lab4

December 27, 2020

[1]:
```
import pandas as pd
import numpy as np
import random
import seaborn as sns
import math
#import copy
from matplotlib import pyplot as plt
INFINITY = 10000000000
```

1

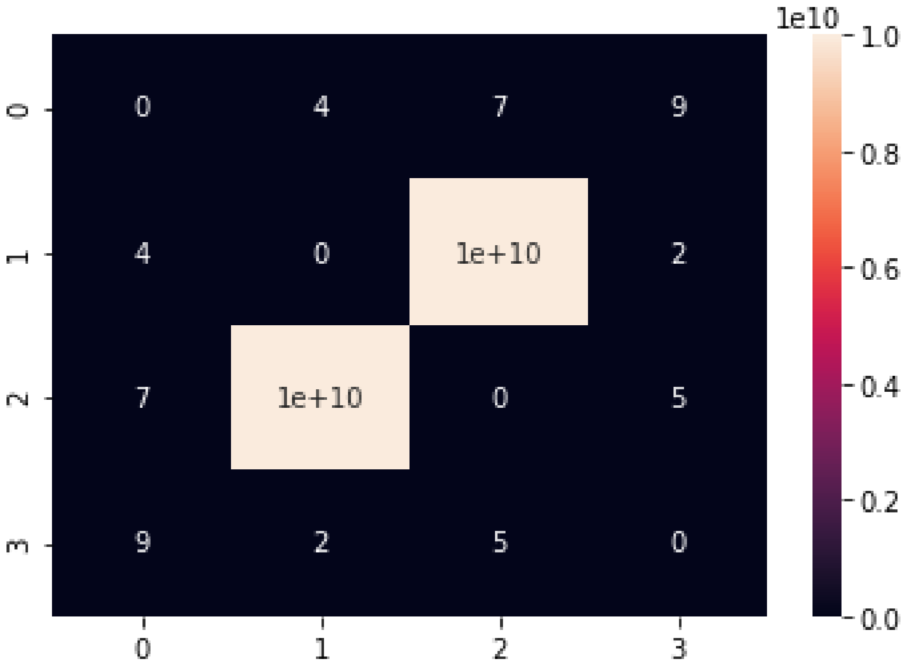
[2]:
```
def initialize_map(p_no_connection, N):

    the_map = np.zeros((N, N))

    for i in range(N):
        for j in range(i):
            if random.random() > p_no_connection:
                value = np.random.randint(1, 10)
            else:
                value = INFINITY
            the_map[i][j] = value
            the_map[j][i] = value

    return the_map
```

[3]:
```
the_map = initialize_map(p_no_connection=0.5, N=4)
the_map
```

[3]:
```
array([[0.e+00, 4.e+00, 7.e+00, 9.e+00],
       [4.e+00, 0.e+00, 1.e+10, 2.e+00],
       [7.e+00, 1.e+10, 0.e+00, 5.e+00],
       [9.e+00, 2.e+00, 5.e+00, 0.e+00]])
```

[4]:
```
sns.heatmap(the_map, annot=True)
```

[4]:
```
<AxesSubplot:>
```



[5]:
```python
def create_new_chromosome(map_len):
    chromosome = []
    for gene in range(map_len - 2):
        chromosome.append(np.random.randint(map_len))

    return chromosome
```

[6]:
```python
some_chromosome = create_new_chromosome(10)
some_chromosome
```

[6]:
```
[1, 3, 2, 6, 9, 8, 1, 4]
```

[7]:
```python
def crossover(a, b):
    if not len(a)== len(b):
        raise IndexError
    new = []
    for i in range(len(a)):
        if random.random() > 0.5:
            new.append(a[i])
        else:
            new.append(b[i])

    return new
```

[8]:

```
crossover([1, 3, 5], [2, 4, 6])
```

[8]:

```
[2, 3, 6]
```

[9]:

```
def create_starting_population(size, map_size):

    #this just creates a population of different routes of a fixed size. Pretty straightforward.

    population = []

    for i in range(0,size):
        population.append(create_new_chromosome(map_size))

    return population
```

[10]:

```
def fitness(chromosome, the_map, start, end):

    #
    score = the_map[start][chromosome[0]]

    for i in range(1, len(chromosome)):
        score += the_map[chromosome[i-1]][chromosome[i]]

    #
    score += the_map[chromosome[len(chromosome)-1]][end]

    return score
```

[11]:

```
some_chromosome = create_new_chromosome(4)
print(some_chromosome)
fitness(some_chromosome, the_map, 0, 3)
```

```
[0, 1]
```

[11]:

```
6.0
```

[12]:

```
def mutate(chromosome, probability, map_len):
    new = []
    for gene in chromosome:
        if random.random() < probability:
            new.append(np.random.randint(map_len))
        else:
            new.append(gene)
    return new
```

[13]:

```
mutate([1, 2, 3 ], 0.3, 4)
```

[13]: [2, 2, 3]

