作业 5

xwzeng

2023 年 6 月 7 日

1

在平面直角坐标系上将以下各点依次标记为顶点 0～5：(1, 3), (2, 1), (6, 5), (3, 4), (3, 7), (5, 3)。取边长度（欧氏距离）为权值，连接以下 9 条边：1–0, 3–5, 5–2, 3–4, 5–1, 0–3, 0–4, 4–2, 2–3。

由这些顶点及边所定义的无向图如图1所示，边的权重四舍五入至 2 位小数。

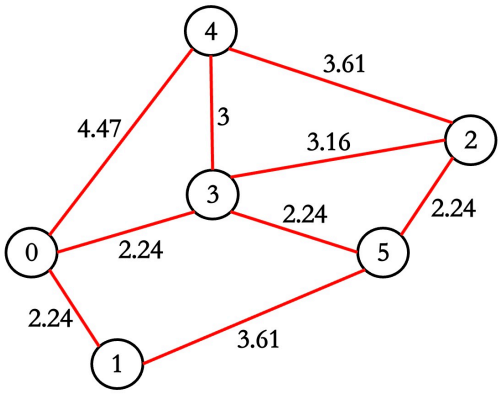

图 1: 第 1 题图示

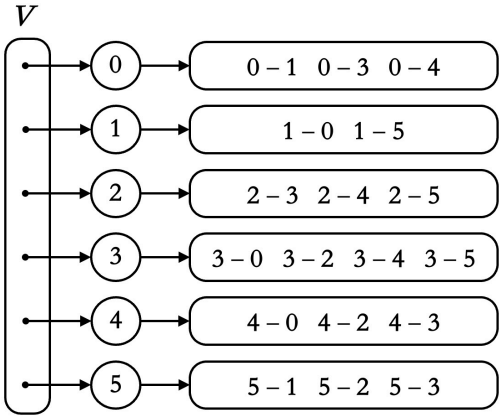

图 2: 邻接表

a) 邻接表

邻接表结构如图2所示。由于没有定义边，这里二级结构的元素都用顶点表示。

b) 最小生成树

由于图中存在多条权重相同的边，向最小生成树中添加边的顺序并不唯一。对于每种算法，我只选择了其中一种顺序展示。

- Prim-Jarnik 算法（见图3）：0–1, 0–3, 3–5, 5–2, 3–4
- Kruskal 算法（见图4）：0-1, 3-5, 0-3, 5-2, 3-4

