算法作业 9

2207070213 赖立勋

November 18, 2024

1

2 图解算法

2.1 问题描述

对图 9-1，设源点是 3，请给出 BFS 之后，每个节点的 parent 和 dis 值。(默认采用数字序)

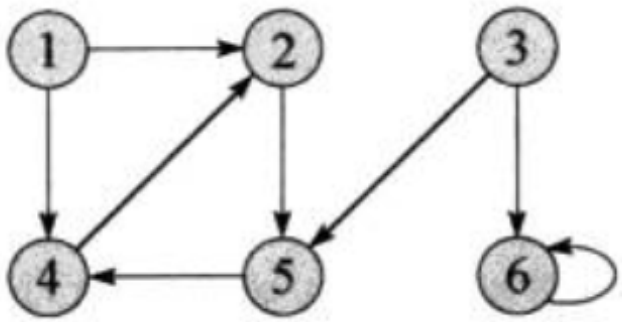

Figure 1: 图 9-1

对图 9-2，设源点是 u，请给出 BFS 之后，每个节点的 parent 和 dis 值。(默认采用字母序)

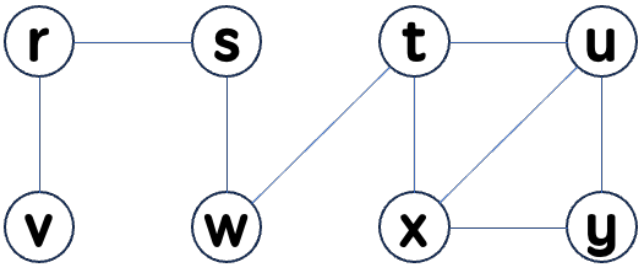

Figure 2: 图 9-2

2.2 解答

2.2.1 对图 9-1 执行 BFS 并确定每个节点的 parent 和 dis 值

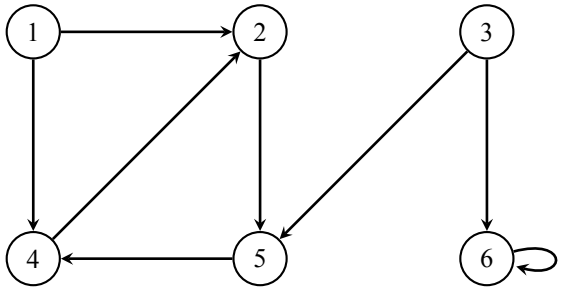

Figure 3: 图 9-1

1. **初始化，开始第一轮遍历：**

   - 所有节点染白，parent = -，dis = ∞
   - 3 入队，dis = 0

```
BFS-WALKER(G) // BFS遍历器
foreach v in [1,2,3,4,5,6] do
    v.color:=WHITE;
    v.parent=NULL
    v.dis:=+ϖ;
foreach v in [3,1,2,4,5,6] do // 从3开始遍历
    if v.color == WHITE then
        BFS(v)
```

```
        BFS(3)
            Initialize an empty queue queNode;
            3.color:=GRAY; // 3 染灰
            3.dis:=0; // 3 dis=0
            queNode.ENQUEUE(3); // 3 入队
```