[14]:
```
def choice_parent(parents):
    index = np.random.randint(len(parents))
    parent = parents[index]
    del parents[index]
    return parent
```

[15]:
```
def score_population(population, the_map, start, end):
    score = 0
    for chromosome in population:
        score += fitness(chromosome, the_map, start, end)
    return score/len(population)
```

[16]:
```
def bounds(x, bounds):
    if bounds[0] < x < bounds[1]:
         return x
    elif x < bounds[0]:
        return bounds[0]
    else:
        return bounds[1]
```

[17]:
```
bounds(4., [1.,3.])
```

[17]: 3.0

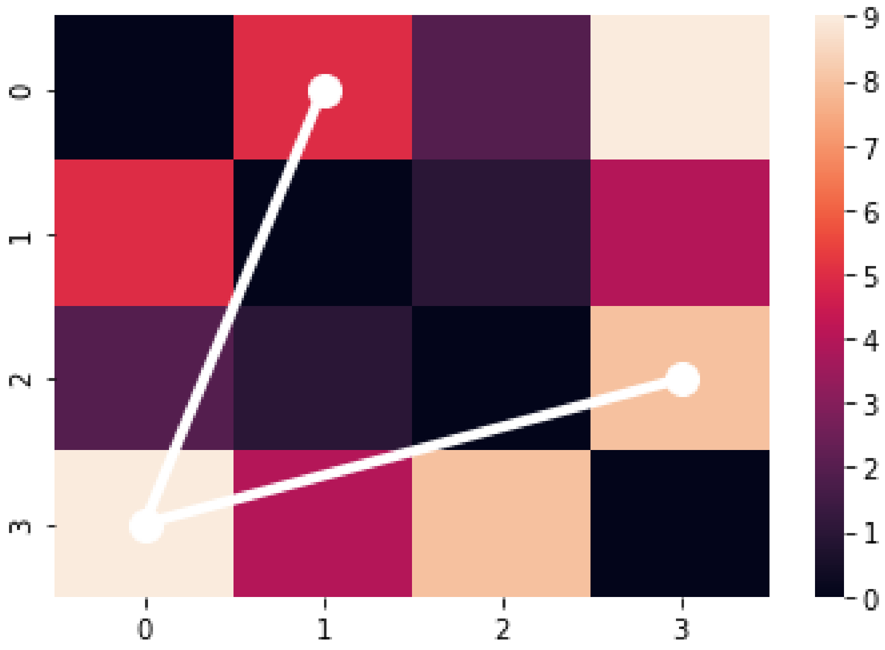
[18]:
```
def selection_loop(the_map, population, start, end, T, mutate_p):
    #
    #population_len = len(population)
    #parents_count = int(T*population_len)
    #population = sorted(population,
    # key=lambda chromosome: fitness(chromosome, the_map,
    #                                 start, end))

    #parents = population[:parents_count].copy()
    parents = population.copy()
    #for gene in population:
    #    if fitness(gene, the_map, start, end) < T:
    #        parents.append(gene)

    #parents_score = score_population(parents, the_map, start, end)
    #        ,
    new_population = []
    #if(fitness(parents[0], the_map, start, end) == 0):
    #    new_population.append(parents[0])
    #    return new_population
    while(len(parents)>1):
        parent_a = choice_parent(parents)
        parent_b = choice_parent(parents)
        #children_num = bounds(int( 1 - float(fitness(parent_a, the_map, start,␣
↪end)+fitness(parent_b, the_map, start, end))/parents_score *␣
↪population_len), [0, int((1 - T)*population_len)])# some difficult function
        #print(children_num)
        for i in range(2):
            new_population.append(mutate(crossover(parent_a,␣
↪parent_b),mutate_p, len(the_map)))
        #check a
        fitness_value = fitness(parent_a, the_map, start, end)
        if fitness_value<T:
            new_population.append(parent_a)
            if fitness_value <=1:
                new_population.clear()
                new_population.append(parent_a)
                return new_population
        #check b
        fitness_value = fitness(parent_b, the_map, start, end)
        if fitness_value<T:
            new_population.append(parent_b)
            if fitness_value <=1:
                new_population.clear()
                new_population.append(parent_b)
                return new_population
    new_population = sorted(new_population,
     key=lambda chromosome: fitness(chromosome, the_map,
                                    start, end))
    min_distance = fitness(new_population[0], the_map, start, end)
    return new_population[:T].copy(), min_distance
```

[19]:

```
def plot_best(the_map, route, start, end):
    ax = sns.heatmap(the_map)
    new_route = [start]
    new_route+=route
    new_route.append(end)

    x= [x + 0.5 for x in new_route[0:len(new_route)-1]]
    y= [x + 0.5 for x in new_route[1:len(new_route)]]

    plt.plot(x, y, marker = 'o', linewidth=4, markersize=12, linestyle = "-",␣
↪color='white')
    plt.show()
```

[20]:

```
the_map = initialize_map(p_no_connection=0, N=4)
plot_best(the_map, [0, 3], 1, 2)
```



