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

```python
#!/usr/bin/python3

import sys
import queue

class BiDij:
    def __init__(self, n):
        self.n = n;                             # Number of nodes
        self.inf = n*10**6                      # All distances in the graph are smaller
        self.d = [[self.inf]*n, [self.inf]*n]   # Initialize distances for forward and backward searches
        self.visited = [False]*n                  # visited[v] == True iff v was visited by forward or backward search
        self.workset = []                       # All the nodes visited by forward or backward search

    def clear(self):
        """Reinitialize the data structures for the next query after the previous query."""
        for v in self.workset:
            self.d[0][v] = self.d[1][v] = self.inf
            self.visited[v] = False;
        del self.workset[0:len(self.workset)]

    def visit(self, q, side, v, dist):
        """Try to relax the distance to node v from direction side by value dist."""
        # Implement this method yourself
        if self.d[side][v] > dist:
            self.d[side][v] = dist
            q[side].put((self.d[side][v], v))
            self.workset.append(v)

    def extract_min(self, q, side):
            _, v = q[side].get()
            return v

    def process(self, q, side, v, adj, cost):
        for u, w in zip(adj[v], cost[v]):
            self.visit(q, side, u, self.d[side][v] + w)

    def shortest_path(self, v):
        distance = self.inf
        for u in self.workset:
            if self.d[0][u] + self.d[1][u] < distance:
                distance = self.d[0][u] + self.d[1][u]
        return (distance if distance != self.inf else -1)

    def query(self, adj, cost, s, t):
        self.clear()
        q = [queue.PriorityQueue(), queue.PriorityQueue()]
        self.visit(q, 0, s, 0)
        self.visit(q, 1, t, 0)
        # Implement the rest of the algorithm yourself
        while not q[0].empty() and not q[1].empty():
            for side in [0,1]:
                v = self.extract_min(q, side)
                self.process(q, side, v, adj[side], cost[side])
                if self.visited[v]:
                    return self.shortest_path(v)
                self.visited[v] = True
        return -1


def readl():
    return map(int, sys.stdin.readline().split())


if __name__ == '__main__':
    n,m = readl()
    adj = [[[] for _ in range(n)], [[] for _ in range(n)]]
    cost = [[[] for _ in range(n)], [[] for _ in range(n)]]
    for e in range(m):
        u,v,c = readl()
        adj[0][u-1].append(v-1)
        cost[0][u-1].append(c)
        adj[1][v-1].append(u-1)
        cost[1][v-1].append(c)
    t, = readl()
    bidij = BiDij(n)
    for i in range(t):
        s, t = readl()
        print(bidij.query(adj, cost, s-1, t-1))
```