problem set 2实验报告

16337183 孟衍璋

Problem 1

　　首先需要创建数据结构来表示地图。地图抽象出来是包含一系列节点与带权重的边的图，题目中不仅给出了从一个地方到另一个地方的路程，还给出了相应的户外移动的距离，这一点为后面寻找最短路径增加了一个约束。

　　在`graph.py`中，给出了`Node`类与`Edge`类，我们需要完成`WeightedEdge`与`Digraph`类的构建。相应的实现如下：

```
class WeightedEdge(Edge):
    def __init__(self, src, dest, total_distance, outdoor_distance):
        self.src = src
        self.dest = dest
        self.total_distance = total_distance
        self.outdoor_distance = outdoor_distance

    def get_total_distance(self):
        return self.total_distance

    def get_outdoor_distance(self):
        return self.outdoor_distance

    def __str__(self):
        return '%s->%s (%s, %s)' % (self.src, self.dest,
self.total_distance, self.outdoor_distance)
```

```
class Digraph(object):
    """Represents a directed graph of Node and Edge objects"""
    def __init__(self):
        self.nodes = set([])
        self.edges = {}  # must be a dict of Node -> list of edges

    def __str__(self):
```

```python
        edge_strs = []
        for edges in self.edges.values():
            for edge in edges:
                edge_strs.append(str(edge))
        edge_strs = sorted(edge_strs)  # sort alphabetically
        return '\n'.join(edge_strs)  # concat edge_strs with "\n"s between
them

    def get_edges_for_node(self, node):
        return self.edges[node]

    def has_node(self, node):
        return node in self.nodes

    def add_node(self, node):
        """Adds a Node object to the Digraph. Raises a ValueError if it is
        already in the graph."""

        # 如果节点已经在图中, raise a ValueError
        if self.has_node(node):
            raise ValueError('node already in the graph')
        # 如果节点不在图中，加入节点信息
        else:
            self.nodes.add(node)
            self.edges[node] = []

    def add_edge(self, edge):
        """Adds an Edge or WeightedEdge instance to the Digraph. Raises a
        ValueError if either of the nodes associated with the edge is not
        in the  graph."""

        weighted_edge = WeightedEdge(edge.get_source(),
edge.get_destination(), edge.get_total_distance(),
edge.get_outdoor_distance())
        # 如果edge中的起始点和终点有一个没有在nodes中, raise a ValueError
        if (edge.get_source() in self.nodes and edge.get_destination() in
self.nodes) == 0:
            raise ValueError('nodes associated with the edge not in the
graph.')
        # 如果edge中的起始点和终点都在nodes中，则在edges中加入对应的信息
        else:
            self.edges[edge.get_source()].append(weighted_edge)
```

Problem 2

Problem 2a

What do the graph's nodes represent in this problem? What do the graph's edges represent? Where are the distances represented?

图的节点在这个问题中代表某栋大楼，图的边在这个问题中代表从某栋大楼前往其他地方的路径，距离体现在类WeightedEdge中。

Problem 2b

这一步需要完成读取文件的函数，实现如下：

```
def load_map(map_filename):
    print("Loading map from file...")
    with open(map_filename, 'r') as mf:
        digraph = Digraph()  # 初始化一个Digraph对象
        nodes = mf.readlines()
        # 将节点信息加入Digraph对象中
        for node in nodes:
            node = node.strip('\n').split(' ')
            # print(node)
            if not digraph.has_node(Node(node[0])):
                digraph.add_node(Node(node[0]))
            if not digraph.has_node(Node(node[1])):
                digraph.add_node(Node(node[1]))
            # 将边的信息加入Digraph对象中
            edge = WeightedEdge(Node(node[0]), Node(node[1]),
int(node[2]), int(node[3]))
            digraph.add_edge(edge)
    return digraph
```

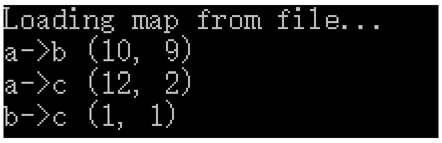
需要注意的是要将起点与终点分别转化为Node对象。

Problem 2c

测试`load_map`函数，只用一行便可以实现：

```python
print(load_map('./test_load_map.txt'))
```

测试结果如下：



```
Loading map from file...
a->b (10, 9)
a->c (12, 2)
b->c (1, 1)
```

Problem 3

Problem 3a

What is the objective function for this problem? What are the constraints?