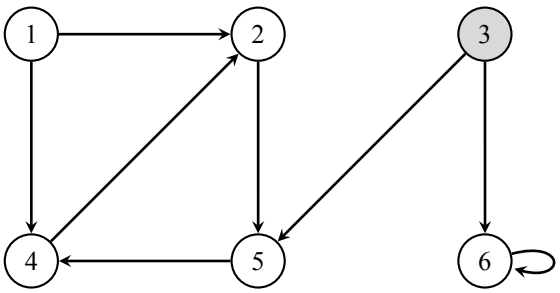

Figure 4: 图 9-1-0

节点编号	Parent	dis 值
1	-	∞
2	-	∞
3	-	0
4	-	∞
5	-	∞
6	-	∞

Figure 5: 图 9-1-0 的 BFS 结果

2. **第一层（dis = 0）：**

- 访问节点 3：
  - 邻居节点为 5 和 6。
  - 设置节点 5 和 6 的 dis = 1，parent = 3。
  - 将节点 5 和 6 入队。

```
while queNode = [3] != empty do
    w:=queNode.DEQUE(); // 队头 3 出队，赋给 w
    foreach neighbor [5,6] of w = 3 do
        if 5.color==WHITE then
            5.color:=GRAY; // 5 染灰
            5.parent:=3; // 5 parent=3
            5.dis:=3.dis+1; // 5 dis=1
            queNode.ENQUEUE(5); // 5 入队
        if 6.color==WHITE then
            6.color:=GRAY; // 6 染灰
            6.parent:=3; // 6 parent=3
            6.dis:=3.dis+1; // 6 dis=1
            queNode.ENQUEUE(6); // 6 入队
    <processing of node w>;
    3.color:=BLACK; // 3 染黑
```

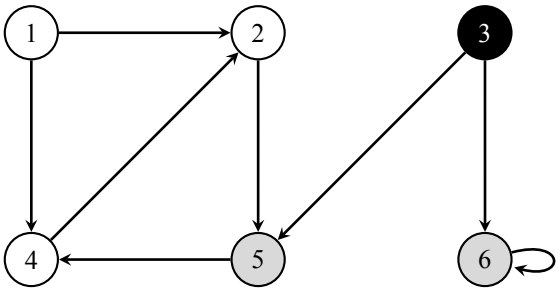

Figure 6: 图 9-1-1

节点编号	Parent	dis 值
1	-	∞
2	-	∞
3	-	0
4	-	∞
5	3	1
6	3	1

Figure 7: 图 9-1-1 的 BFS 结果

3. **第二层（dis = 1）：**

- 访问节点 5：
  - 邻居节点为 4。
  - 设置节点 4 的 dis = 2，parent = 5。
  - 将节点 4 入队。

```
while queNode = [5,6] != empty do // 数字序 5<6
    w:=queNode.DEQUE(); // 队头 5 出队，赋给 w
    foreach neighbor [4] of w = 5 do
        if 4.color==WHITE then
            4.color:=GRAY; // 4 染灰
            4.parent:=5; // 4 parent=5
            4.dis:=5.dis+1; // 4 dis=2
            queNode.ENQUEUE(4); // 4 入队
    <processing of node 5>;
    5.color:=BLACK; // 5 染黑
```

- 访问节点 6：
  - 邻居节点为自身，无需操作。

```
while queNode = [6,4] != empty do
    w:=queNode.DEQUE(); // 队头 6 出队，赋给 w
    foreach neighbor [6] of w = 6 do // 6 邻居是自己
        if 6.color==WHITE then // False
    <processing of node 6>;
    6.color:=BLACK; // 6 染黑
```

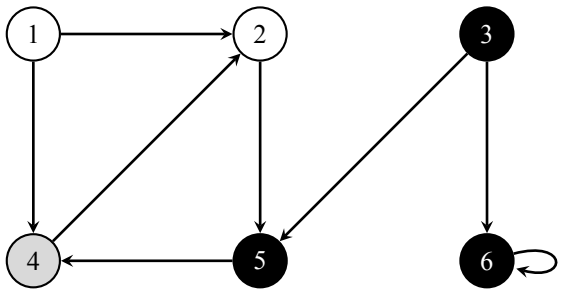

Figure 8: 图 9-1-2

节点编号	Parent	dis 值
1	-	∞
2	-	∞
3	-	0
4	5	2
5	3	1
6	3	1

Figure 9: 图 9-1-2 的 BFS 结果

4. **第三层（dis = 2）：**

- 访问节点 4：
  - 邻居节点为 2。
  - 设置节点 2 的 dis = 3，parent = 4。
  - 将节点 2 入队。

```
while queNode = [4] != empty do
    w:=queNode.DEQUE(); // 队头 4 出队，赋给 w
    foreach neighbor [2] of w = 4 do
        if 2.color==WHITE then
            2.color:=GRAY; // 2 染灰
            2.parent:=4; // 2 parent=4
            2.dis:=4.dis+1; // 2 dis=3
            queNode.ENQUEUE(2); // 2 入队
    <processing of node 4>;
    4.color:=BLACK; // 4 染黑
```