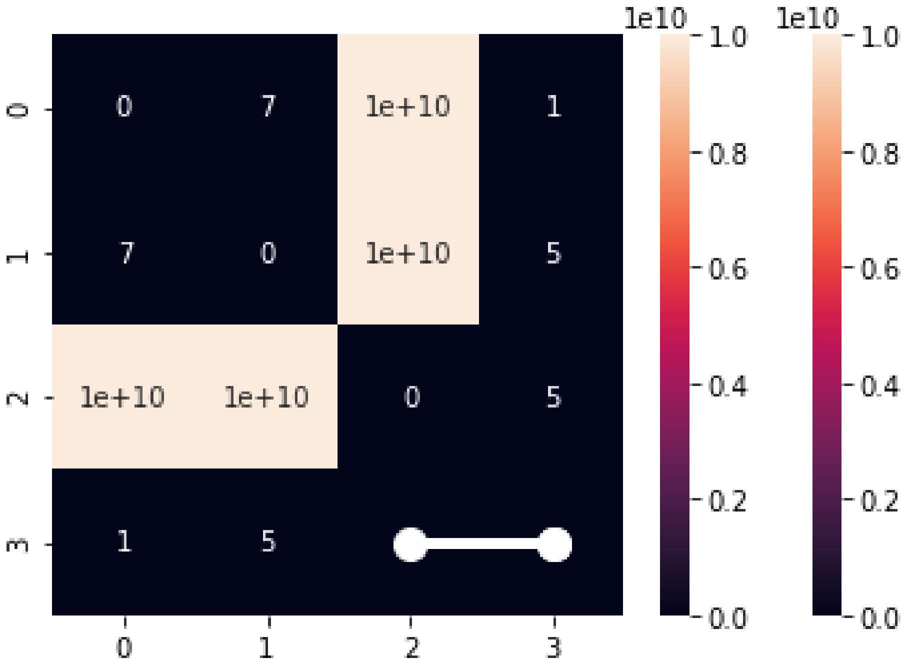
2

[21]:

```
MAP_SIZE = 4
NO_CONNECTION = 0.5
START_POPULATION_SIZE = 10
EARS = 100
EXIT_COUNTER = 10
START = np.random.randint(MAP_SIZE)
END = np.random.randint(MAP_SIZE)
while START == END:
    END = np.random.randint(MAP_SIZE)
print("         = {}".format(START))
print("         = {}".format(END))
MUTATE_P = 0.01
T = START_POPULATION_SIZE

the_map = initialize_map(p_no_connection=NO_CONNECTION, N=MAP_SIZE)
sns.heatmap(the_map, annot=True)
population = create_starting_population(START_POPULATION_SIZE, MAP_SIZE)
best_population = []
prev_distance = math.inf
exit_counter = 0
for i in range(EARS):
    population, distance = selection_loop(the_map, population, START, END, T,␣
 ↪MUTATE_P)
```

```
    print('   {}:     : {}'.format(i, population))
    if(len(population) == 0 or len(population) == 1):
        break
    else:
        best_population = population.copy()

    if(distance == prev_distance):
        exit_counter += 1
    else:
        exit_counter == 0
        prev_distance = distance
    if( exit_counter == EXIT_COUNTER):
        break

best_population = sorted(best_population, key=lambda chromosome:␣
 ↪fitness(chromosome, the_map, START, END))
print("         = {}".format(START))
print("        = {}".format(END))
print('       : {}           {}'.format(best_population[0],␣
 ↪fitness(best_population[0], the_map, START, END)))
plot_best(the_map, best_population[0], START, END)
```

```
         = 2
        = 3
   0:      : [[2, 3], [3, 3], [3, 3], [2, 3], [2, 3], [0, 3], [2, 1], [2,
1], [1, 1], [2, 1]]
   1:      : [[3, 3], [2, 3], [2, 3], [3, 3], [2, 3], [2, 3], [2, 3], [3,
3], [2, 3], [2, 3]]
   2:      : [[2, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2, 3], [3, 3], [2,
3], [2, 3], [2, 3]]
   3:      : [[3, 3], [2, 3], [3, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2,
3], [2, 3], [2, 3]]
   4:      : [[3, 3], [3, 3], [3, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2,
3], [2, 3], [2, 3]]
   5:      : [[2, 3], [3, 3], [3, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2,
3], [2, 3], [3, 3]]
   6:      : [[2, 3], [2, 3], [2, 3], [2, 3], [2, 3], [2, 3], [3, 3], [2,
3], [2, 3], [2, 3]]
   7:      : [[2, 3], [2, 3], [2, 3], [2, 3], [3, 3], [2, 3], [2, 3], [3,
3], [2, 3], [2, 3]]
   8:      : [[2, 3], [2, 3], [2, 3], [3, 3], [2, 3], [2, 3], [2, 3], [2,
3], [3, 3], [3, 3]]
   9:      : [[3, 3], [3, 3], [3, 3], [3, 3], [2, 3], [3, 3], [2, 3], [3,
3], [2, 3], [2, 3]]
   10:      : [[3, 3], [3, 3], [3, 3], [3, 3], [3, 3], [3, 3], [2, 3], [3,
3], [3, 3], [3, 3]]
         = 2
        = 3
      : [3, 3]           5.0
```



```
[22]: def example():
          MAP_SIZE = 10
          NO_CONNECTION = 0.4
          START_POPULATION_SIZE = 10000
          EARS = 10000
          EXIT_COUNTER = 100
          START = np.random.randint(MAP_SIZE)
          END = np.random.randint(MAP_SIZE)
          while START == END:
              END = np.random.randint(MAP_SIZE)
          print("         = {}".format(START))
          print("        = {}".format(END))
          MUTATE_P = 0.01
          T = START_POPULATION_SIZE//100

          the_map = initialize_map(p_no_connection=NO_CONNECTION, N=MAP_SIZE)
          sns.heatmap(the_map, annot=True)
          population = create_starting_population(START_POPULATION_SIZE, MAP_SIZE)
          best_population = []
          prev_distance = math.inf
          exit_counter = 0
```