这个问题的目标函数是找到最短的路径，约束条件是大楼之间的距离、户外行走的距离。

Problem 3b

接下来需要实现寻找最佳路径的函数，我使用了递归的方法，遍历与某个节点相连的所有边，将其对应的节点作为新的起点实现遍历，最后得到路径，实现如下：

```python
def get_best_path(digraph, start, end, path, max_dist_outdoors, best_dist,
                  best_path):
    # TODO
    path = path + [start]
    # 如果开始或结束节点不合法，则raise a ValueError
    if start not in digraph.nodes or end not in digraph.nodes:
        raise ValueError('not in the node list')
    # 如果开始节点与结束节点相同，说明已经找到了想要的路径，则返回对应路径
    elif start == end:
        return path
    else:
        # 遍历与当前节点相连且指向的所有节点
        for node in digraph.get_edges_for_node(start):
            if node.dest not in path:  # 如果那个被指向的节点还没有路过，则构建
一条新的路径包含那个节点
```

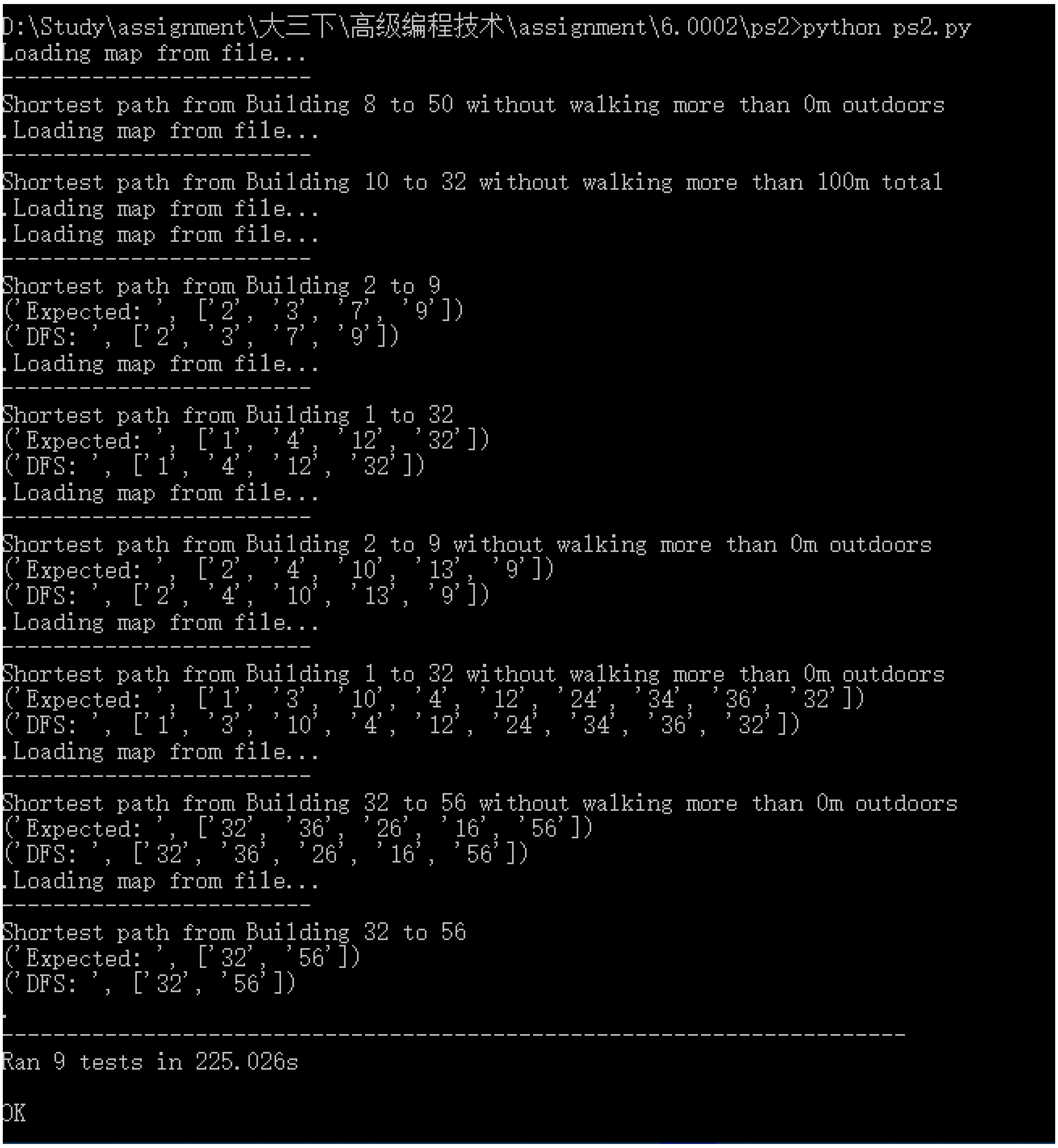
```
                next_path = get_best_path(digraph, node.dest, end, path,
max_dist_outdoors, best_dist, best_path)
                if next_path != None:
                    # 计算新的路径移动的总距离，户外移动的总距离
                    total_dist, outdoor_dist = getDistance(digraph,
next_path)
                    # 如果户外移动的距离不超过最大户外移动距离，且总距离与原来的相比
较小，则加入当前路径
                    if outdoor_dist <= max_dist_outdoors and total_dist <=
best_dist:
                        best_path = next_path
                        best_dist = total_dist
    return best_path

# 计算距离的函数
def getDistance(digraph, path):
    total_dist = outdoor_dist = 0
    for i in range(len(path) - 1):
        for edge in digraph.edges[path[i]]:
            if edge.dest == path[i + 1]:
                total_dist += edge.get_total_distance()
                outdoor_dist += edge.get_outdoor_distance()
    return (total_dist, outdoor_dist)
```

