hw2

January 31, 2025

0.1 hw2 - searching the burning building

[16]:
```
from hw2a import *
from notebook4e import *
import numpy as np
```

[17]:
```
main()
```

```
hours
# is it greater than 0?
 OK  got: 2 expected: <function main.<locals>.<lambda> at 0x7f4d189ea9e0>
temp(node)
 OK  got: 70 expected: 70
 OK  got: 500 expected: 500
 OK  got: 60 expected: 60
No temp for node:  ENTRANCE
 OK  got: None expected: None
room_depth_first
 OK  got: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20'], 125, 26)
expected: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20'], 125, 26)
 OK  got: (['S30', 'S40', 'H40', 'H41', 'H42', 'H43', 'H44', 'H45', 'H46',
'H47', 'H48', 'H49', 'S49', 'S39', 'S29', 'S19', 'EXIT'], 115, 21) expected:
(['S30', 'S40', 'H40', 'H41', 'H42', 'H43', 'H44', 'H45', 'H46', 'H47', 'H48',
'H49', 'S49', 'S39', 'S29', 'S19', 'EXIT'], 115, 21)
 OK  got: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'H30', 'H31',
'H32', 'H33', 'H34', 'H35', 'R35'], 150, 68) expected: (['S19', 'S29', 'S39',
'S49', 'H49', 'H48', 'H47', 'H46', 'R46', 'R45', 'R44', 'H44', 'H43', 'H42',
'H41', 'H40', 'S40', 'S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'H35', 'R35'],
150, 68)
 OK  got: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20'], 125, 26)
expected: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20'], 125, 26)
room_breadth_first
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
```

```
'H11', 'H10', 'S10', 'S20'], 75, 77) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'S10', 'S20'], 75, 77)
 OK  got: (['H20', 'H21', 'H22', 'H23', 'H24', 'H25', 'H26', 'H27', 'H28',
'H29', 'S29', 'S19', 'EXIT'], 75, 83) expected: (['H20', 'H21', 'H22', 'H23',
'H24', 'H25', 'H26', 'H27', 'H28', 'H29', 'S29', 'S19', 'EXIT'], 75, 83)
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'H35', 'R35'], 60,
51) expected: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'H35', 'R35'],
60, 51)
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20'], 75, 77) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'S10', 'S20'], 75, 77)
room_best_first
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20'], 75, 23) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'S10', 'S20'], 75, 23)
 OK  got: (['S10', 'H10', 'H11', 'H12', 'H13', 'H14', 'H15', 'H16', 'H17',
'H18', 'H19', 'S19', 'EXIT'], 75, 23) expected: (['S10', 'H10', 'H11', 'H12',
'H13', 'H14', 'H15', 'H16', 'H17', 'H18', 'H19', 'S19', 'EXIT'], 75, 23)
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20', 'S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'H35',
'R35'], 120, 85) expected: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14',
'H13', 'H12', 'H11', 'H10', 'S10', 'S20', 'S30', 'H30', 'H31', 'H32', 'H33',
'H34', 'H35', 'R35'], 120, 85)
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20'], 75, 23) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'S10', 'S20'], 75, 23)