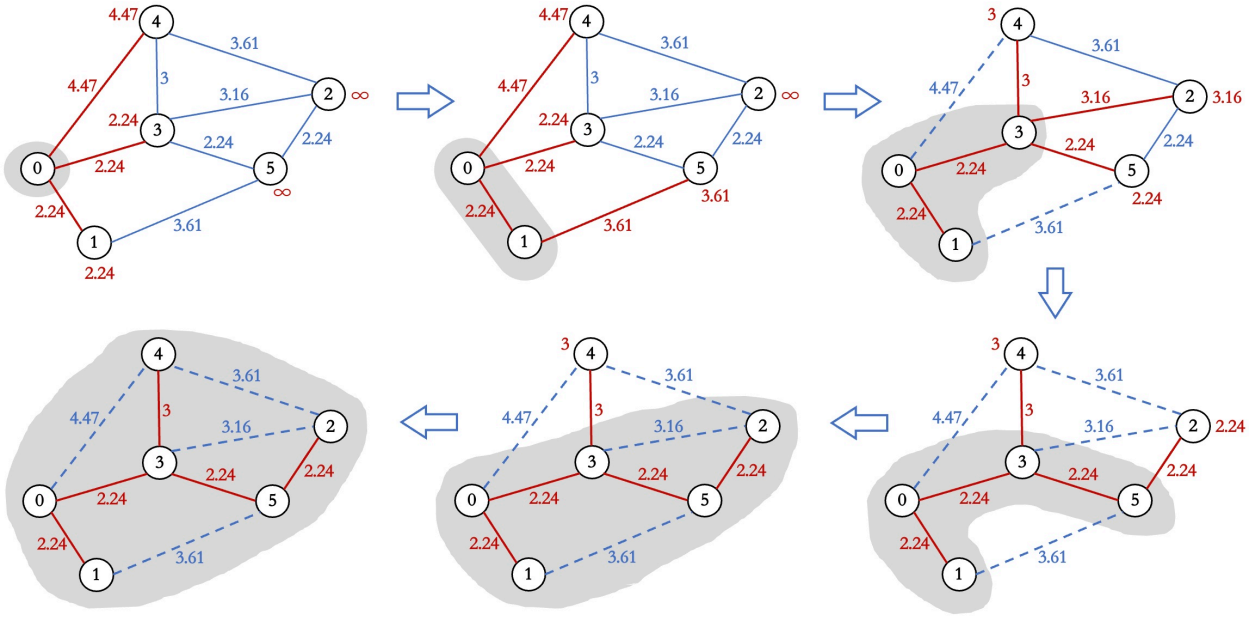

图 3: Prim-Jarnik 算法示意图

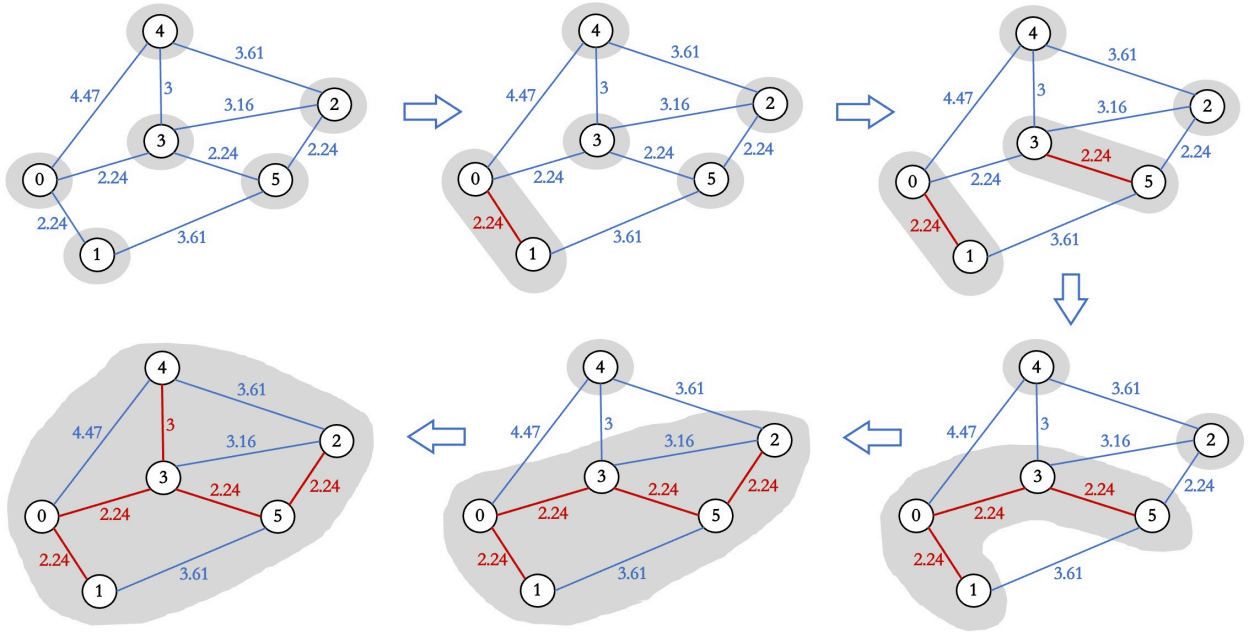

图 4: Kruskal 算法示意图

c) 最短路径树

使用 Dijkstra 算法得到的从顶点 0 出发的最短路径树见图5。

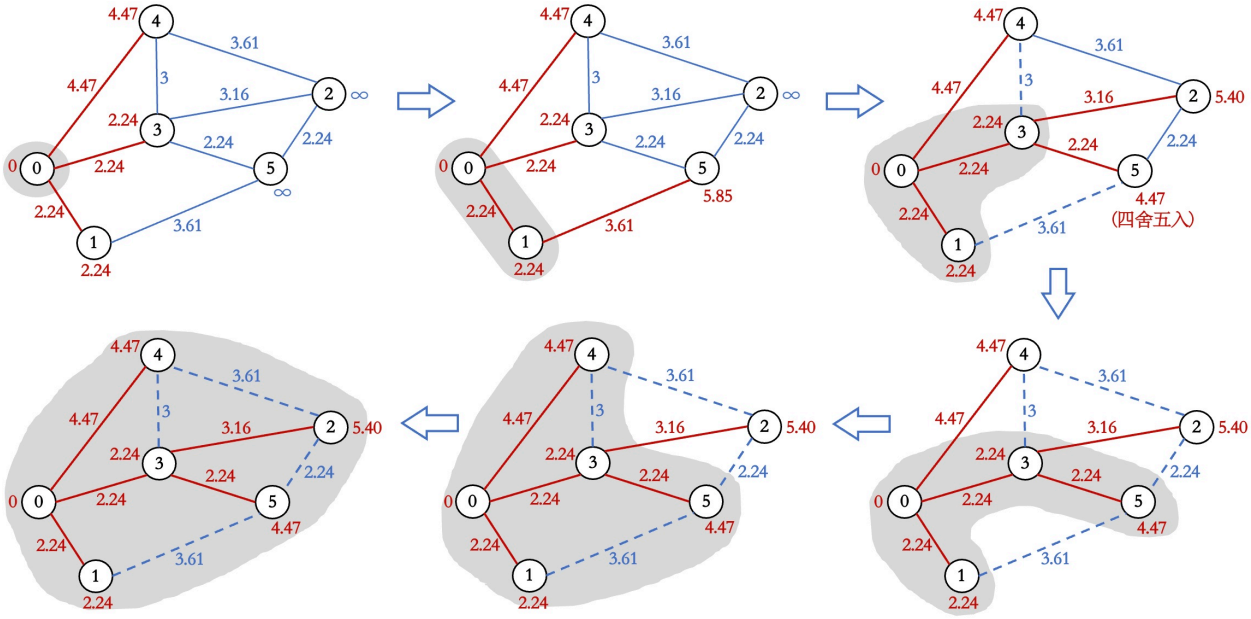

图 5: Dijkstra 算法示意图

d) 邻接矩阵

邻接矩阵结构如表1所示。

表 1: 邻接矩阵

	0	1	2	3	4	5
0	0	2.24	0	2.24	4.47	0
1	2.24	0	0	0	0	3.61
2	0	0	0	3.16	3.61	2.24
3	2.24	0	3.16	0	3	2.24
4	4.47	0	3.61	3	0	0
5	0	3.61	2.24	2.24	0	0

2

用序列（列表、集合）实现 Kruskal 算法中的 Partition 结构（课本上是基于树实现的）。

[ ]:
```python
class Partition:
    """Union-find structure for maintaining disjoint sets."""
    def __init__(self):
        self.group = []
    # ------------------------ nested Position class ----------------------
    class Position:
        __slots__ = '_container', '_element'
        def __init__(self, container, e):
            """Create a new position that is the leader of its own group."""
            self._container = container  # reference to Partition instance
            self._element = e
        def element(self):
            """Return element stored at this position."""
            return self._element
    # -------------------------- nonpublic utility -------------------------
    def _validate(self, p):
        if not isinstance(p, self.Position):