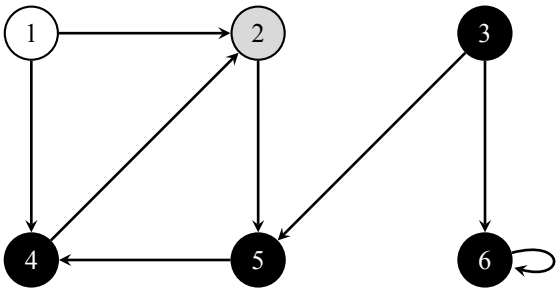

Figure 10: 图 9-1-3

节点编号	Parent	dis 值
1	-	∞
2	4	3
3	-	0
4	5	2
5	3	1
6	3	1

Figure 11: 图 9-1-3 的 BFS 结果

5. **第四层（dis = 3）：**

   - 访问节点 2：
     - 邻居节点为 5，5 已出队，无需操作。

```
                while queNode = [2] != empty do
                    w:=queNode.DEQUE(); // 队头 2 出队，赋给 w
                    foreach neighbor [5] of w = 2 do
                        if 5.color==WHITE then // False
                    <processing of node 2>;
                    2.color:=BLACK; // 2 染黑
```

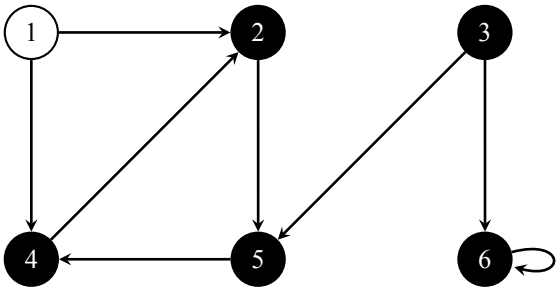

Figure 12: 图 9-1-4

节点编号	Parent	dis 值
1	-	∞
2	4	3
3	-	0
4	5	2
5	3	1
6	3	1

Figure 13: 图 9-1-4 的 BFS 结果

6. **开始第二轮遍历：**

   - 1 入队，dis = 0

```
  foreach v in [3,1,2,4,5,6] do // 1为白色
      if v.color == WHITE then
          BFS(v)
```

```
          BFS(1)
              Initialize an empty queue queNode;
              1.color:=GRAY; // 1 染灰
              1.dis:=0; // 1 dis=0
              queNode.ENQUEUE(1); // 1 入队
```

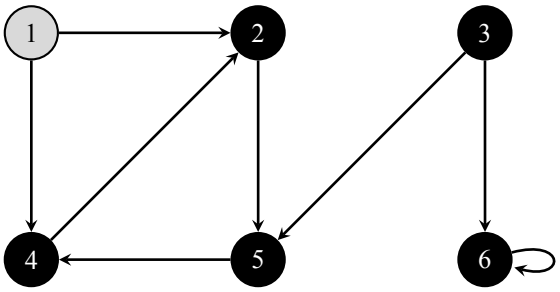

Figure 14: 图 9-1-5

节点编号	Parent	dis 值
1	-	0
2	4	3
3	-	0
4	5	2
5	3	1
6	3	1

Figure 15: 图 9-1-5 的 BFS 结果

7. **第二轮，第一层（dis = 0）：**

- 访问节点 1：
  - 邻居节点为 2 和 4。2 和 4 为黑色，无需操作。

```
                while queNode = [1] != empty do
                    w:=queNode.DEQUE(); // 队头 1 出队，赋给 w
                    foreach neighbor [2,4] of w = 1 do
                        if 2.color==WHITE then // 2.color=BLACK False
                        if 4.color==WHITE then // 4.color=BLACK False
                    <processing of node w>;
                    1.color:=BLACK; // 1 染黑
```