room_worst_first
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35',
'R34', 'H34', 'H33', 'H32', 'H31', 'H30', 'S30', 'S20'], 105, 36) expected:
(['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35', 'R34', 'H34',
'H33', 'H32', 'H31', 'H30', 'S30', 'S20'], 105, 36)
 OK  got: (['S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'R34', 'R35', 'R36',
'H36', 'H37', 'H38', 'H39', 'S39', 'S29', 'S19', 'EXIT'], 105, 85) expected:
(['S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'R34', 'R35', 'R36', 'H36', 'H37',
'H38', 'H39', 'S39', 'S29', 'S19', 'EXIT'], 105, 85)
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'], 60,
10) expected: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'],
60, 10)
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35',
'R34', 'H34', 'H33', 'H32', 'H31', 'H30', 'S30', 'S20'], 105, 36) expected:
(['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35', 'R34', 'H34',
'H33', 'H32', 'H31', 'H30', 'S30', 'S20'], 105, 36)
room_astar
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20'], 75, 18) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'S10', 'S20'], 75, 18)
 OK  got: (['H20', 'H21', 'H22', 'H23', 'H24', 'H25', 'H26', 'H27', 'H28',
'H29', 'S29', 'S19', 'EXIT'], 75, 14) expected: (['H20', 'H21', 'H22', 'H23',
```

```
'H24', 'H25', 'H26', 'H27', 'H28', 'H29', 'S29', 'S19', 'EXIT'], 75, 14)

debug examples

 OK  got: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20', 'S10',
'H10', 'H11', 'H12', 'H13', 'H14', 'H15', 'R15'], 170, 34, ['EXIT', 'S19',
'S29', 'S39', 'S49', 'H49', 'R49', 'H48', 'R48', 'H47', 'R47', 'H46', 'R46',
'R45', 'R44', 'H44', 'H43', 'R43', 'R42', 'H42', 'H41', 'R41', 'H40', 'S40',
'S30', 'S20', 'S10', 'H10', 'H11', 'H12', 'H13', 'H14', 'H15', 'R15']) expected:
(['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46', 'R45', 'R44',
'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'S20', 'S10', 'H10', 'H11',
'H12', 'H13', 'H14', 'H15', 'R15'], 170, 34, ['EXIT', 'S19', 'S29', 'S39',
'S49', 'H49', 'R49', 'H48', 'R48', 'H47', 'R47', 'H46', 'R46', 'R45', 'R44',
'H44', 'H43', 'R43', 'R42', 'H42', 'H41', 'R41', 'H40', 'S40', 'S30', 'S20',
'S10', 'H10', 'H11', 'H12', 'H13', 'H14', 'H15', 'R15'])
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 31, ['EXIT',
'S19', 'H19', 'S29', 'H18', 'R19', 'H29', 'S39', 'H17', 'R18', 'H28', 'R29',
'H39', 'S49', 'H16', 'R17', 'H27', 'R28', 'H38', 'R39', 'H49', 'H15', 'R16',
'H26', 'R27', 'H37', 'R38', 'H48', 'R49', 'H14', 'R15']) expected: (['S19',
'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 31, ['EXIT', 'S19', 'H19', 'S29',
'H18', 'R19', 'H29', 'S39', 'H17', 'R18', 'H28', 'R29', 'H39', 'S49', 'H16',
'R17', 'H27', 'R28', 'H38', 'R39', 'H49', 'H15', 'R16', 'H26', 'R27', 'H37',
'R38', 'H48', 'R49', 'H14', 'R15'])
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 17, ['EXIT',
