lec1_step8

October 6, 2022

[ ]:
```
## Python basics for novice data scientists, supported by Wagatsuma Lab@Kyutech
#
# The MIT License (MIT): Copyright (c) 2020 Hiroaki Wagatsuma and Wagatsuma␣
 ↪Lab@Kyutech
#
# Permission is hereby granted, free of charge, to any person obtaining a copy␣
 ↪of this software and associated documentation files (the "Software"), to␣
 ↪deal in the Software without restriction, including without limitation the␣
 ↪rights to use, copy, modify, merge, publish, distribute, sublicense, and/or␣
 ↪sell copies of the Software, and to permit persons to whom the Software is␣
 ↪furnished to do so, subject to the following conditions:
# The above copyright notice and this permission notice shall be included in␣
 ↪all copies or substantial portions of the Software.
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR␣
 ↪IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,␣
 ↪FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE␣
 ↪AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER␣
 ↪LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING␣
 ↪FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS␣
 ↪IN THE SOFTWARE. */
#
# # @Time    : 2020-10-14
# # @Author  : Hiroaki Wagatsuma
# # @Site    : https://github.com/hirowgit/2A_python_basic_course
# # @IDE     : Python 3.7.7 (default, Mar 10 2020, 15:43:27) [Clang 10.0.0␣
 ↪(clang-1000.11.45.5)] on darwin
# # @File    : lec1_step8.py
```

[ ]:
```
# Practice 3-1 (page 13/29)
# https://www.slideshare.net/tadahirotaniguchi0624/3-46861684
```

[1]:
```
TargetGraph={
    'S':['A','B'],
    'A':['S','B','C'],
    'B':['S','A','E' ,'F'],
    'C':['A','E','D'],
    'D':['C','E' ,'G'],
    'E':['B','C' ,'D' ,'G'],
    'F':['B'],
    'G':['D','E']
}
```

[39]:

```
state=[]
OpenList=['S']
ClosedList=[]
while OpenList:
    print(state)
    #print(OpenList)
    state=OpenList[0]
    del OpenList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    OpenList.insert(0, activeNodes)  # the first item
    #s1 = ','.join(OpenList);
    print('OpenList(1): ',OpenList)
    #pprint.pprint(OpenList)
    OpenList=[item for i in OpenList for item in i if item not in ClosedList]
    print('OpenList(2): ',OpenList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
[]
OpenList(1):  [['A', 'B']]
OpenList(2):  ['A', 'B']
ClosedList:  ['S']
S
OpenList(1):  [['B', 'C'], 'B']
OpenList(2):  ['B', 'C', 'B']
ClosedList:  ['S', 'A']
A
OpenList(1):  [['E', 'F'], 'C', 'B']
OpenList(2):  ['E', 'F', 'C']
ClosedList:  ['S', 'A', 'B']
B
OpenList(1):  [['C', 'D', 'G'], 'F', 'C']
OpenList(2):  ['C', 'D', 'G', 'F', 'C']
ClosedList:  ['S', 'A', 'B', 'E']
E
OpenList(1):  [['D'], 'D', 'G', 'F', 'C']
OpenList(2):  ['D', 'D', 'G', 'F']
ClosedList:  ['S', 'A', 'B', 'E', 'C']
C
```

```
OpenList(1):  [['G'], 'D', 'G', 'F']
OpenList(2):  ['G', 'G', 'F']
ClosedList:  ['S', 'A', 'B', 'E', 'C', 'D']
D
completed
```

[75]:

```
H = {'S': 0,'A': 5, 'B': 8, 'C': 1, 'D': 2, 'E': 6}
sorted(H)
```

[75]:

```
['A', 'B', 'C', 'D', 'E', 'S']
```

[76]:

```
sorted(H.keys())
```

[76]:

```
['A', 'B', 'C', 'D', 'E', 'S']
```

[42]:

```
sorted(H.values())
```

[42]:

```
[0, 1, 2, 5, 6, 8]
```

[44]:

```
sorted(H.items(), key = lambda x:x[0])
```

[44]:

```
[('A', 5), ('B', 8), ('C', 1), ('D', 2), ('E', 6), ('S', 0)]
```

[74]:

```
sorted(H.items(), key = lambda x:x[1])
```

[74]:

```
[('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]
```

[77]:

```
H2=sorted(H.items(), key = lambda x:x[1])
print(H2)
```

```
[('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]
```

