Leetcode(python) 图简单

743网络延迟时间

题目：有 N 个网络节点，标记为 1 到 N。

给定一个列表 times，表示信号经过**有向**边的传递时间。 times[i] = (u, v, w)，其中 u 是源节点，v 是目标节点， w 是一个信号从源节点传递到目标节点的时间。

现在，我们向当前的节点 K 发送了一个信号。需要多久才能使所有节点都收到信号？如果不能使所有节点收到信号，返回 -1。

**注意:**

1. N 的范围在 [1, 100] 之间。
2. K 的范围在 [1, N] 之间。
3. times 的长度在 [1, 6000] 之间。
4. 所有的边 times[i] = (u, v, w) 都有 1 <= u, v <= N 且 1 <= w <= 100。

思路一：Dijkstra算法，纯模板题。

时间复杂度是O(N ^ 2 + E)，空间复杂度是O(N+E).

代码一：

1. **class** Solution(object):
2. **def** networkDelayTime(self, times, N, K):
3. """
4. :type times: List[List[int]]
5. :type N: int
6. :type K: int
7. :rtype: int
8. """
9. K-=1
10. nodes=collections.defaultdict(list)
11. **for** u,v,w **in** times:
12. nodes[u-1].append((v-1,w))
13. dist=[float('inf')]\*N
14. dist[K]=0
15. done=set()
16. **for** \_ **in** range(N):
17. smallest=min((d,i) **for** (i,d) **in** enumerate(dist) **if** i **not** **in** done)[1]
18. **for** v,w **in** nodes[smallest]:
19. **if** v **not** **in** done **and** dist[smallest]+w<dist[v]:
20. dist[v]=dist[smallest]+w
21. done.add(smallest)
22. **return** -1 **if** float('inf') **in** dist **else** max(dist)

思路二：Floyd-Warshall算法。这个算法TLE.

代码二：

1. **class** Solution(object):
2. **def** networkDelayTime(self, times, N, K):
3. """
4. :type times: List[List[int]]
5. :type N: int
6. :type K: int
7. :rtype: int
8. """
9. d=[[float('inf')]\*N **for** \_ **in** range(N)]
10. **for** time **in** times:
11. u,v,w=time[0]-1,time[1]-1,time[2]
12. d[u][v]=w
13. **for** i **in** range(N):
14. d[i][i]=0
15. **for** k **in** range(N):
16. **for** i **in** range(N):
17. **for** j **in** range(N):
18. d[i][j]=min(d[i][j],d[i][k]+d[k][j])
19. **return** -1 **if** float('inf') **in** d[K-1] **else** max(d[K-1])

思路三：Bellman-Ford算法，这个算法TLE。

代码三：

1. **class** Solution(object):
2. **def** networkDelayTime(self, times, N, K):
3. """
4. :type times: List[List[int]]
5. :type N: int
6. :type K: int
7. :rtype: int
8. """
9. dist=[float('inf')]\*N
10. dist[K-1]=0
11. **for** i **in** range(N):
12. **for** time **in** times:
13. u=time[0]-1
14. v=time[1]-1
15. w=time[2]
16. dist[v]=min(dist[v],dist[u]+w)
17. **return** -1 **if** float('inf') **in** dist **else** max(dist)