修改 Dijkstra 算法，使得当某一结点存在多条最短路径时，保留经过结点最少的那条路径

【解】这只需要对 Dijkstra 算法做一个小小的修改。在 Dijkstra 算法中，每次在尚未找到最短路径的结点集中选择了路径最短的结点 u 后，需要检查它的每一个后继 v。如果已知的源点到后继的距离大于源点经过 u 再到 v 的距离，则更新 v 的距离和路径信息。而当已知的源点到后继的距离小于等于源点经过 u 再到 v 的距离时，则不作任何操作。代码清单 14-5 对 Dijkstra 算法的修改就在于当已知的源点到后继的距离等于源点经过 u 再到 v 的距离时，进一步检查经过 u 到 v 的经过的结点数是否比已知路径少。如果少的话则更新路径。为此，在 Dijkstra 算法中增加了了一个数组 hop，记录从源点到每一个结点的已知最短路径上的结点数。每次更新路径信息时也同时更新 hop 数组。

代码清单 14-5 程序设计题 3 的解

```
1.  template <class TypeOfVer, class TypeOfEdge>
2.  void adjListGraph<TypeOfVer, TypeOfEdge>::dijkstra(TypeOfVer start,
    TypeOfEdge noEdge) const
3.  { TypeOfEdge *distance = new TypeOfEdge[Vers];
4.    int *prev = new int [Vers];
5.    int *hop = new int[Vers];                    // 保存经过的结点数
6.    bool *known = new bool[Vers];
7.
8.    int u, sNo, i, j;
9.    edgeNode *p;
10.   TypeOfEdge min;
11.
12.   for (i = 0; i< Vers; ++i) {                  // 初始化
13.         known[i] = false;
14.      distance[i] = noEdge;
15.         hop[i] = 0;
16.   }
17.
18.   for (sNo = 0; sNo < Vers; ++sNo)             // 寻找起始结点
19.         if (verList[sNo].ver == start) break;
20.   if (sNo == Vers){
21.         cout << "起始结点不存在" << endl;
22.         return;
23.   }
24.
25.   distance[sNo] = 0;
26.   prev[sNo] = sNo;
27.
28.   for (i = 1; i < Vers; ++i) {                 // 寻找 vers-1 个结点的路径
29.         min = noEdge;
30.         for (j = 0; j < Vers; ++j)
31.               if (!known[j] && distance[j] < min) {
```

```
32.                     min = distance[j];
33.                     u = j;
34.                 }
35.         known[u] = true;
36.             for (p = verList[u].head; p != NULL; p = p->next)
37.                 if (!known[p->end] && (distance[p->end] > min + p->weight ||
38.             distance[p->end] == min + p->weight && hop[p->end] > hop[u] + 1)) {
39.                 distance[p->end] = min + p->weight;
40.                 hop[p->end] = hop[u] + 1;
41.                 prev[p->end] = u;
42.                     }
43.     }
44.
45.     for (i = 0; i < Vers; ++i) {                        // 输出路径
46.             cout << "从" << start << "到" << verList[i].ver << "的路径为:" << endl;
47.             printPath(sNo, i, prev);
48.             cout << "\t 长度为 :  " << distance[i] << endl;
49.     }
50.   }
```