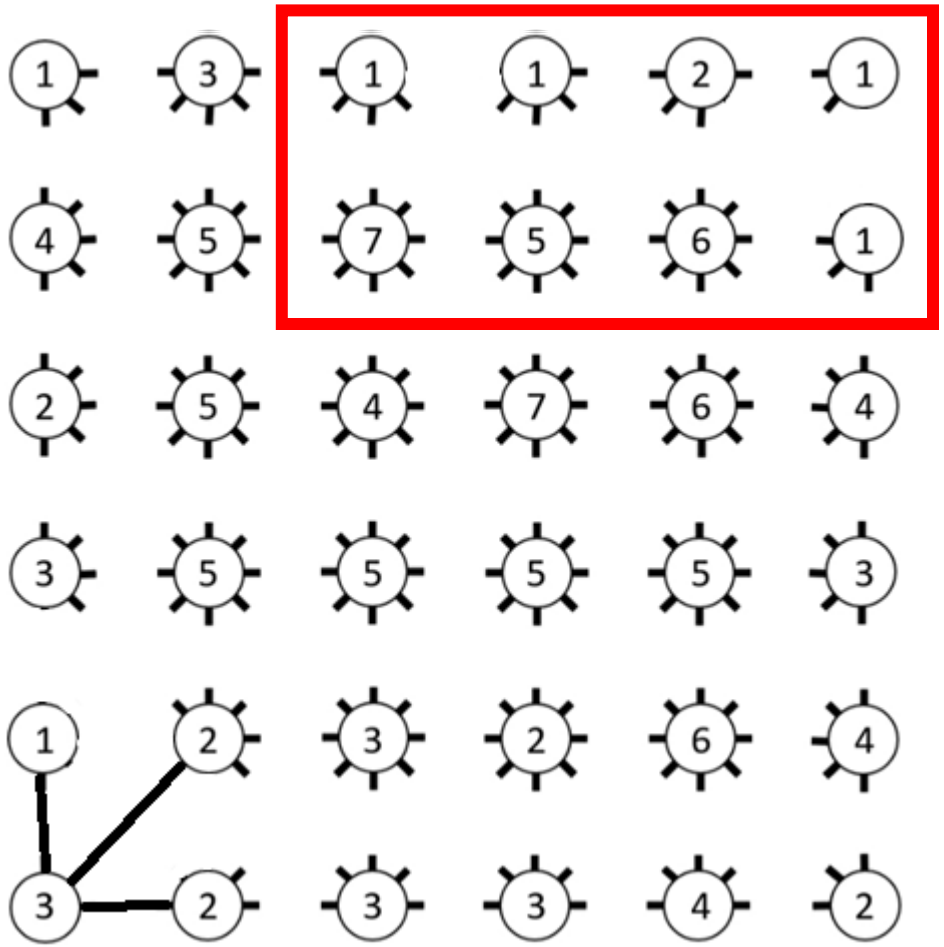
Trying to solve «Spokes» with programming and TDD

