第3章 圈和遍历

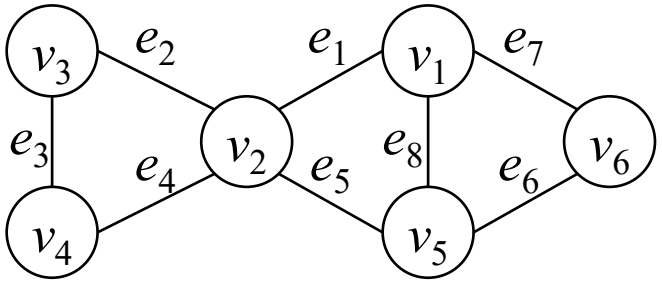
程龚
南京大学 计算机学院
gcheng@nju.edu.cn
http://ws.nju.edu.cn/~gcheng

本章内容

- 第3.1节 圈和树
- 第3.2节 二分图
- 第3.3节 欧拉图
  - 第3.3.1节 理论
  - **第3.3.2节 算法**
- 第3.4节 哈密尔顿图

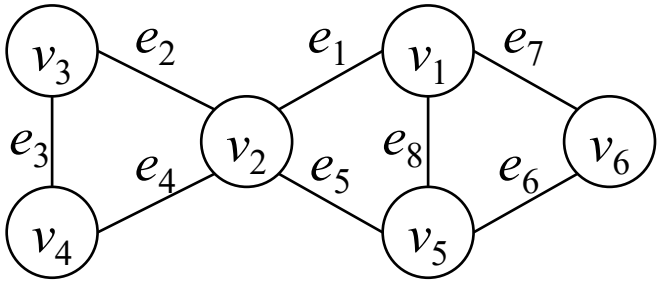
如何找出图中的一条欧拉迹?

- 弗勒里算法
- 希尔霍尔策算法



如何找出图中的一条欧拉迹?

- **弗勒里算法**
- 希尔霍尔策算法



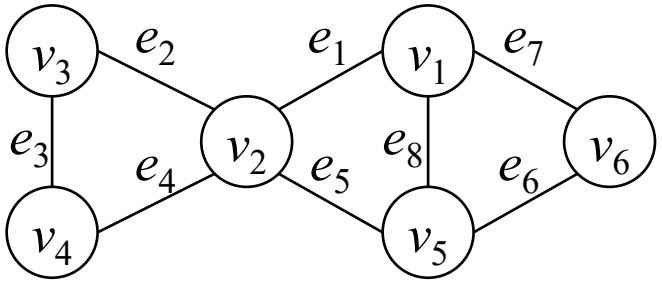
弗勒里算法

- 基本思路：逐步构造欧拉迹，每步将当前迹延长一条边，
  并尽可能避免选择剩余图的割边，从而避免剩余的边不能被全部增加到迹中。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃v ∈ V, d(v)是奇数 then
    u ← v;
else
    u ← V 中任意一个非孤立点;
输出 (u);
while u 非孤立点 do
    if u 关联一条非割边 e′ then
        e ← e′;
    else
        e ← u 关联的任意一条边;
    输出 (e);
    u ← e 的另一个端点;
    输出 (u);
    G ← G − e;
```



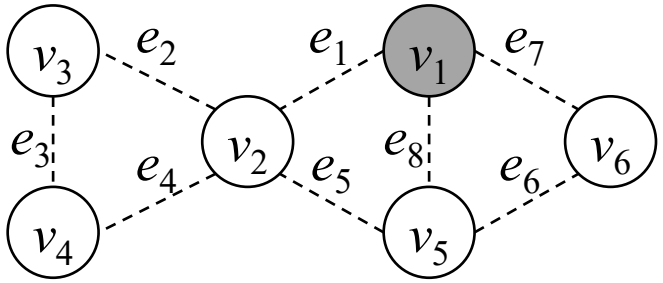
弗勒里算法

- 从顶点u出发：
  - 若顶点集V中有顶点的度为奇数，则以该顶点为出发点；
  - 否则，以任意非孤立点为出发点。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2      u ← v;
3  else
4      u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7      if u 关联一条非割边 e′ then
8          e ← e′;
9      else
10         e ← u 关联的任意一条边;
11     输出 (e);
12     u ← e 的另一个端点;
13     输出 (u);
14     G ← G − e;
```



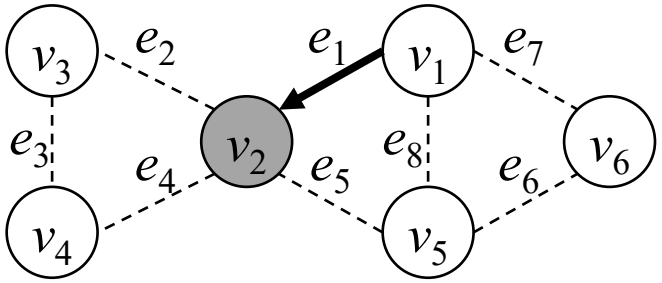
弗勒里算法

- 每轮while循环从u经过其关联的边e到达e的另一个端点，作为下轮while循环的u，并将e从G中删除，直至到达孤立点：

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e' then
8  |  |  e ← e';
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



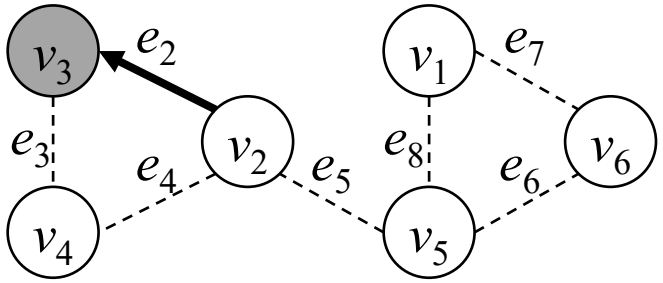
弗勒里算法

- 每轮while循环从u经过其关联的边e到达e的另一个端点，作为下轮while循环的u，并将e从G中删除，直至到达孤立点：
  - e的选取原则是优先选择非割边。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