            raise TypeError('p must be proper Position type')
        if p._container is not self:
            raise ValueError('p does not belong to this container')
    # ---------------------- public Partition methods ----------------------
    def make_group(self, e):
        """Makes a new group containing element e, and returns its Position."""
        self.group.append({e})
        return self.Position(self, e)

    def find(self, p):
        """Finds the group containging p and return this group."""
        self._validate(p)
        for s in self.group:
            if p.element() in s:
                return s
        raise ValueError('p does not belong to this container')
```

```
    def union(self, p, q):
        """Merges the groups p and q (if distinct)."""
        if p is not q:  # only merge if different groups
            p |= q
            self.group.remove(q)
```

在第 1 题的图1上测试代码，结果正确。

[1]:
```
%run week14_2.py
```

```
| 2<->5 | 0<->1 | 3<->5 | 0<->3 | 3<->4 |
```

3

现有文本字符串 abaababaababababaca 和模式串 ababac，模式串的失败函数见表2。

表 2: 失败函数表

j	0	1	2	3	4	5
$P[j]$	a	b	a	b	a	c
$F(j)$	0	0	1	2	3	0

KMP 算法进行匹配的过程见图6，共比较了 22 次。

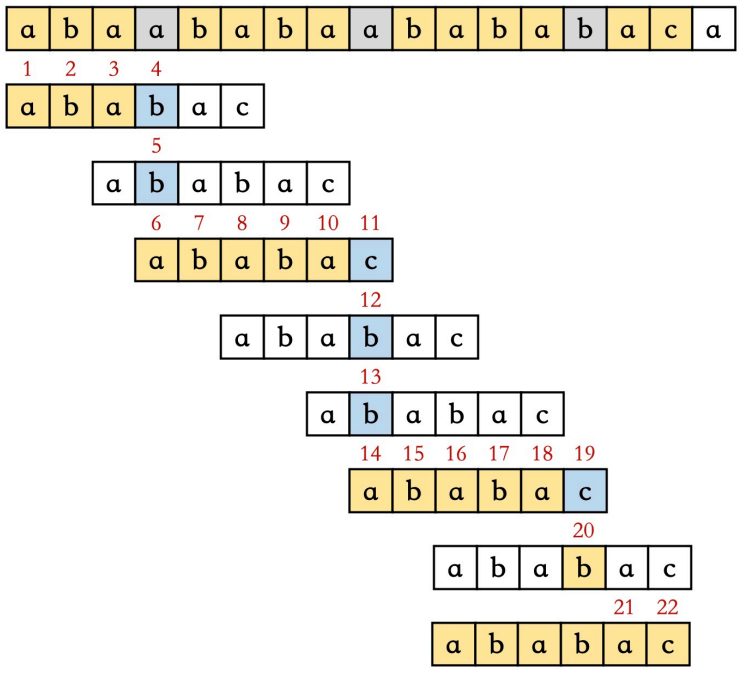

图 6: KMP 算法示意图