[78]:

```
sorted(H2, key = lambda x:x[1])
```

[78]:

```
[('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]
```

[79]:

```
sorted(H2, key = lambda x:x[0])
```

[79]:

```
[('A', 5), ('B', 8), ('C', 1), ('D', 2), ('E', 6), ('S', 0)]
```

[80]:

```
[i[0] for i in H2 ]
```

[80]:

```
['S', 'C', 'D', 'A', 'E', 'B']
```

[69]:

```
[i[1] for i in H2 ]
```

[69]:

```
[0, 1, 2, 5, 6, 8]
```

[81]:

```
hh1=[i[0] for i in H2 ]
hh2=[i[1] for i in H2 ]
```

[87]:

```
[(hh1[i],hh2[i]) for i in range(len(hh1)) ]
```

[87]:

```
[('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]
```

[86]:

```
[(hh1[i],hh2[i]) for i in range(len(hh1)) ]
```

[86]:

```
[('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]
```

[5]:

```
['S','A','B','C','D','E','F','G']
```

[5]:

```
['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']
```

[3]:

```
C=[[0, 2, 6, 0, 0, 0, 0, 0],
      [2, 0, 2, 1, 0, 0, 0, 0] ,
      [6, 2, 0, 0, 0, 5, 4, 0] ,
      [0, 1, 0, 0, 5, 2, 0, 0] ,
      [0, 0, 0, 5, 0, 1, 0, 1] ,
      [0, 0, 5, 2, 1, 0, 0, 5] ,
      [0, 0, 4, 0, 0, 0, 0, 0] ,
      [0, 0, 0, 0, 1, 5, 0, 0]
]
```

[91]:

```
print(C)
```

```
[[0, 2, 6, 0, 0, 0, 0, 0], [2, 0, 2, 1, 0, 0, 0, 0], [6, 2, 0, 0, 0, 5, 4, 0],
[0, 1, 0, 0, 5, 2, 0, 0], [0, 0, 0, 5, 0, 1, 0, 1], [0, 0, 5, 2, 1, 0, 0, 5],
[0, 0, 4, 0, 0, 0, 0, 0], [0, 0, 0, 0, 1, 5, 0, 0]]
```

[92]:

```
pprint.pprint(C)
```

```
[[0, 2, 6, 0, 0, 0, 0, 0],
 [2, 0, 2, 1, 0, 0, 0, 0],
 [6, 2, 0, 0, 0, 5, 4, 0],
 [0, 1, 0, 0, 5, 2, 0, 0],
 [0, 0, 0, 5, 0, 1, 0, 1],
 [0, 0, 5, 2, 1, 0, 0, 5],
 [0, 0, 4, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 1, 5, 0, 0]]
```

[53]:

```
N=7
Node=[chr(i) for i in range(65,65+N)]
Node=['S']+Node
print(Node)
```

```
['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']
```

[11]:

```python
OpenList=['B','D']
# Node.index('B')
indexList=[Node.index(L)  for L in OpenList]
indexList
```

[11]:

```
[2, 4]
```

[37]:

```python
Node.index(state)
```

[37]:

```
1
```

[39]:

```python
key=Node.index(state)
Cost=C[key]
Cost
```

[39]:

```
[2, 0, 2, 1, 0, 0, 0, 0]
```

[70]:

```python
OpenList=['S','A','E','F']
state='B'
key=Node.index(state)
Cost=C[key]
print(Cost)
print(' ')
indexList=[Node.index(L)  for L in OpenList]
print(indexList)
CList=[C[Node.index(state)][i] for i in indexList]
# a[[0,1]]
print(C[Node.index(state)])
print(CList)
print(sorted(CList))
```

```
[6, 2, 0, 0, 0, 5, 4, 0]

[0, 1, 5, 6]
[6, 2, 0, 0, 0, 5, 4, 0]
[6, 2, 5, 4]
[2, 4, 5, 6]
```

[71]:

```python
LL=[7,6,5,4,3,2,1]
LL.sort(key=lambda x: x)
print(LL)

LL=[7,6,5,4,3,2,1]
LL.sort()
print(LL)
```

```
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7]
```

[72]:
```
LL=[7,6,5,4,3,2,1]
aa=['a','b','c','d','e','f','g']
aa.sort(key=LL)
aa
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In [72], line 3
      1 LL=[7,6,5,4,3,2,1]
      2 aa=['a','b','c','d','e','f','g']
----> 3 aa.sort(key=LL)
      4 aa

TypeError: 'list' object is not callable
```