Problem 3c

使用`get_best_path`函数，输出最短的路径，实现如下：

```
def directed_dfs(digraph, start, end, max_total_dist, max_dist_outdoors):
    # TODO
    best_path = get_best_path(digraph, Node(start), Node(end), [],
max_dist_outdoors, max_total_dist, None)
    # 如果没有路径满足约束，则raise a ValueError
    if best_path == None:
        raise ValueError
    # 获取路径
    route = []
    for node in best_path:
        route.append(node.get_name())
    return route
```

测试结果如下:



```
D:\Study\assignment\大三下\高级编程技术\assignment\6.0002\ps2>python ps2.py
Loading map from file...
------------------------
Shortest path from Building 8 to 50 without walking more than 0m outdoors
.Loading map from file...
------------------------
Shortest path from Building 10 to 32 without walking more than 100m total
.Loading map from file...
.Loading map from file...
------------------------
Shortest path from Building 2 to 9
('Expected: ', ['2', '3', '7', '9'])
('DFS: ', ['2', '3', '7', '9'])
.Loading map from file...
------------------------
Shortest path from Building 1 to 32
('Expected: ', ['1', '4', '12', '32'])
('DFS: ', ['1', '4', '12', '32'])
.Loading map from file...
------------------------
Shortest path from Building 2 to 9 without walking more than 0m outdoors
('Expected: ', ['2', '4', '10', '13', '9'])
('DFS: ', ['2', '4', '10', '13', '9'])
.Loading map from file...
------------------------
Shortest path from Building 1 to 32 without walking more than 0m outdoors
('Expected: ', ['1', '3', '10', '4', '12', '24', '34', '36', '32'])
('DFS: ', ['1', '3', '10', '4', '12', '24', '34', '36', '32'])
.Loading map from file...
------------------------
Shortest path from Building 32 to 56 without walking more than 0m outdoors
('Expected: ', ['32', '36', '26', '16', '56'])
('DFS: ', ['32', '36', '26', '16', '56'])
.Loading map from file...
------------------------
Shortest path from Building 32 to 56
('Expected: ', ['32', '56'])
('DFS: ', ['32', '56'])
.
----------------------------------------------------------------------
Ran 9 tests in 225.026s

OK
```