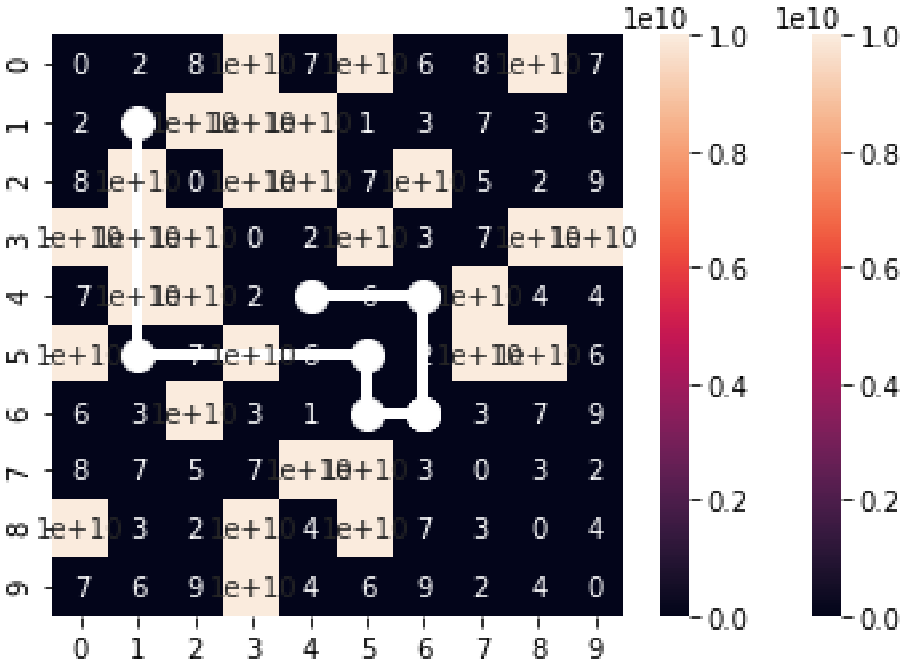
```
    try:
        for i in range(EARS):
            population, distance = selection_loop(the_map, population, START,␣
↪END, T, MUTATE_P)
            if(len(population) == 0 or len(population) == 1):
                break
            else:
                best_population = population.copy()

            if(distance == prev_distance):
                exit_counter += 1
            else:
                exit_counter == 0
                prev_distance = distance
            if( exit_counter == EXIT_COUNTER):
                break

    except (KeyboardInterrupt):
        pass
    best_population = sorted(best_population, key=lambda chromosome:␣
↪fitness(chromosome, the_map, START, END))
    print("         = {}".format(START))
    print("        = {}".format(END))
    print('       : {}          {}'.format(best_population[0],␣
↪fitness(best_population[0], the_map, START, END)))
    plot_best(the_map, best_population[0], START, END)
```

3 1

[23]:

```
%%time
example()
```

```
        = 1
       = 4
        = 1
       = 4
     : [1, 5, 5, 6, 6, 6, 6, 4]           4.0
```