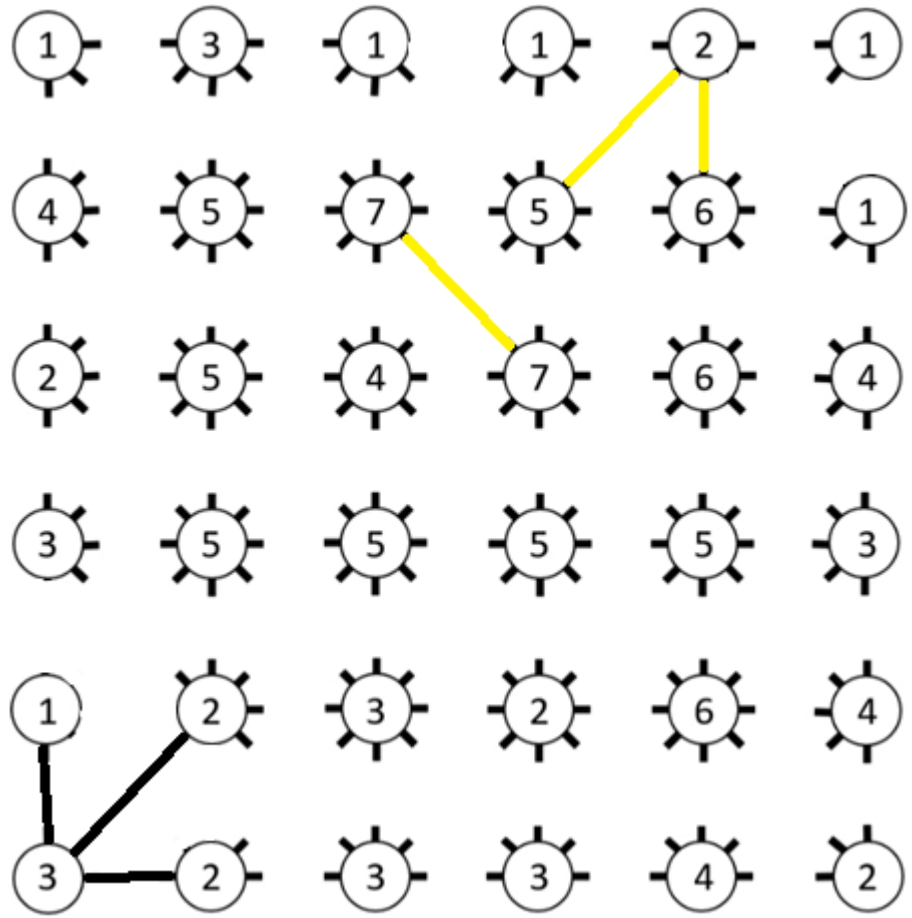
Sacha Peter

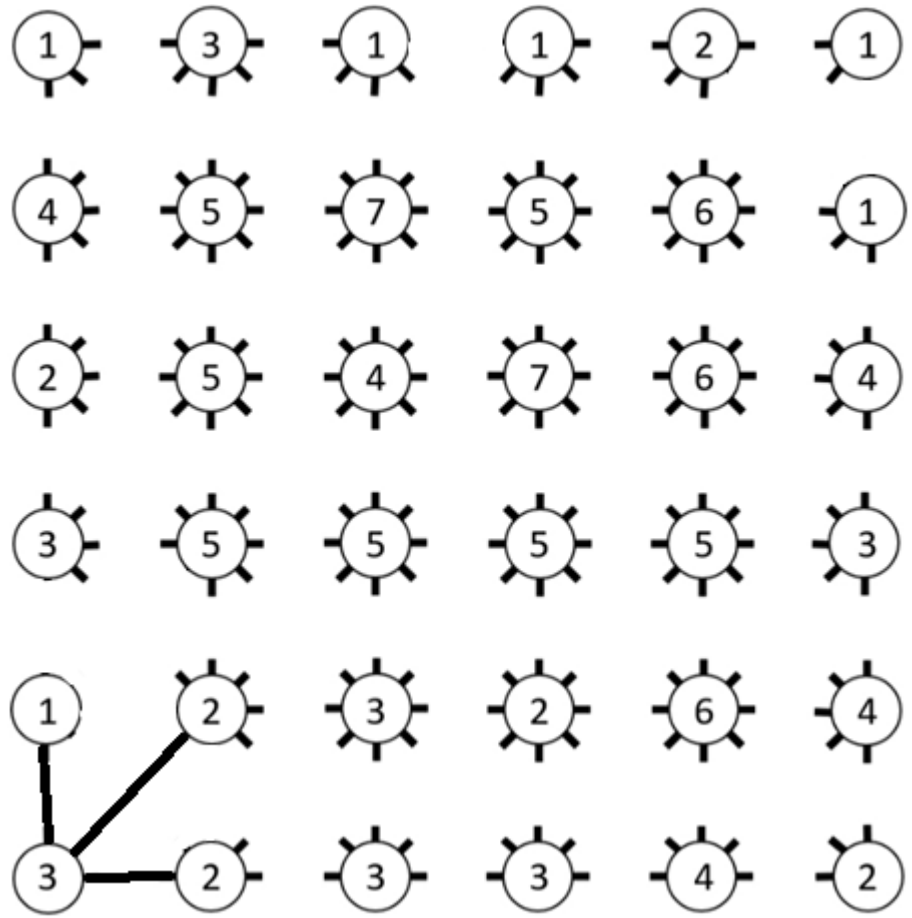
What is „Spokes“

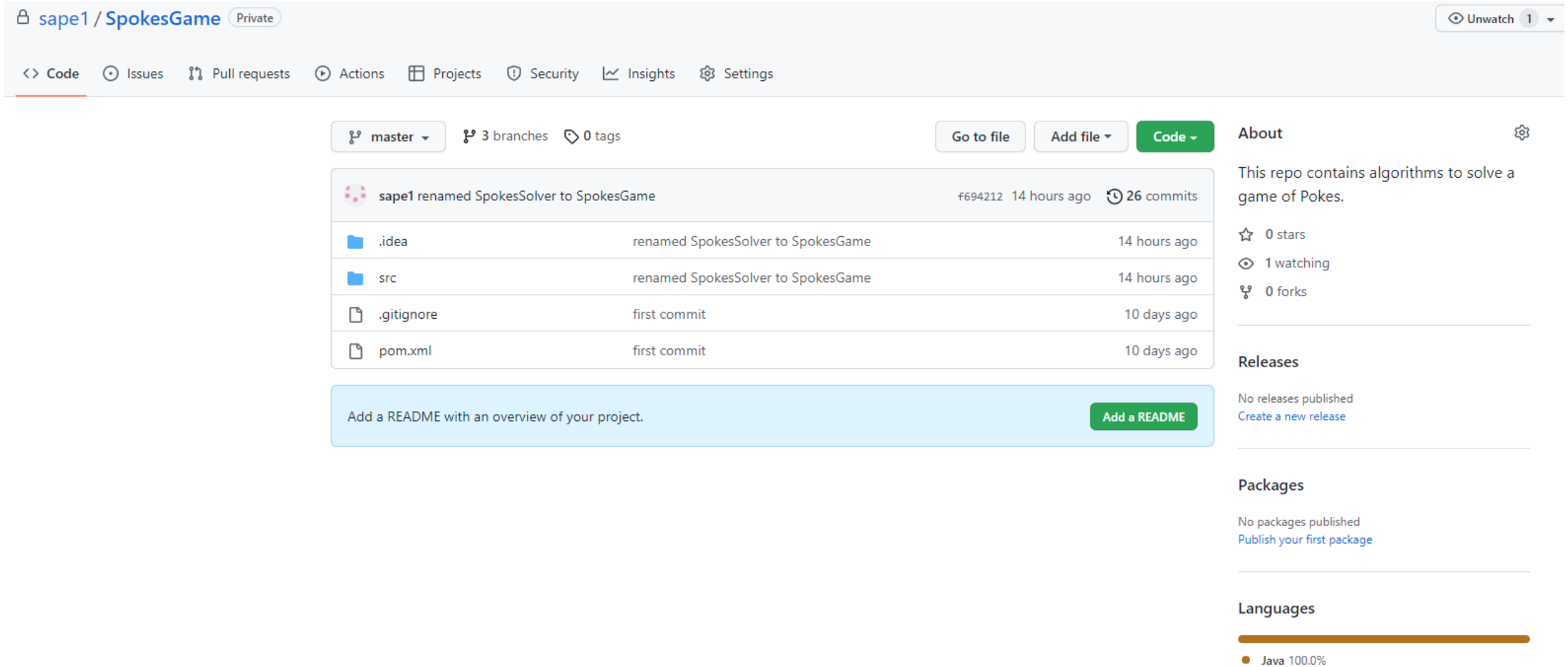



1 3 1 1 2 1
4 5 7 5 6 1
2 5 4 7 6 4
3 5 5 5 5 3
1 2 3 2 6 4
3 2 3 3 4 2


1 3 1 1 2 1
4 5 7 5 6 1
2 5 4 7 6 4
3 5 5 5 5 3
1 2 3 2 6 4
3 2 3 3 4 2


1 3 1 1 2 1
4 5 7 5 6 1
2 5 4 7 6 4
3 5 5 5 5 3
1 2 3 2 6 4
3 2 3 3 4 2


1 3 1 1 2 1
4 5 7 5 6 1
2 5 4 7 6 4
3 5 5 5 5 3
1 2 3 2 6 4
3 2 3 3 4 2


sape1 / SpokesGame
Private
Unwatch 1
Code
Issues
Pull requests
Actions
Projects
Security
Insights
Settings
master
3 branches
0 tags
Go to file
Add file
Code
sape1 renamed SpokesSolver to SpokesGame
f694212 14 hours ago
26 commits
.idea
renamed SpokesSolver to SpokesGame
14 hours ago
src
renamed SpokesSolver to SpokesGame
14 hours ago
.gitignore
first commit
10 days ago
pom.xml
first commit
10 days ago
Add a README with an overview of your project.
Add a README
About
This repo contains algorithms to solve a game of Pokes.
0 stars
1 watching
0 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package
Languages
Java 100.0%

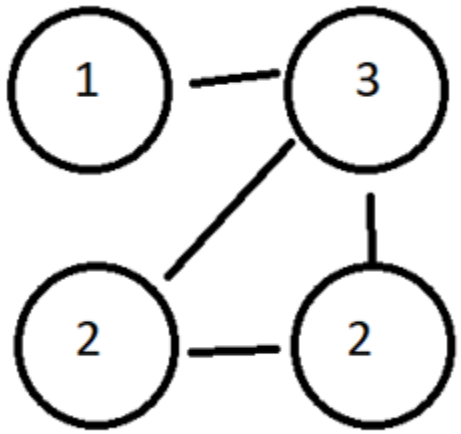
```java
class SpokesGameShould {

    sape1
    @Test
    void createEmptyBoard() {
        SpokesGame spokesGame = new SpokesGame( boardLayout: "");

        Board board = spokesGame.getBoard();
        Map<Coordinate, Node> nodes = board.getNodes();