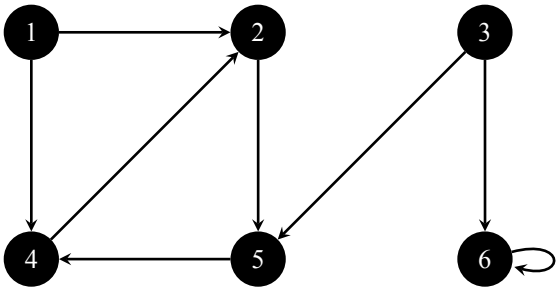

Figure 16: 图 9-1-6

节点编号	Parent	dis 值
1	-	0
2	4	3
3	-	0
4	5	2
5	3	1
6	3	1

Figure 17: 图 9-1-6 的 BFS 结果

8. **结束：**

- 队列为空，while 循环结束。
- 外层节点全黑，BFS 结束。

```
                while queNode = [] != empty do    // queNode = [] False
```

```
    foreach v in [3,1,2,4,5,6] do //全部为黑色
        if v.color == WHITE then
            BFS(v)
```

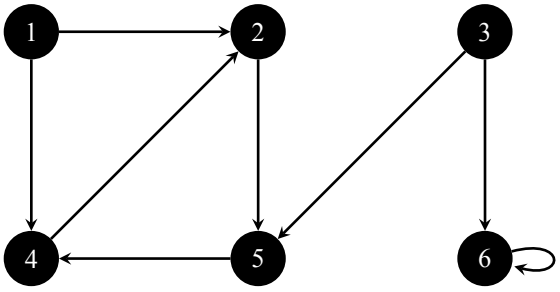

Figure 18: 图 9-1-7

节点编号	Parent	dis 值
1	-	0
2	4	3
3	-	0
4	5	2
5	3	1
6	3	1

Figure 19: 图 9-1-7 的 BFS 结果

2.2.2 对图 9-2 执行 BFS 并确定每个节点的 parent 和 dis 值

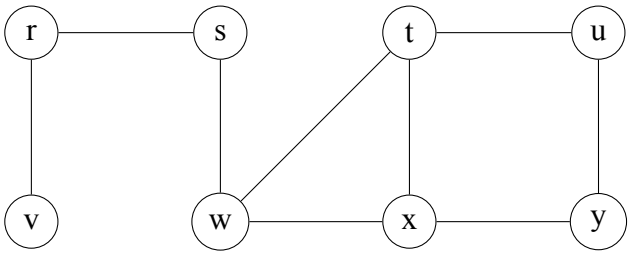

Figure 20: 图 9-2

1. **初始化：**

   - 所有节点染白，parent = -，dis = ∞
   - u 入队，dis = 0

```
1        BFS-WARRER(G)
2        foreach node v in [r,x,t,u,v,w,x,y] do
3            v.color:=WHITE;
4            v.parent=NULL
5            v.dis:=+ϖ;
6        foreach node v in [u,r,x,t,v,w,x,y] do
7            if v.color == WHITE then
8                BFS(v)
9                BFS(u)
10                   Initialize an empty queue queNode;
11                   u.color:=GRAY; // u 染灰
12                   u.dis:=0; // u dis=0
13                   queNode.ENQUEUE(u); // u 入队
```

