



Learn Git and GitHub without any code!

Using the Hello World guide, you'll start a branch, write comments, and open a pull request.

Read the guide

akueisara / algo-on-graphs

<> Code ⓘ Issues 1 🔄 Pull requests 1 ⏮ Actions 📁 Projects 📖 Wiki 🛡 Security 📈 Insights

🔗 master ▾ algo-on-graphs / week 6 / friend_suggestion / friend_suggestion.py3 / <> Jump to ▾

Go to file ⋮

👤 akueisara Add week 6 friend_suggestion

Latest commit ac2dbc0 on Mar 23, 2017 🕒 History

🔍 1 contributor

78 lines (66 sloc) 2.69 KB

Raw Blame 🖨 ✎ 🗑

```
1  #!/usr/bin/python3
2
3  import sys
4  import queue
5
6  class BiDij:
7      def __init__(self, n):
8          self.n = n;                # Number of nodes
9          self.inf = n*10**6         # All distances in the graph are smaller
10         self.d = [[self.inf]*n, [self.inf]*n] # Initialize distances for forward and backward searches
11         self.visited = [False]*n    # visited[v] == True iff v was visited by forward or backward search
12         self.workset = []           # All the nodes visited by forward or backward search
13
14     def clear(self):
15         """Reinitialize the data structures for the next query after the previous query."""
16         for v in self.workset:
17             self.d[0][v] = self.d[1][v] = self.inf
18             self.visited[v] = False;
19         del self.workset[0:len(self.workset)]
20
21     def visit(self, q, side, v, dist):
22         """Try to relax the distance to node v from direction side by value dist."""
23         # Implement this method yourself
24         if self.d[side][v] > dist:
25             self.d[side][v] = dist
26             q[side].put((self.d[side][v], v))
27             self.workset.append(v)
28
29     def extract_min(self, q, side):
30         _, v = q[side].get()
31         return v
32
33     def process(self, q, side, v, adj, cost):
34         for u, w in zip(adj[v], cost[v]):
35             self.visit(q, side, u, self.d[side][v] + w)
36
37     def shortest_path(self, v):
38         distance = self.inf
39         for u in self.workset:
40             if self.d[0][u] + self.d[1][u] < distance:
41                 distance = self.d[0][u] + self.d[1][u]
42         return (distance if distance != self.inf else -1)
43
44     def query(self, adj, cost, s, t):
45         self.clear()
46         q = [queue.PriorityQueue(), queue.PriorityQueue()]
47         self.visit(q, 0, s, 0)
48         self.visit(q, 1, t, 0)
49         # Implement the rest of the algorithm yourself
50         while not q[0].empty() and not q[1].empty():
51             for side in [0,1]:
52                 v = self.extract_min(q, side)
53                 self.process(q, side, v, adj[side], cost[side])
54                 if self.visited[v]:
55                     return self.shortest_path(v)
56                 self.visited[v] = True
57         return -1
58
59
60     def readl():
61         return map(int, sys.stdin.readline().split())
62
63
64     if __name__ == '__main__':
65         n,m = readl()
66         adj = [[[ for _ in range(n)], [ for _ in range(n)]]
67         cost = [[[ for _ in range(n)], [ for _ in range(n)]]
68         for e in range(m):
69             u,v,c = readl()
70             adj[0][u-1].append(v-1)
71             cost[0][u-1].append(c)
72             adj[1][v-1].append(u-1)
73             cost[1][v-1].append(c)
74         t, = readl()
75         bidij = BiDij(n)
76         for i in range(t):
77             s, t = readl()
78             print(bidij.query(adj, cost, s-1, t-1))
```