        assertThat(nodes).isEmpty();
    }
```

```java
public class SpokesGame {

    4 usages
    private final Board board;

    12 usages   sape1
    public SpokesGame(String boardLayout) { board = new Board(boardLayout); }
```

```java
public class Board {
    14 usages
    private Map<Coordinate, Node> nodes = new HashMap<>();
    1 usage   sape1
    public Board(String boardLayout) {
        if (boardLayout.isEmpty()) {
            return;
        }
        intitializeNodes(boardLayout);
    }

    5 usages   sape1
    public Map<Coordinate, Node> getNodes() { return nodes; }
```

```java
@Test
void create2x2Board() {
    SpokesGame spokesGame = new SpokesGame( boardLayout: "2\n2\n13\n22");

    Board board = spokesGame.getBoard();
    Map<Coordinate, Node> actualNodes = board.getNodes();
    Map<Coordinate, Node> expectedNodes = expectedNodesProvider.getExpectedNodesOf2_2_13_22();

    expectedNodes.forEach((coordinate, node) -> assertThat(actualNodes.get(coordinate)).isEqualTo(node));
}
```

```java
public class Board {
    1 usage  new *
    public Board(String boardLayout) {
        if (boardLayout.isEmpty()) {
            return;
        }
        intitializeNodes(boardLayout);
    }
```




```java
private void intitializeNodes(String boardLayout) {
    List<String> input = new LinkedList<>(Arrays.asList(boardLayout.split( regex: "\n")));