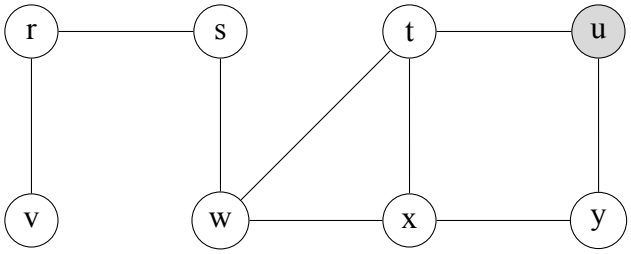

Figure 21: 图 9-2-0

节点	Parent	dis 值
r	-	∞
s	-	∞
t	-	∞
u	-	0
v	-	∞
w	-	∞
x	-	∞
y	-	∞

Figure 22: 图 9-2-0 的 BFS 结果

2. **第一层（dis = 0）：**

- 访问节点 u，发现邻居 t 和 y
- 设置 t 和 y 的 dis 值为 1，parent 为 u

```
while queNode = [u] != empty do
    w1:=queNode.DEQUEUE(); // 队头 u 出队，赋给 w1
    foreach neighbor [t,y] of w1 = u do
        if t.color==WHITE then
            t.color:=GRAY; // t 染灰
            t.parent:=u; // t parent=u
            t.dis:=u.dis+1; // t dis=1
            queNode.ENQUEUE(t); // t 入队
        if y.color==WHITE then
            y.color:=GRAY; // y 染灰
            y.parent:=u; // y parent=u
            y.dis:=u.dis+1; // y dis=1
            queNode.ENQUEUE(y); // y 入队
    <processing of node u>;
    u.color:=BLACK; // u 染黑
```

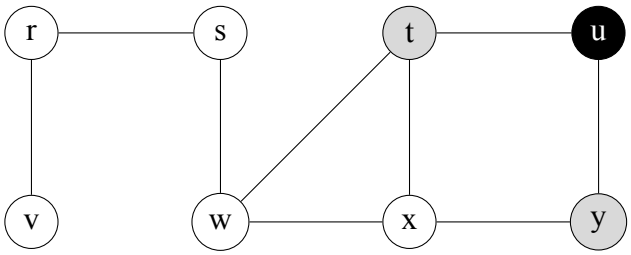

Figure 23: 图 9-2-1

节点	Parent	dis 值
r	-	∞
s	-	∞
t	u	1
u	-	0
v	-	∞
w	-	∞
x	-	∞
y	u	1

Figure 24: 图 9-2-1 的 BFS 结果

3. **第二层（dis = 1）：**

- 访问 t：发现 w 和 x，设置它们的 dis 值为 2，parent 为 t

```
while queNode = [t,y] != empty do
    w1:=queNode.DEQUE(); // 队头 t 出队，赋给 w1
    foreach neighbor [u,w,x] of w1 = t do
        if u.color==WHITE then// u.color=BLACK False
        if w.color==WHITE then
            w.color:=GRAY; // w 染灰
            w.parent:=t; // w parent=t
            w.dis:=t.dis+1; // w dis=2
            queNode.ENQUEUE(w); // w 入队
        if x.color==WHITE then
            x.color:=GRAY; // x 染灰
            x.parent:=t; // x parent=t
            x.dis:=t.dis+1; // x dis=2
            queNode.ENQUEUE(x); // x 入队
    <processing of node t>;
    t.color:=BLACK; // t 染黑
```

- 访问 y：邻居均已访问或在队列中

```
while queNode = [y,w,x] != empty do
    w1:=queNode.DEQUE(); // 队头 y 出队，赋给 w1
    foreach neighbor [u,x] of w1 = y do
        if u.color==WHITE then // u.color=BLACK False
        if x.color==WHITE then // x.color=GRAY False
    <processing of node y>;
    y.color:=BLACK; // y 染黑
```