[73]:
```
keys = ['node','cost']
d_all=[]
for i in range(len(Node)):
    values=[Node[i],Cost[i]]
    d = {k: v for k, v in zip(keys, values)}
    d_all.append(d)
print(d_all)
d_all.sort(key=lambda x: x['cost'])
d_all
```

```
[{'node': 'S', 'cost': 6}, {'node': 'A', 'cost': 2}, {'node': 'B', 'cost': 0},
{'node': 'C', 'cost': 0}, {'node': 'D', 'cost': 0}, {'node': 'E', 'cost': 5},
{'node': 'F', 'cost': 4}, {'node': 'G', 'cost': 0}]
```

[73]:
```
[{'node': 'B', 'cost': 0},
 {'node': 'C', 'cost': 0},
 {'node': 'D', 'cost': 0},
 {'node': 'G', 'cost': 0},
 {'node': 'A', 'cost': 2},
 {'node': 'F', 'cost': 4},
 {'node': 'E', 'cost': 5},
 {'node': 'S', 'cost': 6}]
```

[75]:
```
keys = ['node','cost']
d_all=[]
for i in range(len(Node)):
    values=[Node[i],Cost[i]]
    d = {k: v for k, v in zip(keys, values)}
    d_all.append(d)
print(d_all)
d_all.sort(key=lambda x: x['cost'], reverse=True)
```

```
d_all
```

```
[{'node': 'S', 'cost': 6}, {'node': 'A', 'cost': 2}, {'node': 'B', 'cost': 0},
{'node': 'C', 'cost': 0}, {'node': 'D', 'cost': 0}, {'node': 'E', 'cost': 5},
{'node': 'F', 'cost': 4}, {'node': 'G', 'cost': 0}]
```

[75]:

```
[{'node': 'S', 'cost': 6},
 {'node': 'E', 'cost': 5},
 {'node': 'F', 'cost': 4},
 {'node': 'A', 'cost': 2},
 {'node': 'B', 'cost': 0},
 {'node': 'C', 'cost': 0},
 {'node': 'D', 'cost': 0},
 {'node': 'G', 'cost': 0}]
```

[77]:

```
print(OpenList)
print(CList)
print(' ')
keys = ['node','cost']
# keys2 = ['node','cost','h']
d_all=[]
for i in range(len(OpenList)):
    values=[OpenList[i],CList[i]]
    d = {k: v for k, v in zip(keys, values)}
    d_all.append(d)
print(d_all)
d_all.sort(key=lambda x: x['cost'])
print(d_all)
print([d['node'] for d in d_all])
```

```
['S', 'A', 'E', 'F']
[6, 2, 5, 4]

[{'node': 'S', 'cost': 6}, {'node': 'A', 'cost': 2}, {'node': 'E', 'cost': 5},
{'node': 'F', 'cost': 4}]
[{'node': 'A', 'cost': 2}, {'node': 'F', 'cost': 4}, {'node': 'E', 'cost': 5},
{'node': 'S', 'cost': 6}]
['A', 'F', 'E', 'S']
```

[122]:

```
mergedList[0]
 [mergedList[j][0] for j in range(len(mergedList))]
print(mergedList)
print(len(mergedList[0]))
```

```
[['S', 'A', 'E', 'F'], [0, 1, 5, 6], [6, 2, 5, 4]]
4
```

[125]:

```
OpenList=['S','A','E','F']
state='B'
key=Node.index(state)
Cost=C[key]
indexList=[Node.index(L)  for L in OpenList]
# print(Node[[0,1]])
CList=[C[Node.index(state)][i] for i in indexList]
mergedList=[OpenList,indexList,CList]
print(mergedList)

mergedList2=[]
for i in range(len(mergedList[0])):
    mergedList2.append([mergedList[j][i] for j in range(len(mergedList))])

    print([i])
    print(mergedList2)


print(mergedList2)
print(' ')

mergedList2.sort(key=lambda x: x[2])
print(mergedList2)
[mergedList2[i][0] for i in range(len(mergedList2))]
```

```
[['S', 'A', 'E', 'F'], [0, 1, 5, 6], [6, 2, 5, 4]]
[0]
[['S', 0, 6]]
[1]
[['S', 0, 6], ['A', 1, 2]]
[2]
[['S', 0, 6], ['A', 1, 2], ['E', 5, 5]]
[3]
[['S', 0, 6], ['A', 1, 2], ['E', 5, 5], ['F', 6, 4]]
[['S', 0, 6], ['A', 1, 2], ['E', 5, 5], ['F', 6, 4]]

[['A', 1, 2], ['F', 6, 4], ['E', 5, 5], ['S', 0, 6]]
```