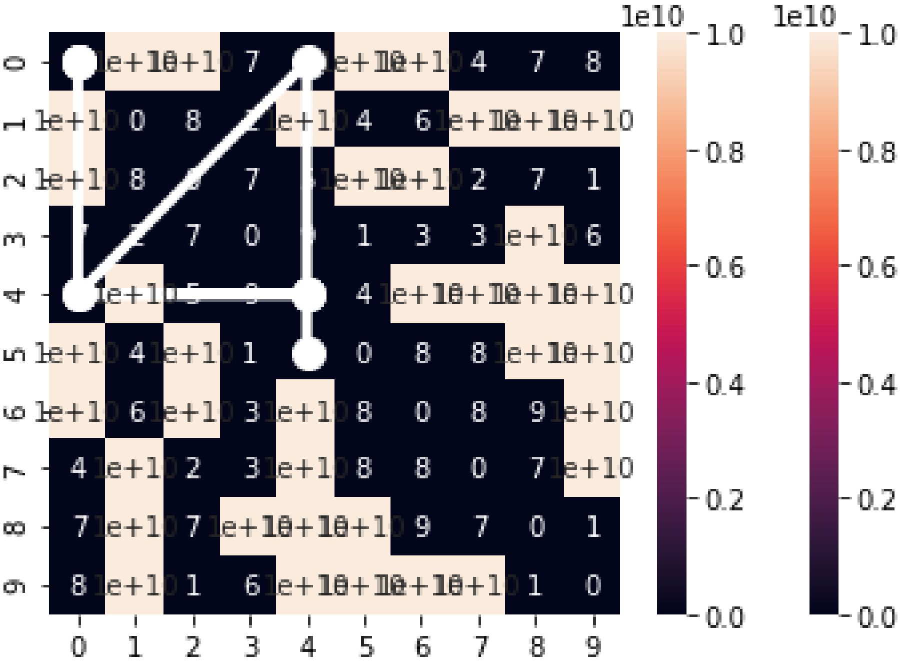


```
CPU times: user 971 ms, sys: 31.9 ms, total: 1 s
Wall time: 1 s
```

4 2

[24]:
```
%%time
example()
```

```
          = 0
         = 5
          = 0
         = 5
    : [0, 0, 0, 4, 4, 0, 4, 4]           7.0
```

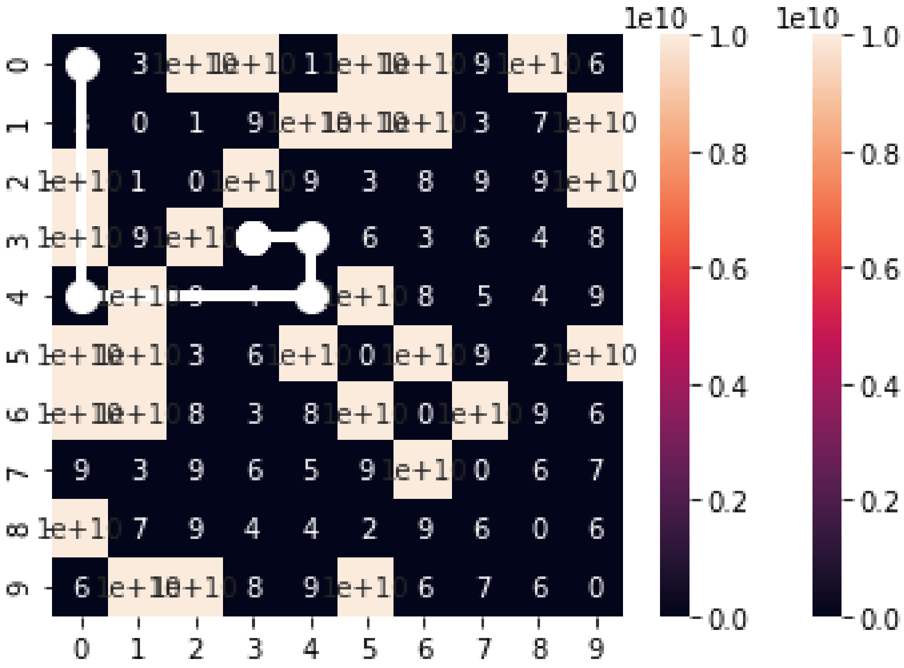


```
CPU times: user 1 s, sys: 35.9 ms, total: 1.04 s
Wall time: 1.04 s
```

5 3

[25]:
```
%%time
example()
```

```
            = 0
           = 3
            = 0
           = 3
        : [0, 4, 4, 3, 3, 3, 3, 3]           5.0
```



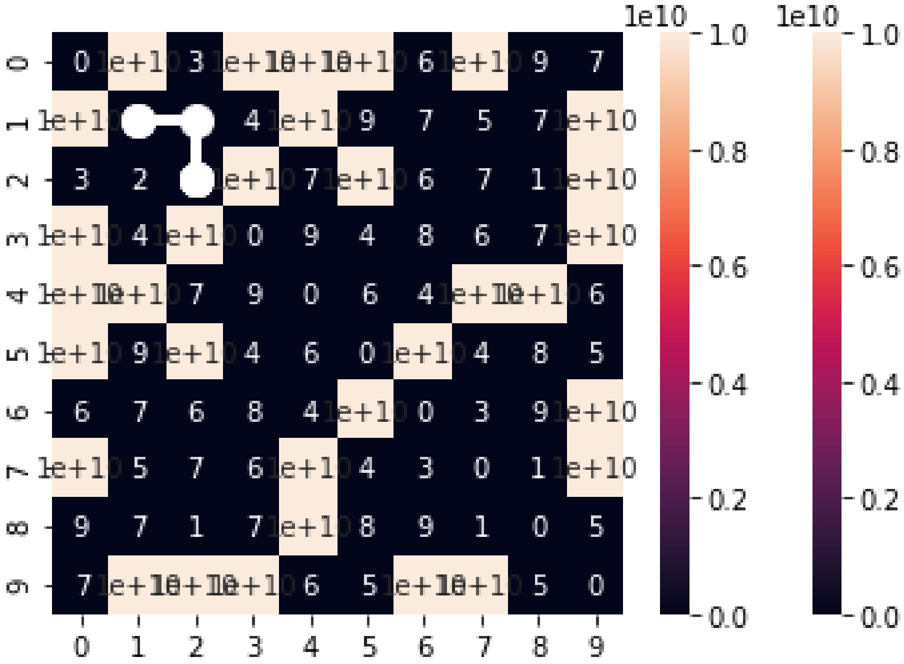
```
CPU times: user 971 ms, sys: 32.1 ms, total: 1 s
Wall time: 1 s
```