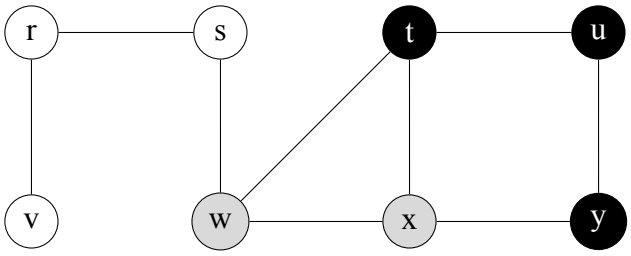

Figure 25: 图 9-2-1

节点	Parent	dis 值
r	-	∞
s	-	∞
t	u	1
u	-	0
v	-	∞
w	t	2
x	t	2
y	u	1

Figure 26: 图 9-2-1 的 BFS 结果

4. **第三层（dis = 2）：**

- 访问 w：发现 s，设置其 dis 值为 3，parent 为 w

```
while queNode = [w,x] != empty do
    w1:=queNode.DEQUE(); // 队头 w 出队，赋给 w1
    foreach neighbor [s,t,x] of w1 = w do
        if s.color==WHITE then
            s.color:=GRAY; // s 染灰
            s.parent:=t; // s parent=w
            s.dis:=t.dis+1; // s dis=2
            queNode.ENQUEUE(s); // s 入队
        if t.color==WHITE then // t.color=BLACK False
        if x.color==WHITE then// x.color=GRAY False
    <processing of node w>;
    w.color:=BLACK; // w 染黑
```

- 访问 x：邻居均已访问

```
while queNode = [x,s] != empty do
    w1:=queNode.DEQUE(); // 队头 x 出队，赋给 w1
    foreach neighbor [t,w,y] of w1 = x do
        if t.color==WHITE then // t.color=BLACK False
        if w.color==WHITE then // w.color=BLACK False
        if y.color==WHITE then // y.color=BLACK False
    <processing of node x>;
    x.color:=BLACK; // x 染黑
```

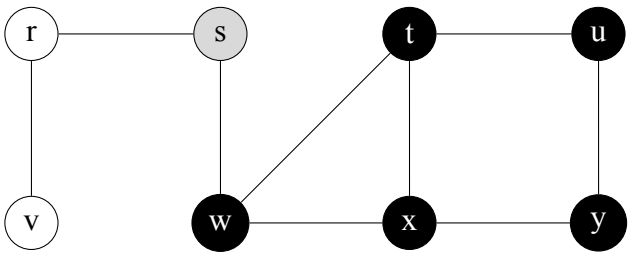

Figure 27: 图 9-2-3

节点	Parent	dis 值
r	-	∞
s	w	3
t	u	1
u	-	0
v	-	∞
w	t	2
x	t	2
y	u	1

Figure 28: 图 9-2-3 的 BFS 结果

5. **第四层（dis = 3）：**

   - 访问 s：发现 r，设置其 dis 值为 4，parent 为 s

```
                while queNode = [s] != empty do
                    w1:=queNode.DEQUE(); // 队头 s 出队，赋给 w1
                    foreach neighbor [r,w] of w1 = s do
                        if r.color==WHITE then
                            r.color:=GRAY; // r 染灰
                            r.parent:=s; // r parent=s
                            r.dis:=s.dis+1; // r dis=4
                            queNode.ENQUEUE(r); // r 入队
                        if w.color==WHITE then // w.color=BLACK False
                    <processing of node s>;
                    s.color:=BLACK; // s 染黑
```

