

# lec1\_step8

October 22, 2020

```
In [ ]: # Practice 3-2 (page 18/29)
        # https://www.slideshare.net/tadahirotaniguchi0624/3-46861684
```

```
In [38]: 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']
    }
```

```
In [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

```

```

In [75]: H = {'S': 0, 'A': 5, 'B': 8, 'C': 1, 'D': 2, 'E': 6}
         sorted(H)

Out[75]: ['A', 'B', 'C', 'D', 'E', 'S']

In [76]: sorted(H.keys())

Out[76]: ['A', 'B', 'C', 'D', 'E', 'S']

In [42]: sorted(H.values())

Out[42]: [0, 1, 2, 5, 6, 8]

In [44]: sorted(H.items(), key = lambda x:x[0])

Out[44]: [('A', 5), ('B', 8), ('C', 1), ('D', 2), ('E', 6), ('S', 0)]

In [74]: sorted(H.items(), key = lambda x:x[1])

Out[74]: [('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]

In [77]: H2=sorted(H.items(), key = lambda x:x[1])
         print(H2)

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

```

```

In [78]: sorted(H2, key = lambda x:x[1])
Out[78]: [('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]

In [79]: sorted(H2, key = lambda x:x[0])
Out[79]: [('A', 5), ('B', 8), ('C', 1), ('D', 2), ('E', 6), ('S', 0)]

In [80]: [i[0] for i in H2 ]
Out[80]: ['S', 'C', 'D', 'A', 'E', 'B']

In [69]: [i[1] for i in H2 ]
Out[69]: [0, 1, 2, 5, 6, 8]

In [81]: hh1=[i[0] for i in H2 ]
          hh2=[i[1] for i in H2 ]

In [87]: [(hh1[i],hh2[i]) for i in range(len(hh1)) ]
Out[87]: [('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]

In [86]: [(hh1[i],hh2[i]) for i in range(len(hh1)) ]
Out[86]: [('S', 0), ('C', 1), ('D', 2), ('A', 5), ('E', 6), ('B', 8)]

In [90]: 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]
           ]

In [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, 5],

In [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]]

```

```

In [99]: 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']

In [110]: [s for s in range(len(Node)) if 'E' in Node[s]][0]

Out[110]: 5

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

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

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

S
A

In [118]: C[1][2]

Out[118]: 2

In [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]

Out[121]: 2

In [129]: 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']
```

Out[129]: 2

In [142]: *# 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

```

```

In [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]
    #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

```

In [252]: *# 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

```

```

In [214]: # 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

```

```

In [243]: # 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')]
[2, 6]
OpenList(1):  ['A', 'B']
CostList(1):  [2, 6]
OpenList(2):  ['A', 'B']
CostList(2):  [2, 6]
ClosedList:  ['S']

```

A

```

[('B', 'A'), ('C', 'A')]
[2, 1]
OpenList(1):  ['B', 'C', 'B']
CostList(1):  [4, 3, 6]
OpenList(2):  ['B', 'C', 'B']
CostList(2):  [4, 3, 6]
ClosedList:  ['S', 'A']

```

B

```

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

```

E

```

[('C', 'E'), ('D', 'E'), ('G', 'E')]
[2, 1, 5]
OpenList(1):  ['C', 'D', 'G', 'F', 'C']
CostList(1):  [11, 10, 14, 8, 3]
OpenList(2):  ['C', 'D', 'G', 'F', 'C']
CostList(2):  [11, 10, 14, 8, 3]
ClosedList:  ['S', 'A', 'B', 'E']

```

C

```

[('D', 'C')]

```

```

[5]
OpenList(1):  ['D', 'D', 'G', 'F', 'C']
CostList(1):  [16, 10, 14, 8, 3]
OpenList(2):  ['D', 'D', 'G', 'F']
CostList(2):  [16, 10, 14, 8]
ClosedList:   ['S', 'A', 'B', 'E', 'C']
D
[('G', 'D')]
[1]
OpenList(1):  ['G', 'D', 'G', 'F']
CostList(1):  [17, 10, 14, 8]
OpenList(2):  ['G', 'G', 'F']
CostList(2):  [17, 14, 8]
ClosedList:   ['S', 'A', 'B', 'E', 'C', 'D']
G
completed

```

```

In [257]: 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]

```

```

In [259]: 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]

```

```

In [253]: # 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(sorted):  ['A', 'B']
CostList(sorted):  [2, 6]
ClosedList:  ['S']

```

A

```

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

```

```
In [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]]

```

```
In [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

```
In [151]: keyT=[1,4,5]
         [Node[i] for i in keyT]
```

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

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

```
Out[168]: False
```

```
In [ ]:
```