'S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12', 'H11', 'H10',
'R11', 'R12', 'R13', 'R14', 'R15']) expected: (['S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'R15'], 40, 17, ['EXIT', 'S19', 'H19', 'H18', 'H17', 'H16', 'H15',
'H14', 'H13', 'H12', 'H11', 'H10', 'R11', 'R12', 'R13', 'R14', 'R15'])
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 67, ['EXIT',
'S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35', 'R34', 'H34',
'H35', 'H33', 'H32', 'H31', 'H30', 'R31', 'R32', 'R33', 'R37', 'R38', 'R39',
'S30', 'H29', 'H28', 'H27', 'H26', 'H25', 'R25', 'H24', 'H23', 'H22', 'H21',
'H20', 'S20', 'S40', 'H40', 'H41', 'H42', 'H43', 'H44', 'R44', 'R45', 'R46',
'H45', 'H46', 'H47', 'H48', 'H49', 'R49', 'S49', 'H19', 'H18', 'H17', 'H16',
'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'R11', 'R12', 'R13', 'R14', 'R15'])
expected: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 67, ['EXIT',
'S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35', 'R34', 'H34',
'H35', 'H33', 'H32', 'H31', 'H30', 'R31', 'R32', 'R33', 'R37', 'R38', 'R39',
'S30', 'H29', 'H28', 'H27', 'H26', 'H25', 'R25', 'H24', 'H23', 'H22', 'H21',
'H20', 'S20', 'S40', 'H40', 'H41', 'H42', 'H43', 'H44', 'R44', 'R45', 'R46',
'H45', 'H46', 'H47', 'H48', 'H49', 'R49', 'S49', 'H19', 'H18', 'H17', 'H16',
'H15', 'H14', 'H13', 'H12', 'H11', 'H10', 'R11', 'R12', 'R13', 'R14', 'R15'])
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15'], 40, 8, ['EXIT',
'S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'R15']) expected: (['S19', 'H19',
'H18', 'H17', 'H16', 'H15', 'R15'], 40, 8, ['EXIT', 'S19', 'H19', 'H18', 'H17',
'H16', 'H15', 'R15'])
 OK  got: (['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46',
```

```
'R45', 'R44', 'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'H30', 'H31',
'H32', 'H33', 'H34', 'H35', 'R35'], 150, 68, ['EXIT', 'S19', 'S29', 'S39',
'S49', 'H49', 'R49', 'H48', 'R48', 'H47', 'R47', 'H46', 'R46', 'R45', 'R44',
'H44', 'H43', 'R43', 'R42', 'H42', 'H41', 'R41', 'H40', 'S40', 'S30', 'S20',
'S10', 'H10', 'H11', 'H12', 'H13', 'H14', 'H15', 'R15', 'R16', 'H16', 'H17',
'H18', 'R18', 'R17', 'R14', 'R13', 'R12', 'R11', 'H20', 'H21', 'H22', 'H23',
'H24', 'H25', 'R25', 'R26', 'H26', 'H27', 'H28', 'R28', 'R27', 'R24', 'R23',
'R22', 'R21', 'H30', 'H31', 'H32', 'H33', 'H34', 'H35', 'R35']) expected:
(['S19', 'S29', 'S39', 'S49', 'H49', 'H48', 'H47', 'H46', 'R46', 'R45', 'R44',
'H44', 'H43', 'H42', 'H41', 'H40', 'S40', 'S30', 'H30', 'H31', 'H32', 'H33',
'H34', 'H35', 'R35'], 150, 68, ['EXIT', 'S19', 'S29', 'S39', 'S49', 'H49',
'R49', 'H48', 'R48', 'H47', 'R47', 'H46', 'R46', 'R45', 'R44', 'H44', 'H43',
'R43', 'R42', 'H42', 'H41', 'R41', 'H40', 'S40', 'S30', 'S20', 'S10', 'H10',
'H11', 'H12', 'H13', 'H14', 'H15', 'R15', 'R16', 'H16', 'H17', 'H18', 'R18',
'R17', 'R14', 'R13', 'R12', 'R11', 'H20', 'H21', 'H22', 'H23', 'H24', 'H25',