[125]:

```
['A', 'F', 'E', 'S']
```

[102]:

```
import numpy as np

OpenList=['S','A','E','F']
state='B'
key=Node.index(state)
Cost=C[key]
indexList=[Node.index(L)  for L in OpenList]
```

```
# print(Node[[0,1]])
CList=[C[Node.index(state)][i] for i in indexList]
mergedList=[OpenList,indexList,CList]
print(mergedList)

mergedList_np=np.array(mergedList)
print(mergedList_np)
# np.transpose(mergedList_np)
print(mergedList_np.T)
print(' ')
mergedList2=(mergedList_np.T).tolist()
print(mergedList2[0])
print(' ')
mergedList2.sort(key=lambda x: x[2])
print(mergedList2)
[mergedList2[i][0] for i in range(len(mergedList2))]
```

```
[['S', 'A', 'E', 'F'], [0, 1, 5, 6], [6, 2, 5, 4]]
[['S' 'A' 'E' 'F']
 ['0' '1' '5' '6']
 ['6' '2' '5' '4']]
[['S' '0' '6']
 ['A' '1' '2']
 ['E' '5' '5']
 ['F' '6' '4']]

['S', '0', '6']

[['A', '1', '2'], ['F', '6', '4'], ['E', '5', '5'], ['S', '0', '6']]
```

[102]:
```
['A', 'F', 'E', 'S']
```

[ ]:
```

```

[ ]:
```
# ==============================
```

[113]:
```
[('S','A')]
```

[113]:
```
[('S', 'A')]
```

[120]:
```
g=('S', 'A')
print(g[0])
print(g[1])
```

```
S
A
```

[118]:

```
C[1][2]
```

[118]: 2

[121]:

```
g=('S', 'A')
i=[s for s in range(len(Node)) if g[0] in Node[s]][0]
j=[s for s in range(len(Node)) if g[1] in Node[s]][0]
C[i][j]
```

[121]: 2

[7]:

```
def eachCost(Pair,Node,C):
    i=[s for s in range(len(Node)) if Pair[0] in Node[s]][0]
    j=[s for s in range(len(Node)) if Pair[1] in Node[s]][0]
    return C[i][j]

C=[[0, 2, 6, 0, 0, 0, 0, 0],
      [2, 0, 2, 1, 0, 0, 0, 0] ,
      [6, 2, 0, 0, 0, 5, 4, 0] ,
      [0, 1, 0, 0, 5, 2, 0, 0] ,
      [0, 0, 0, 5, 0, 1, 0, 1] ,
      [0, 0, 5, 2, 1, 0, 0, 5] ,
      [0, 0, 4, 0, 0, 0, 0, 0] ,
      [0, 0, 0, 0, 1, 5, 0, 0]
]
N=7
Node=[chr(i) for i in range(65,65+N)]
Node=['S']+Node
print(Node)
g=('S', 'A')
eachCost(g,Node,C)
```

```
['S', 'A', 'B', 'C', 'D', 'E', 'F', 'G']
```

[7]: 2

[10]:

```
# New with the cost calculation
CostList=[]
state=[]
OpenList=['S']
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    print(state)
    del OpenList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    print(costM)
    print(costM[0])
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    print(costMat)
    OpenList.insert(0, activeNodes)  # the first item
    CostList.insert(0, costMat)  # the first item
    print('OpenList(1): ',OpenList)
    #OpenList=[item for i in OpenList for item in i if i not in ClosedList]
    OpenList=[item for i in OpenList for item in i]
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    print('key: ',key)
    print('OpenList(2): ',OpenList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
[('A', 'S'), ('B', 'S')]
('A', 'S')
[2, 6]
OpenList(1):  [['A', 'B']]
key:  [0, 1]
OpenList(2):  ['A', 'B']
ClosedList:  ['S']
A
[('B', 'A'), ('C', 'A')]
('B', 'A')
[2, 1]
OpenList(1):  [['B', 'C'], 'B']
key:  [0, 1, 2]
OpenList(2):  ['B', 'C', 'B']
ClosedList:  ['S', 'A']
B
[('E', 'B'), ('F', 'B')]
('E', 'B')
[5, 4]
OpenList(1):  [['E', 'F'], 'C', 'B']
key:  [0, 1, 2]
OpenList(2):  ['E', 'F', 'C', 'B']
ClosedList:  ['S', 'A', 'B']
E
[('C', 'E'), ('D', 'E'), ('G', 'E')]
('C', 'E')
[2, 1, 5]
```

```
OpenList(1):  [['C', 'D', 'G'], 'F', 'C', 'B']
key:  [0, 1, 2, 3, 4]
OpenList(2):  ['C', 'D', 'G', 'F', 'C', 'B']
ClosedList:  ['S', 'A', 'B', 'E']
C
[('D', 'C')]
('D', 'C')
[5]
OpenList(1):  [['D'], 'D', 'G', 'F', 'C', 'B']
key:  [0, 1, 2, 3]
OpenList(2):  ['D', 'D', 'G', 'F', 'C', 'B']
ClosedList:  ['S', 'A', 'B', 'E', 'C']
D
[('G', 'D')]
('G', 'D')
[1]
OpenList(1):  [['G'], 'D', 'G', 'F', 'C', 'B']
key:  [0, 2, 3]
OpenList(2):  ['G', 'D', 'G', 'F', 'C', 'B']
ClosedList:  ['S', 'A', 'B', 'E', 'C', 'D']
G
completed
```