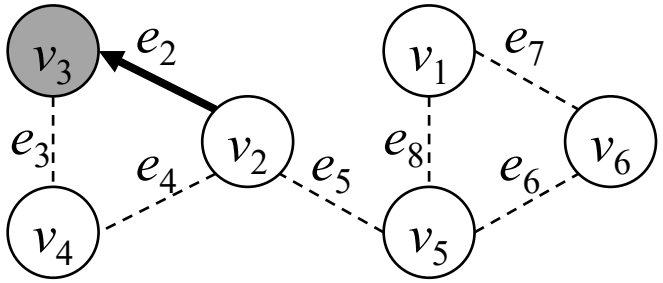
弗勒里算法

- 算法运行过程中，按序输出经过的顶点和边，组成一条欧拉迹。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e' then
8  |  |  e ← e';
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



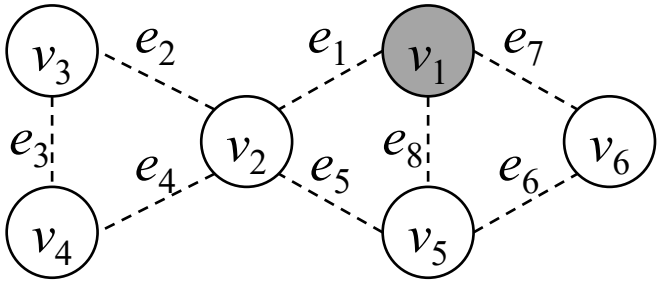
弗勒里算法

- 例如：$d(v_1)$是奇数
  - $u \leftarrow v_1$，输出v_1

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e' then
8  |  |  e ← e';
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1

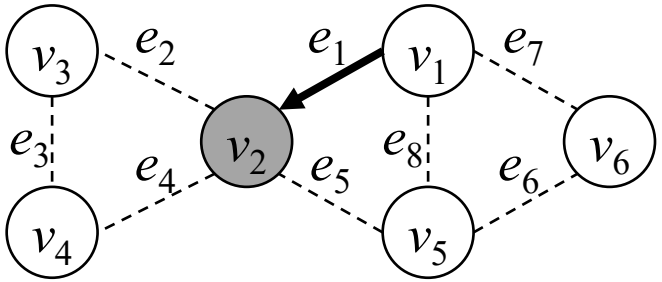
弗勒里算法

- v_1非孤立点
  - $e \leftarrow v_1$关联一条非割边e_1，输出e_1
  - $u \leftarrow e_1$的另一个端点v_2，输出v_2
  - 删除e_1

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1, v_2

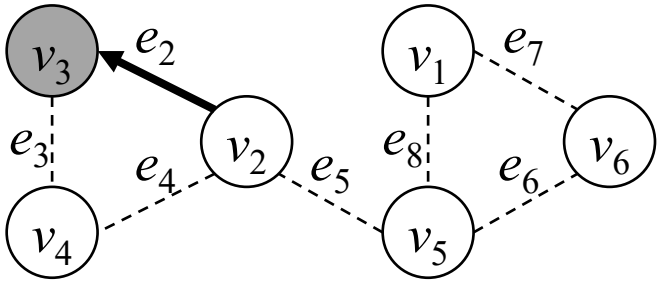
弗勒里算法

- v_2非孤立点
  - $e \leftarrow v_2$关联一条非割边e_2，输出e_2
  - $u \leftarrow e_2$的另一个端点v_3，输出v_3
  - 删除e_2

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1, v_2, v_3

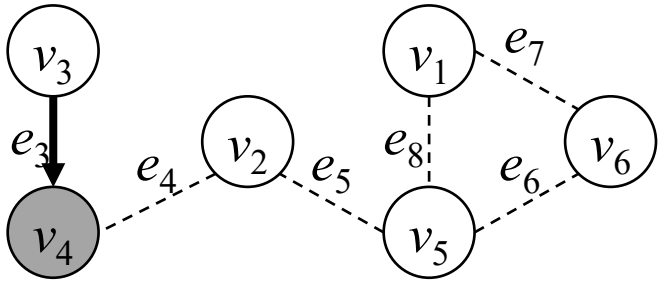
弗勒里算法

- v_3非孤立点
  - $e \leftarrow v_3$关联一条割边e_3，输出e_3
  - $u \leftarrow e_3$的另一个端点v_4，输出v_4
  - 删除e_3

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1, v_2, v_3, v_4

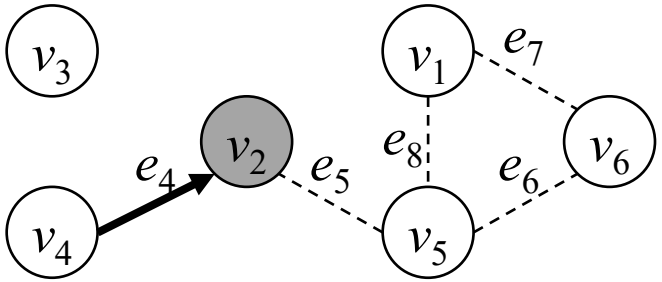
弗勒里算法

- v_4非孤立点
  - $e \leftarrow v_4$关联一条割边e_4，输出e_4
  - $u \leftarrow e_4$的另一个端点v_2，输出v_2
  - 删除e_4

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1, v_2, v_3, v_4, v_2

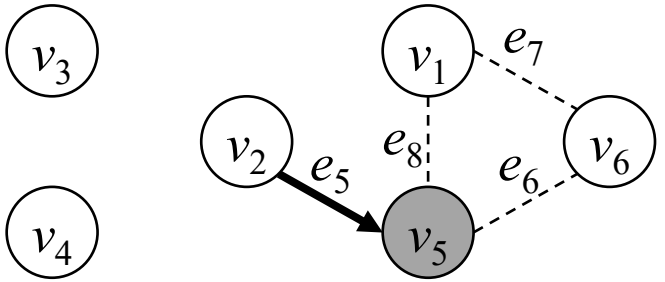
弗勒里算法

- v_2非孤立点
  - $e \leftarrow v_2$关联一条割边e_5，输出e_5
  - $u \leftarrow e_5$的另一个端点v_5，输出v_5
  - 删除e_5

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：$v_1, v_2, v_3, v_4, v_2, v_5$

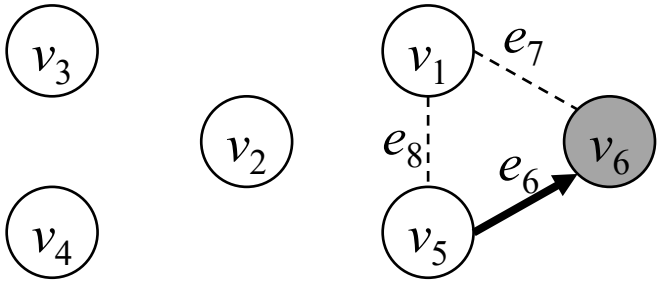
弗勒里算法

- v_5非孤立点
  - $e \leftarrow v_5$关联一条非割边e_6，输出e_6
  - $u \leftarrow e_6$的另一个端点v_6，输出v_6
  - 删除e_6

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：$v_1, v_2, v_3, v_4, v_2, v_5, v_6$

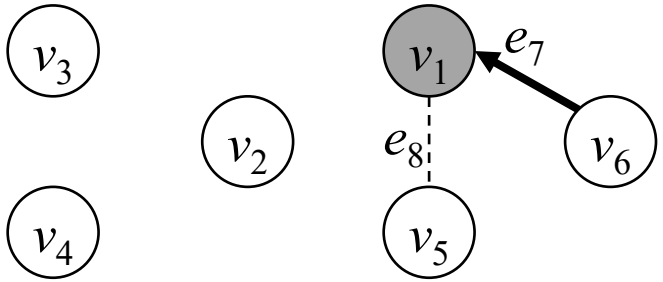
弗勒里算法

- v_6非孤立点
  - $e \leftarrow v_6$关联一条割边e_7，输出e_7
  - $u \leftarrow e_7$的另一个端点v_1，输出v_1
  - 删除e_7

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：$v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1$

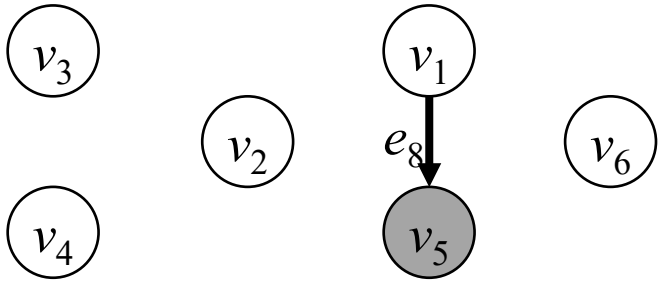
弗勒里算法

- v_1非孤立点
  - $e \leftarrow v_1$关联一条割边e_8，输出e_8
  - $u \leftarrow e_8$的另一个端点v_5，输出v_5
  - 删除e_8

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：$v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1, v_5$

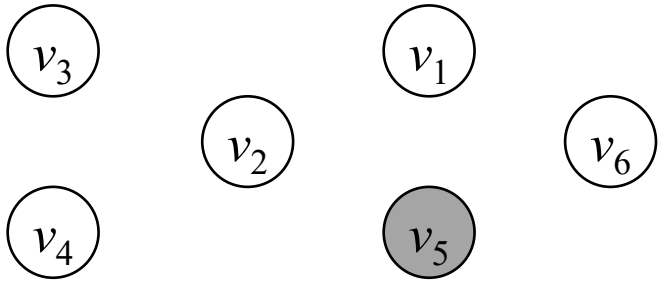
弗勒里算法

- v_5是孤立点，算法运行结束

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出: $v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1, v_5$

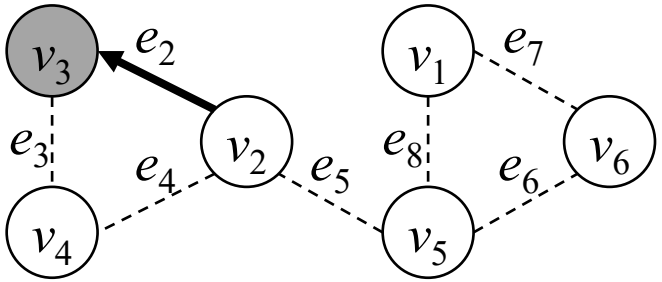
弗勒里算法的正确性

- 算法每轮while循环开始前：
  - 或者图G为空图，则原图的欧拉迹已找到；
  - 或者G含以顶点u为起点的欧拉迹，则该欧拉迹和已输出的序列可拼接得到原图的欧拉迹。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```



输出：v_1, v_2, v_3

弗勒里算法的正确性

- 算法每轮while循环开始前：
  - 或者图G为空图，则原图的欧拉迹已找到；
  - 或者G含以顶点u为起点的欧拉迹，则该欧拉迹和已输出的序列可拼接得到原图的欧拉迹。

思考题3.33 图G含欧拉迹的充要条件是什么？
是空图 或 边集的边导出子图非空连通，且有至多2个顶点的度为奇数。

- 只需证明：
  - 推论3.2 弗勒里算法每轮while循环条件判定前，
    图G或者为空图，或者边集的边导出子图连通且含顶点u。
  - 定理3.6 弗勒里算法每轮while循环条件判定前，
    图G或者没有顶点的度为奇数，或者恰有2个顶点（包括顶点u）的度为奇数。

定理3.6

- 弗勒里算法每轮while循环条件判定前，
  图G或者没有顶点的度为奇数，或者恰有2个顶点（包括顶点u）的度为奇数。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e' then
8  |  |  e ← e';
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```

定理3.6

- 弗勒里算法每轮while循环条件判定前，
  图G或者没有顶点的度为奇数，或者恰有2个顶点（包括顶点u）的度为奇数。
  - 第1轮while循环条件判定前成立。

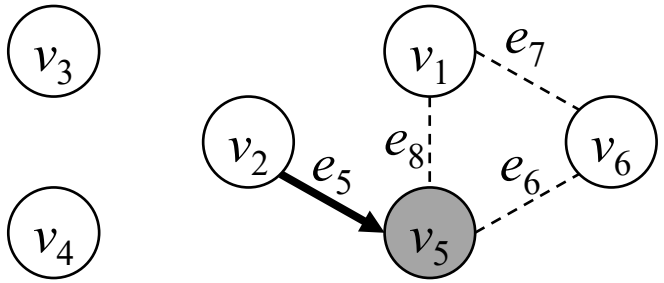
算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

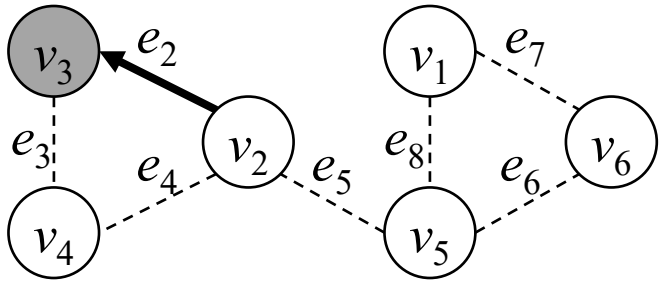
```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```

定理3.6

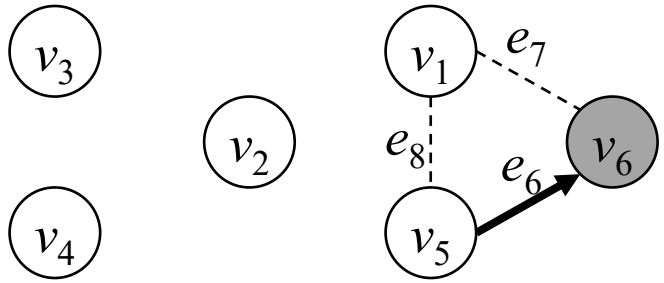
- 弗勒里算法每轮while循环条件判定前，
  图G或者没有顶点的度为奇数，或者恰有2个顶点（包括顶点u）的度为奇数。
  - 第1轮while循环条件判定前成立。
  - 接下来，若本轮while循环条件判定前成立，将下轮while循环条件判定前的顶点u记作u'，
    则本轮while循环从图G中删除边e后，u和u'的度的奇偶性都发生改变：
    - 若u的度由奇变偶，u'的度由奇变偶，则下轮while循环条件判定前，G没有顶点的度为奇数；
    - 若u的度由奇变偶，u'的度由偶变奇，则下轮while循环条件判定前，G恰有2个顶点（包括u'）的度为奇数；
    - 若u的度由偶变奇，则u'的度只能也由偶变奇，下轮while循环条件判定前，G恰有2个顶点（包括u'）的度为奇数。



u的度由奇变偶，u'的度由奇变偶



u的度由奇变偶，u'的度由偶变奇



u的度由偶变奇，u'的度由偶变奇

推论3.2

- 弗勒里算法每轮while循环条件判定前，
  图G或者为空图，或者边集的边导出子图连通且含顶点u。

算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```

推论3.2

- 弗勒里算法每轮while循环条件判定前，
  图G或者为空图，或者边集的边导出子图连通且含顶点u。
  - 第1轮while循环条件判定前成立。

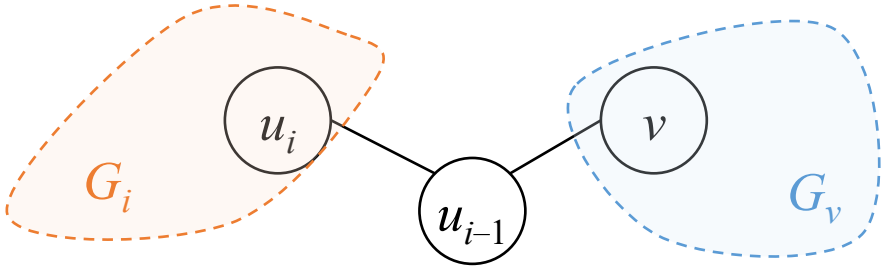
算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2      u ← v;
3  else
4      u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7      if u 关联一条非割边 e′ then
8          e ← e′;
9      else
10         e ← u 关联的任意一条边;
11     输出 (e);
12     u ← e 的另一个端点;
13     输出 (u);
14     G ← G − e;
```

推论3.2

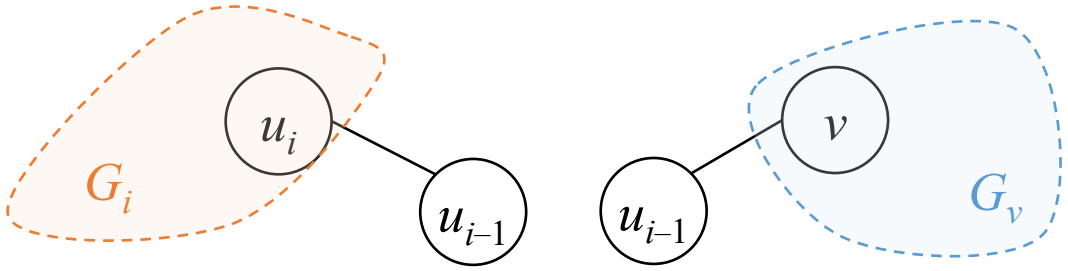
- 弗勒里算法每轮while循环条件判定前，
  图G或者为空图，或者边集的边导出子图连通且含顶点u。
  - 第1轮while循环条件判定前成立。
  - 接下来，采用反证法，对于非空图G，
    先假设G的边集的边导出子图不连通首次发生在第i轮while循环条件判定前（$i > 1$），
    将第$i-1$轮和第i轮while循环条件判定前的顶点u分别记作u_{i-1}和u_i，
    则第$i-1$轮while循环条件判定前，u_{i-1}关联至少2条边(u_{i-1}, u_i)和(u_{i-1}, v)，
    其中(u_{i-1}, u_i)是割边，因此，(u_{i-1}, v)也是割边，
    将图$G - u_{i-1}$中含顶点u_i和含顶点v的连通分支分别记作G_i和G_v。



推论3.2

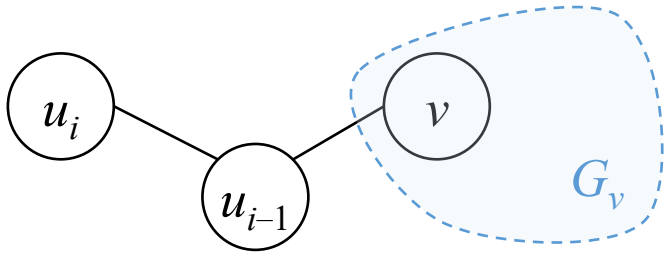
- 弗勒里算法每轮while循环条件判定前，
  图G或者为空图，或者边集的边导出子图连通且含顶点u。
  - 第1轮while循环条件判定前成立。
  - 接下来，采用反证法，对于非空图G，
    先假设G的边集的边导出子图不连通首次发生在第i轮while循环条件判定前（$i > 1$），
    将第$i-1$轮和第i轮while循环条件判定前的顶点u分别记作u_{i-1}和u_i，
    则第$i-1$轮while循环条件判定前，u_{i-1}关联至少2条边(u_{i-1}, u_i)和(u_{i-1}, v)，
    其中(u_{i-1}, u_i)是割边，因此，(u_{i-1}, v)也是割边，
    将图$G - u_{i-1}$中含顶点u_i和含顶点v的连通分支分别记作G_i和G_v。
    在G_i和边(u_{i-1}, u_i)组成的子图中，u_{i-1}的度为1，因此，G_i含至少1个度为奇数的顶点；
    同理，G_v含至少1个度为奇数的顶点。
    因此，第$i-1$轮while循环条件判定前，无论u_{i-1}的度是否为奇数，都与定理3.6矛盾。

定理3.6 弗勒里算法每轮while循环条件判定前，图G或者没有顶点的度为奇数，或者恰有2个顶点（包括顶点u）的度为奇数。



推论3.2

- 弗勒里算法每轮while循环条件判定前，
  图G或者为空图，或者边集的边导出子图连通且含顶点u。
  - 再假设图G的边集的边导出子图连通但不含顶点u首次发生在第i轮while循环条件判定前（$i>1$），
    将第$i-1$轮和第i轮while循环条件判定前的u分别记作u_{i-1}和u_i，
    则第i轮while循环条件判定前，u_i为孤立点，
    第$i-1$轮while循环条件判定前，u_{i-1}关联至少2条边(u_{i-1}, u_i)和(u_{i-1}, v)，
    其中(u_{i-1}, u_i)是割边，后续证明同上。



弗勒里算法

- 时间复杂度：$O(n + m(n + m))$
  - 每轮while循环：$O(n + m)$
  - 循环的轮数：$O(m)$

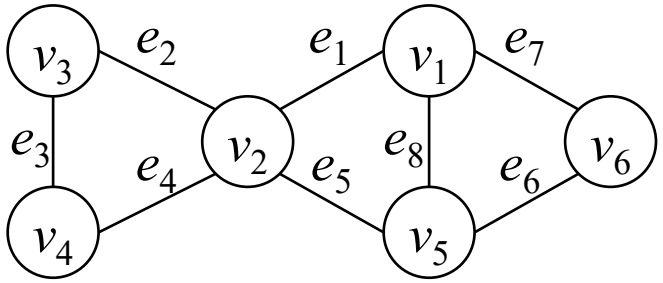
算法 3.2: 弗勒里算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃v ∈ V, d(v)是奇数 then
2  |  u ← v;
3  else
4  |  u ← V 中任意一个非孤立点;
5  输出 (u);
6  while u 非孤立点 do
7  |  if u 关联一条非割边 e′ then
8  |  |  e ← e′;
9  |  else
10 |  |  e ← u 关联的任意一条边;
11 |  输出 (e);
12 |  u ← e 的另一个端点;
13 |  输出 (u);
14 |  G ← G − e;
```

如何找出图中的一条欧拉迹?

- 弗勒里算法
- **希尔霍尔策算法**



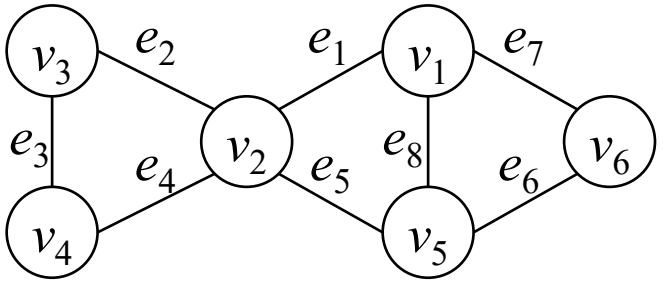
希尔霍尔策算法

- 基本思路：逐步构造欧拉迹，每步将当前迹与一条闭迹拼接。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)



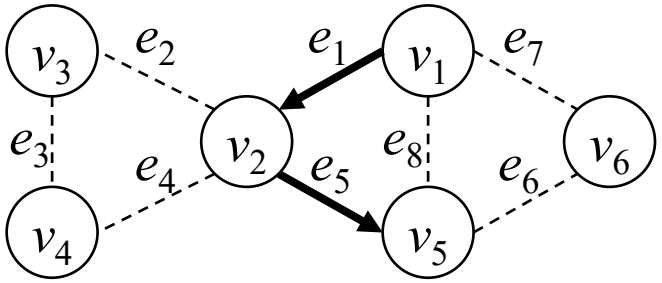
希尔霍尔策算法

- 首先找一条迹T:
  - 若顶点集V中有两个顶点的度为奇数，则任找一条以它们为起点和终点的迹作为T;
  - 否则，任找一条闭迹作为T。
- 将T经过的边从G中删除。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



T: v_1, v_2, v_5

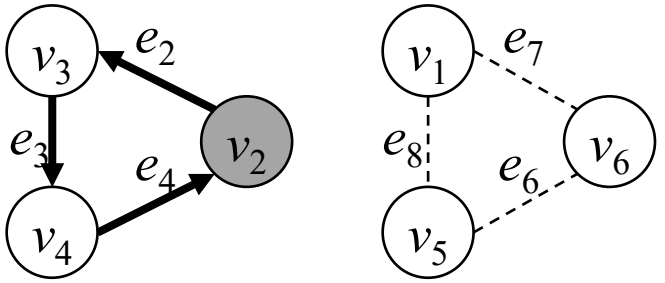
希尔霍尔策算法

- 每轮while循环从T经过的顶点中选择w，满足w在当前G中非孤立点，即w关联至少一条T尚未经过的边，直至找不到满足条件的w;
- 在当前G中任找一条以w为起点和终点的闭迹T'，将T和T'在w处拼接形成一条更长的迹作为T，将T'经过的边从G中删除。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



T: $v_1, v_2, v_3, v_4, v_2, v_5$

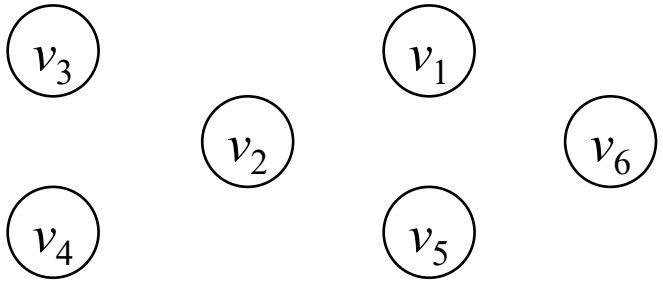
希尔霍尔策算法

- 算法运行结束时，输出欧拉迹T。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



T: $v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1, v_5$

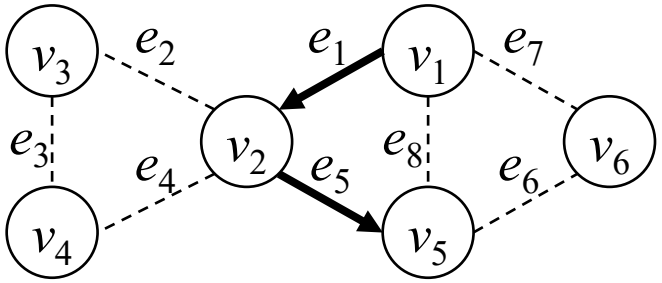
希尔霍尔策算法

- 例如：$d(v_1)$和$d(v_5)$是奇数
  - $T \leftarrow v_1, v_2, v_5$
  - 删除e_1、e_5

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
2  |  T ← G 中任意一条 u-v 迹;
3  else
4  |  T ← G 中任意一条闭迹;
5  G ← G − T经过的边的集合;
6  while ∃w ∈ V, w 非孤立点且 T 经过 w do
7  |  T′ ← G 中任意一条 w-w 闭迹;
8  |  T ← T 和 T′ 拼接;
9  |  G ← G − T′经过的边的集合;
10 输出 (T)
```



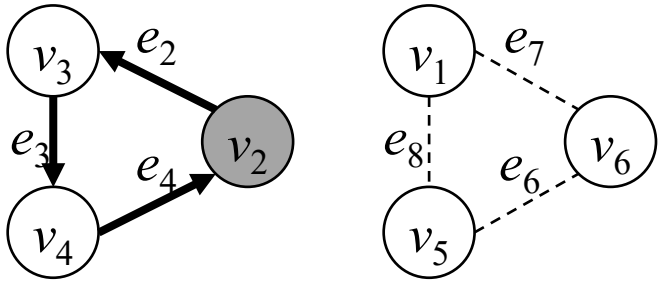
希尔霍尔策算法

- $w \leftarrow v_2$
  - $T' \leftarrow v_2, v_3, v_4, v_2$
  - $T \leftarrow v_1, v_2, v_3, v_4, v_2, v_5$
  - 删除e_2、e_3、e_4

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



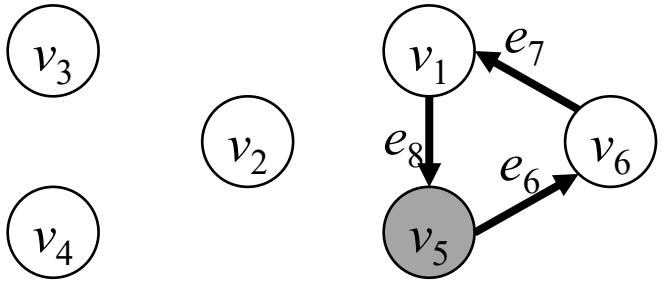
希尔霍尔策算法

- $w \leftarrow v_5$
  - $T' \leftarrow v_5, v_6, v_1, v_5$
  - $T \leftarrow v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1, v_5$
  - 删除e_6、e_7、e_8

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
2  |  T ← G 中任意一条 u-v 迹;
3  else
4  |  T ← G 中任意一条闭迹;
5  G ← G − T经过的边的集合;
6  while ∃w ∈ V, w 非孤立点且 T 经过 w do
7  |  T′ ← G 中任意一条 w-w 闭迹;
8  |  T ← T 和 T′ 拼接;
9  |  G ← G − T′经过的边的集合;
10 输出 (T)
```



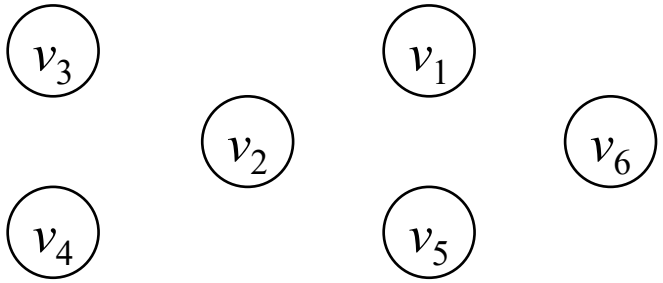
希尔霍尔策算法

- w不存在
  - 输出$T = v_1, v_2, v_3, v_4, v_2, v_5, v_6, v_1, v_5$

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
1  if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
2  |  T ← G 中任意一条 u-v 迹;
3  else
4  |  T ← G 中任意一条闭迹;
5  G ← G − T经过的边的集合;
6  while ∃w ∈ V, w 非孤立点且 T 经过 w do
7  |  T′ ← G 中任意一条 w-w 闭迹;
8  |  T ← T 和 T′ 拼接;
9  |  G ← G − T′经过的边的集合;
10 输出 (T)
```



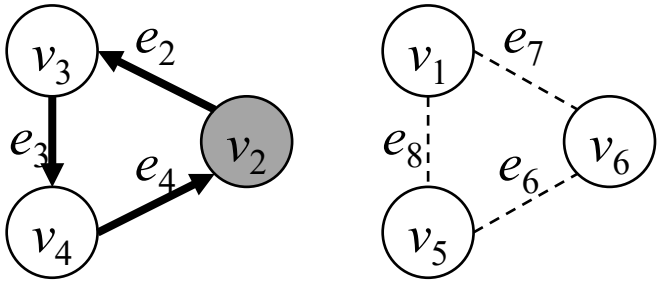
希尔霍尔策算法的正确性

- 算法每轮while循环开始前：
  - 或者图G为空图，则原图的欧拉迹已找到；
  - 或者G的每个非平凡连通分支都含（该分支自身的）欧拉回路，且都和迹T有公共顶点，则这些欧拉回路和T可拼接得到原图的欧拉迹。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



希尔霍尔策算法的正确性

- 算法每轮while循环开始前：
  - 或者图G为空图，则原图的欧拉迹已找到；
  - 或者G的每个非平凡连通分支都含（该分支自身的）欧拉回路，且都和迹T有公共顶点，则这些欧拉回路和T可拼接得到原图的欧拉迹。

思考题3.33 图G含欧拉回路的充要条件是什么？
边集的边导出子图非空连通，且没有顶点的度为奇数。

- 只需证明：
  - 定理3.7 希尔霍尔策算法每轮while循环条件判定前，图G没有顶点的度为奇数。
  - 定理3.8 希尔霍尔策算法每轮while循环条件判定前，图G或者为空图，或者边集的边导出子图的每个连通分支都和迹T有公共顶点。

定理3.7

- 希尔霍尔策算法每轮while循环条件判定前，图G没有顶点的度为奇数。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)

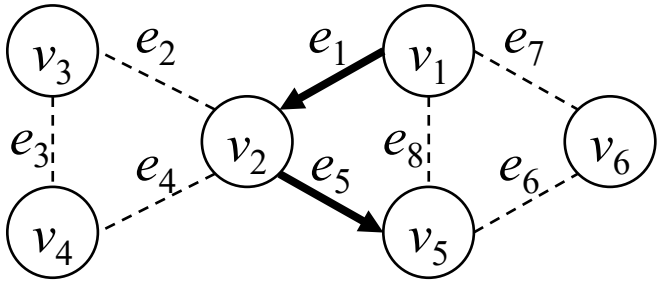
定理3.7

- 希尔霍尔策算法每轮while循环条件判定前，图G没有顶点的度为奇数。
  - 第1轮while循环条件判定前成立。

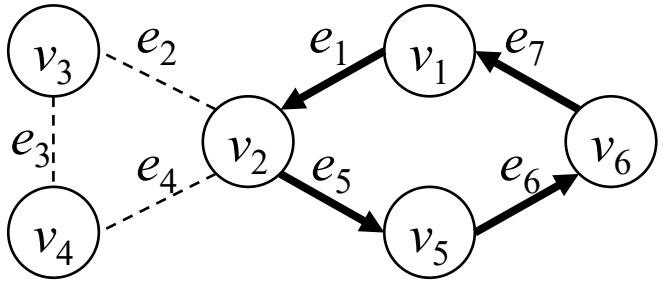
算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)



删除一条u-v迹经过的边的集合



删除一条闭迹经过的边的集合

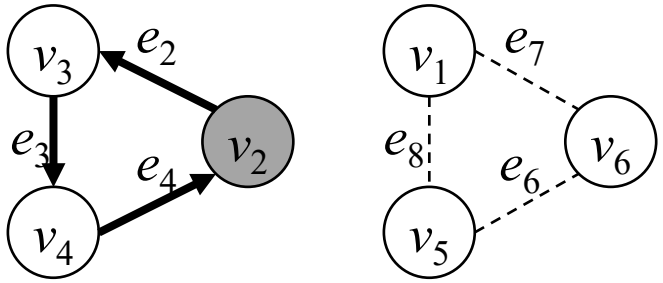
定理3.7

- 希尔霍尔策算法每轮while循环条件判定前，图G没有顶点的度为奇数。
  - 第1轮while循环条件判定前成立。
  - 接下来，若本轮while循环条件判定前成立，则本轮while循环从图G中删除一条闭迹经过的所有边后，所有顶点的奇偶性不变，因此，下轮while循环条件判定前仍成立。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)



定理3.8

- 希尔霍尔策算法每轮while循环条件判定前，图G或者为空图，或者边集的边导出子图的每个连通分支都和迹T有公共顶点。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)

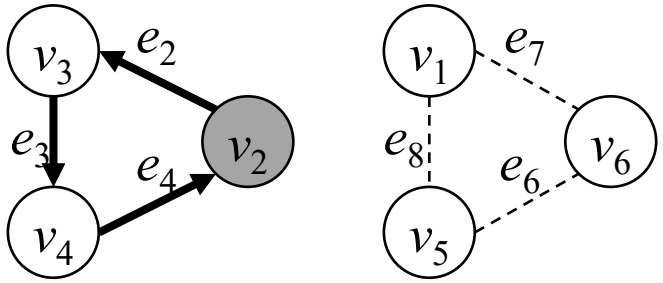
定理3.8

- 希尔霍尔策算法每轮while循环条件判定前，图G或者为空图，或者边集的边导出子图的每个连通分支都和迹T有公共顶点。
  - 采用反证法：
    假设某轮while循环条件判定前，非空图G的边集的边导出子图的某个连通分支和迹T没有公共顶点，则原图的边集的边导出子图不连通，与原图含欧拉迹矛盾。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```



思考题3.41

- 在希尔霍尔策算法中，要找的迹一定存在吗？如何找出这样的迹？

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 | $T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 | $T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 | $T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 | $T \leftarrow T$ 和 T' 拼接;
9 | $G \leftarrow G - T'$经过的边的集合;
10 输出 (T)

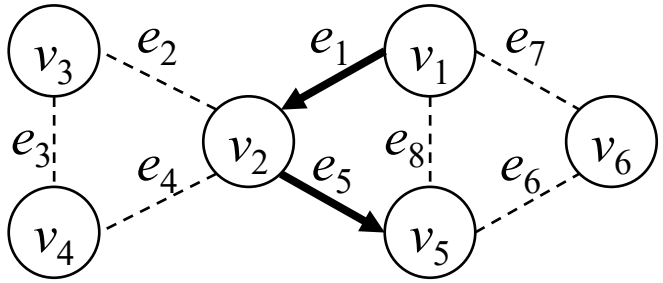
思考题3.41

- 在希尔霍尔策算法中，要找的迹一定存在吗？如何找出这样的迹？
  - 以度为奇数的顶点为起点的极长迹的终点一定是另一个度为奇数的顶点：其它顶点的度为偶数，能进必能出，所以不是终点。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)



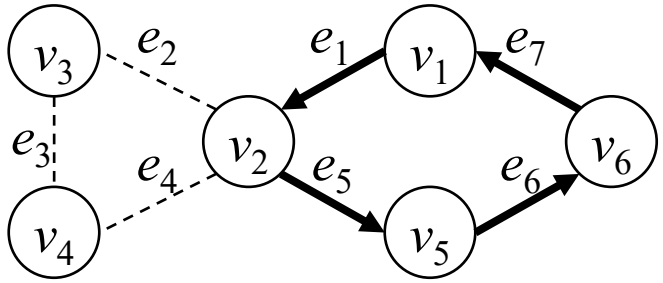
思考题3.41

- 在希尔霍尔策算法中，要找的迹一定存在吗？如何找出这样的迹？
  - 以度为正偶数的顶点为起点的极长迹一定是闭迹：
    其它顶点的度为偶数，能进必能出，所以不是终点。

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

1 **if** $\exists u, v \in V$, $d(u)$是奇数 且 $d(v)$是奇数 **then**
2 　$T \leftarrow G$ 中任意一条 u-v 迹;
3 **else**
4 　$T \leftarrow G$ 中任意一条闭迹;
5 $G \leftarrow G - T$经过的边的集合;
6 **while** $\exists w \in V$, w 非孤立点且 T 经过 w **do**
7 　$T' \leftarrow G$ 中任意一条 w-w 闭迹;
8 　$T \leftarrow T$ 和 T' 拼接;
9 　$G \leftarrow G - T'$经过的边的集合;
10 输出 (T)



希尔霍尔策算法

- 时间复杂度：$O(n+m)$
  - 每轮while循环：$O(1)$，除找闭迹外
    - 采用双向链表等数据结构来存储迹、维护所有满足条件的顶点w、维护每个w关联的迹T尚未经过的边
  - 循环的轮数：$O(m)$

算法 3.3: 希尔霍尔策算法

输入: 含欧拉迹的图 $G = \langle V, E \rangle$

```
if ∃u, v ∈ V, d(u)是奇数 且 d(v)是奇数 then
    T ← G 中任意一条 u-v 迹;
else
    T ← G 中任意一条闭迹;
G ← G − T经过的边的集合;
while ∃w ∈ V, w 非孤立点且 T 经过 w do
    T′ ← G 中任意一条 w-w 闭迹;
    T ← T 和 T′ 拼接;
    G ← G − T′经过的边的集合;
输出 (T)
```

请认真完成课后练习