'R25', 'R26', 'H26', 'H27', 'H28', 'R28', 'R27', 'R24', 'R23', 'R22', 'R21',
'H30', 'H31', 'H32', 'H33', 'H34', 'H35', 'R35'])
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'H35', 'R35'], 60,
51, ['EXIT', 'S19', 'H19', 'S29', 'H18', 'R19', 'H29', 'S39', 'H17', 'R18',
'H28', 'R29', 'H39', 'S49', 'H16', 'R17', 'H27', 'R28', 'H38', 'R39', 'H49',
'H15', 'R16', 'H26', 'R27', 'H37', 'R38', 'H48', 'R49', 'H14', 'R15', 'H25',
'R26', 'H36', 'R37', 'H47', 'R48', 'H13', 'R14', 'H24', 'R25', 'H35', 'R36',
'H46', 'R47', 'H12', 'R13', 'H23', 'R24', 'H34', 'R35']) expected: (['S19',
'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'H35', 'R35'], 60, 51, ['EXIT', 'S19',
'H19', 'S29', 'H18', 'R19', 'H29', 'S39', 'H17', 'R18', 'H28', 'R29', 'H39',
'S49', 'H16', 'R17', 'H27', 'R28', 'H38', 'R39', 'H49', 'H15', 'R16', 'H26',
'R27', 'H37', 'R38', 'H48', 'R49', 'H14', 'R15', 'H25', 'R26', 'H36', 'R37',
'H47', 'R48', 'H13', 'R14', 'H24', 'R25', 'H35', 'R36', 'H46', 'R47', 'H12',
'R13', 'H23', 'R24', 'H34', 'R35'])
 OK  got: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20', 'S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'H35',
'R35'], 120, 85, ['EXIT', 'S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14',
'H13', 'H12', 'H11', 'H10', 'R11', 'R12', 'R13', 'R14', 'R15', 'R16', 'R17',
'R18', 'R19', 'S10', 'S20', 'H20', 'H21', 'R21', 'H22', 'R22', 'R23', 'H23',
'H24', 'R24', 'H25', 'H26', 'R26', 'H27', 'R27', 'H28', 'R28', 'H29', 'R29',
'S29', 'R25', 'S30', 'S40', 'H40', 'H41', 'R41', 'H42', 'R42', 'R43', 'H43',
'H44', 'H45', 'H46', 'H47', 'R47', 'H48', 'R48', 'H49', 'R49', 'S49', 'H30',
'H31', 'H32', 'H33', 'R31', 'R32', 'R33', 'R44', 'R46', 'S39', 'H39', 'H38',
'H37', 'R37', 'R38', 'R39', 'H34', 'H35', 'H36', 'R45', 'R34', 'R36', 'R35'])
expected: (['S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14', 'H13', 'H12',
'H11', 'H10', 'S10', 'S20', 'S30', 'H30', 'H31', 'H32', 'H33', 'H34', 'H35',
'R35'], 120, 85, ['EXIT', 'S19', 'H19', 'H18', 'H17', 'H16', 'H15', 'H14',
'H13', 'H12', 'H11', 'H10', 'R11', 'R12', 'R13', 'R14', 'R15', 'R16', 'R17',
'R18', 'R19', 'S10', 'S20', 'H20', 'H21', 'R21', 'H22', 'R22', 'R23', 'H23',
'H24', 'R24', 'H25', 'H26', 'R26', 'H27', 'R27', 'H28', 'R28', 'H29', 'R29',
'S29', 'R25', 'S30', 'S40', 'H40', 'H41', 'R41', 'H42', 'R42', 'R43', 'H43',
'H44', 'H45', 'H46', 'H47', 'R47', 'H48', 'R48', 'H49', 'R49', 'S49', 'H30',
'H31', 'H32', 'H33', 'R31', 'R32', 'R33', 'R44', 'R46', 'S39', 'H39', 'H38',
```

```
'H37', 'R37', 'R38', 'R39', 'H34', 'H35', 'H36', 'R45', 'R34', 'R36', 'R35'])
 OK  got: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'], 60,
10, ['EXIT', 'S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'])
expected: (['S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'], 60,
10, ['EXIT', 'S19', 'S29', 'S39', 'H39', 'H38', 'H37', 'H36', 'R36', 'R35'])
```