6 4

[26]:

```
%%time
example()
```

```
         = 2
        = 1
         = 2
        = 1
     : [2, 2, 2, 2, 2, 2, 1, 1]           2.0
```



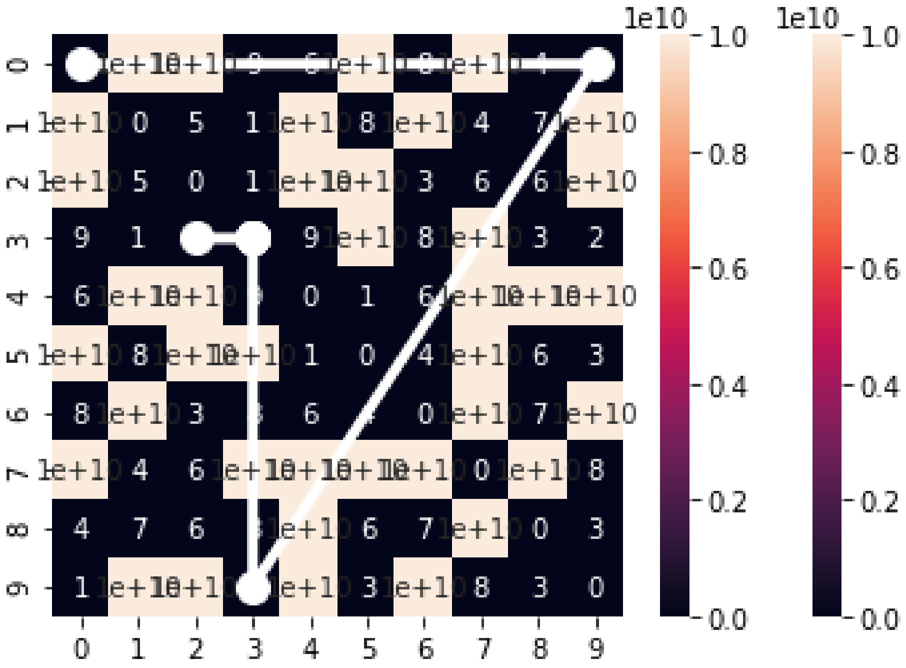
```
CPU times: user 1 s, sys: 36.1 ms, total: 1.04 s
Wall time: 1.04 s
```

7 5

[27]:

```
%%time
example()
```

```
          = 2
         = 0
          = 2
         = 0
    : [3, 3, 3, 3, 3, 3, 9, 0]           4.0
```



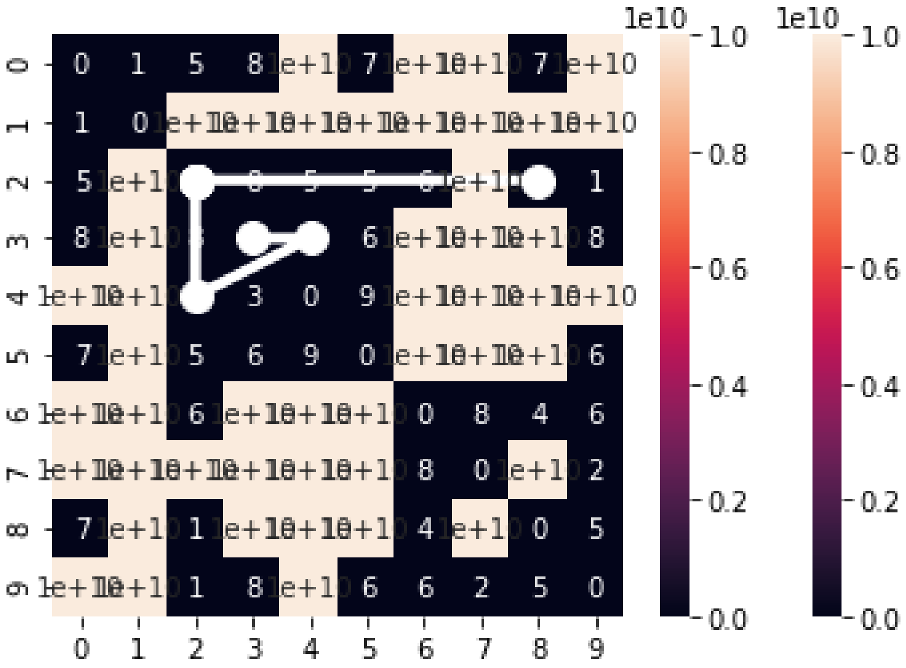
```
CPU times: user 972 ms, sys: 23.9 ms, total: 996 ms
Wall time: 994 ms
```

