Homework-4, (2 marks), EE-677-2016

SBP, EE, IITB

26th Aug 2016

1 Arguably my best artwork so far ...

There are 13 points to be "wired together" with minimum cost Manhatten wiring (using only horizontal and vertical segments).

In the Python script given below, the list **gCostLowerTriangular** is intended to store the costs of the best wiring between point **i** and point **j**, for all $j < i$.

However presently I have filled it with random integers.

Note that the length of a Manhatten path should include **both square cells** at both the **endpoints**. For example, the shortest Manhatten path between point 0, at location (0,0) and point 1, at location (0,6) is equal to 13. One such path is

[(0,0), (0,1), (0,2), (0,3), (1,3), (2,3), (3,3),(3,4),(3,5),(2,5), (2,6),(1,6),(0,6)]

Make a note of this ... wrong answers due to failure to follow this convention would lead to 0 marks. The TA's will not be permitting any variations in the answers.

EXERCISE 1 : Your task is to fill this list **gCostLowerTriangular** with (absolutely) correct entries. There are 78 entries to be filled up. You may form two groups and roughly divide this work among yourself so that each of you need not compute more than 2 such entries. Of course I would not mind if you work out all 78 entries by yourself. After all, you need to appreciate the amount of donkey-effort that went into preparing this homework. It would have been easier to write a graphics program to create random obstacles and position 13 points randomly.

EXERCISE 2 : Using Prim's algorithm find a minimum a good wiring that connects all the given 13 locations. Please note that due to possible sharing of square-cells, the MST approach might not give the optimum answer. However, we will accept any MST as a good wiring solution.

EXERCISE 3 : Depict your solution on the same grid.

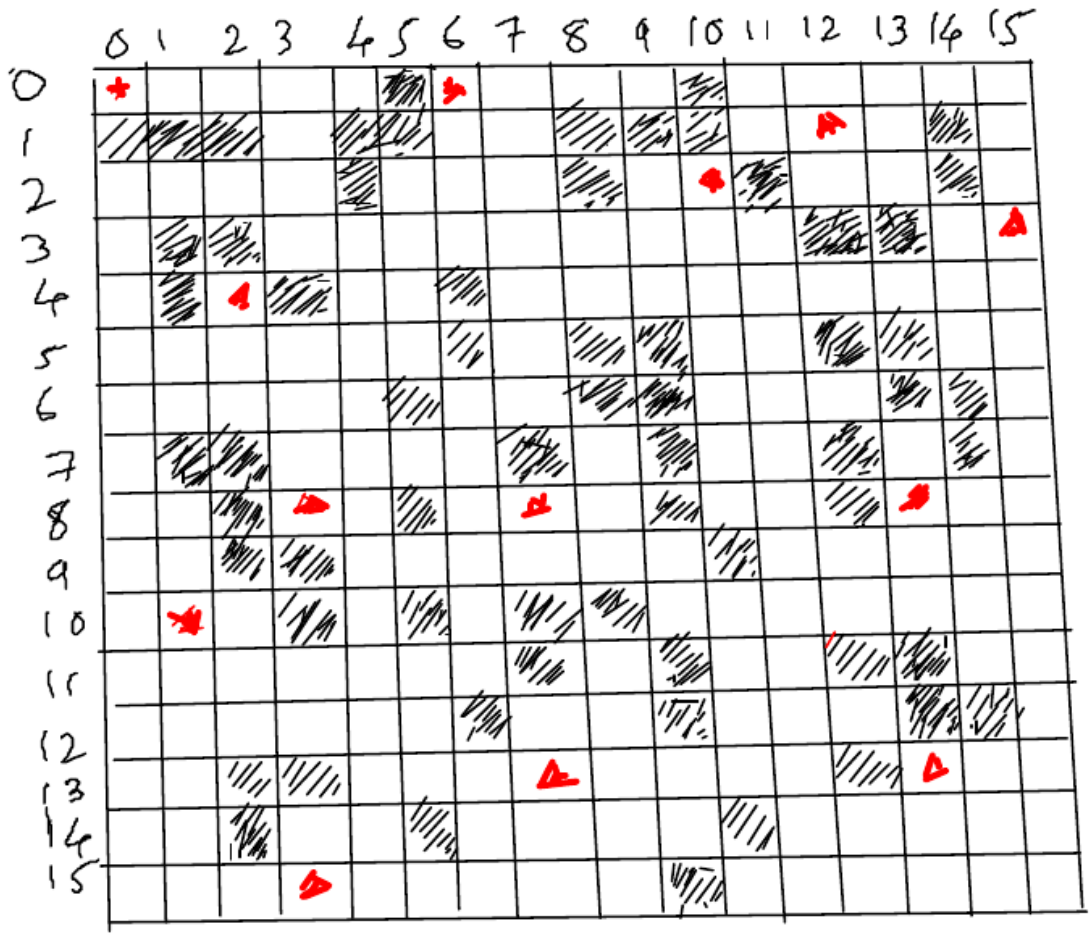

Figure 1: FindMinimumManhattenLengthWiringProblemInstance

```
# hw4_ee677_2016.py
# -- Sachin B. Patkar ( for ee-677-2016 ... Friday 26th Aug 2016 )

import random

numV = 13
V = range( numV )
locationsV = [ \
```

```
    (0,0), (0,6), (1,12), (2,10), (3,15), (4,2), \
    (8,3), (8,7), (8,13), (10,1), (13,7), (13,13), (14,3) ]

gNeighbors =  \
    [ [ j for j in range(numV) if j != i ] for i in range( numV ) ]

gCostLowerTriangular = [ \
      [ int( random.random() * 64 )  for j in range( i )  ] \
          for i in range ( numV )
   ]

gCost = [ \
        [ gCostLowerTriangular[i][j] if i > j else \
            0 if i == j else gCostLowerTriangular[j][i] \
              for j in range( numV ) ] \
                for i in range( numV)
    ]

gCostSymmetricalFlag = True
for i in range(numV) :
  for j in range(i) :
    if ( gCost[i][j] != gCost[j][i] ) :
      gCostSymmetricalFlag = False
      raise Exception ( "gCost not symmetrical" )
for i in range( numV ) :
  if ( gCost[i][i] != 0 ) :
    raise Exception ( "gCost nonzero on diagonal" )

print "gCost is symmetrical and has zeros on diagonal "
print gCost
```