[18]:
```
np.int = int        # Turnaround to resolve deprecated conflict
# import networkx as nx
```

[19]:
```
%matplotlib inline
import networkx as nx
import matplotlib.pyplot as plt
from matplotlib import lines

from ipywidgets import interact
import ipywidgets as widgets
from IPython.display import display
import time
```

Data for burning building

[20]:
```
xnode_colors = {node: 'white' for node in room_map.locations.keys()}
xnode_positions = room_map.locations
xnode_label_pos = { k:[v[0],v[1]-10]  for k,v in room_map.locations.items() }
xedge_weights = {(k, k2) : v2 for k, v in room_map.graph_dict.items() for k2,␣
 ↪v2 in v.items()}

room_graph_data = {  'graph_dict' : room_map.graph_dict,
                     'node_colors': xnode_colors,
                     'node_positions': xnode_positions,
                     'node_label_positions': xnode_label_pos,
                     'edge_weights': xedge_weights
}
```

[21]:
```
room_locations = room_map.locations
print(room_locations)
```

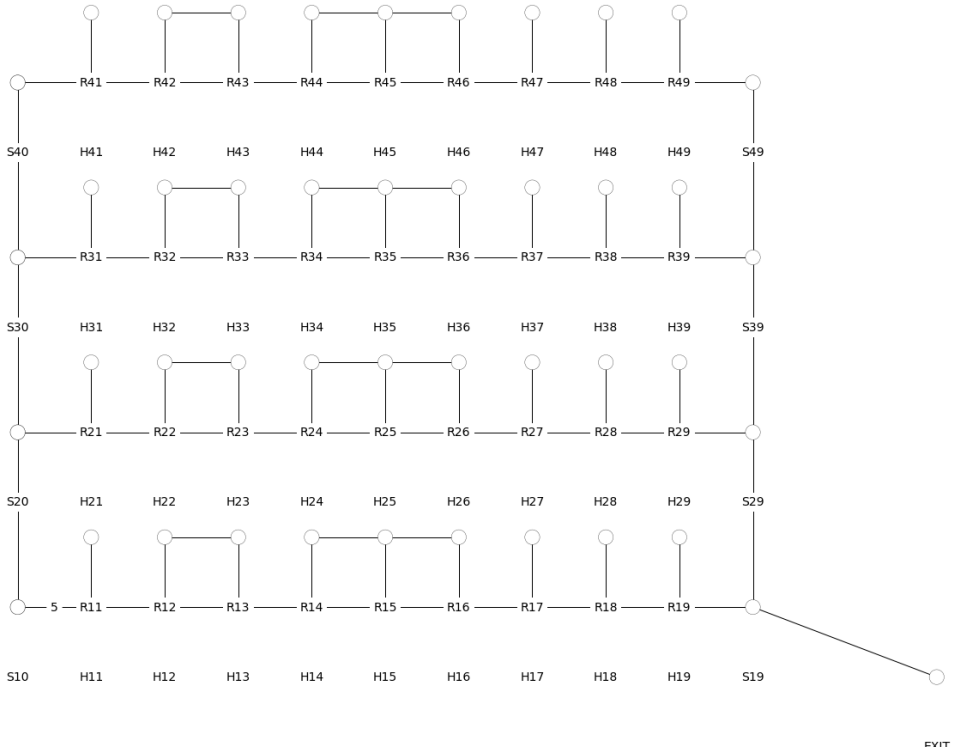
```
{'H10': (40, 50), 'H11': (50, 50), 'H12': (60, 50), 'H13': (70, 50), 'H14': (80,
50), 'H15': (90, 50), 'H16': (100, 50), 'H17': (110, 50), 'H18': (120, 50),
'H19': (130, 50), 'S10': (40, 50), 'R11': (50, 60), 'R12': (60, 60), 'R13': (70,
60), 'R14': (80, 60), 'R15': (90, 60), 'R16': (100, 60), 'R17': (110, 60),
'R18': (120, 60), 'R19': (130, 60), 'S19': (140, 50), 'EXIT': (165, 40), 'H20':
(40, 75), 'H21': (50, 75), 'H22': (60, 75), 'H23': (70, 75), 'H24': (80, 75),
'H25': (90, 75), 'H26': (100, 75), 'H27': (110, 75), 'H28': (120, 75), 'H29':
(130, 75), 'S20': (40, 75), 'R21': (50, 85), 'R22': (60, 85), 'R23': (70, 85),
'R24': (80, 85), 'R25': (90, 85), 'R26': (100, 85), 'R27': (110, 85), 'R28':
(120, 85), 'R29': (130, 85), 'S29': (140, 75), 'H30': (40, 100), 'H31': (50,
100), 'H32': (60, 100), 'H33': (70, 100), 'H34': (80, 100), 'H35': (90, 100),
```

```
'H36': (100, 100), 'H37': (110, 100), 'H38': (120, 100), 'H39': (130, 100),
'S30': (40, 100), 'R31': (50, 110), 'R32': (60, 110), 'R33': (70, 110), 'R34':
(80, 110), 'R35': (90, 110), 'R36': (100, 110), 'R37': (110, 110), 'R38': (120,
110), 'R39': (130, 110), 'S39': (140, 100), 'H40': (40, 125), 'H41': (50, 125),
'H42': (60, 125), 'H43': (70, 125), 'H44': (80, 125), 'H45': (90, 125), 'H46':
(100, 125), 'H47': (110, 125), 'H48': (120, 125), 'H49': (130, 125), 'S40': (40,
125), 'R41': (50, 135), 'R42': (60, 135), 'R43': (70, 135), 'R44': (80, 135),
'R45': (90, 135), 'R46': (100, 135), 'R47': (110, 135), 'R48': (120, 135),
'R49': (130, 135), 'S49': (140, 125)}
```