8 6

[28]:

```
%%time
example()
```

```
          = 8
         = 3
          = 8
         = 3
    : [2, 2, 2, 2, 2, 2, 4, 3]          9.0
```

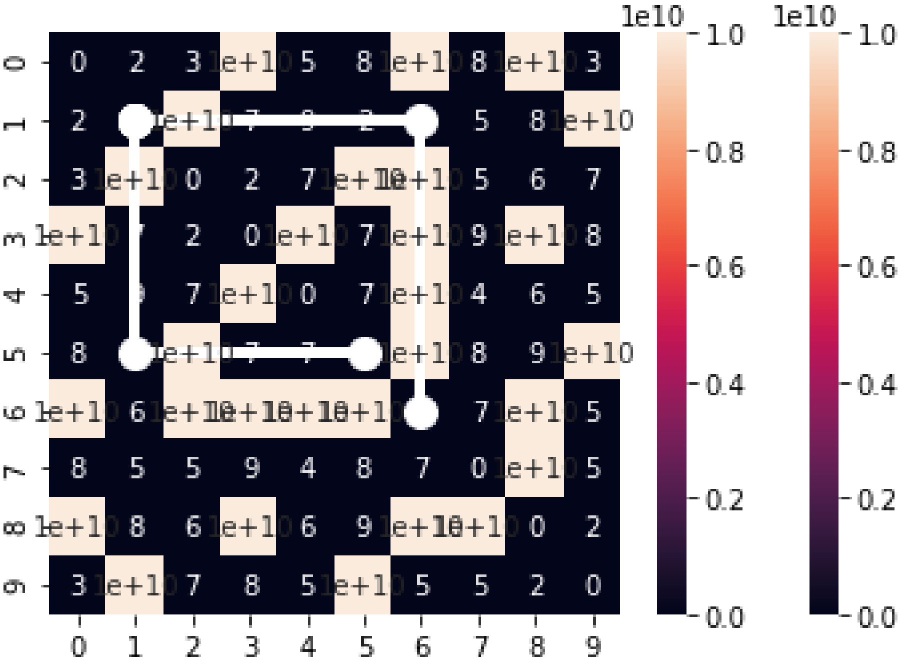


```
CPU times: user 1.02 s, sys: 31.5 ms, total: 1.05 s
Wall time: 1.05 s
```

9 7

[29]:
```
%%time
example()
```

```
          = 6
         = 5
          = 6
         = 5
    : [6, 1, 1, 1, 1, 1, 1, 5]         8.0
```

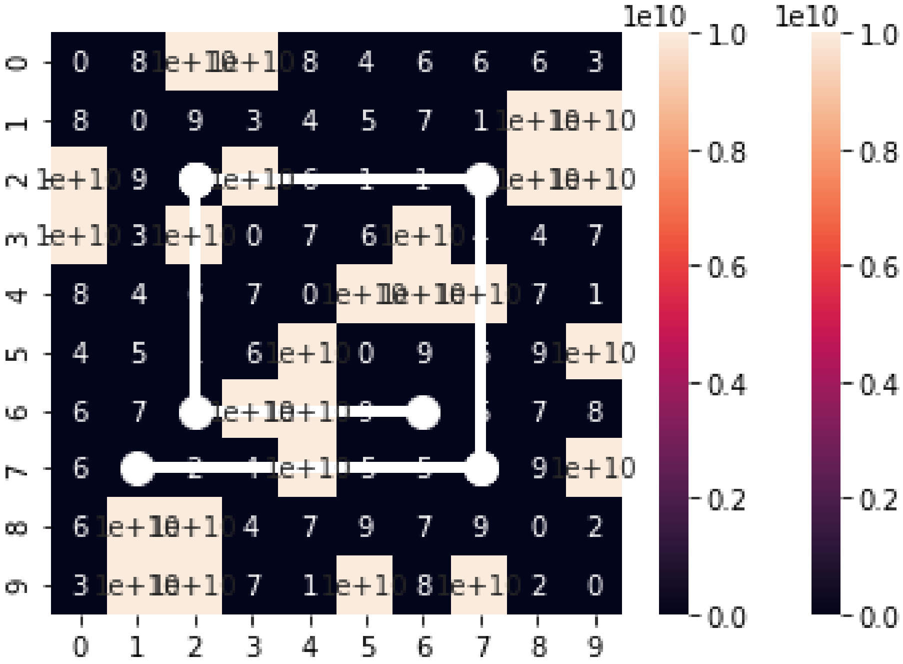


```
CPU times: user 954 ms, sys: 44.1 ms, total: 998 ms
Wall time: 996 ms
```