[ ]:

```
# 2022/10/05
# New with the cost calculation
CostList=[]
state=[]
OpenList=['S']
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    print(state)
    del OpenList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    print(costM)
    print(costM[0])
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    print(costMat)
    OpenList.insert(0, activeNodes)  # the first item
    CostList.insert(0, costMat)  # the first item
    print('OpenList(1): ',OpenList)
    #OpenList=[item for i in OpenList for item in i if i not in ClosedList]
    OpenList=[item for i in OpenList for item in i]
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    print('key: ',key)
    print('OpenList(2): ',OpenList)
    print('ClosedList: ',ClosedList)
print('completed')
```

[2]:

```
OpenList=['S']
ClosedList=[]
while OpenList:
    state=OpenList[0]
    del OpenList[0]
    ClosedList=ClosedList+[state]
    ClosedList=list(set(ClosedList))
    print(['state',state])
    print(['OpenList(1)',OpenList])
    print(['ClosedList',ClosedList])
    if state=='G':
        break
    tmpSt=set(TargetGraph[state]) -set(ClosedList)
    activeNodes=list(tmpSt -set(OpenList))
    OpenList=OpenList+activeNodes
  #  OpenList=list(set(OpenList))
    print(['OpenList(2)',OpenList])
    print('')
print('Completed')
```

```
['state', 'S']
['OpenList(1)', []]
['ClosedList', ['S']]
['OpenList(2)', ['A', 'B']]

['state', 'A']
['OpenList(1)', ['B']]
['ClosedList', ['A', 'S']]
['OpenList(2)', ['B', 'C']]

['state', 'B']
['OpenList(1)', ['C']]
['ClosedList', ['B', 'A', 'S']]
['OpenList(2)', ['C', 'F', 'E']]

['state', 'C']
['OpenList(1)', ['F', 'E']]
['ClosedList', ['C', 'A', 'S', 'B']]
['OpenList(2)', ['F', 'E', 'D']]
```

```
['state', 'F']
['OpenList(1)', ['E', 'D']]
['ClosedList', ['F', 'A', 'S', 'C', 'B']]
['OpenList(2)', ['E', 'D']]

['state', 'E']
['OpenList(1)', ['D']]
['ClosedList', ['F', 'A', 'S', 'C', 'E', 'B']]
['OpenList(2)', ['D', 'G']]

['state', 'D']
['OpenList(1)', ['G']]
['ClosedList', ['F', 'A', 'S', 'D', 'C', 'E', 'B']]
['OpenList(2)', ['G']]

['state', 'G']
['OpenList(1)', []]
['ClosedList', ['F', 'A', 'S', 'G', 'D', 'C', 'E', 'B']]
Completed
```

[ ]:

[208]:

```
# New with the cost calculation
CostList=[]
state=[]
OpenList=['S']
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    print(state)
    del OpenList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    print(costM)
    print(costM[0])
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    print(costMat)
    OpenList.insert(0, activeNodes)  # the first item
    CostList=costMat+CostList  # the first item
    print('OpenList(1): ',OpenList)
    print('CostList(1): ',CostList)
    #OpenList=[item for i in OpenList for item in i if i not in ClosedList]
    OpenList=[item for i in OpenList for item in i]
```

```python
    #CostList=[item for i in CostList for item in i]
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    OpenList=[OpenList[i] for i in key]
    #CostList=[CostList[i] for i in key]
    print('key: ',key)
    print('OpenList(2): ',OpenList)
    print('CostList(2): ',CostList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
[('A', 'S'), ('B', 'S')]
('A', 'S')
[2, 6]
OpenList(1):  [['A', 'B']]
CostList(1):  [2, 6]
key:  [0, 1]
OpenList(2):  ['A', 'B']
CostList(2):  [2, 6]
ClosedList:  ['S']
A
[('B', 'A'), ('C', 'A')]
('B', 'A')
[2, 1]
OpenList(1):  [['B', 'C'], 'B']
CostList(1):  [2, 1, 2, 6]
key:  [0, 1, 2]
OpenList(2):  ['B', 'C', 'B']
CostList(2):  [2, 1, 2, 6]
ClosedList:  ['S', 'A']
B
[('E', 'B'), ('F', 'B')]
('E', 'B')
[5, 4]
OpenList(1):  [['E', 'F'], 'C', 'B']
CostList(1):  [5, 4, 2, 1, 2, 6]
key:  [0, 1, 2]
OpenList(2):  ['E', 'F', 'C']
CostList(2):  [5, 4, 2, 1, 2, 6]
ClosedList:  ['S', 'A', 'B']
E
[('C', 'E'), ('D', 'E'), ('G', 'E')]
('C', 'E')
[2, 1, 5]
OpenList(1):  [['C', 'D', 'G'], 'F', 'C']
CostList(1):  [2, 1, 5, 5, 4, 2, 1, 2, 6]
key:  [0, 1, 2, 3, 4]
```

```
OpenList(2):  ['C', 'D', 'G', 'F', 'C']
CostList(2):  [2, 1, 5, 5, 4, 2, 1, 2, 6]
ClosedList:  ['S', 'A', 'B', 'E']
C
[('D', 'C')]
('D', 'C')
[5]
OpenList(1):  [['D'], 'D', 'G', 'F', 'C']
CostList(1):  [5, 2, 1, 5, 5, 4, 2, 1, 2, 6]
key:  [0, 1, 2, 3]
OpenList(2):  ['D', 'D', 'G', 'F']
CostList(2):  [5, 2, 1, 5, 5, 4, 2, 1, 2, 6]
ClosedList:  ['S', 'A', 'B', 'E', 'C']
D
[('G', 'D')]
('G', 'D')
[1]
OpenList(1):  [['G'], 'D', 'G', 'F']
CostList(1):  [1, 5, 2, 1, 5, 5, 4, 2, 1, 2, 6]
key:  [0, 2, 3]
OpenList(2):  ['G', 'G', 'F']
CostList(2):  [1, 5, 2, 1, 5, 5, 4, 2, 1, 2, 6]
ClosedList:  ['S', 'A', 'B', 'E', 'C', 'D']
G
completed
```

[252]:

```python
# New version with sort
CostList=[]
state=[]
stateC=[]
OpenList=['S']
CostList=[0]
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    stateC=CostList[0]
    print(state)
    del OpenList[0]
    del CostList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    OpenList=activeNodes+OpenList  # the first item
    CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the first item
    print('OpenList(1): ',OpenList)
    print('CostList(1): ',CostList)
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    OpenList=[OpenList[i] for i in key]
    CostList=[CostList[i] for i in key]
    print('OpenList(2): ',OpenList)
    print('CostList(2): ',CostList)
    mergeM=[(OpenList[i],CostList[i]) for i in range(len(OpenList)) ]
    mergeMs=sorted(mergeM, key = lambda x:x[1])
    OpenList=[i[0] for i in mergeMs]
    CostList=[i[1] for i in mergeMs]
    print('OpenList(sorted): ',OpenList)
    print('CostList(sorted): ',CostList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
OpenList(1):  ['A', 'B']
CostList(1):  [2, 6]
OpenList(2):  ['A', 'B']
CostList(2):  [2, 6]
OpenList(sorted):  ['A', 'B']
CostList(sorted):  [2, 6]
ClosedList:  ['S']
A
OpenList(1):  ['B', 'C', 'B']
CostList(1):  [4, 3, 6]
OpenList(2):  ['B', 'C', 'B']
CostList(2):  [4, 3, 6]
OpenList(sorted):  ['C', 'B', 'B']
CostList(sorted):  [3, 4, 6]
ClosedList:  ['S', 'A']
C
OpenList(1):  ['E', 'D', 'B', 'B']
CostList(1):  [5, 8, 4, 6]
OpenList(2):  ['E', 'D', 'B', 'B']
CostList(2):  [5, 8, 4, 6]
OpenList(sorted):  ['B', 'E', 'B', 'D']
CostList(sorted):  [4, 5, 6, 8]
ClosedList:  ['S', 'A', 'C']
B
OpenList(1):  ['E', 'F', 'E', 'B', 'D']
CostList(1):  [9, 8, 5, 6, 8]
OpenList(2):  ['E', 'F', 'E', 'D']
CostList(2):  [9, 8, 5, 8]
OpenList(sorted):  ['E', 'F', 'D', 'E']
```

```
CostList(sorted):  [5, 8, 8, 9]
ClosedList:  ['S', 'A', 'C', 'B']
E
OpenList(1):  ['D', 'G', 'F', 'D', 'E']
CostList(1):  [6, 10, 8, 8, 9]
OpenList(2):  ['D', 'G', 'F', 'D']
CostList(2):  [6, 10, 8, 8]
OpenList(sorted):  ['D', 'F', 'D', 'G']
CostList(sorted):  [6, 8, 8, 10]
ClosedList:  ['S', 'A', 'C', 'B', 'E']
D
OpenList(1):  ['G', 'F', 'D', 'G']
CostList(1):  [7, 8, 8, 10]
OpenList(2):  ['G', 'F', 'G']
CostList(2):  [7, 8, 10]
OpenList(sorted):  ['G', 'F', 'G']
CostList(sorted):  [7, 8, 10]
ClosedList:  ['S', 'A', 'C', 'B', 'E', 'D']
G
completed
```