    columns = Integer.parseInt(input.get(0));
    input.remove( index: 0);
    rows = Integer.parseInt(input.get(0));
    input.remove( index: 0);

    for (int x = 0; x < columns; x++) {
        for (int y = 0; y < rows; y++) {
            int nodeValue = Integer.parseInt(input.get(y).substring(x, x + 1));
            Node node = createNodeWithValuesAndConnectors(nodeValue, x, y);
            nodes.put(new Coordinate(x, y), node);
        }
    }
}
```

```
@Test
void createConnectorsInA2x2Board()
```

```
@Test
void createConnectorsInA4x4Board()
```

```
@Test
void createLegalSpokeInA2x2Board()
```

```
@Test
void checkIf2x2BoardIsSolved()
```

```
@Test
void checkIf2x2BoardIsNotSolved()
```

```
@Test
void checkIf4x4BoardIsSolved()
```

```
@Test
void throwAnExceptionForPlacingASpokeToItself()
```

```
@Test
void throwAnExceptionForPlacingASpokeForNodesThatAreNotNeighbours()
```

```
@Test
void throwAnExceptionForPlacingASpokeForAConnectorThatDoesNotExist()
```

```
@Test
void throwAnExceptionForPlacingADiagonalSpokeOverAnotherSpoke()
```

```java
class BrutusShould {

    sape1 *
    @Test
    void solveA2x2Board(){
        Brutus brutus = new Brutus( boardLayout: "2\n2\n13\n22");

        brutus.solve();

        SpokesGame spokesGame = brutus.getSolution();
        boolean isSolved = spokesGame.isSolved();
        assertThat(isSolved).isTrue();
    }
```

