GWV Projekt: 15-Puzzle

F. Degenhardt, B. Cordt, P. Hölzen

Grid Repräsentation

- “CSV” Format
  - 1;2;3;4 [...] 13;14;15;
- grid = [x.strip().split(";") for x in lines]
  - Leere Kachel empty string

possible_moves I

```
def possible_moves(grid):
    x, y = find(grid)
    ret = []
    if y != 0:
        ret += 'u'
    if y != len(grid)-1:
        ret += 'd'
```

possible_moves II

```
[...]
if x != 0:
        ret += 'l'
    if x != len(grid[0])-1:
        ret += 'r'
    return ret
```

move I

```
def move(grid, d):
    ret = copy.deepcopy(grid)
    x, y = find(grid)
    if d == 'u':
            ret[y][x], ret[y-1][x] = ret[y-1][x], ret[y][x]
    elif d == 'd':
            ret[y][x], ret[y+1][x] = ret[y+1][x], ret[y][x]
```

move II

```
[...]
elif d == 'l':
        ret[y][x], ret[y][x-1] = ret[y][x-1], ret[y][x]
    elif d == 'r':
        ret[y][x], ret[y][x+1] = ret[y][x+1], ret[y][x]
    return ret
```

heuristic

```
def heuristic(grid):
    s = 0
    for x, y in [(x, y) for x in range(4) for y in range(4)]:
        try:
            n = int(grid[y][x])
        except ValueError:
            n = 16
        s += abs(y - ((n-1)/4)) + abs(x - ((n-1)%4))
    return s
```

f_add / search I

- def f_add(frontier, path):

    return sorted(frontier + [path], key=lambda p: heuristic(p[-1]) + len(p))

- def search(grid):

    frontier = [[grid]]

    count = 0

    while frontier:

        count += 1

        path = frontier[0]

        del frontier[0]

        current_grid = path[-1]

search II

```
[...]
if heuristic(current_grid) == 0:
        for g in path: print_grid(g)
        return True
for m in possible_moves(current_grid):
        next_grid = move(current_grid, m)
        if not next_grid in path:
            frontier = f_add(frontier, path + [next_grid])
```