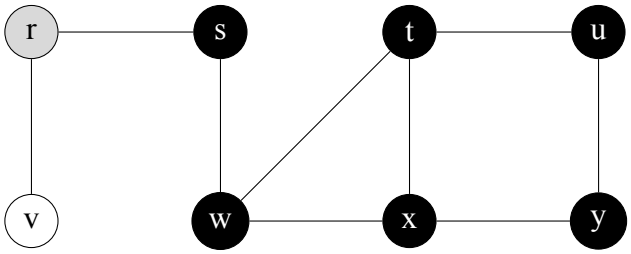

Figure 29: 图 9-2-4

节点	Parent	dis 值
r	s	4
s	w	3
t	u	1
u	-	0
v	r	5
w	t	2
x	t	2
y	u	1

Figure 30: 图 9-2-4 的 BFS 结果

6. **第五层（dis = 4）：**

   - 访问 r：发现 v，设置其 dis 值为 5，parent 为 r

```
                while queNode = [r] != empty do
                    w1:=queNode.DEQUE(); // 队头 r 出队，赋给 w1
                    foreach neighbor [s,v] of w1 = r do
                        if s.color==WHITE then // s.color=BLACK False
                        if v.color==WHITE then
                            v.color:=GRAY; // v 染灰
                            v.parent:=r; // v parent=r
                            v.dis:=r.dis+1; // v dis=5
                            queNode.ENQUEUE(v); // v 入队
                    <processing of node r>;
                    r.color:=BLACK; // r 染黑
```

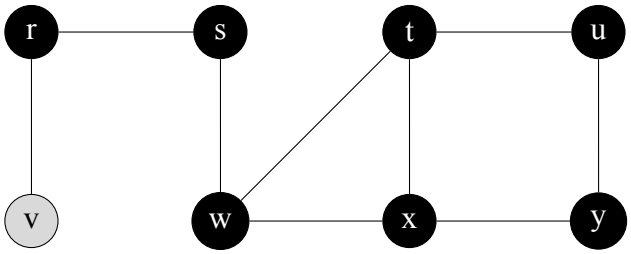

Figure 31: 图 9-2-5

节点	Parent	dis 值
r	s	4
s	w	3
t	u	1
u	-	0
v	r	5
w	t	2
x	t	2
y	u	1

Figure 32: 图 9-2-5 的 BFS 结果

7. **第六层（dis = 5）：**

   - 访问 v：邻居均已访问

```
                while queNode = [v] != empty do
                    w1:=queNode.DEQUE(); // 队头 v 出队，赋给 w1
                    foreach neighbor [r] of w1 = v do
                        if r.color==WHITE then // r.color=BLACK False
                    <processing of node v>;
                    v.color:=BLACK; // v 染黑
```

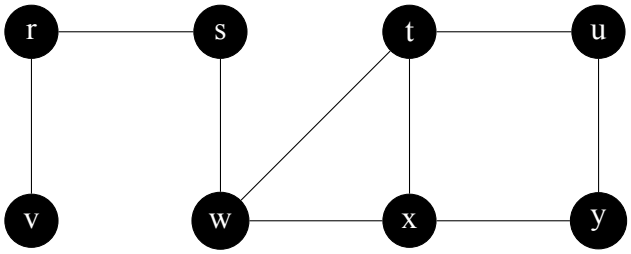

Figure 33: 图 9-2-6

节点	Parent	dis 值
r	s	4
s	w	3
t	u	1
u	-	0
v	r	5
w	t	2
x	t	2
y	u	1

Figure 34: 图 9-2-6 的 BFS 结果

8. **结束**

   - 队列为空，while 循环结束。
   - 外层节点全黑，BFS 结束。

```
                while queNode = [] != empty do    // queNode = [] False
```

```
    foreach v in [u,r,s,t,v,w,x,y] do //全部为黑色
        if v.color == WHITE then
            BFS(v)
```

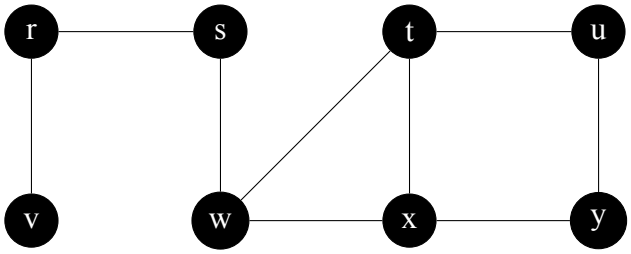

Figure 35: 图 9-2-7

节点	Parent	dis 值
r	s	4
s	w	3
t	u	1
u	-	0
v	r	5
w	t	2
x	t	2
y	u	1

Figure 36: 图 9-2-7 的 BFS 结果