10 8

[30]:
```
%%time
example()
```

```
              = 1
             = 6
              = 1
             = 6
        : [7, 7, 7, 2, 2, 2, 2, 6]           4.0
```



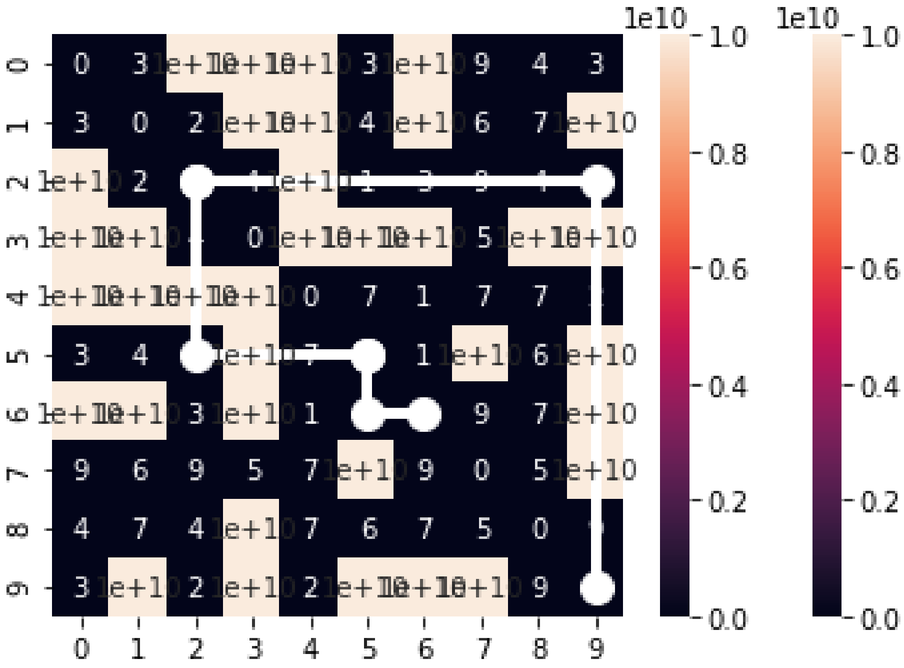
```
CPU times: user 975 ms, sys: 56.2 ms, total: 1.03 s
Wall time: 1.03 s
```

11 9

[33]:

```
%%time
example()
```

```
         = 9
        = 6
         = 9
        = 6
    : [9, 2, 2, 2, 5, 5, 5, 6]           4.0
```

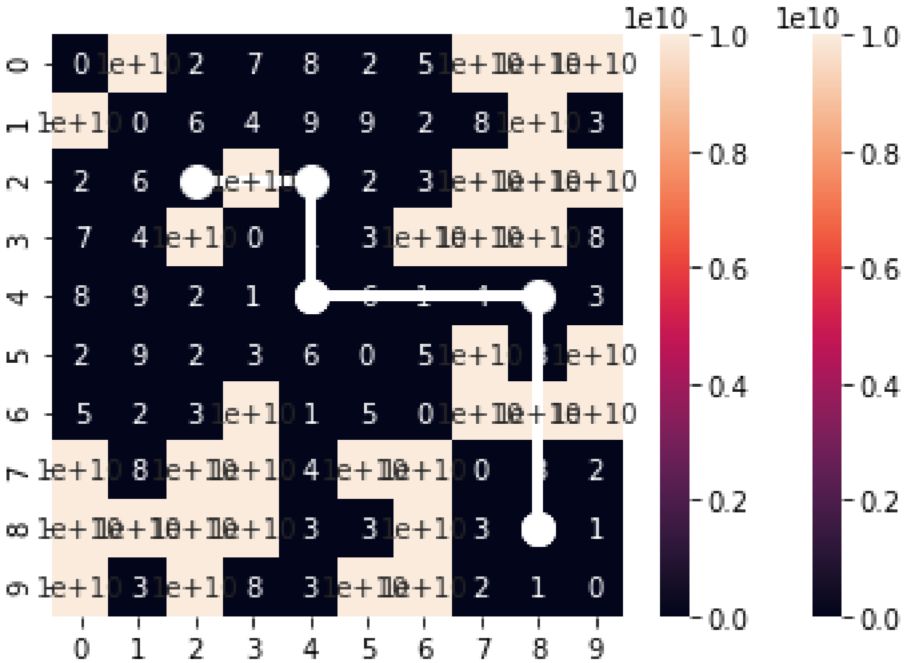


```
CPU times: user 986 ms, sys: 43.8 ms, total: 1.03 s
Wall time: 1.03 s
```

12   10

[32]:
```
%%time
example()
```

```
        = 8
       = 2
        = 8
       = 2
  : [8, 8, 8, 4, 4, 4, 4, 2]          5.0
```



```
CPU times: user 952 ms, sys: 40 ms, total: 992 ms
Wall time: 990 ms
```

```
[ ]:
```