[22]:
```
show_map(room_graph_data)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
/tmp/ipykernel_893716/4033054162.py in <module>
----> 1 show_map(room_graph_data)

/home/httpd/html/zoo/classes/cs370/aima/notebook4e.py in show_map(graph_data,␣
 ↪node_colors)
    964
    965     # add edge lables to the graph
--> 966     nx.draw_networkx_edge_labels(G, pos=node_positions,␣
 ↪edge_labels=edge_weights, font_size=14)
    967
    968     # add a legend

~/.local/lib/python3.10/site-packages/networkx/drawing/nx_pylab.py in␣
 ↪draw_networkx_edge_labels(G, pos, edge_labels, label_pos, font_size,␣
 ↪font_color, font_family, font_weight, alpha, bbox, horizontalalignment,␣
 ↪verticalalignment, ax, rotate, clip_on, node_size, nodelist, connectionstyle,␣
 ↪hide_ticks)
   1555             )
   1556         else:
-> 1557             text_items[edge] = CurvedArrowText(
   1558                 arrow,
   1559                 label,

~/.local/lib/python3.10/site-packages/networkx/drawing/nx_pylab.py in␣
 ↪__init__(self, arrow, label_pos, labels_horizontal, ax, *args, **kwargs)
   1370                 ax = plt.gca()
   1371             self.ax = ax
-> 1372             self.x, self.y, self.angle = self.
 ↪_update_text_pos_angle(arrow)
   1373
   1374             # Create text object

~/.local/lib/python3.10/site-packages/networkx/drawing/nx_pylab.py in␣
 ↪_update_text_pos_angle(self, arrow)
```

```
   1408             # Fractional label position
   1409             path_disp = self._get_arrow_path_disp(arrow)
-> 1410             (x1, y1), (cx, cy), (x2, y2) = path_disp.vertices
   1411             # Text position at a proportion t along the line in display
 ↪coords
   1412             # default is 0.5 so text appears at the halfway point

ValueError: too many values to unpack (expected 3)
```



[23]:
```
xnode_label_pos
```

[23]:
```
{'H10': [40, 40],
 'H11': [50, 40],
 'H12': [60, 40],
 'H13': [70, 40],
 'H14': [80, 40],
 'H15': [90, 40],
 'H16': [100, 40],
 'H17': [110, 40],
 'H18': [120, 40],
 'H19': [130, 40],
```

```
'S10': [40, 40],
'R11': [50, 50],
'R12': [60, 50],
'R13': [70, 50],
'R14': [80, 50],
'R15': [90, 50],
'R16': [100, 50],
'R17': [110, 50],
'R18': [120, 50],
'R19': [130, 50],
'S19': [140, 40],
'EXIT': [165, 30],
'H20': [40, 65],
'H21': [50, 65],
'H22': [60, 65],
'H23': [70, 65],
'H24': [80, 65],
'H25': [90, 65],
'H26': [100, 65],
'H27': [110, 65],
'H28': [120, 65],
'H29': [130, 65],
'S20': [40, 65],
'R21': [50, 75],
'R22': [60, 75],
'R23': [70, 75],
'R24': [80, 75],
'R25': [90, 75],
'R26': [100, 75],
'R27': [110, 75],
'R28': [120, 75],
'R29': [130, 75],
'S29': [140, 65],
'H30': [40, 90],
'H31': [50, 90],
'H32': [60, 90],
'H33': [70, 90],
'H34': [80, 90],
'H35': [90, 90],
'H36': [100, 90],
'H37': [110, 90],
'H38': [120, 90],
'H39': [130, 90],
'S30': [40, 90],
'R31': [50, 100],
'R32': [60, 100],
'R33': [70, 100],
```

```
 'R34': [80, 100],
 'R35': [90, 100],
 'R36': [100, 100],
 'R37': [110, 100],
 'R38': [120, 100],
 'R39': [130, 100],
 'S39': [140, 90],
 'H40': [40, 115],
 'H41': [50, 115],
 'H42': [60, 115],
 'H43': [70, 115],
 'H44': [80, 115],
 'H45': [90, 115],
 'H46': [100, 115],
 'H47': [110, 115],
 'H48': [120, 115],
 'H49': [130, 115],
 'S40': [40, 115],
 'R41': [50, 125],
 'R42': [60, 125],
 'R43': [70, 125],
 'R44': [80, 125],
 'R45': [90, 125],
 'R46': [100, 125],
 'R47': [110, 125],
 'R48': [120, 125],
 'R49': [130, 125],
 'S49': [140, 115]}
```