[214]:

```python
# New version
CostList=[]
state=[]
OpenList=['S']
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    print(state)
    del OpenList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    #print(costM)
    #print(costM[0])
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    print(costMat)
    OpenList=activeNodes+OpenList  # the first item
    CostList=costMat+CostList  # the first item
    print('OpenList(1): ',OpenList)
    print('CostList(1): ',CostList)
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    OpenList=[OpenList[i] for i in key]
    CostList=[CostList[i] for i in key]
    #print('key: ',key)
    print('OpenList(2): ',OpenList)
    print('CostList(2): ',CostList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
[2, 6]
OpenList(1):  ['A', 'B']
CostList(1):  [2, 6]
OpenList(2):  ['A', 'B']
CostList(2):  [2, 6]
ClosedList:  ['S']
A
[2, 1]
OpenList(1):  ['B', 'C', 'B']
CostList(1):  [2, 1, 2, 6]
OpenList(2):  ['B', 'C', 'B']
CostList(2):  [2, 1, 2]
ClosedList:  ['S', 'A']
B
[5, 4]
OpenList(1):  ['E', 'F', 'C', 'B']
CostList(1):  [5, 4, 2, 1, 2]
OpenList(2):  ['E', 'F', 'C']
CostList(2):  [5, 4, 2]
ClosedList:  ['S', 'A', 'B']
E
[2, 1, 5]
OpenList(1):  ['C', 'D', 'G', 'F', 'C']
CostList(1):  [2, 1, 5, 5, 4, 2]
OpenList(2):  ['C', 'D', 'G', 'F', 'C']
CostList(2):  [2, 1, 5, 5, 4]
ClosedList:  ['S', 'A', 'B', 'E']
C
[5]
OpenList(1):  ['D', 'D', 'G', 'F', 'C']
CostList(1):  [5, 2, 1, 5, 5, 4]
OpenList(2):  ['D', 'D', 'G', 'F']
CostList(2):  [5, 2, 1, 5]
ClosedList:  ['S', 'A', 'B', 'E', 'C']
D
[1]
OpenList(1):  ['G', 'D', 'G', 'F']
CostList(1):  [1, 5, 2, 1, 5]
OpenList(2):  ['G', 'G', 'F']
CostList(2):  [1, 2, 1]
```

```
ClosedList:  ['S', 'A', 'B', 'E', 'C', 'D']
G
completed
```

[3]:

```
# New version
CostList=[]
state=[]
stateC=[]
OpenList=['S']
CostList=[0]
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    stateC=CostList[0]
    print(state)
    del OpenList[0]
    del CostList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    #activeNodes=[item for item in TargetGraph[state] ]
    costM=[(s,state) for s in activeNodes]
    print(costM)
    #print(costM[0])
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    print(costMat)
    OpenList=activeNodes+OpenList  # the first item
    #print(stateC*costMat)
    #CostList=stateC*costMat+CostList  # the first item
    CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the first item
    print('OpenList(1): ',OpenList)
    print('CostList(1): ',CostList)
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    OpenList=[OpenList[i] for i in key]
    CostList=[CostList[i] for i in key]
    #print('key: ',key)
    print('OpenList(2): ',OpenList)
    print('CostList(2): ',CostList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
[('A', 'S'), ('B', 'S')]
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
```

```
/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_29209/1580446706.py␣
 ↪in <module>
     21     print(costM)
     22     #print(costM[0])
---> 23     costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
     24     print(costMat)
     25     OpenList=activeNodes+OpenList  # the first item

/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_29209/1580446706.py␣
 ↪in <listcomp>(.0)
     21     print(costM)
     22     #print(costM[0])
---> 23     costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
     24     print(costMat)
     25     OpenList=activeNodes+OpenList  # the first item

NameError: name 'eachCost' is not defined
```

[257]:

```python
stateC=2
costMat=[2,4]
CostList=[1,2,3]
CostList=stateC*costMat+CostList  # the first item
print(stateC*costMat)
print(CostList)
```

```
[2, 4, 2, 4]
[2, 4, 2, 4, 1, 2, 3]
```

[259]:

```python
stateC=2
costMat=[2,4]
CostList=[1,2,3]
CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the first item
print(list(map(lambda x: x + stateC, costMat)))
print(CostList)
```

```
[4, 6]
[4, 6, 1, 2, 3]
```

[2]:

```python
# New version with sort
CostList=[]
state=[]
stateC=[]
OpenList=['S']
CostList=[0]
ClosedList=[]
while OpenList:
    #print(OpenList)
    state=OpenList[0]
    stateC=CostList[0]
    print(state)
    del OpenList[0]
    del CostList[0]
    ClosedList.append(state)
    if state=='G':
        break
    activeNodes=[item for item in TargetGraph[state] if item not in ClosedList]
    costM=[(s,state) for s in activeNodes]
    costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
    OpenList=activeNodes+OpenList  # the first item
    CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the first item
    print('OpenList(1): ',OpenList)
    print('CostList(1): ',CostList)
    key=[k for k in range(len(OpenList)) if OpenList[k] not in ClosedList]
    OpenList=[OpenList[i] for i in key]
    CostList=[CostList[i] for i in key]
    #print('OpenList(2): ',OpenList)
    #print('CostList(2): ',CostList)
    mergeM=[(OpenList[i],CostList[i]) for i in range(len(OpenList)) ]
    mergeMs=sorted(mergeM, key = lambda x:x[1])
    OpenList=[i[0] for i in mergeMs]
    CostList=[i[1] for i in mergeMs]
    print('OpenList(sorted): ',OpenList)
    print('CostList(sorted): ',CostList)
    print('ClosedList: ',ClosedList)
print('completed')
```

```
S
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_29209/3817949483.py␣
 ↪in <module>
     18      activeNodes=[item for item in TargetGraph[state] if item not in␣
 ↪ClosedList]
     19      costM=[(s,state) for s in activeNodes]
---> 20      costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
     21      OpenList=activeNodes+OpenList  # the first item
     22      CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the␣
 ↪first item

/var/folders/mg/w5t8lkhc8xj79f001s7kzpfh0000gp/T/ipykernel_29209/3817949483.py␣
 ↪in <listcomp>(.0)
     18      activeNodes=[item for item in TargetGraph[state] if item not in␣
 ↪ClosedList]
```

```
     19     costM=[(s,state) for s in activeNodes]
---> 20     costMat=[eachCost(costM[i],Node,C) for i in range(len(costM))]
     21     OpenList=activeNodes+OpenList  # the first item
     22     CostList=list(map(lambda x: x + stateC, costMat))+CostList  # the␣
 ↪first item

NameError: name 'eachCost' is not defined
```

[246]:

```
import itertools

CostList=[[2, 1], [2, 6]]
print(CostList)
print([item  for i in CostList for item in [i] ])
print([item  for i in CostList for item in i if type(i)==list])
print([item for i in CostList for item in [i] if type(i)!=list])


#list(itertools.chain.from_iterable(CostList))
print(CostList)
```

```
[[2, 1], [2, 6]]
[[2, 1], [2, 6]]
[2, 1, 2, 6]
[]
[[2, 1], [2, 6]]
```

[247]:

```
import itertools

l_2d = [[0, 1], 2, 3]

print(list(itertools.chain.from_iterable(l_2d)))
# [0, 1, 2, 3]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-247-17c4a5d48cf1> in <module>()
      3 l_2d = [[0, 1], 2, 3]
      4
----> 5 print(list(itertools.chain.from_iterable(l_2d)))
      6 # [0, 1, 2, 3]

TypeError: 'int' object is not iterable
```

[151]:

```
keyT=[1,4,5]
[Node[i] for i in keyT]
```

```
[151]: ['A', 'D', 'E']
```

```
[168]: type(1)
       type([1])
       i=[1]
       type(i)==list
       type(i)!=list
```

```
[168]: False
```

```
[ ]:
```