k = value	n = connectors	possibilites: n! / (k! * (n-k)!)
1	3	3
3	5	10
1	5	5
1	5	5
2	5	10
1	3	3
4	5	5
5	8	56
7	8	8
5	8	56
6	8	28
1	5	5
2	5	10
5	8	56
4	8	70
7	8	8
6	8	28
4	5	5
3	5	10
5	8	56
5	8	56
5	8	56
5	8	56
3	5	10
1	5	5
2	8	28
3	8	56
2	8	28
6	8	28
4	5	5
3	3	1
2	5	10
3	5	10
3	5	10
4	5	5
2	3	3
	sum:	7.86495E+39

5 billions variants per second with 12 cores

5x 10^{21} years


Q&A


THANK
YOU
GRACIAS
ARIGATO
SHUKURIA
BOLZÏN
MERCI
BÏYAN
DANKSCHEEN
SHUKRIA
TINGKI
YAQHANYELAY
TASHAKKUR ATU
SUKSAMA
EKHMET
MEHRBANI
GRAZIE
MAAKE
KOMAPSUMNIDA
JUSPAXAR
GOZAIMASHITA
EFCHARISTO
PALDIES
SPASIBO
DENKUJA
NENACHALHYA
SINACHALHUYA
NUHUN
CHALTU
SPASSIBO
WABEEJA
HAITEKA
YUSPAGARBAYAM
HUI
ATTO
ANIHA
DHANYABAAD
GUI
HATUR
UNALCHÉESH
EKOJU
SIKOMO
MAKETAI
MINMONCHAR
SANCO
MERASTAHHY
GAEJTHO
LAH
TAVTAPUCH
MEDAWAGSE
AGUYJE
FAKAAUE
DAIKA

Keep in touch

- linkedin.com/in/sacha-peter/
- sacha.peter@css.ch
- github.com/sape1

References

- Mensa International https://en.wikipedia.org/wiki/Mensa_International
- Mensa Switzerland https://mensa.ch/