[24]:
```
psource(show_map)
```

```
<IPython.core.display.HTML object>
```

[25]:
```
def tree_breadth_search_for_vis(problem):
    """Search through the successors of a problem to find a goal.
    The argument frontier should be an empty queue.
    Don't worry about repeated paths to a state. [Figure 3.7]"""

    # we use these two variables at the time of visualisations
    iterations = 0
    all_node_colors = []
    node_colors = {k : 'white' for k in problem.graph.nodes()}

    #Adding first node to the queue
    frontier = deque([Node(problem.initial)])

    node_colors[Node(problem.initial).state] = "orange"
```

```
    iterations += 1
    all_node_colors.append(dict(node_colors))

    while frontier:
        #Popping first node of queue
        node = frontier.popleft()

        # modify the currently searching node to red
        node_colors[node.state] = "red"
        iterations += 1
        all_node_colors.append(dict(node_colors))

        if problem.goal_test(node.state):
            # modify goal node to green after reaching the goal
            node_colors[node.state] = "green"
            iterations += 1
            all_node_colors.append(dict(node_colors))
            return(iterations, all_node_colors, node)

        frontier.extend(node.expand(problem))

        for n in node.expand(problem):
            node_colors[n.state] = "orange"
            iterations += 1
            all_node_colors.append(dict(node_colors))

        # modify the color of explored nodes to gray
        node_colors[node.state] = "gray"
        iterations += 1
        all_node_colors.append(dict(node_colors))

    return None

def breadth_first_tree_search(problem):
    "Search the shallowest nodes in the search tree first."
    iterations, all_node_colors, node = tree_breadth_search_for_vis(problem)
    return(iterations, all_node_colors, node)
```

[26]:
```
from ipywidgets import *
```

[27]:
```
all_node_colors = []
room_problem = GraphProblem('R18', 'EXIT', room_map)
a, b, c = breadth_first_tree_search(room_problem)
display_visual(room_graph_data, user_input=False,
               algorithm=breadth_first_tree_search,
               problem=room_problem)
```

```
interactive(children=(IntSlider(value=0, description='iteration', max=1),␣
↪Output()), _dom_classes=('widget-int…

interactive(children=(ToggleButton(value=False, description='visualize'),␣
↪Output()), _dom_classes=('widget-int…
```

[28]:
```
def tree_depth_search_for_vis(problem):
    """Search through the successors of a problem to find a goal.
    The argument frontier should be an empty queue.
    Don't worry about repeated paths to a state. [Figure 3.7]"""

    # we use these two variables at the time of visualisations
    iterations = 0
    all_node_colors = []
    node_colors = {k : 'white' for k in problem.graph.nodes()}

    #Adding first node to the stack
    frontier = [Node(problem.initial)]

    node_colors[Node(problem.initial).state] = "orange"
    iterations += 1
    all_node_colors.append(dict(node_colors))

    while frontier:
        #Popping first node of stack
        node = frontier.pop()

        # modify the currently searching node to red
        node_colors[node.state] = "red"
        iterations += 1
        all_node_colors.append(dict(node_colors))

        if problem.goal_test(node.state):
            # modify goal node to green after reaching the goal
            node_colors[node.state] = "green"
            iterations += 1
            all_node_colors.append(dict(node_colors))
            return(iterations, all_node_colors, node)

        frontier.extend(node.expand(problem))

        for n in node.expand(problem):
            node_colors[n.state] = "orange"
            iterations += 1
            all_node_colors.append(dict(node_colors))

        # modify the color of explored nodes to gray
        node_colors[node.state] = "gray"
```

```
        iterations += 1
        all_node_colors.append(dict(node_colors))

    return None

def depth_first_tree_search(problem):
    "Search the deepest nodes in the search tree first."
    iterations, all_node_colors, node = tree_depth_search_for_vis(problem)
    return(iterations, all_node_colors, node)
```

[29]:
```
all_node_colors = []
room_problem = GraphProblem('R35', 'EXIT', room_map)
a, b, c = breadth_first_tree_search(room_problem)
display_visual(room_graph_data, user_input=False,
               algorithm=depth_first_tree_search,
               problem=room_problem)
```

```
interactive(children=(IntSlider(value=0, description='iteration', max=1),␣
↪Output()), _dom_classes=('widget-int…

interactive(children=(ToggleButton(value=False, description='visualize'),␣
↪Output()), _dom_classes=('widget-int…
```

[ ]: