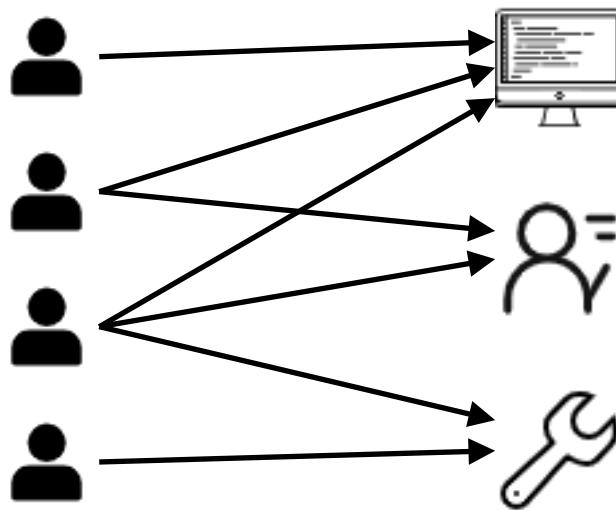


3-5 匹配

程龚

除了任务分配和组队参赛，你还能想到匹配的哪些应用场景？



你能解释这些术语吗？

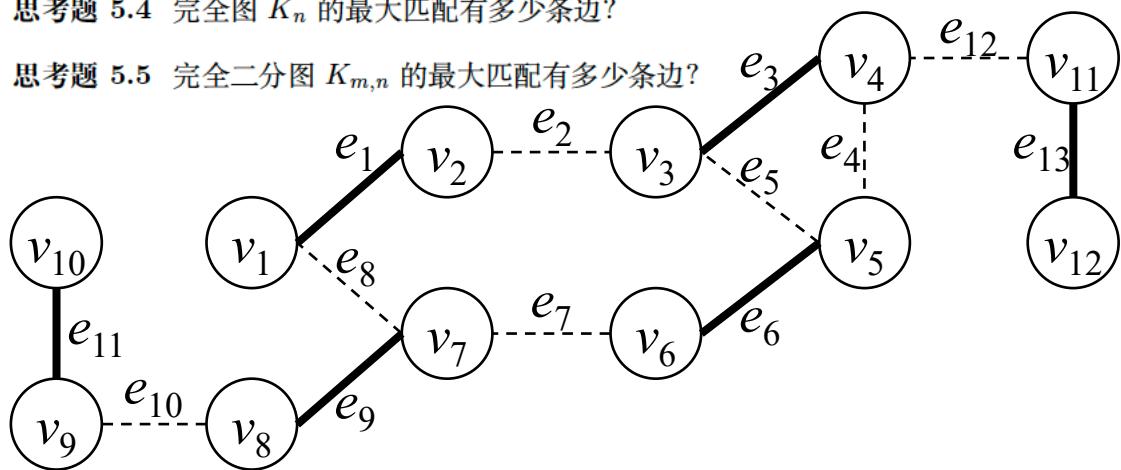
- 匹配
 - 饱和（已匹配）
- 极大匹配、最大匹配

思考题 5.1 每个图都有匹配吗？

思考题 5.3 阶为 n 的图的最大匹配至多有多少条边？

思考题 5.4 完全图 K_n 的最大匹配有多少条边？

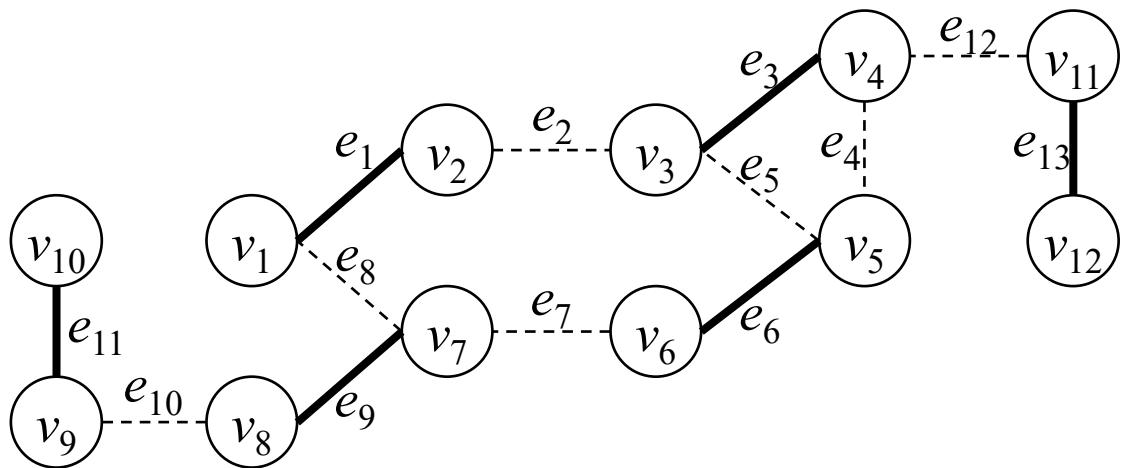
思考题 5.5 完全二分图 $K_{m,n}$ 的最大匹配有多少条边？



匹配的运算

思考题 5.6 图的两个匹配的并集的边导出子图的每个连通分支的结构有什么特征?

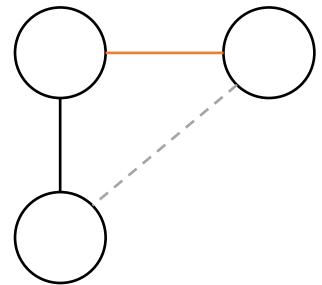
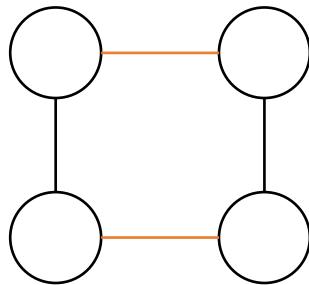
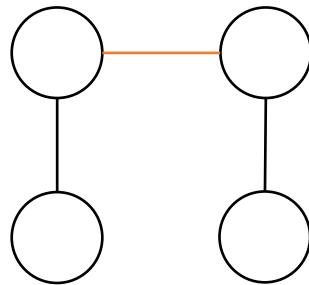
思考题 5.7 图的两个匹配的对称差的边导出子图的每个连通分支的结构有什么特征?



匹配的运算

思考题 5.6 图的两个匹配的并集的边导出子图的每个连通分支的结构有什么特征?

思考题 5.7 图的两个匹配的对称差的边导出子图的每个连通分支的结构有什么特征?

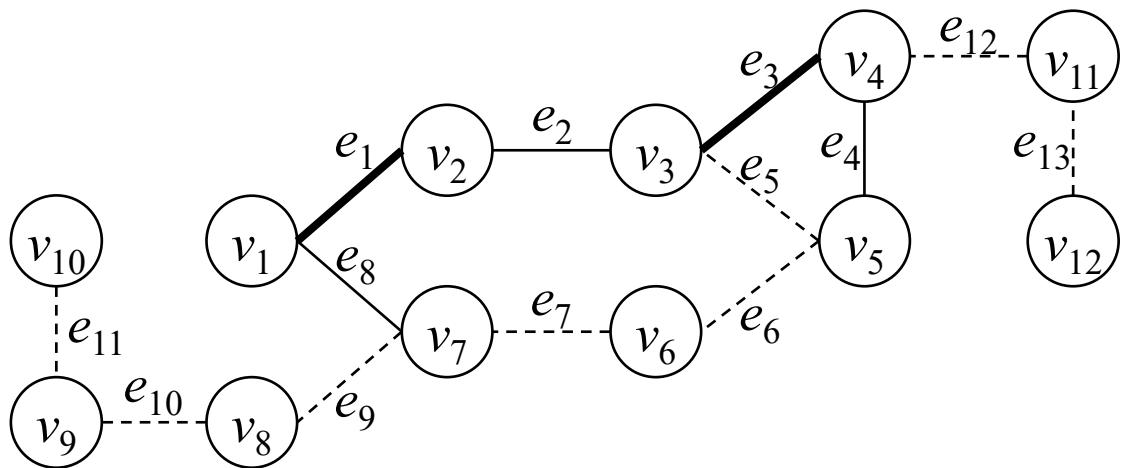


你能解释这些术语吗？

- M 交错路
- M 增广路

思考题 5.8 每个匹配都有交错路和增广路吗？若有，则唯一吗？

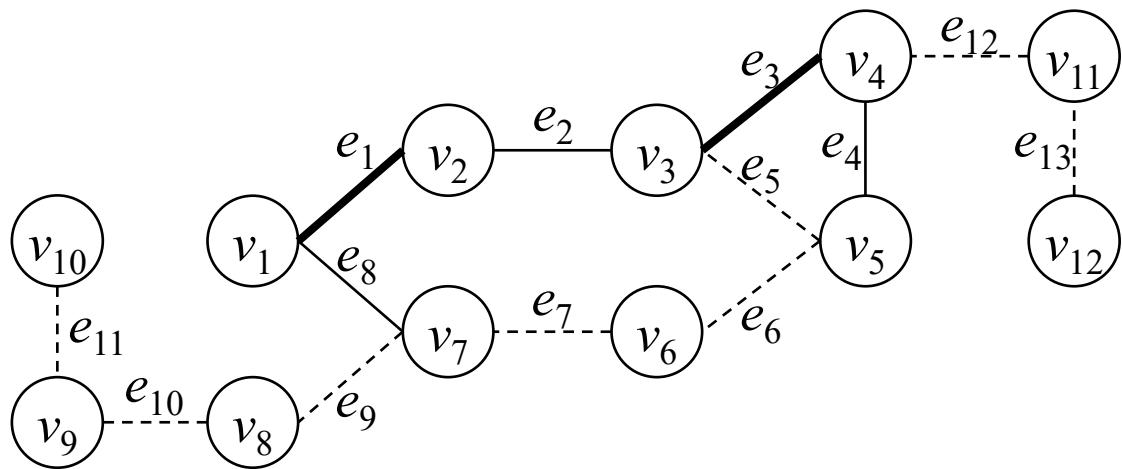
思考题 5.9 如何利用增广路得到一个更大的匹配？



请证明下述定理

定理 5.1 (贝尔热定理) 对于图 $G = \langle V, E \rangle$ 和匹配 $M \subseteq E$, M 是最大匹配当且仅当 G 不含 M 增广路。

■ 必要性 (反证法)

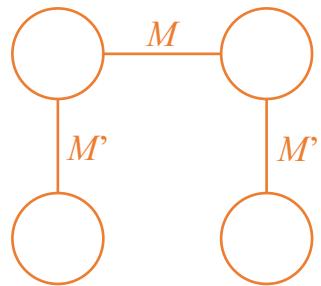
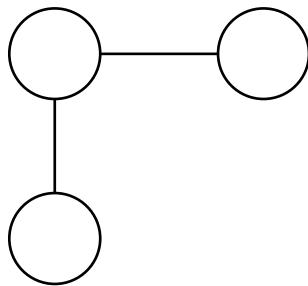
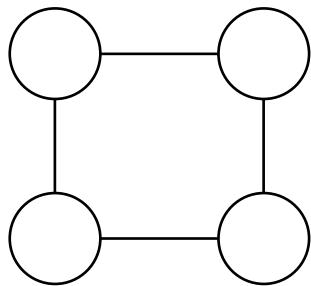


请证明下述定理

定理 5.1 (贝尔热定理) 对于图 $G = \langle V, E \rangle$ 和匹配 $M \subseteq E$, M 是最大匹配当且仅当 G 不含 M 增广路。

■ 充分性 (反证法)

- 假设最大匹配 $|M'| > |M|$, 则 M' 和 M 的对称差的边导出子图的连通分支有什么特征?

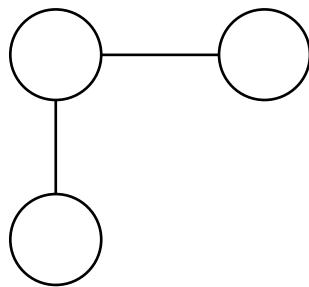
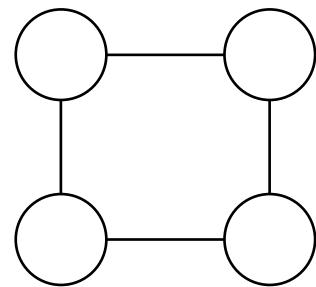


请证明下述定理

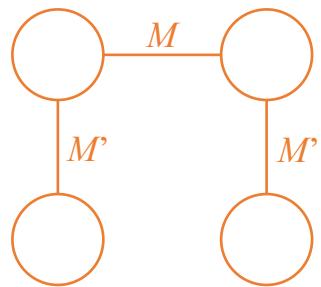
定理 5.1 (贝尔热定理) 对于图 $G = \langle V, E \rangle$ 和匹配 $M \subseteq E$, M 是最大匹配当且仅当 G 不含 M 增广路。

■ 充分性 (反证法)

- 假设最大匹配 $|M'| > |M|$, 则 M' 和 M 的对称差的边导出子图的连通分支有什么特征?



为什么是 M 增广路?



你理解这个算法了吗？

思考题 5.11 do-while 循环运行多少轮？

思考题 5.12 如何高效地判定 DFS 树中从根顶点到顶点 v 的路是 M 交错路？

算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.visited \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.visited = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10      while  $P \neq \text{null};$ 
11    输出  $(M);$ 
```

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq \text{DFS}$  树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     | if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交错路 then
7       | |  $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       | | if  $P_v \neq \text{null}$  then
9         | | | return  $P_v;$ 
10    return null;
```

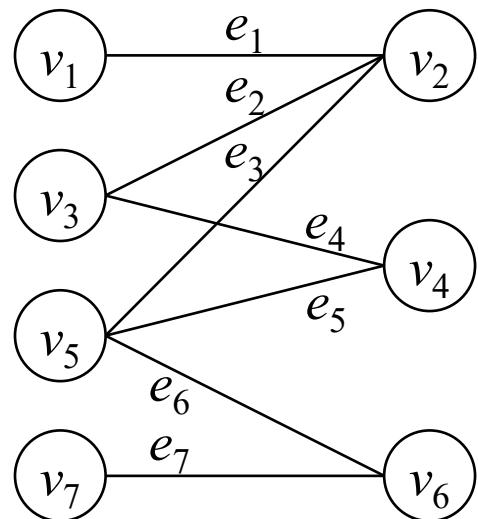
你理解这个算法了吗？

算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```



你理解这个算法了吗？

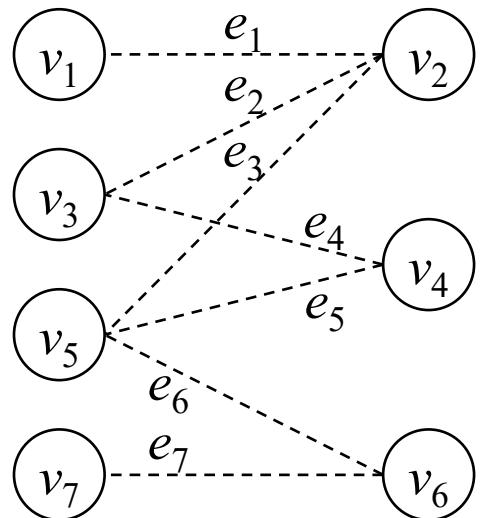
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{\}$$



你理解这个算法了吗？

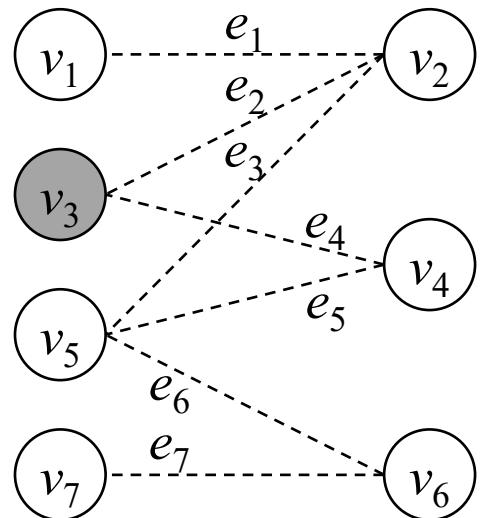
■ 第1轮do-while循环开始

算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{\}$$



你理解这个算法了吗？

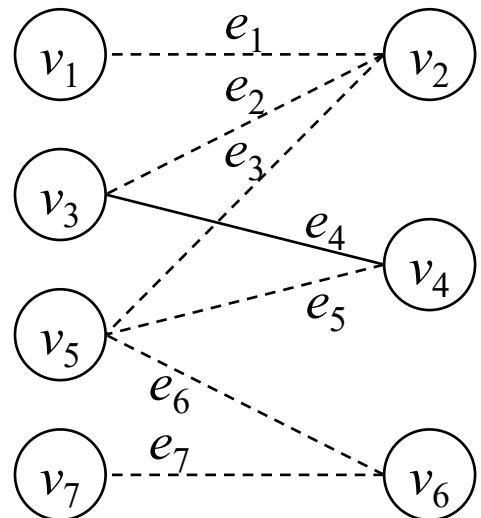
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{\}$$



你理解这个算法了吗？

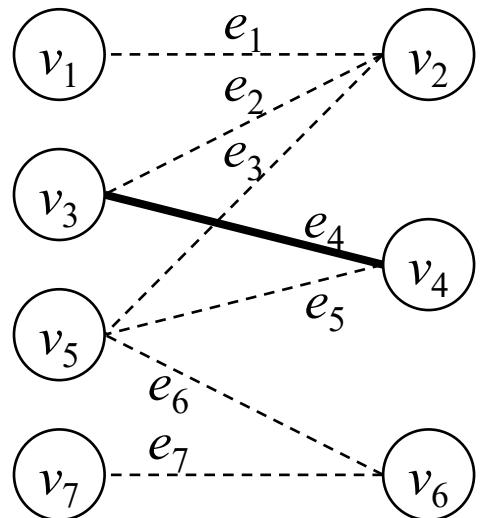
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_4\}$$



你理解这个算法了吗？

■ 第2轮do-while循环开始

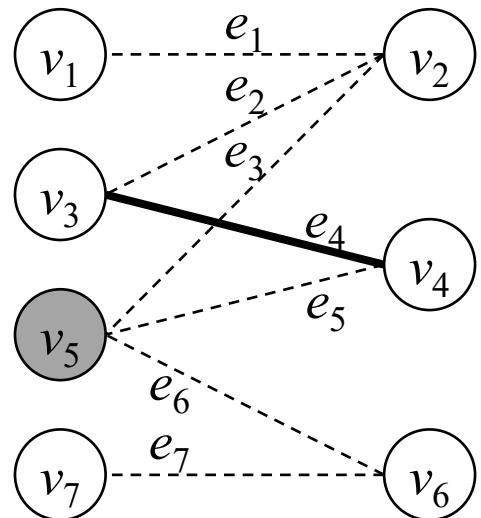
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_4\}$$



你理解这个算法了吗？

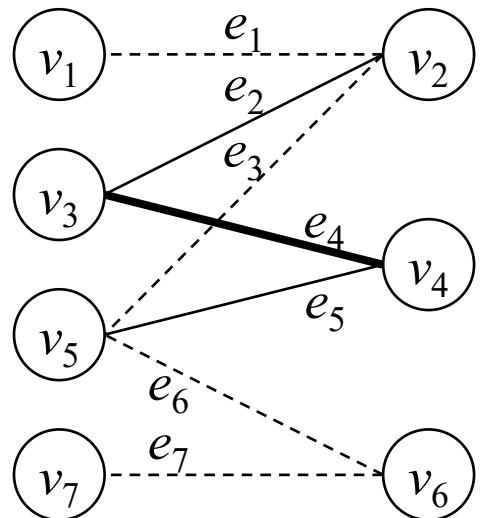
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_4\}$$



你理解这个算法了吗？

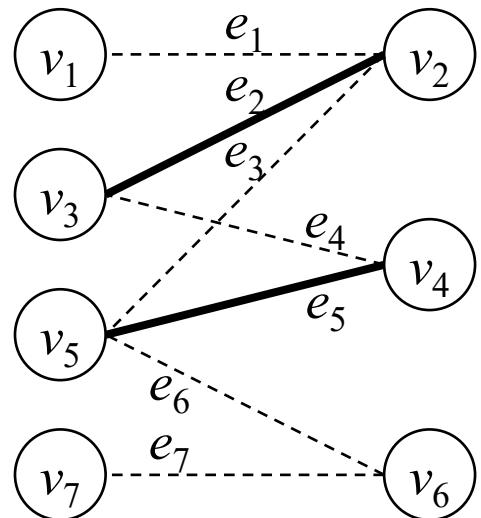
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

■ 第3轮do-while循环开始

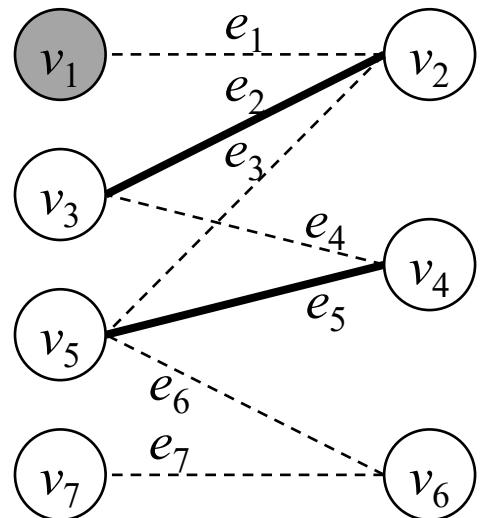
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

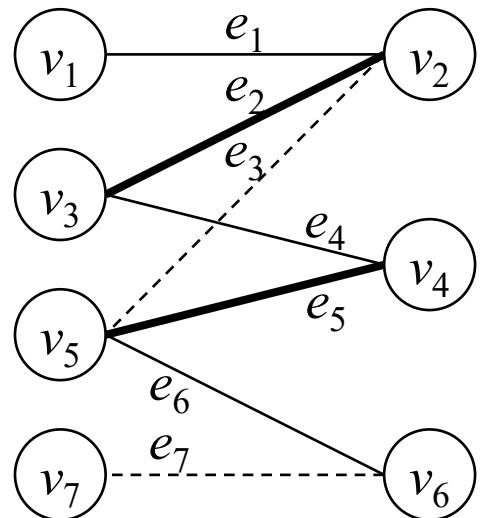
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

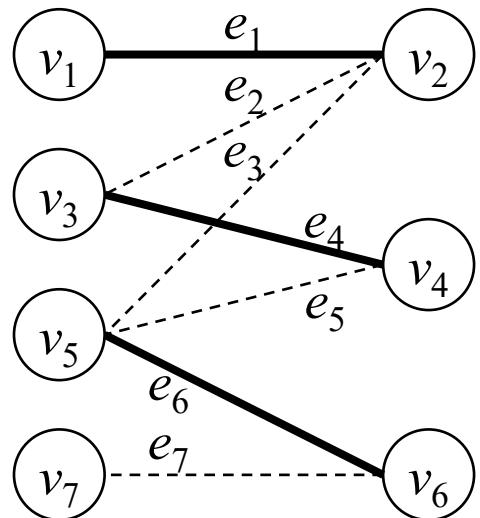
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_1, e_4, e_6\}$$



你理解这个算法了吗？

■ 第4轮do-while循环开始

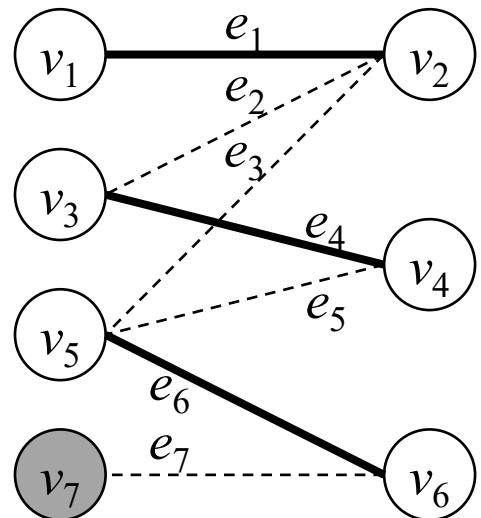
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_1, e_4, e_6\}$$



你理解这个算法了吗？

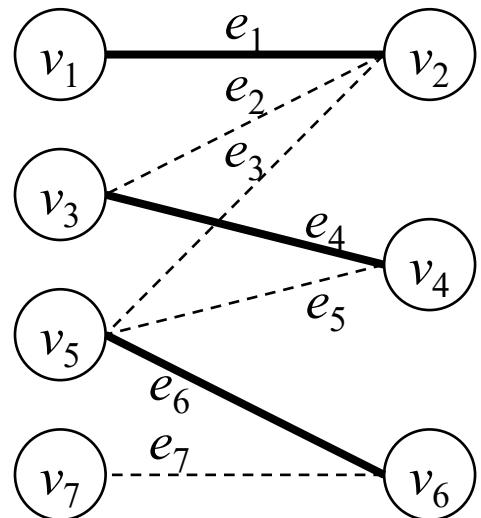
算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

$$M = \{e_1, e_4, e_6\}$$



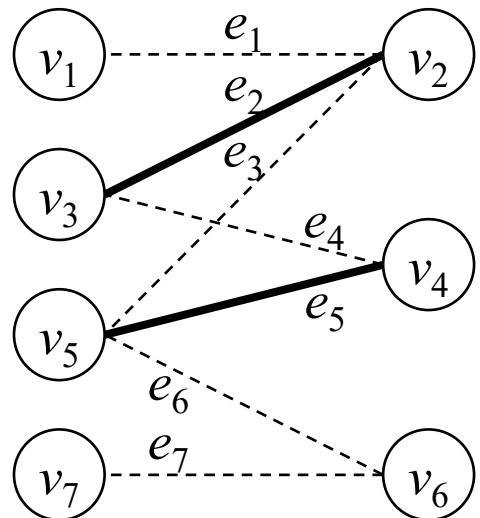
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



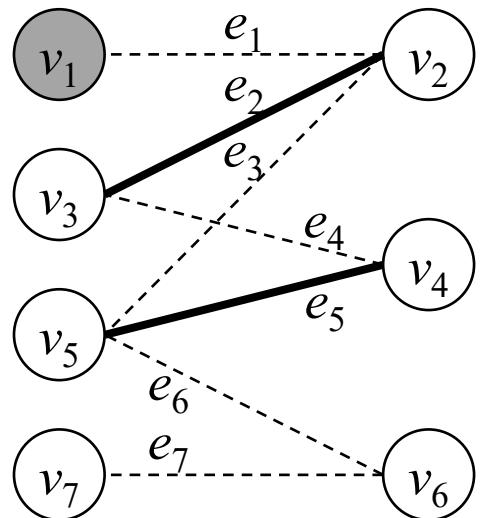
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

1 $u.visited \leftarrow \text{true};$
2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
3 | return DFS 树中从根顶点到 u 的路;
4 else
5 | foreach $(u, v) \in E$ do
6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交
| | 错路 then
7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
8 | | | if $P_v \neq \text{null}$ then
9 | | | | return $P_v;$
10 | return null;

$$M = \{e_5, e_2\}$$



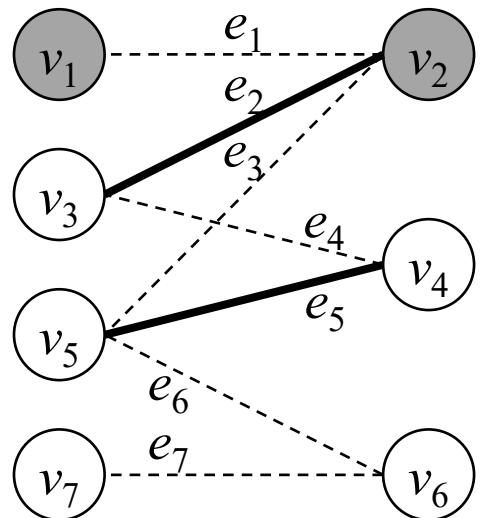
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



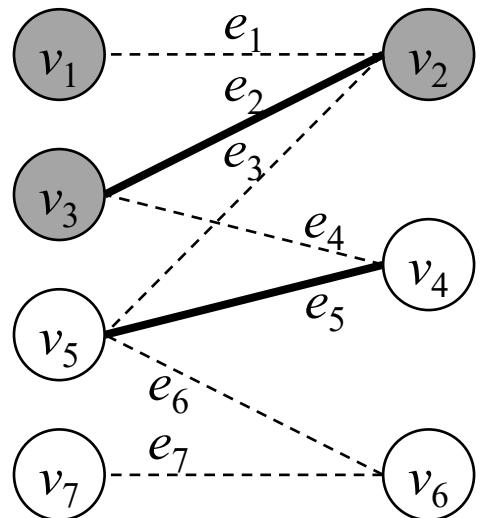
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



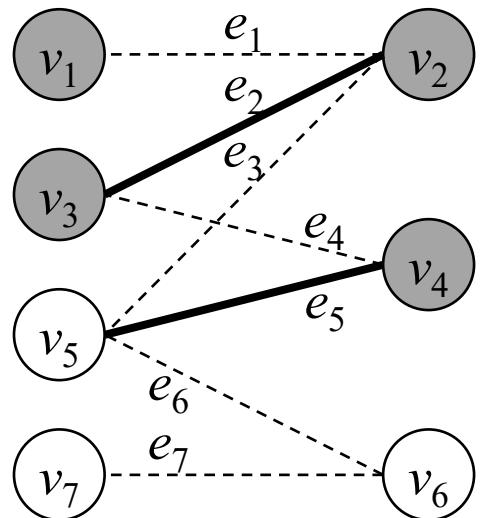
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



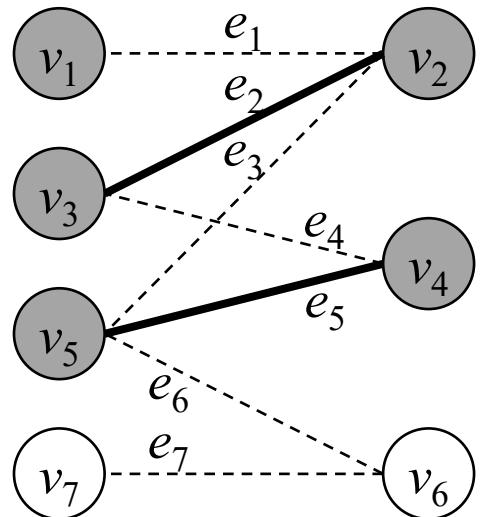
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

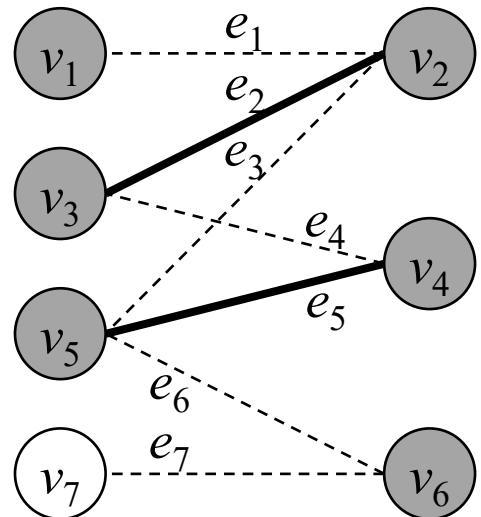
- 若非根顶点 u 未被 M 饱和，则DFS树中从根顶点到 u 的 M 交错路是 M 增广路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

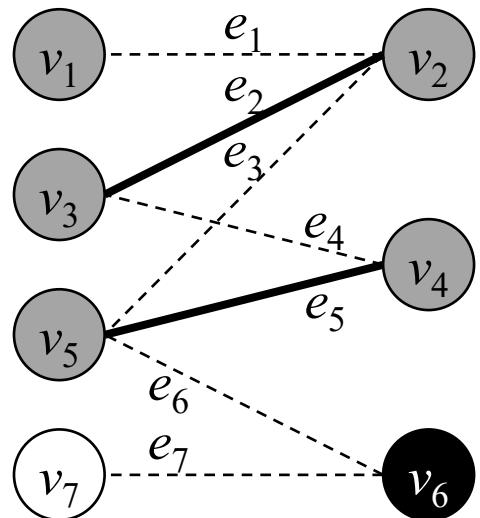
■ 算法返回这条路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

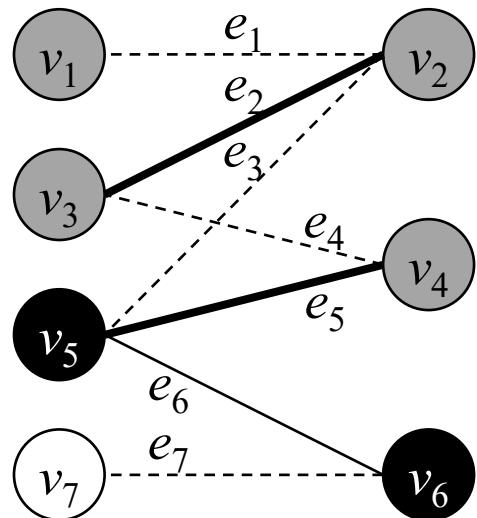
■ 算法返回这条路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

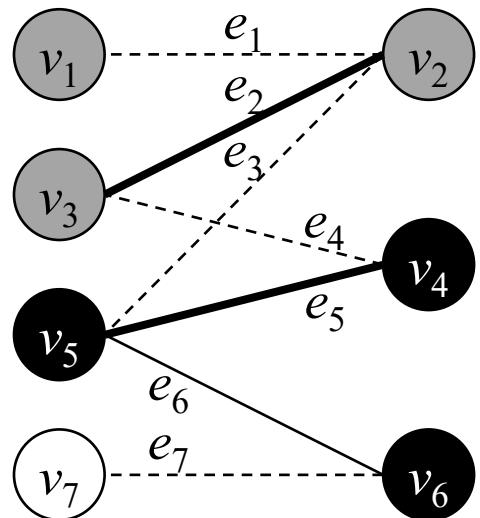
■ 算法返回这条路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

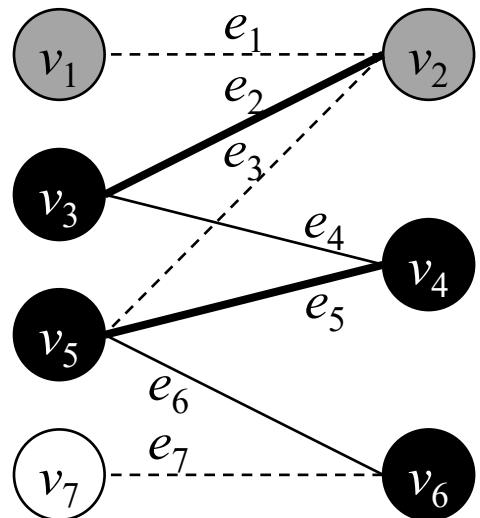
■ 算法返回这条路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_5, e_2\}$$



你理解这个算法了吗？

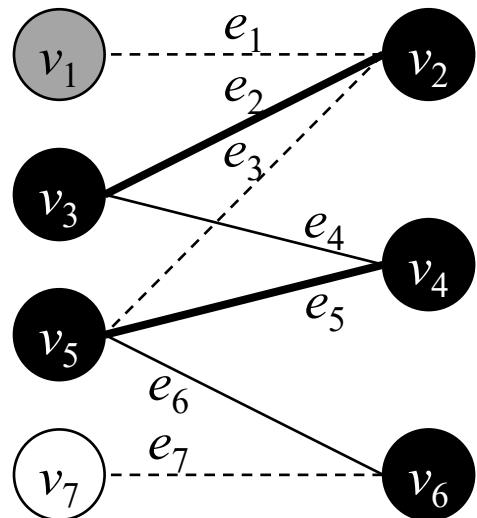
■ 算法返回这条路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_5, e_2\}$$



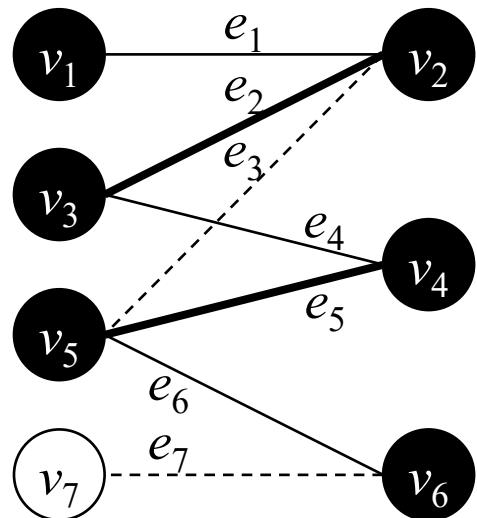
你理解这个算法了吗？

■ 算法返回这条路

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_5, e_2\}$$



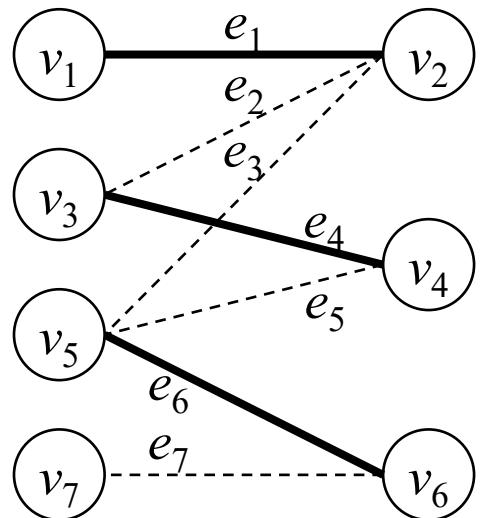
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



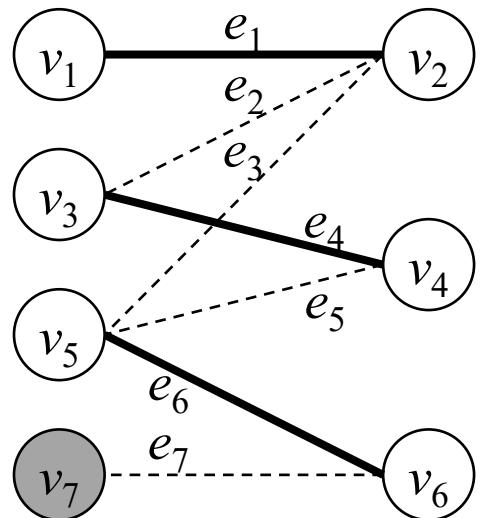
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



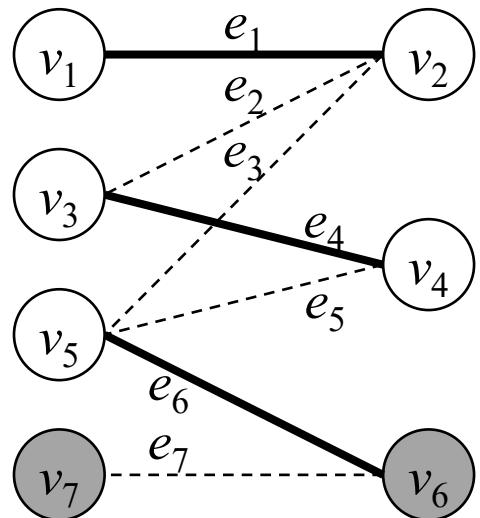
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



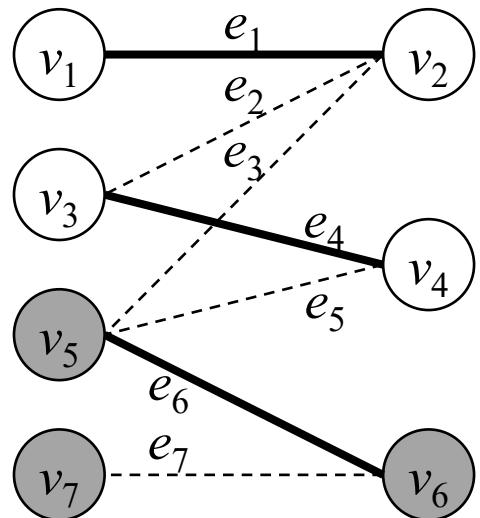
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



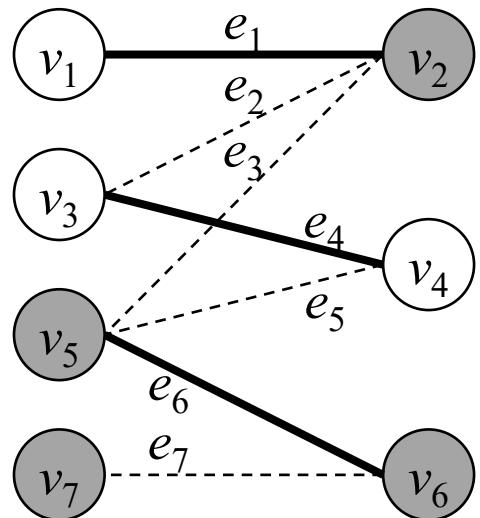
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



你理解这个算法了吗？

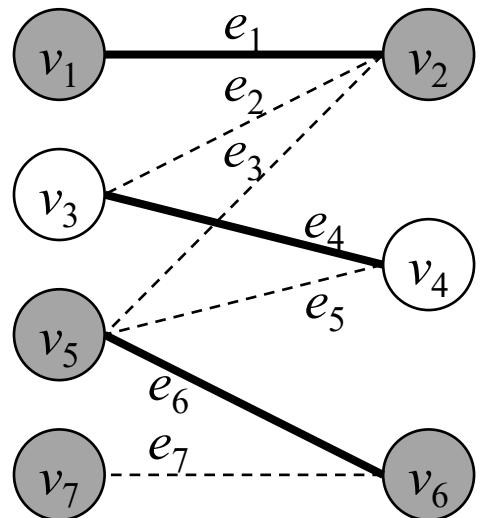
■ 未找到 M 增广路

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



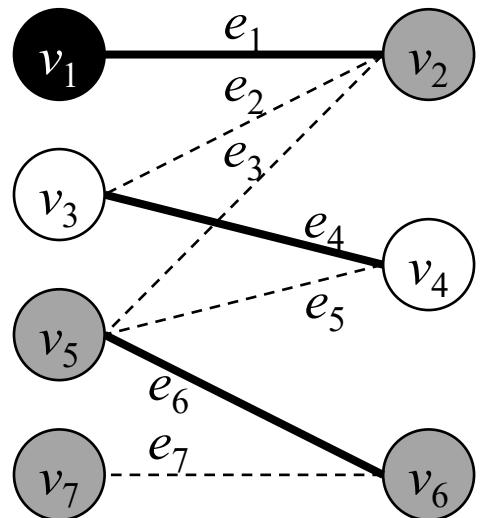
你理解这个算法了吗？

■ 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_1, e_4, e_6\}$$



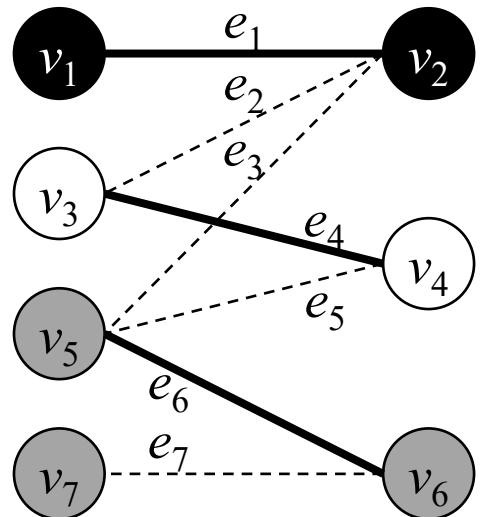
你理解这个算法了吗？

- 未找到 M 增广路， 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_1, e_4, e_6\}$$



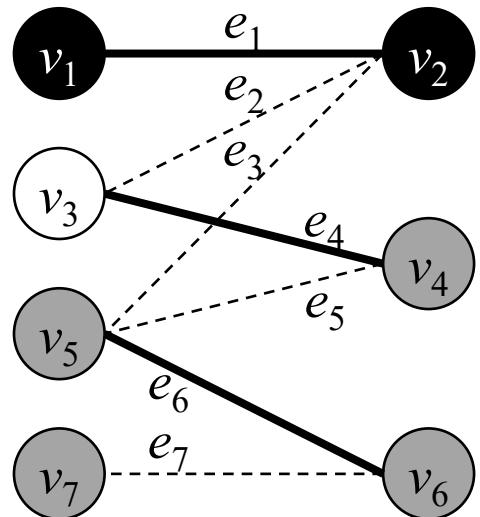
你理解这个算法了吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



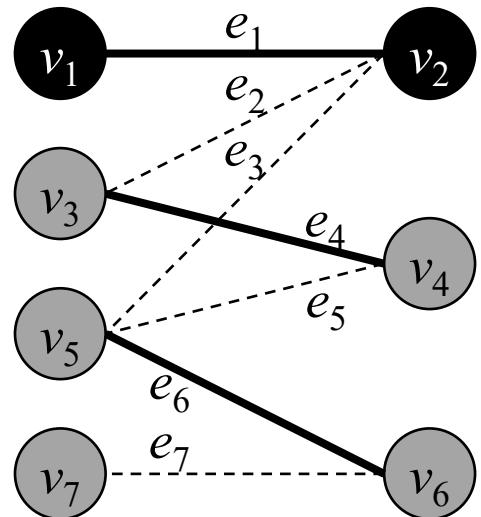
你理解这个算法了吗？

■ 未找到 M 增广路

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_1, e_4, e_6\}$$



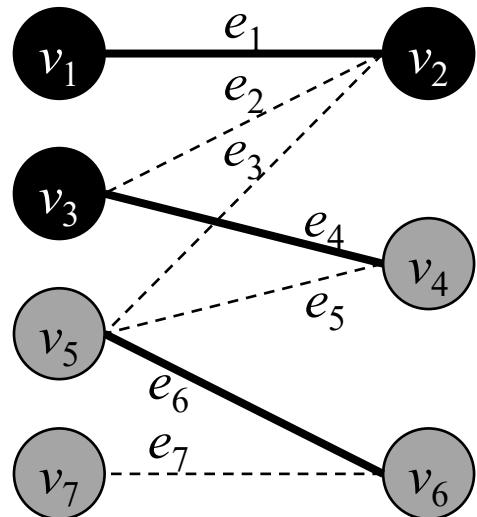
你理解这个算法了吗？

■ 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_1, e_4, e_6\}$$



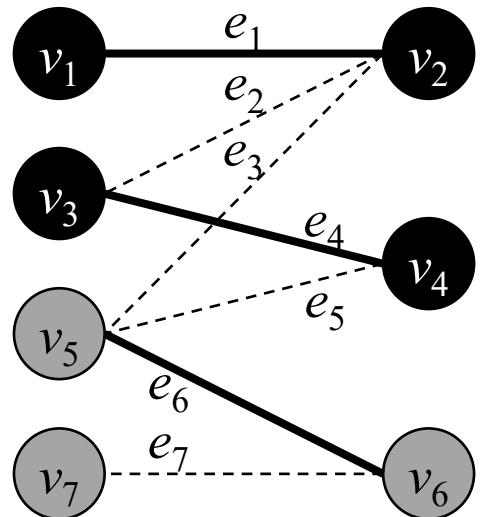
你理解这个算法了吗？

- 未找到 M 增广路， 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   | return DFS 树中从根顶点到  $u$  的路;
4 else
5   | foreach  $(u, v) \in E$  do
6     |   | if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
7       |   |   错路 then
8         |   |   |  $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
9         |   |   | if  $P_v \neq \text{null}$  then
10        |   |   |   | return  $P_v;$ 
10      |   | return null;
```

$$M = \{e_1, e_4, e_6\}$$



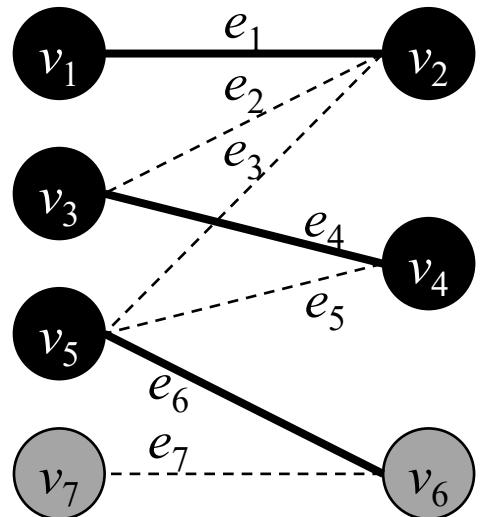
你理解这个算法了吗？

- 未找到 M 增广路， 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   | return DFS 树中从根顶点到  $u$  的路;
4 else
5   | foreach  $(u, v) \in E$  do
6     |   | if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
7       |   |   错路 then
8         |   |   |  $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
9         |   |   | if  $P_v \neq \text{null}$  then
10        |   |   |   | return  $P_v;$ 
10      |   | return null;
```

$$M = \{e_1, e_4, e_6\}$$



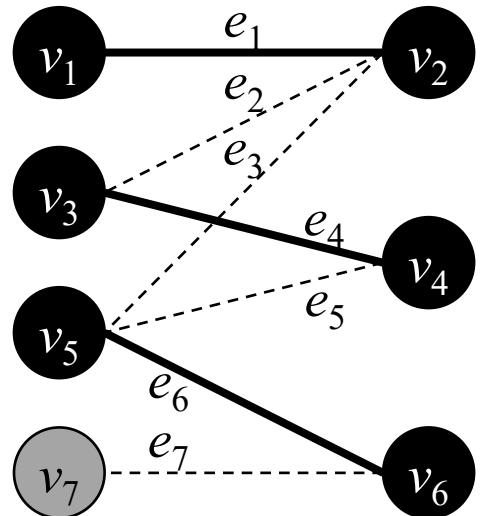
你理解这个算法了吗？

- 未找到 M 增广路， 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
      错路 then
7        $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
8       if  $P_v \neq \text{null}$  then
9         return  $P_v;$ 
10  return null;
```

$$M = \{e_1, e_4, e_6\}$$



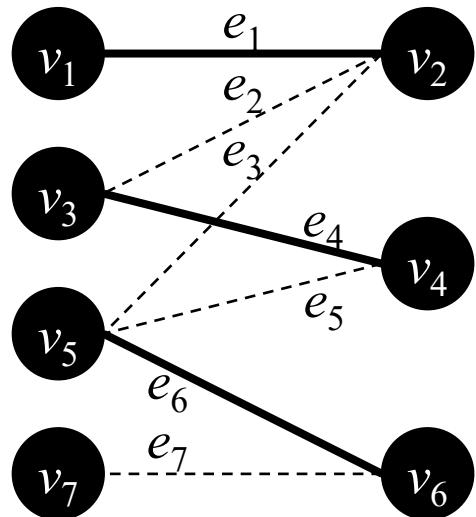
你理解这个算法了吗？

- 未找到 M 增广路， 算法返回null

算法 5.2: DFSAP

```
输入: 图  $G = \langle X \cup Y, E \rangle$ , 顶点  $u$ , 匹配  $M$ 
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq$  DFS 树的根顶点 then
3   | return DFS 树中从根顶点到  $u$  的路;
4 else
5   | foreach  $(u, v) \in E$  do
6     |   | if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
7       |   |   错路 then
8         |   |   |  $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
9         |   |   | if  $P_v \neq \text{null}$  then
10        |   |   |   | return  $P_v;$ 
10      |   | return null;
```

$$M = \{e_1, e_4, e_6\}$$



你理解这个算法了吗？

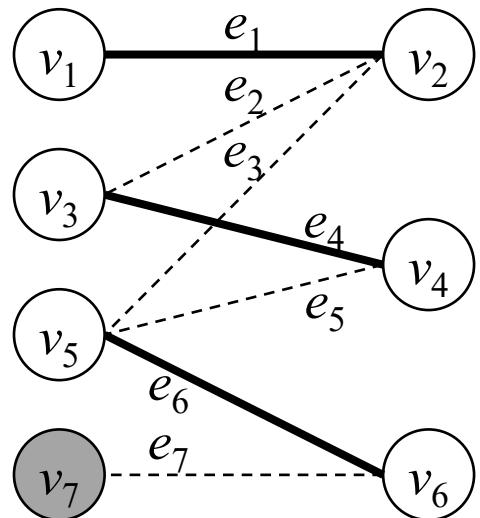
思考题 5.13 (♦) 对顶点 r 调用 DFSAP 算法返回 null 时，是否已尝试所有以 r 为起点的 M 交错路？会遗漏 M 增广路吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



你理解这个算法了吗？

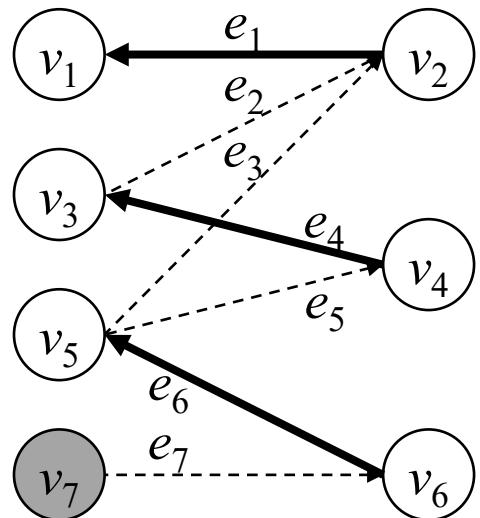
思考题 5.13 (♦) 对顶点 r 调用 DFSAP 算法返回 null 时，是否已尝试所有以 r 为起点的 M 交错路？会遗漏 M 增广路吗？

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

- 1 $u.visited \leftarrow \text{true};$
- 2 if u 未被 M 饱和且 $u \neq$ DFS 树的根顶点 then
- 3 | return DFS 树中从根顶点到 u 的路;
- 4 else
- 5 | foreach $(u, v) \in E$ do
- 6 | | if $v.visited = \text{false}$ 且 DFS 树中从根顶点到 v 的路是 M 交错路 then
- 7 | | | $P_v \leftarrow \text{DFSAP}(G, v, M);$
- 8 | | | if $P_v \neq \text{null}$ then
- 9 | | | | return $P_v;$
- 10 | return null;

$$M = \{e_1, e_4, e_6\}$$



你能解释这个算法的时间复杂度吗？

- 对于阶为 n 、边数为 m 的图，匈牙利算法的时间复杂度为 $O(n(n + m))$

算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.visited \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.visited = \text{false}$  且  $r$  未被  $M$  饱和 then
6       | |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       | | if  $P \neq \text{null}$  then
8         | | |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         | | | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

算法 5.2: DFSAP

输入: 图 $G = \langle X \cup Y, E \rangle$, 顶点 u , 匹配 M

```
1  $u.visited \leftarrow \text{true};$ 
2 if  $u$  未被  $M$  饱和且  $u \neq \text{DFS 树的根顶点}$  then
3   return DFS 树中从根顶点到  $u$  的路;
4 else
5   foreach  $(u, v) \in E$  do
6     | if  $v.visited = \text{false}$  且 DFS 树中从根顶点到  $v$  的路是  $M$  交
7       | | 错路 then
8       | | |  $P_v \leftarrow \text{DFSAP}(G, v, M);$ 
9       | | | if  $P_v \neq \text{null}$  then
10      | | | | return  $P_v;$ 
10  return null;
```

霍普克罗夫特-卡普算法做出了哪两项改进？

算法 5.1: 匈牙利算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2   foreach  $u \in (X \cup Y)$  do
3     |  $u.\text{visited} \leftarrow \text{false};$ 
4   foreach  $r \in X$  do
5     | if  $r.\text{visited} = \text{false}$  且  $r$  未被  $M$  饱和 then
6       |   |  $P \leftarrow \text{DFSAP}(G, r, M);$ 
7       |   | if  $P \neq \text{null}$  then
8         |   |   |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
9         |   |   | 中止 ForEach 循环;
10  while  $P \neq \text{null};$ 
11  输出  $(M);$ 
```

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2   |  $Q \leftarrow \text{HKInit}(G, M);$ 
3   |  $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4   |  $\mathcal{P} \leftarrow \text{HCPPaths}(G, Y');$ 
5   | foreach  $P \in \mathcal{P}$  do
6     |   |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

你理解这个算法了吗？

- HKInit的功能? Q 的含义?
- HKBFS的功能? Y' 的含义?
- HKPaths的功能? P 的含义?

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

```
1 do
2   |    $Q \leftarrow \text{HKInit}(G, M);$ 
3   |    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4   |    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   |   foreach  $P \in \mathcal{P}$  do
6     |     |    $M \leftarrow P$ 经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

你理解这个算法了吗？

■ d的含义？

算法 5.4: HKInit

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M
初值: 队列 Q 初值为空

```
1 foreach  $u \in (X \cup Y)$  do
2   if  $u \in X$  且  $u$  未被  $M$  饱和 then
3      $u.visited \leftarrow \text{true};$ 
4      $u.d \leftarrow 0;$ 
5     入队列  $(Q, u);$ 
6   else
7      $u.visited \leftarrow \text{false};$ 
8      $u.d \leftarrow \infty;$ 
9 return  $Q;$ 
```

你理解这个算法了吗？

- d' 的含义和作用？
- 你能基于 d' 给出 Y' 的含义吗？

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

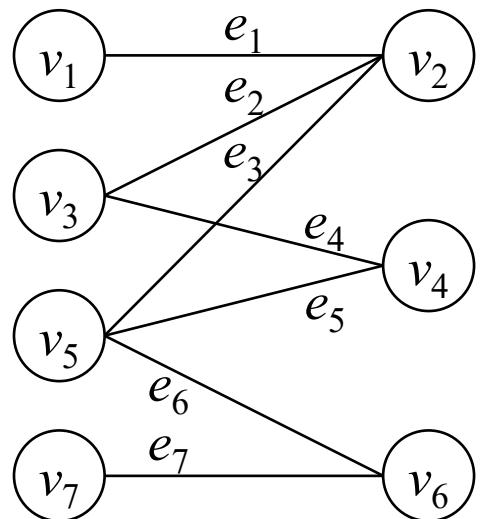
```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15
16 return  $Y'$ ;
```

你理解这个算法了吗？

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



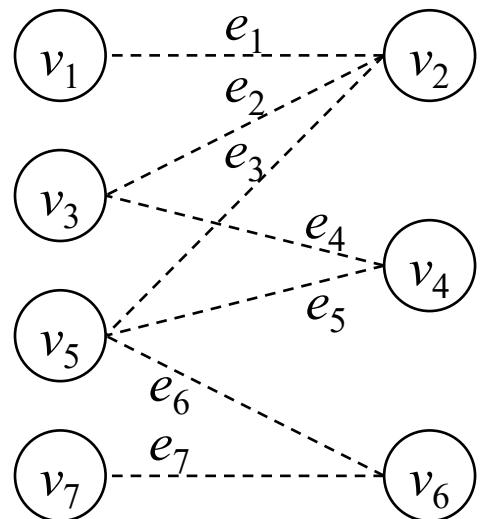
你理解这个算法了吗？

■ 第1轮do-while循环开始

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7 while  $\mathcal{P} \neq \emptyset;$ 
8 输出  $(M);$ 
```

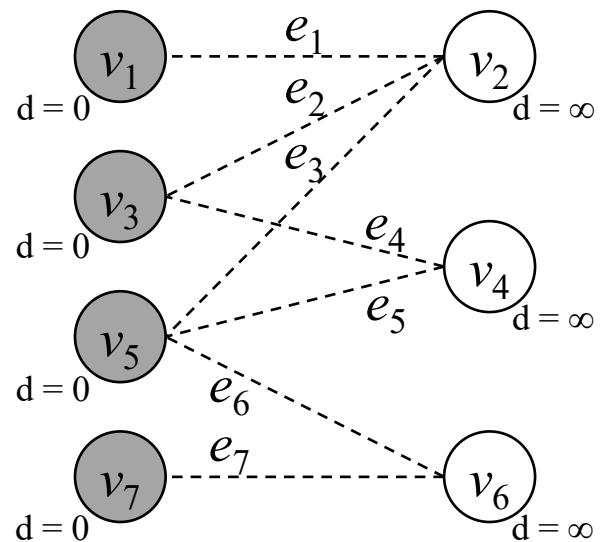


你理解这个算法了吗？

$$Q \quad \overline{v_1 \quad v_3 \quad v_5 \quad v_7}$$

算法 5.4: HKInit

```
输入: 二分图  $G = \langle X \cup Y, E \rangle$ 
初值:  $Q$  初值为空队列
1 foreach  $u \in (X \cup Y)$  do
2   if  $u \in X$  且  $u$  未被  $M$  饱和 then
3      $u.\text{visited} \leftarrow \text{true};$ 
4      $u.d \leftarrow 0;$ 
5     入队列  $(Q, u);$ 
6   else
7      $u.\text{visited} \leftarrow \text{false};$ 
8      $u.d \leftarrow \infty;$ 
9 return  $Q;$ 
```



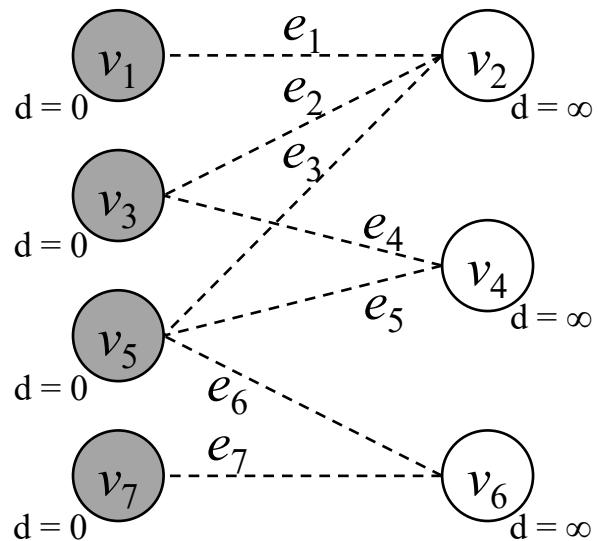
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_1 \quad v_3 \quad v_5 \quad v_7} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



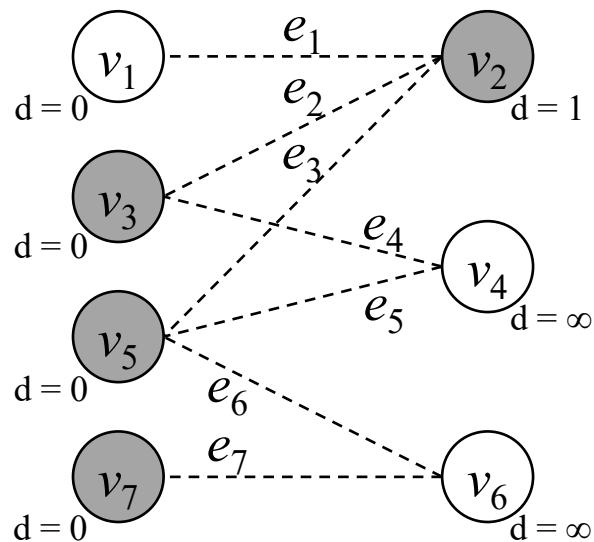
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_3 \quad v_5 \quad v_7 \quad v_2} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



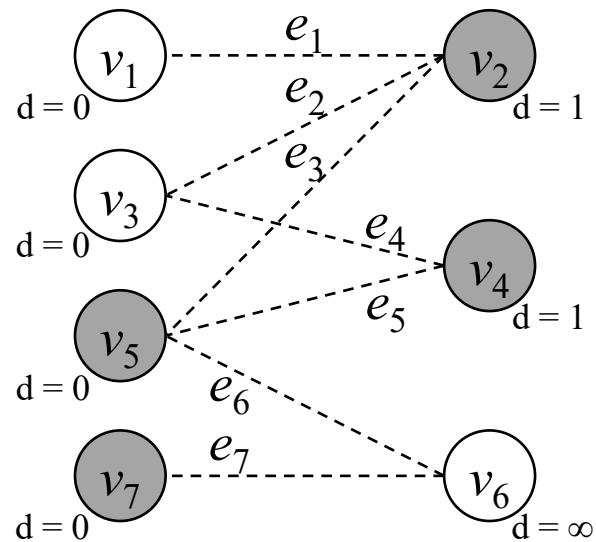
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_5 \quad v_7 \quad v_2 \quad v_4} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



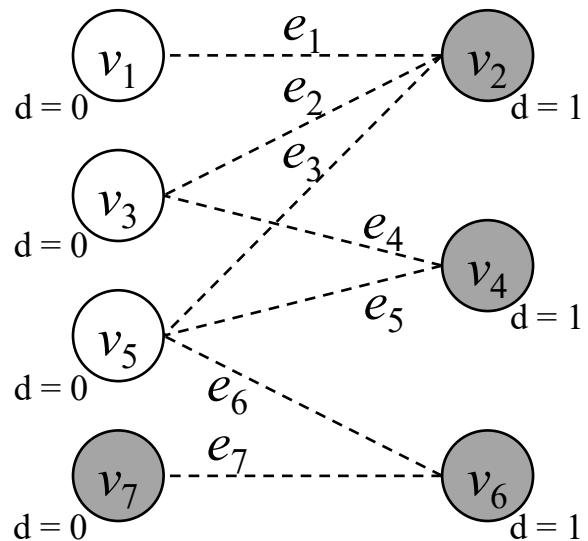
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_7 \quad v_2 \quad v_4 \quad v_6} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



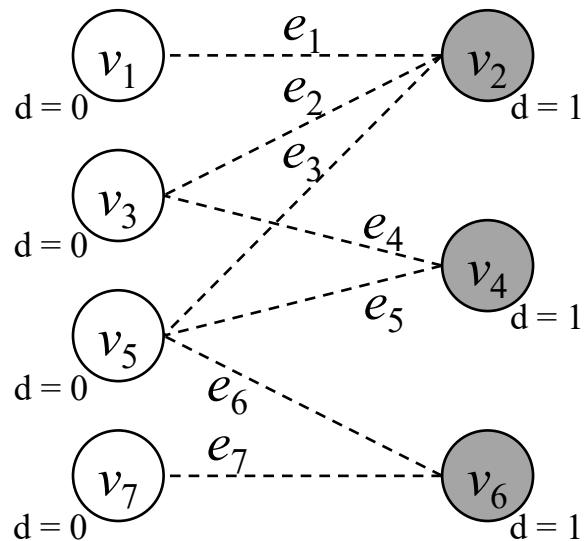
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_2 \quad v_4 \quad v_6} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



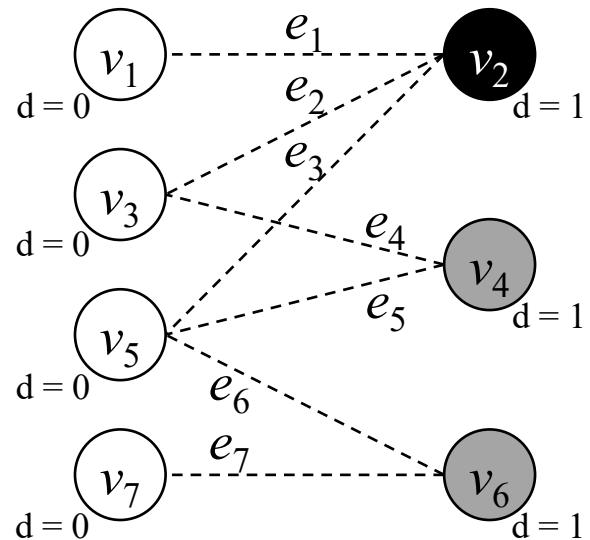
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_4 & v_6 \\ Y' = \{v_2\} \\ d' = 1 \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



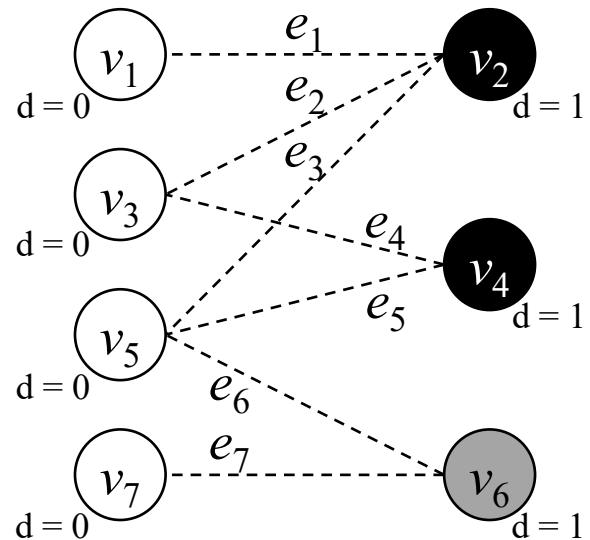
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_6 \\ Y' = \{v_2, v_4\} \\ d' = 1 \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



你理解这个算法了吗？

Q

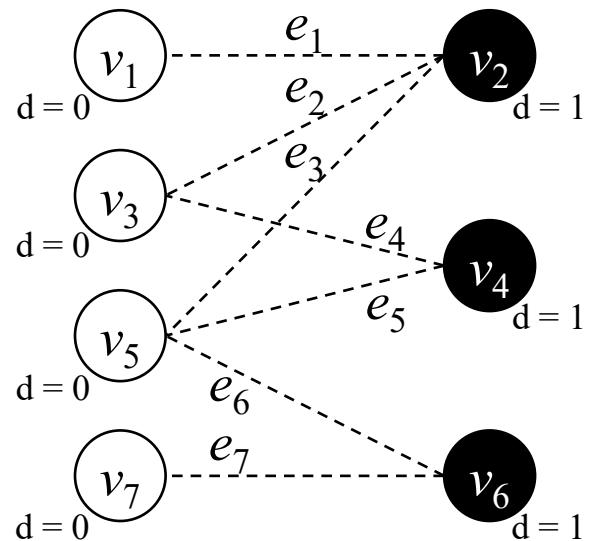
$Y' = \{v_2, v_4, v_6\}$

$d' = 1$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



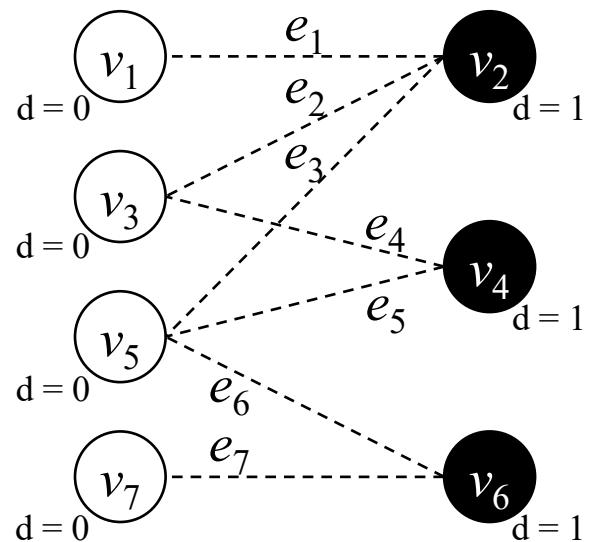
你理解这个算法了吗？

$$Y = \{v_2, v_4, v_6\}$$

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



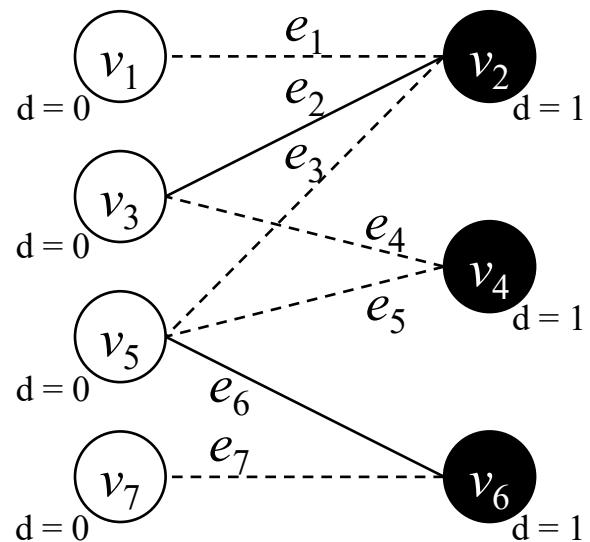
你理解这个算法了吗？

$$Y = \{v_2, v_4, v_6\}$$

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

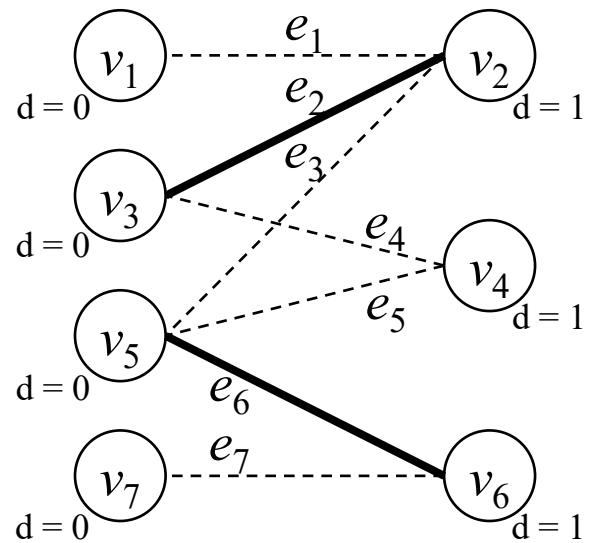


你理解这个算法了吗？

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



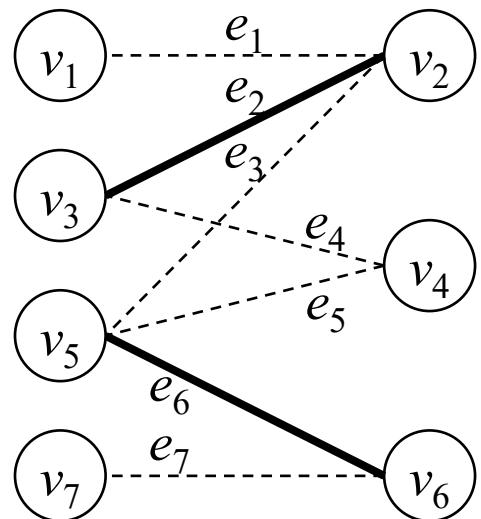
你理解这个算法了吗？

- 第2轮do-while循环开始

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

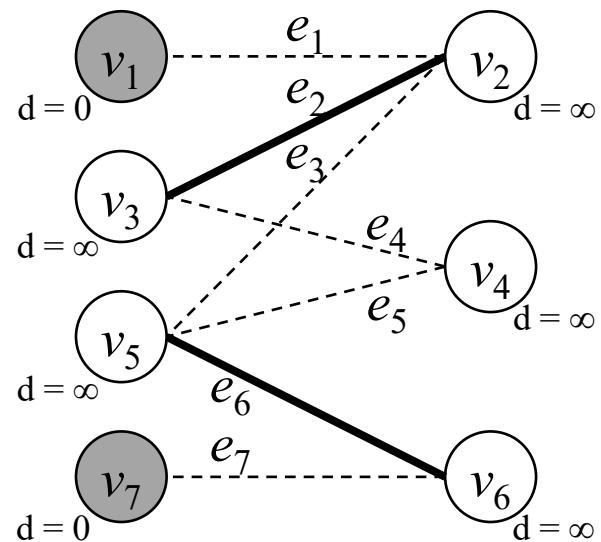


你理解这个算法了吗？

$$Q \quad \frac{v_1 \quad v_7}{}$$

算法 5.4: HKInit

```
输入: 二分图  $G = \langle X \cup Y, E \rangle$ 
初值:  $Q$  初值为空队列
1 foreach  $u \in (X \cup Y)$  do
2   if  $u \in X$  且  $u$  未被  $M$  饱和 then
3      $u.visited \leftarrow \text{true};$ 
4      $u.d \leftarrow 0;$ 
5     入队列  $(Q, u);$ 
6   else
7      $u.visited \leftarrow \text{false};$ 
8      $u.d \leftarrow \infty;$ 
9 return  $Q;$ 
```



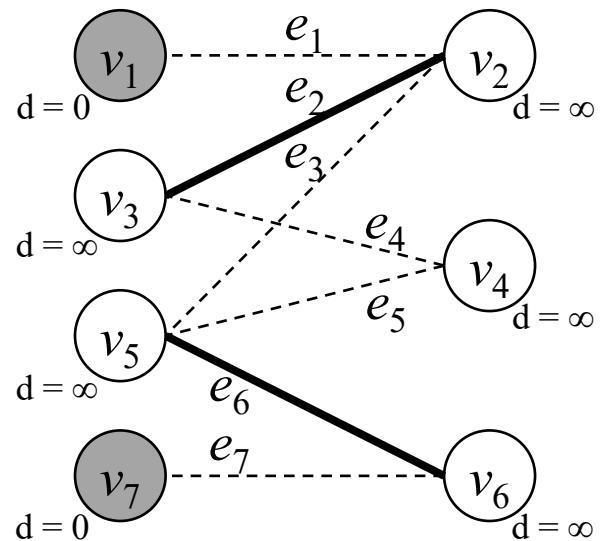
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_1 & v_7 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



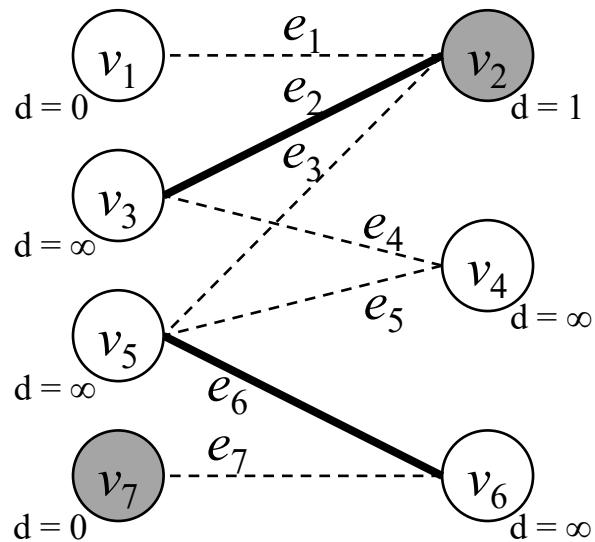
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_7 \quad v_2} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



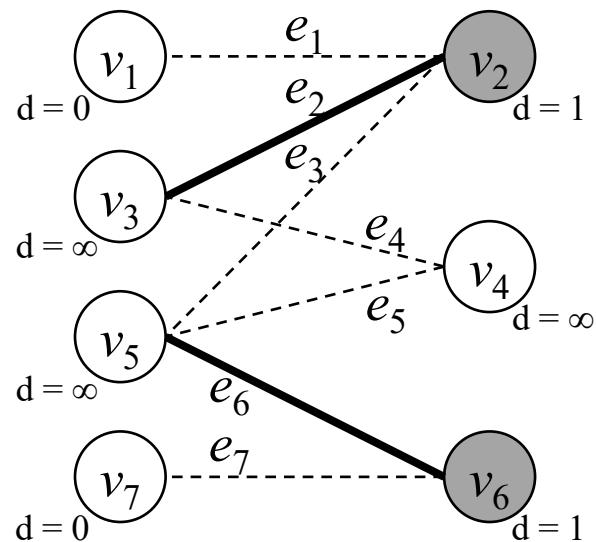
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_2 & v_6 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



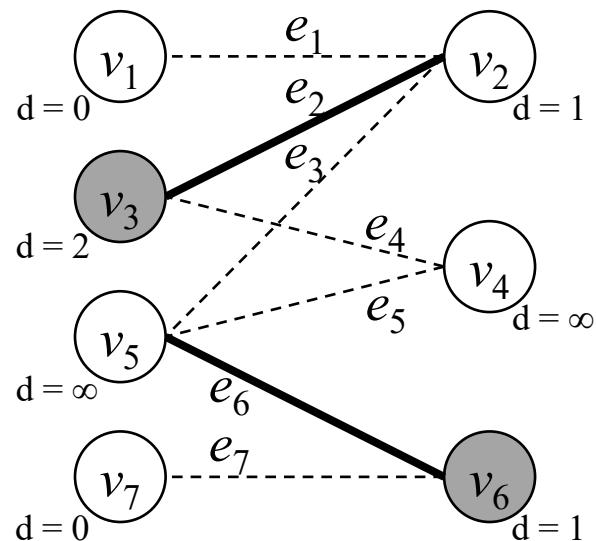
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_6 \quad v_3} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



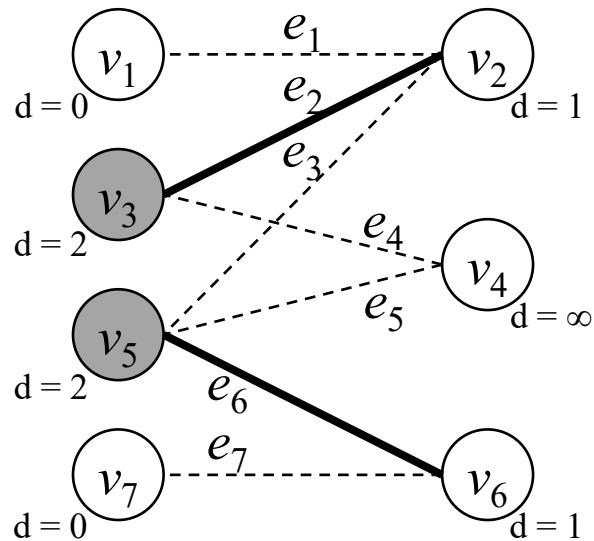
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_3 & v_5 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



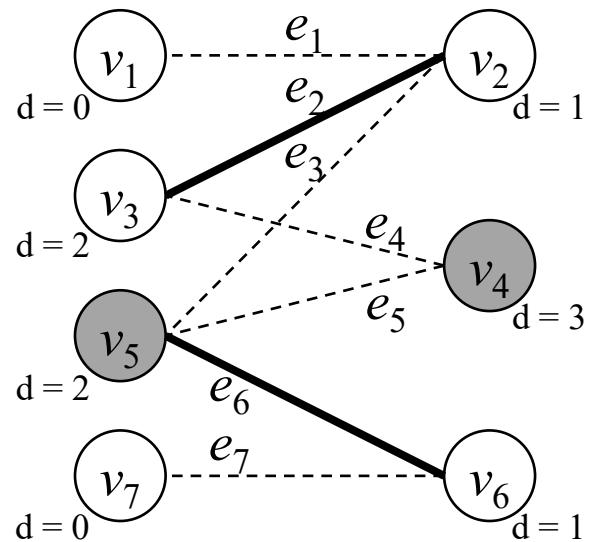
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_5 & v_4 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



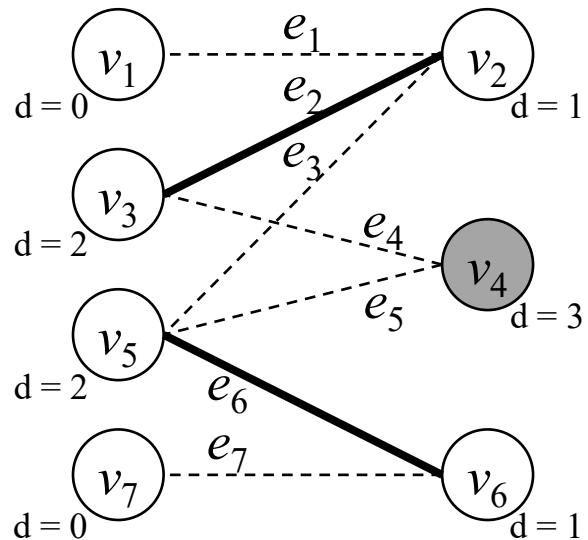
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_4 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



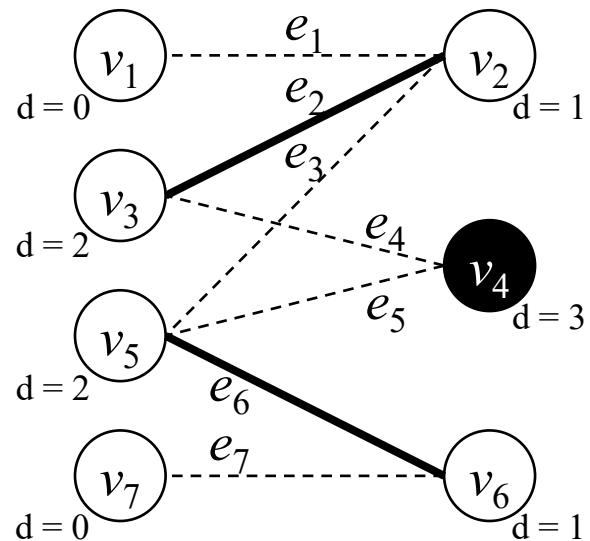
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline Y' = \{v_4\} \\ d' = 3 \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



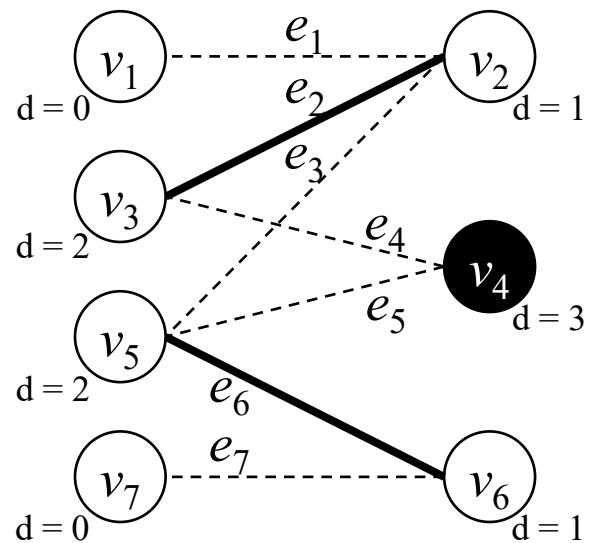
你理解这个算法了吗？

$$Y = \{v_4\}$$

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



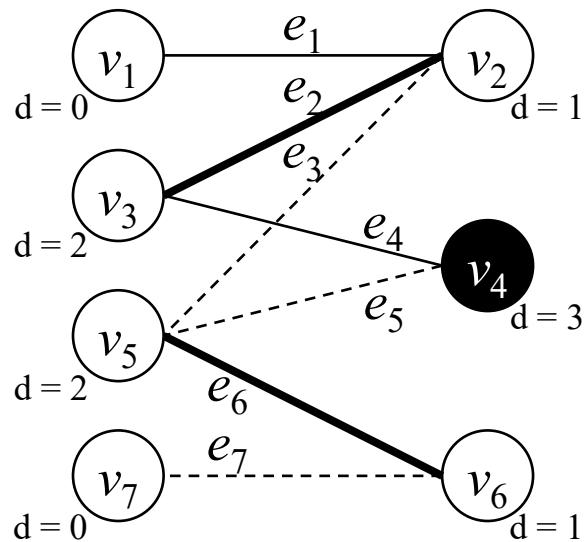
你理解这个算法了吗？

$$Y = \{v_4\}$$

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

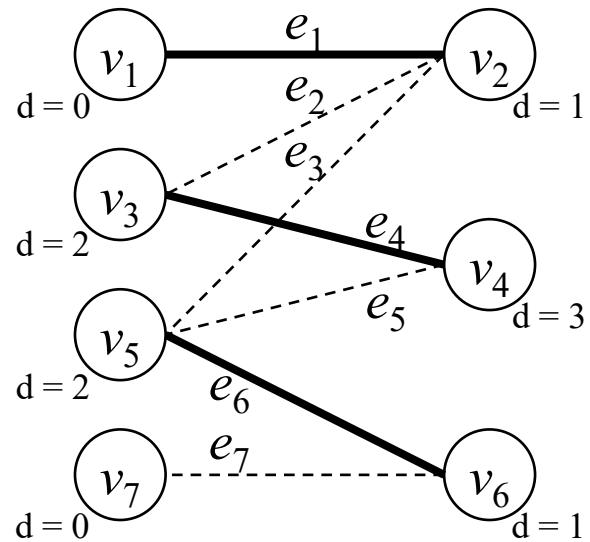


你理解这个算法了吗？

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



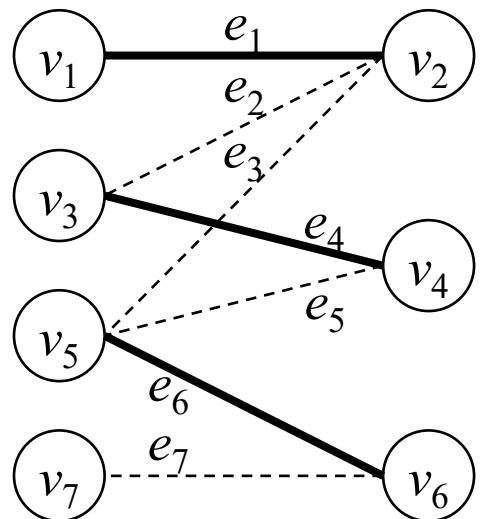
你理解这个算法了吗？

■ 第3轮do-while循环开始

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2      $Q \leftarrow \text{HKInit}(G, M);$ 
3      $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4      $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5     foreach  $P \in \mathcal{P}$  do
6          $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7     while  $\mathcal{P} \neq \emptyset;$ 
8     输出  $(M);$ 
```

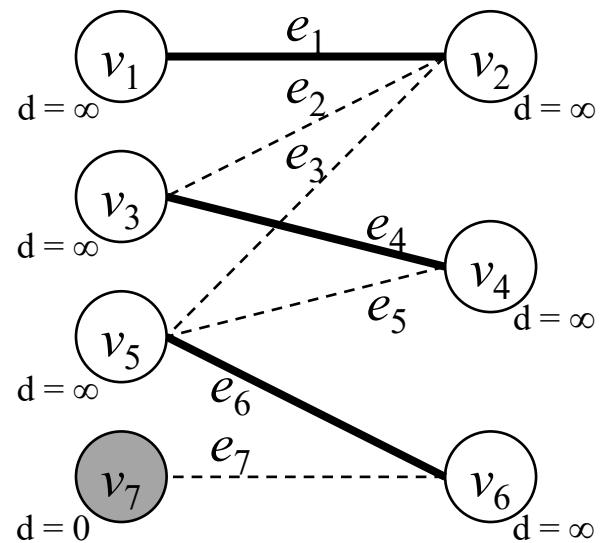


你理解这个算法了吗？

$$Q \quad \overline{v_7}$$

算法 5.4: HKInit

```
输入: 二分图  $G = \langle X \cup Y, E \rangle$ 
初值:  $Q$  初值为空队列
1 foreach  $u \in (X \cup Y)$  do
2   if  $u \in X$  且  $u$  未被  $M$  饱和 then
3      $u.\text{visited} \leftarrow \text{true};$ 
4      $u.d \leftarrow 0;$ 
5     入队列  $(Q, u);$ 
6   else
7      $u.\text{visited} \leftarrow \text{false};$ 
8      $u.d \leftarrow \infty;$ 
9 return  $Q;$ 
```



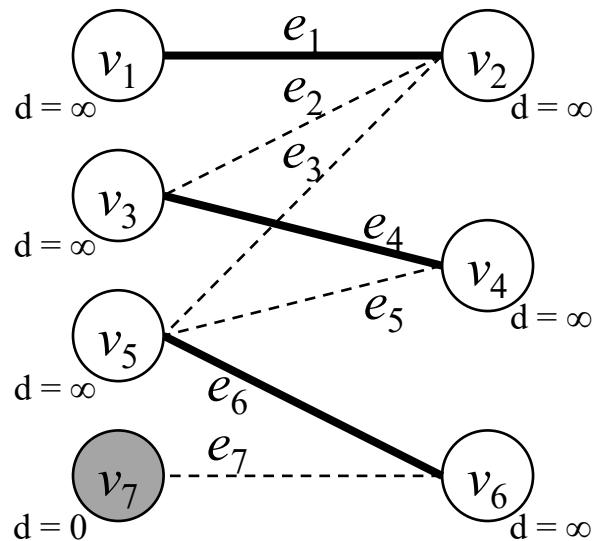
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_7 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



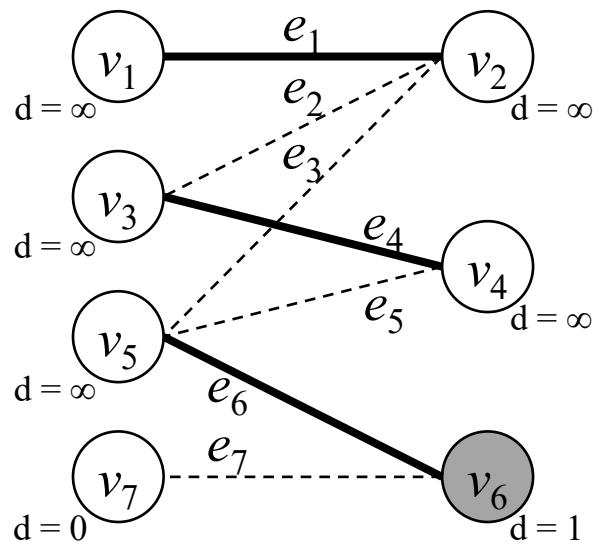
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_6 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



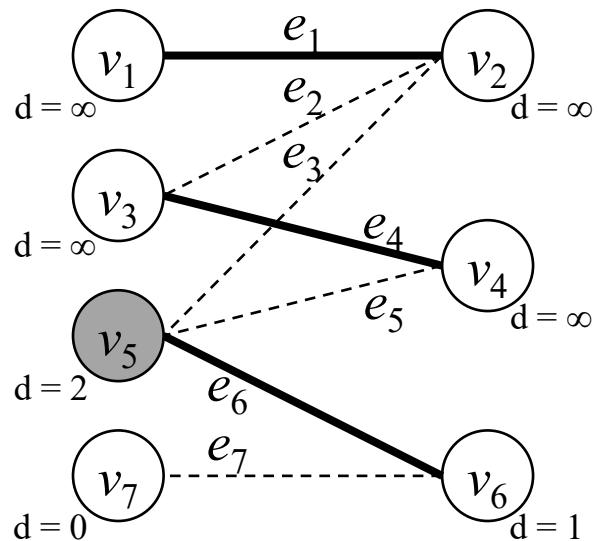
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_5 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



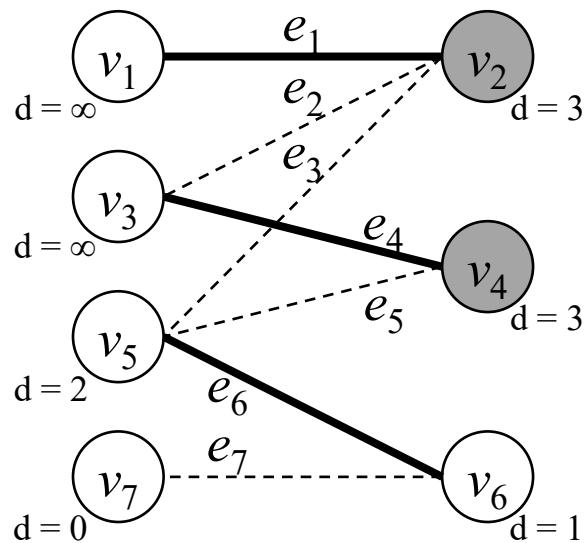
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_2 & v_4 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



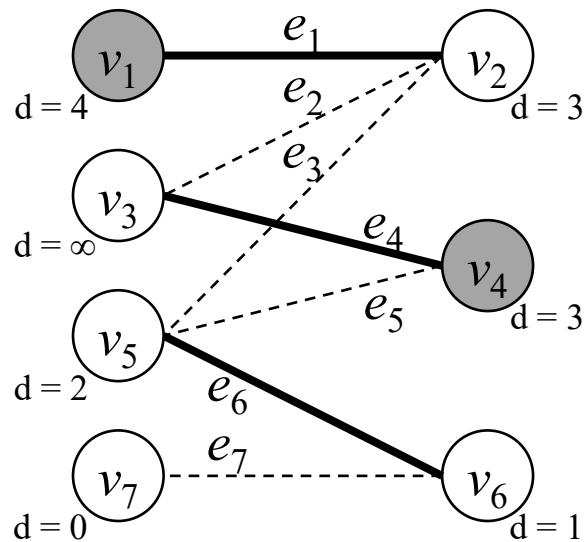
你理解这个算法了吗？

$$\begin{array}{c} Q \quad \overline{v_4 \quad v_1} \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



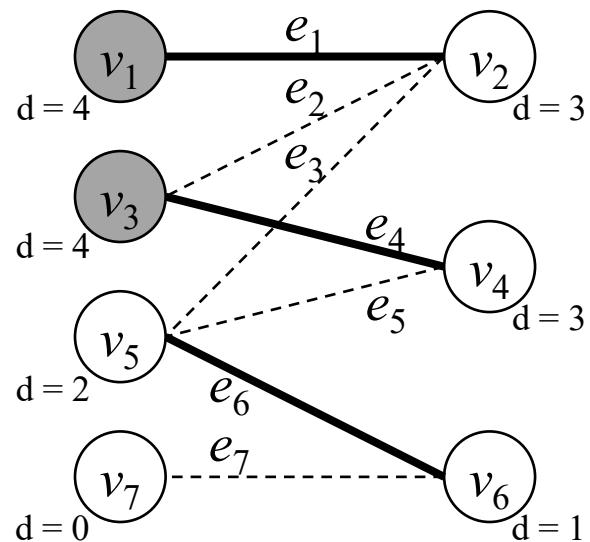
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_1 & v_3 \\ \hline Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



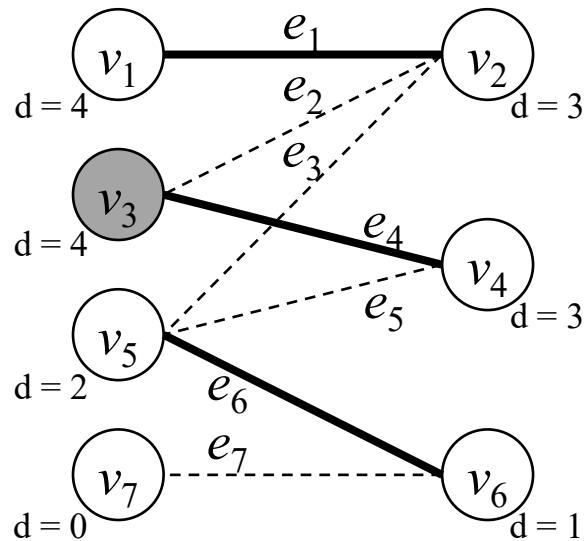
你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline v_3 \\ Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

输入: 二分图 $G = \langle X \cup Y, E \rangle$, 匹配 M , 队列 Q
初值: 顶点子集 Y' 初值为 \emptyset ; 变量 d' 初值为 ∞

```
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```

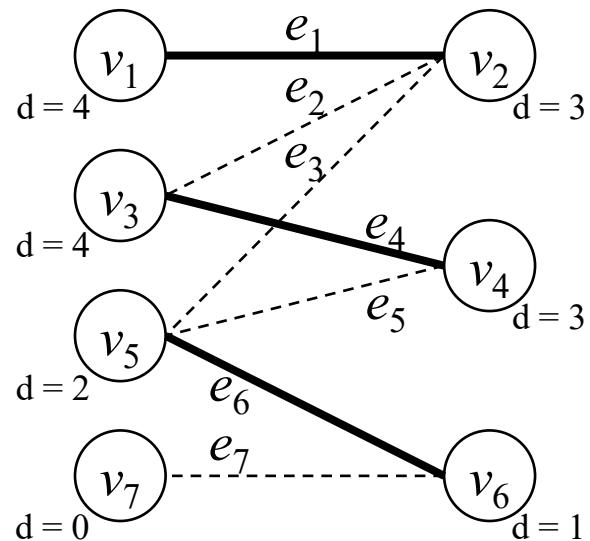


你理解这个算法了吗？

$$\begin{array}{c} Q \\ \hline Y' = \{\} \\ d' = \infty \end{array}$$

算法 5.5: HKBFS

```
输入: 二分图  $G = \langle X \cup Y, E \rangle$ , 匹配  $M$ , 队列  $Q$ 
初值: 顶点子集  $Y'$  初值为  $\emptyset$ ; 变量  $d'$  初值为  $\infty$ 
1 while  $Q$  非空 do
2    $v \leftarrow$  出队列 ( $Q$ );
3   if  $v.d > d'$  then
4     中止 while 循环;
5   else if  $v$  未被  $M$  饱和且  $v.d > 0$  then
6      $Y' \leftarrow Y' \cup \{v\}$ ;
7      $d' \leftarrow v.d$ ;
8   else
9     foreach  $(v, w) \in E$  do
10       if  $w.visited = \text{false}$  且 BFS 树中从根顶点到  $w$  的路是
11          $M$  交错路 then
12            $w.visited \leftarrow \text{true}$ ;
13            $w.d \leftarrow v.d + 1$ ;
14           入队列 ( $Q, w$ );
15 return  $Y'$ ;
```



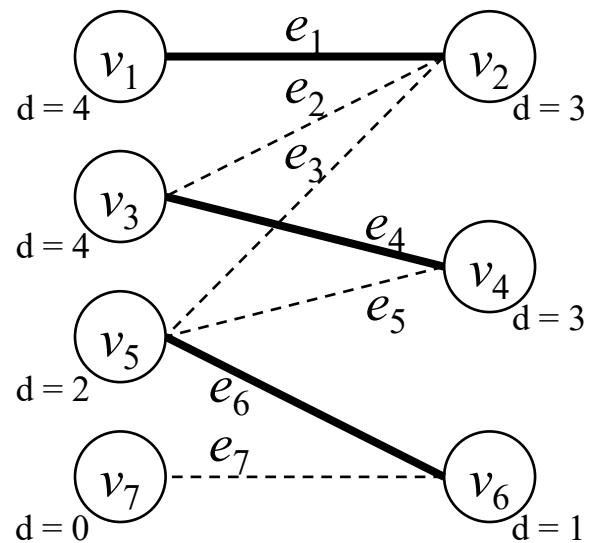
你理解这个算法了吗？

$$Y = \{\}$$

算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$
初值: 集合 M 初值为 \emptyset

```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HKPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6      $| M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```



你理解这个算法的时间复杂度了吗？

- 对于阶为 n 、边数为 m 的图，霍普克罗夫特-卡普算法的时间复杂度为 $O(\sqrt{n}(n+m))$

定理 5.3 随着 do-while 循环轮数的增加，变量 d' 的值严格单调增。

定理 5.4 对于阶为 n 的图，所有 do-while 循环的变量 d' 有至多 $2\lfloor\sqrt{\frac{n}{2}}\rfloor + 2$ 种值。

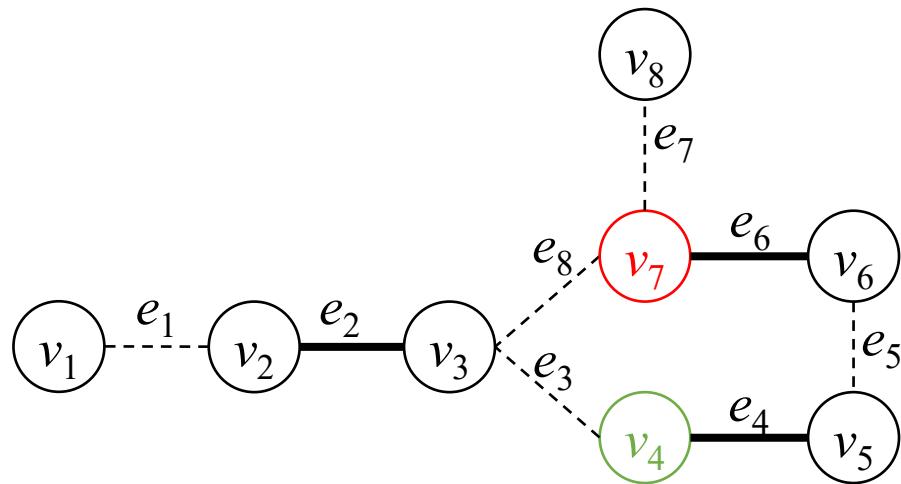
算法 5.3: 霍普克罗夫特-卡普算法

输入: 二分图 $G = \langle X \cup Y, E \rangle$

初值: 集合 M 初值为 \emptyset

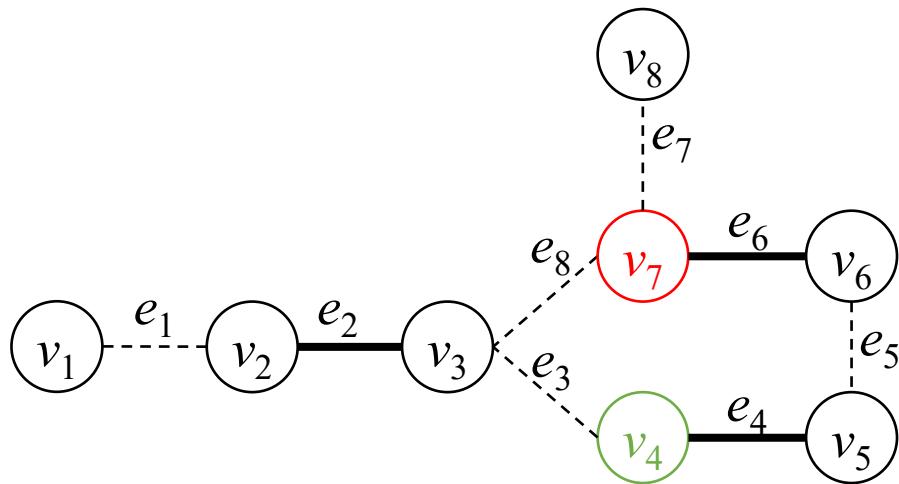
```
1 do
2    $Q \leftarrow \text{HKInit}(G, M);$ 
3    $Y' \leftarrow \text{HKBFS}(G, M, Q);$ 
4    $\mathcal{P} \leftarrow \text{HCPPaths}(G, Y');$ 
5   foreach  $P \in \mathcal{P}$  do
6     |  $M \leftarrow P$  经过的边的集合  $\Delta M;$ 
7   while  $\mathcal{P} \neq \emptyset;$ 
8   输出  $(M);$ 
```

为什么匈牙利/霍普克罗夫特-卡普算法只适用于二分图?



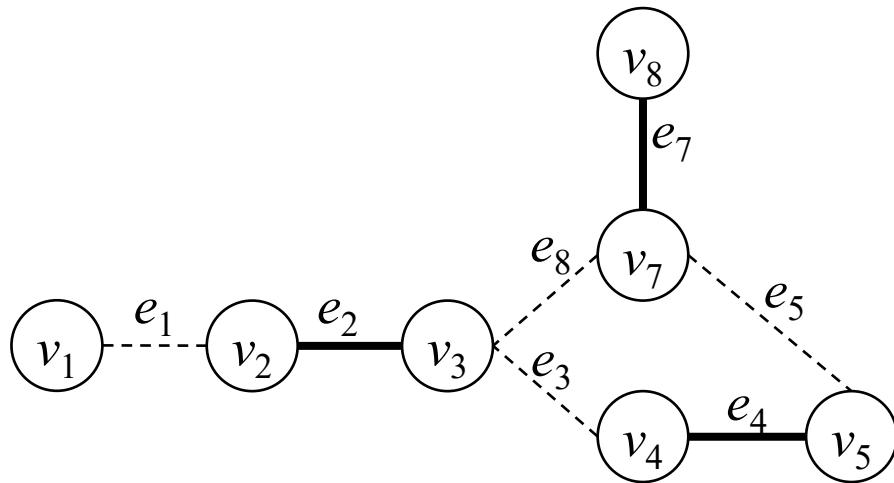
为什么匈牙利/霍普克罗夫特-卡普算法只适用于二分图?

思考题 5.16 为什么二分图不存在上述问题?



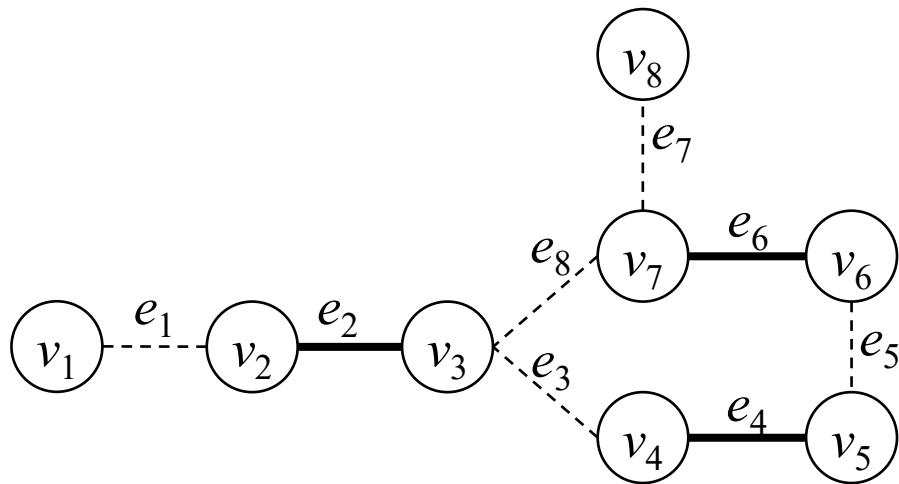
为什么匈牙利/霍普克罗夫特-卡普算法只适用于二分图?

思考题 5.16 为什么二分图不存在上述问题?

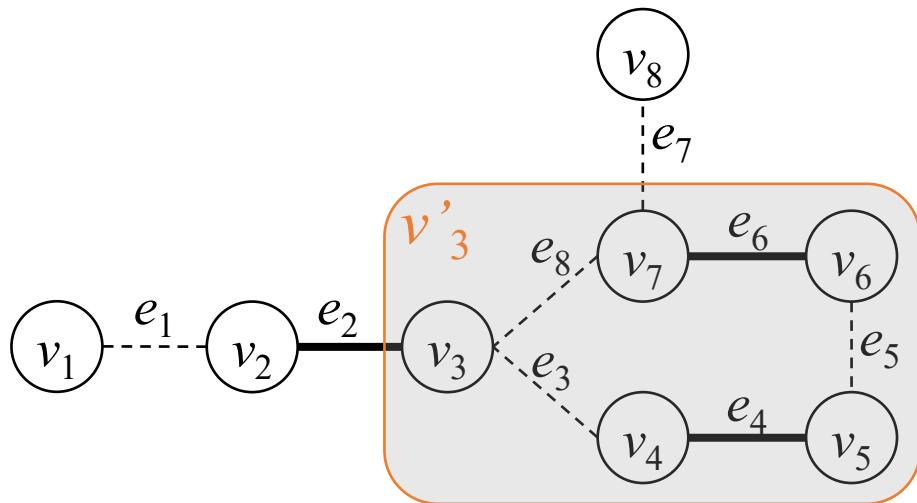


你能解释这些术语吗？

- 花
- 花梗
- 花托

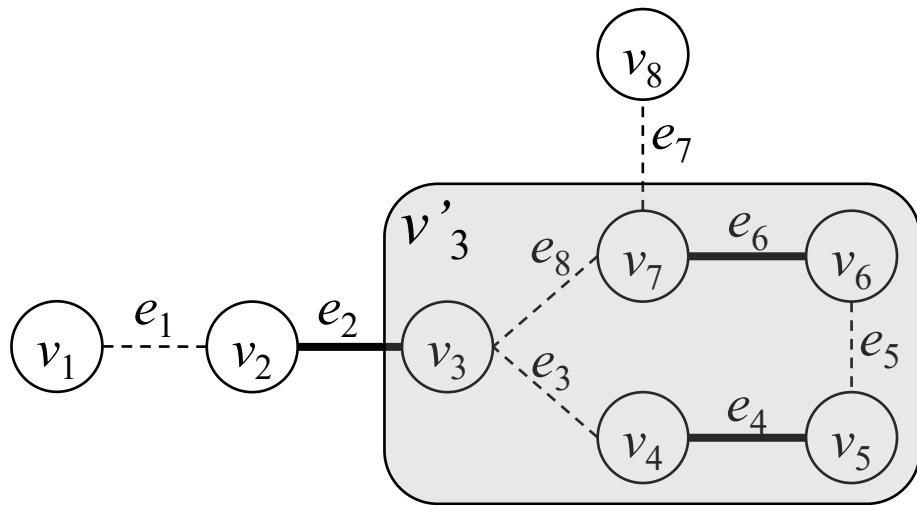


你理解花算法了吗？



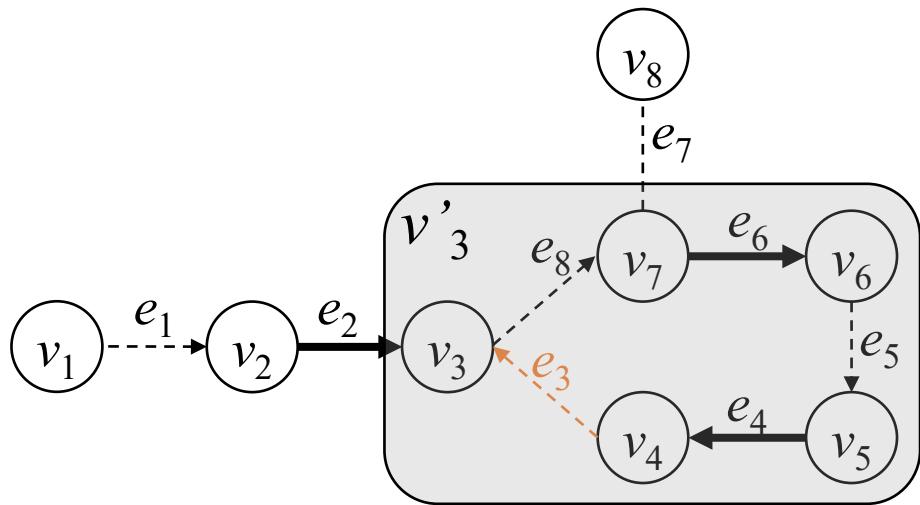
你理解花算法了吗？

思考题 5.17 (◊) 在花算法中，如何识别花？



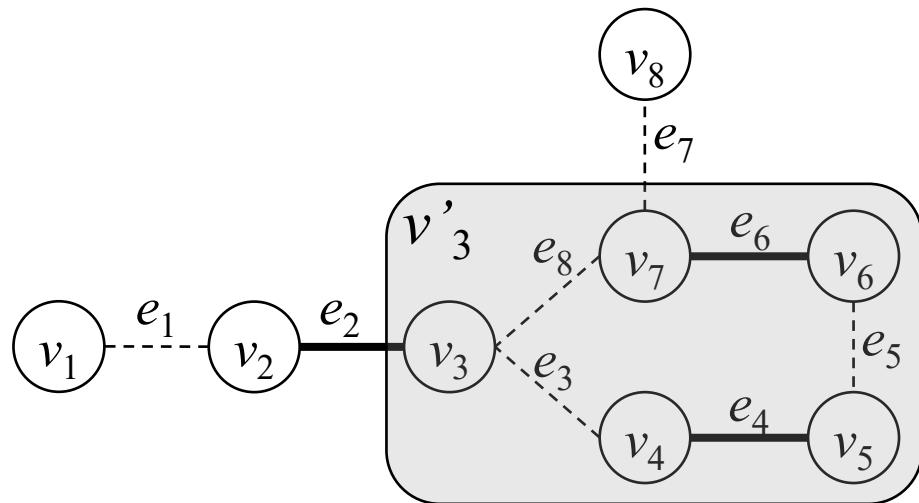
你理解花算法了吗？

思考题 5.17 (◊) 在花算法中，如何识别花？



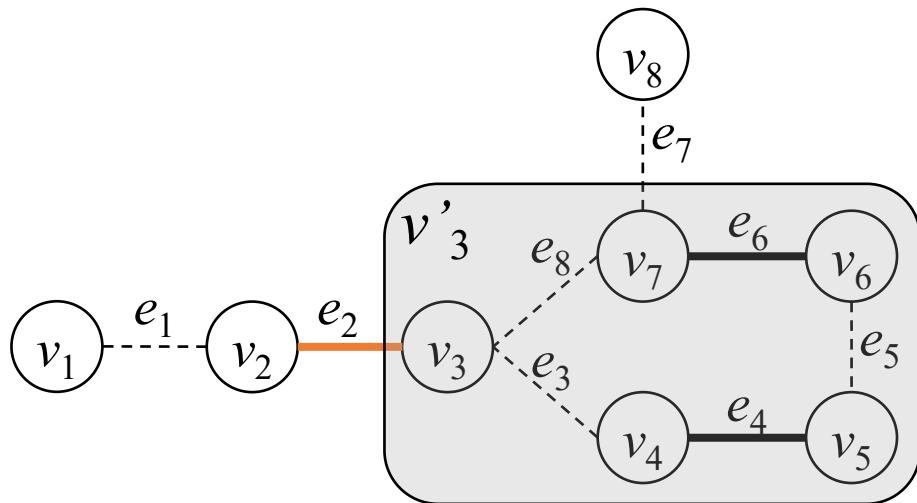
你理解花算法了吗？

思考题 5.18 在花算法运行过程中，所有花都会被收缩吗？



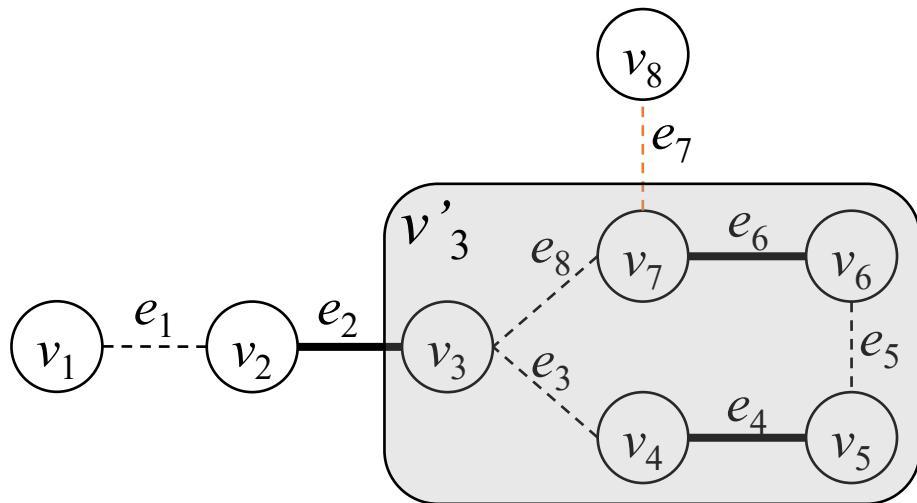
你理解花算法了吗？

思考题 5.19 (♦) 花梗的最后一条边有可能不在匹配 M 中吗？

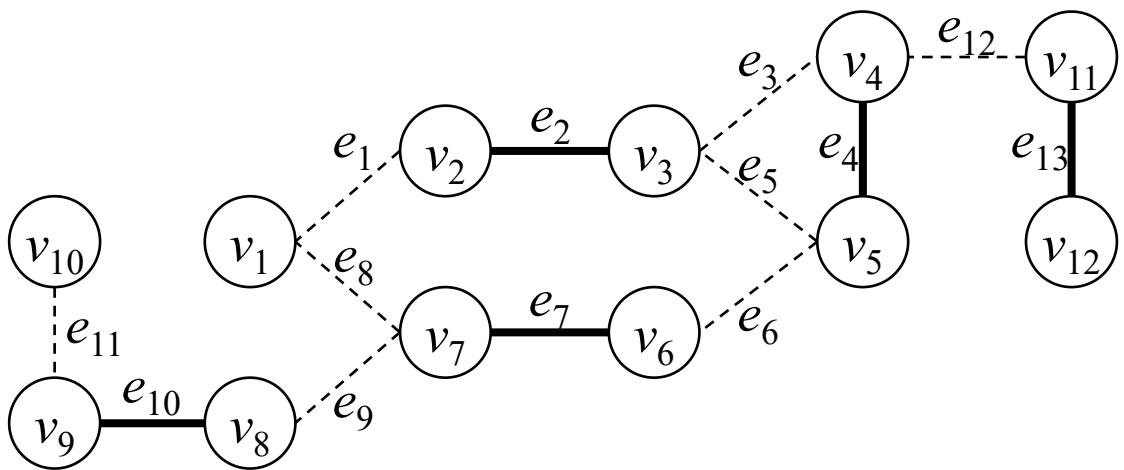


你理解花算法了吗？

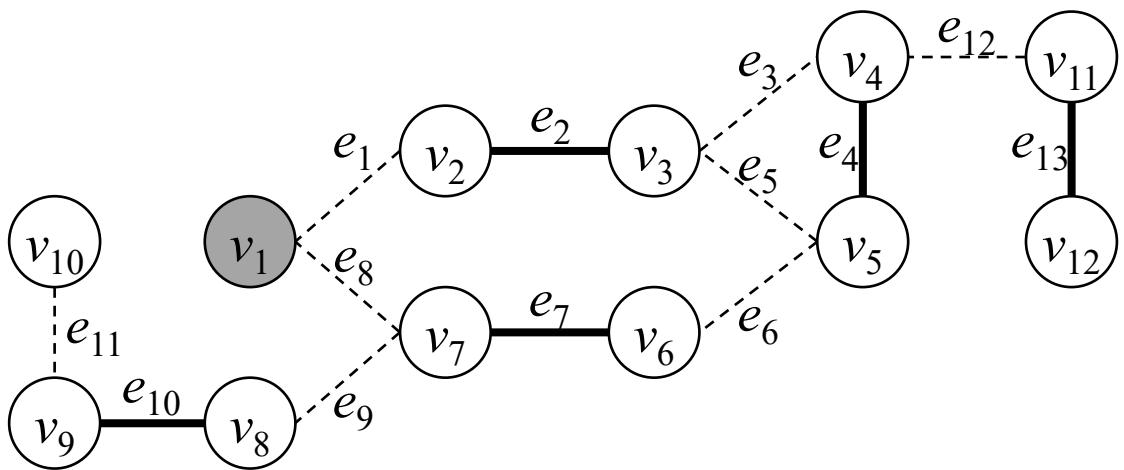
思考题 5.20 (♦) 花中顶点与花外顶点间的边（除花梗的最后一条边外），
有可能在匹配 M 中吗？



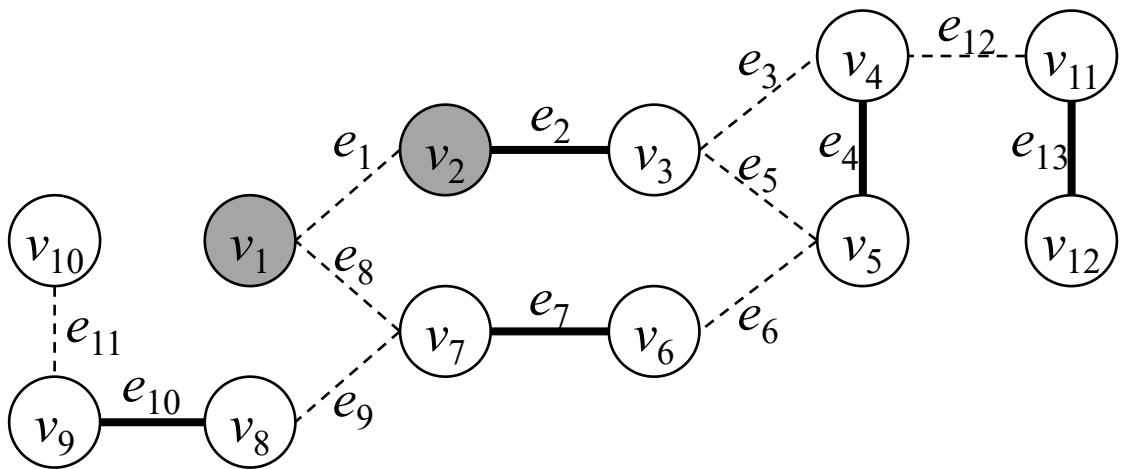
你理解花算法了吗？



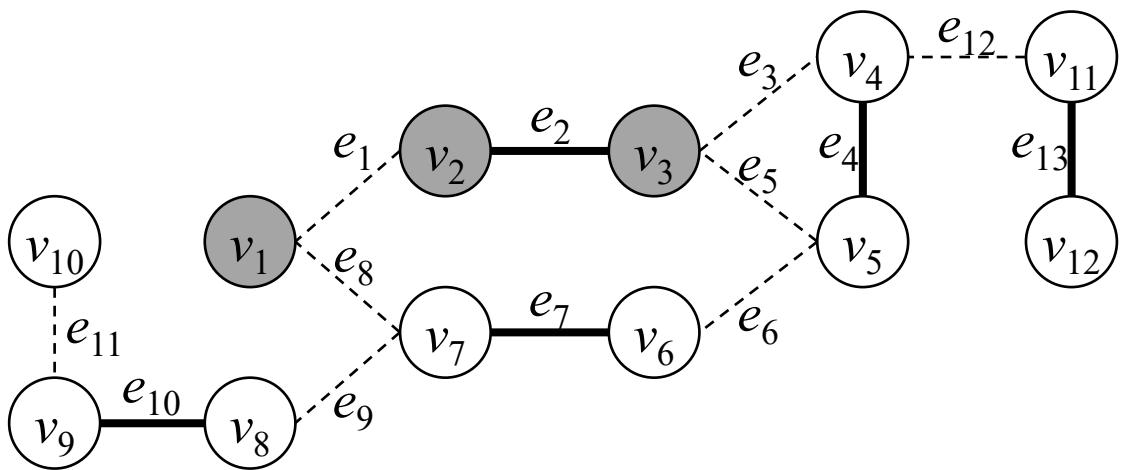
你理解花算法了吗？



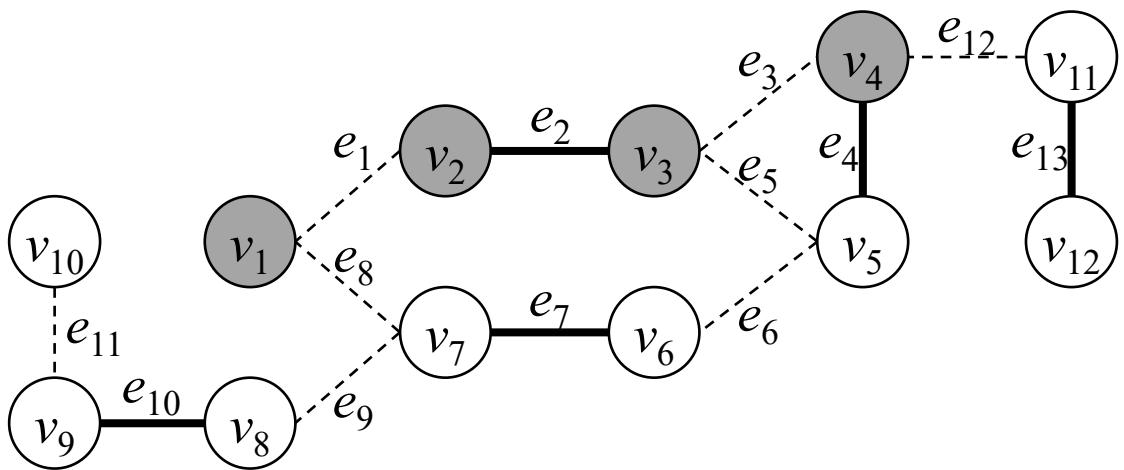
你理解花算法了吗？



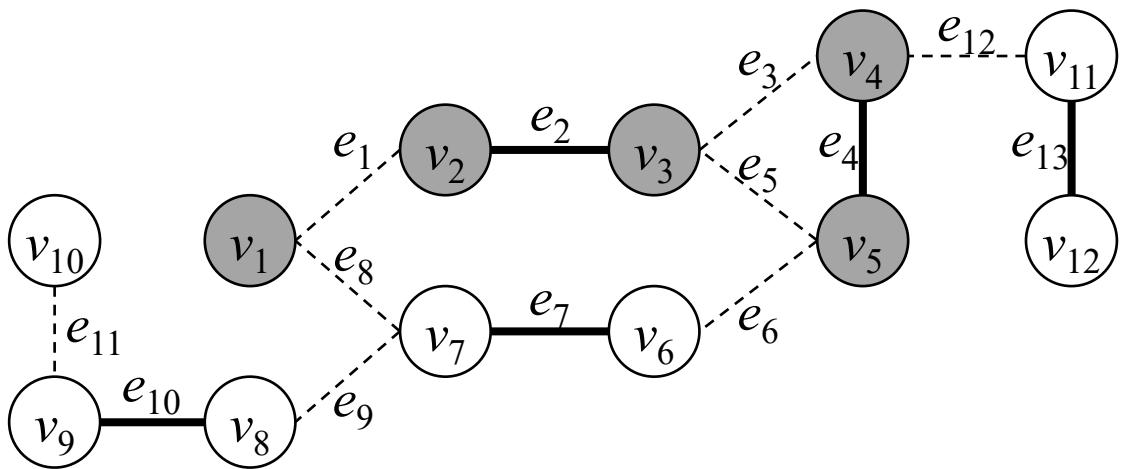
你理解花算法了吗？



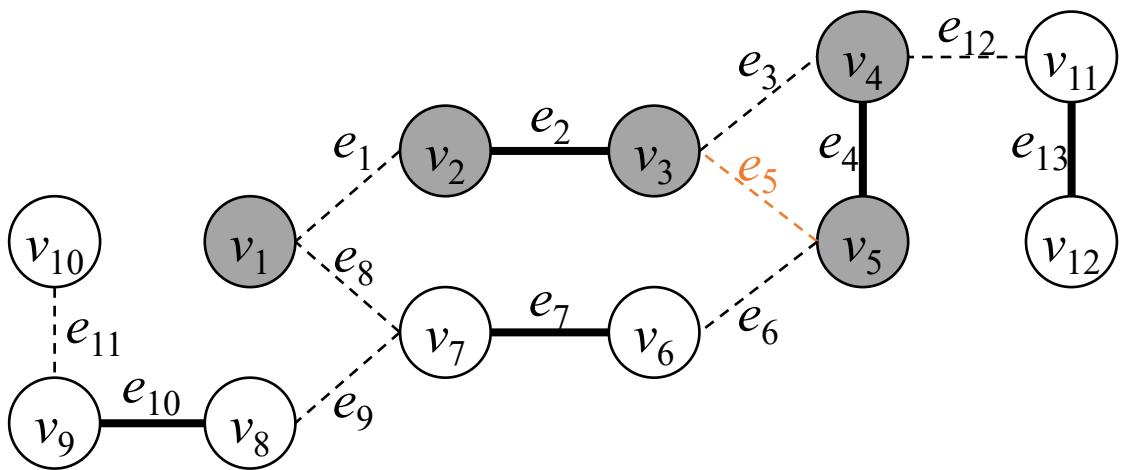
你理解花算法了吗？



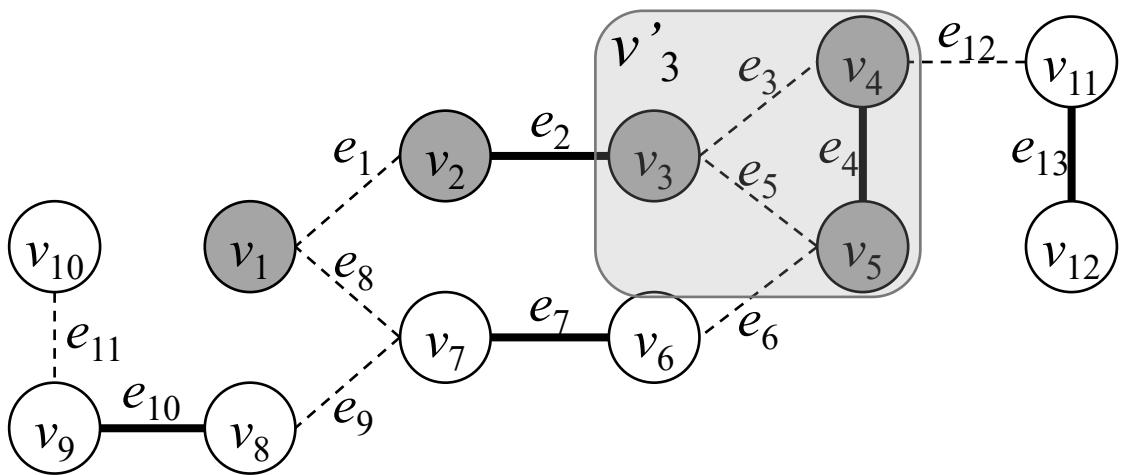
你理解花算法了吗？



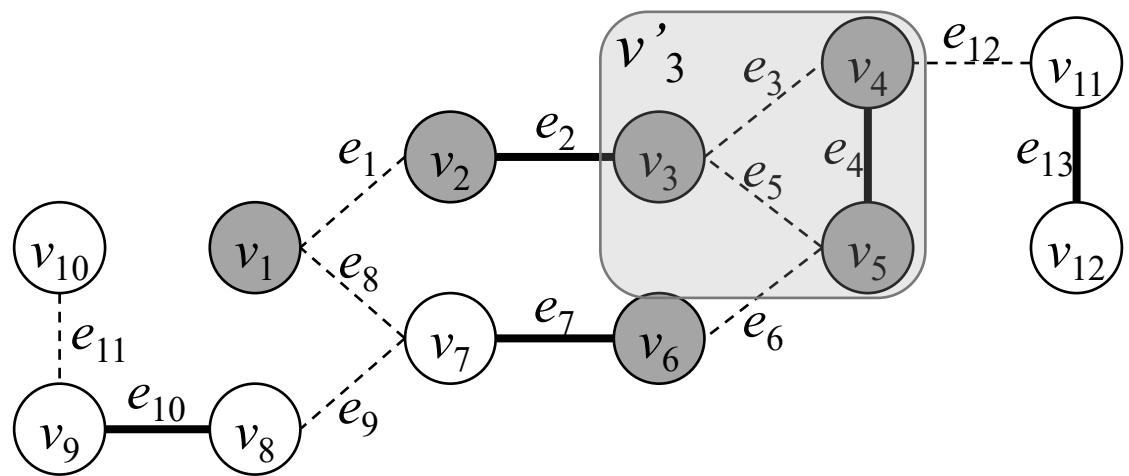
你理解花算法了吗？



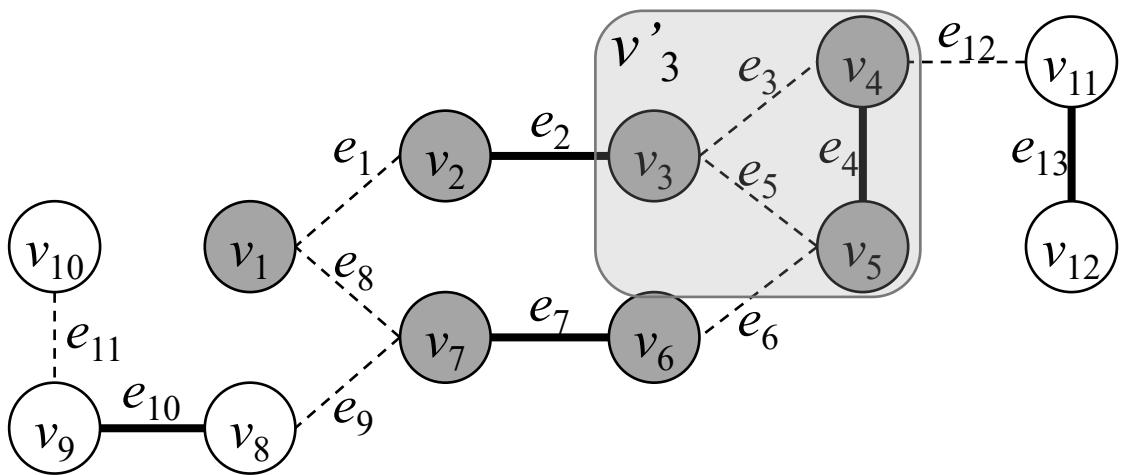
你理解花算法了吗？



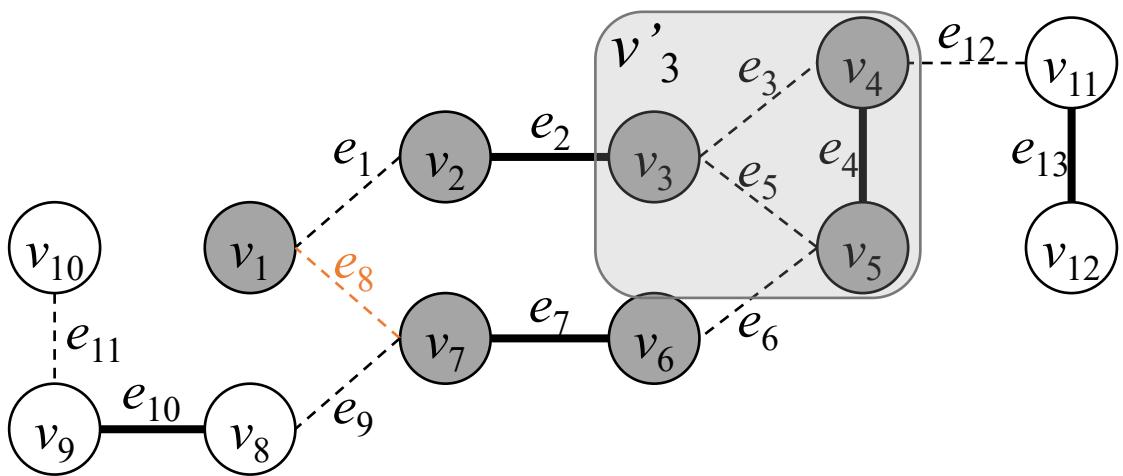
你理解花算法了吗？



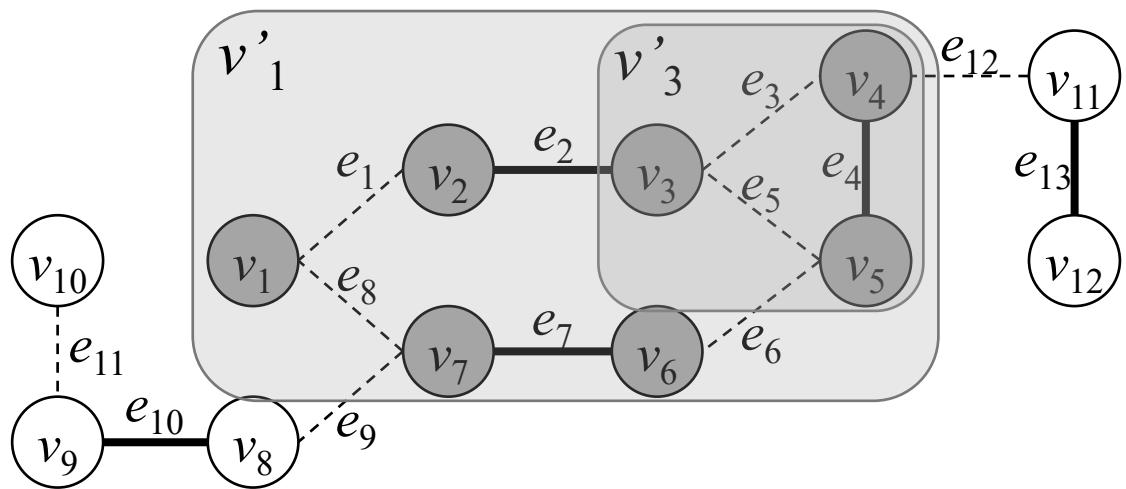
你理解花算法了吗？



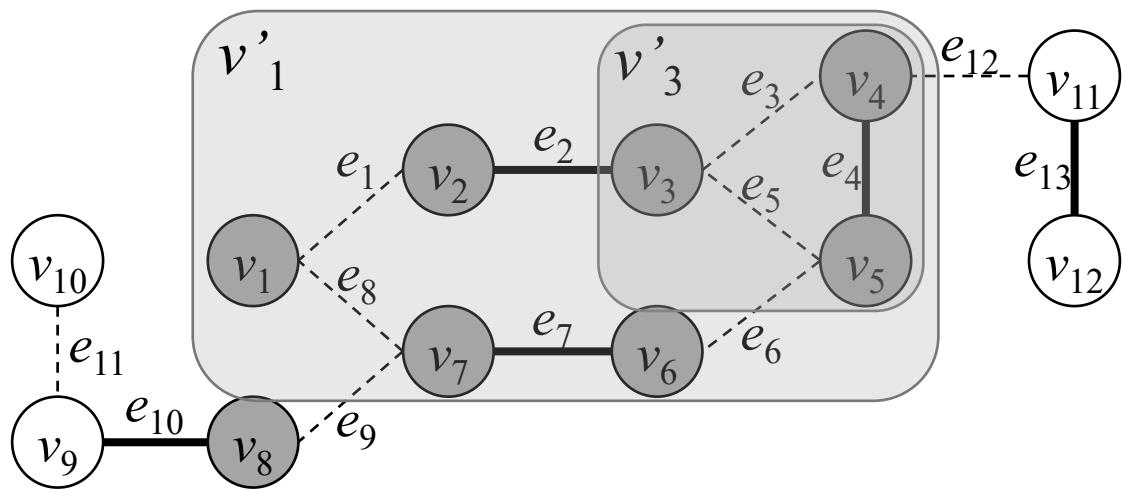
你理解花算法了吗？



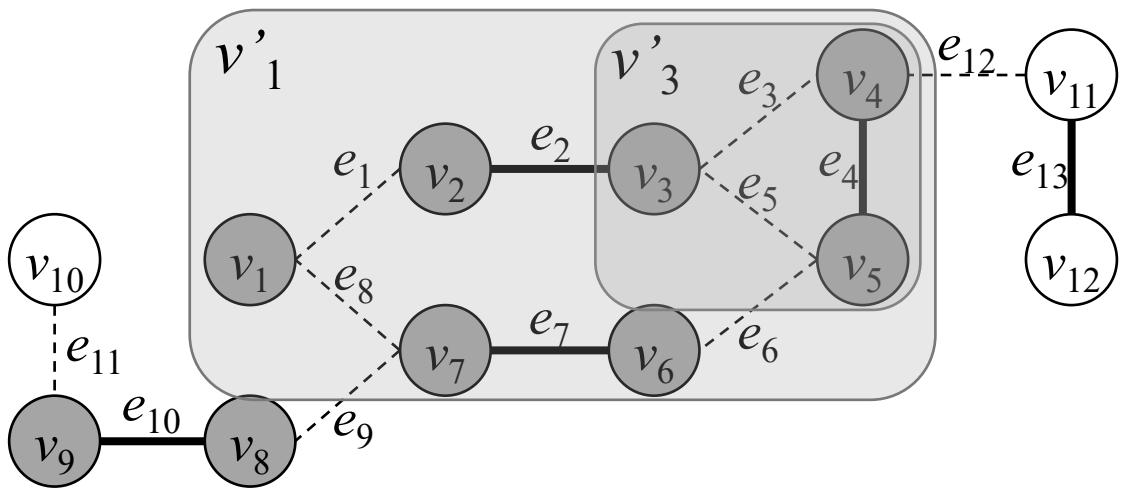
你理解花算法了吗？



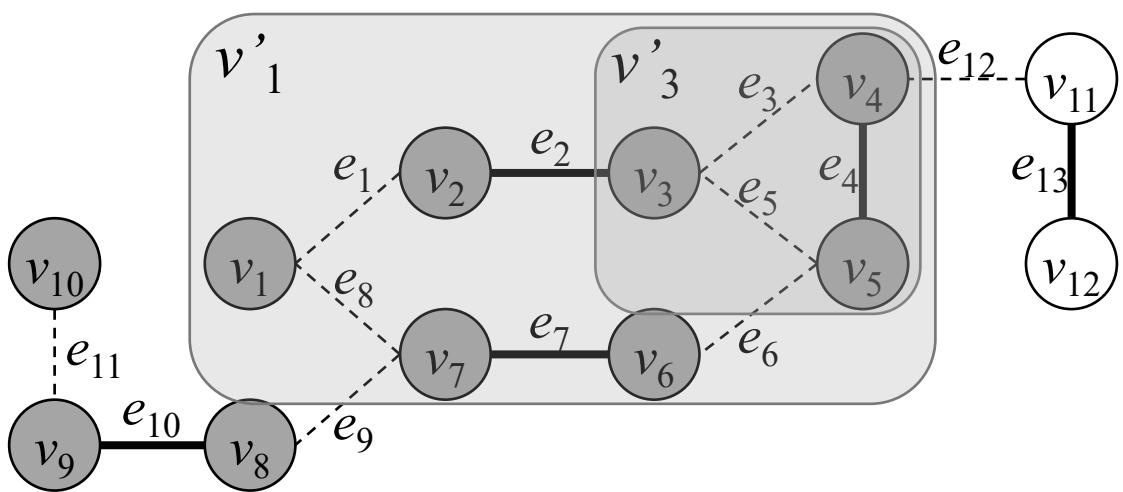
你理解花算法了吗？



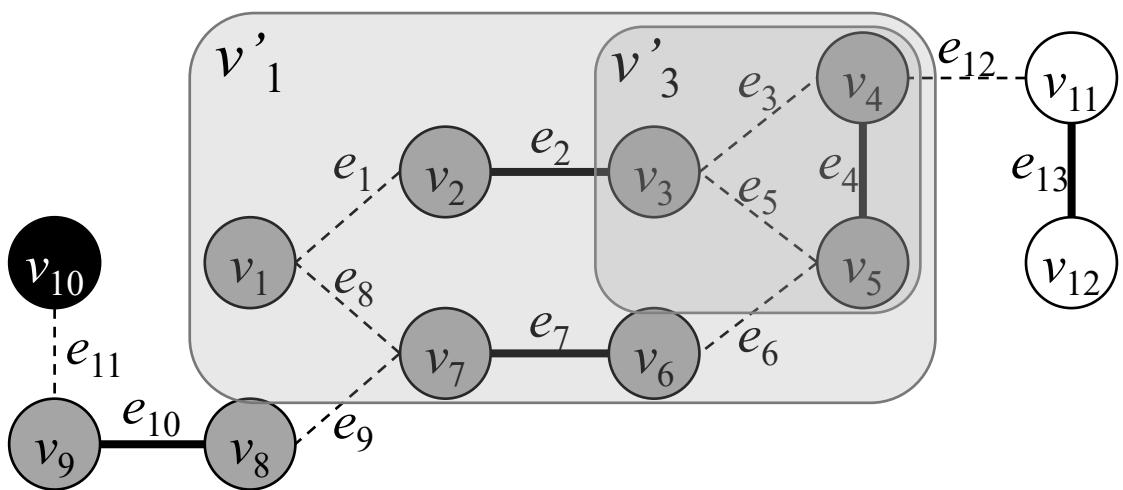
你理解花算法了吗？



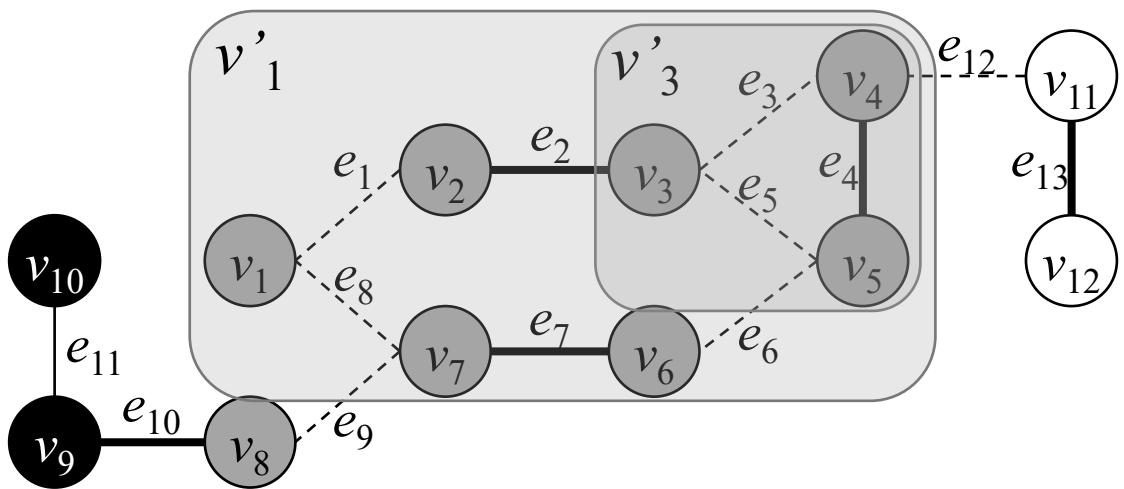
你理解花算法了吗？



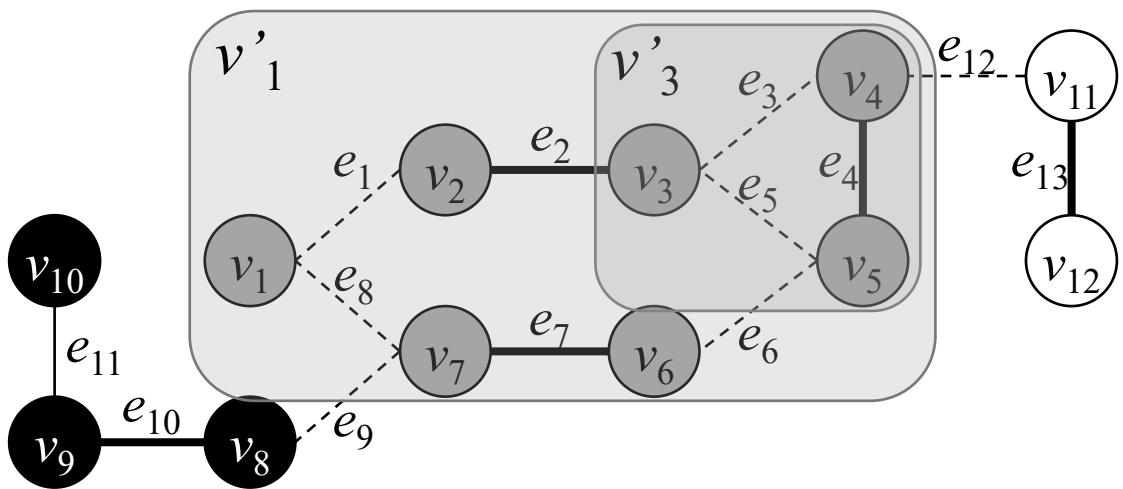
你理解花算法了吗？



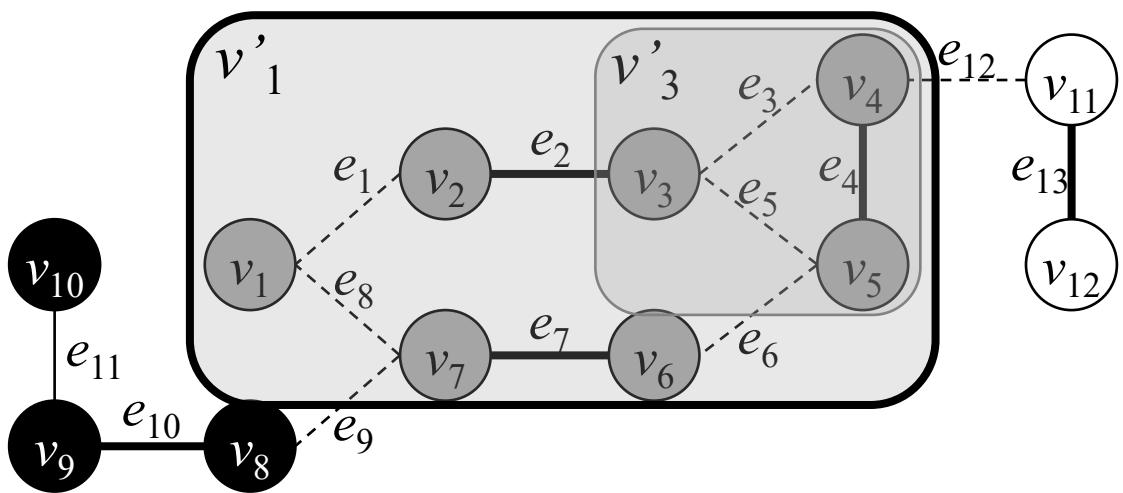
你理解花算法了吗？



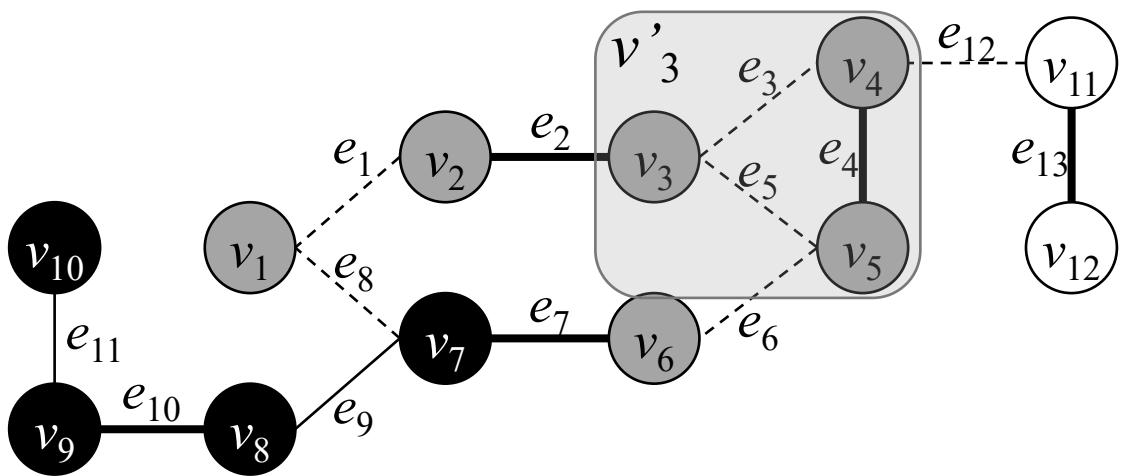
你理解花算法了吗？



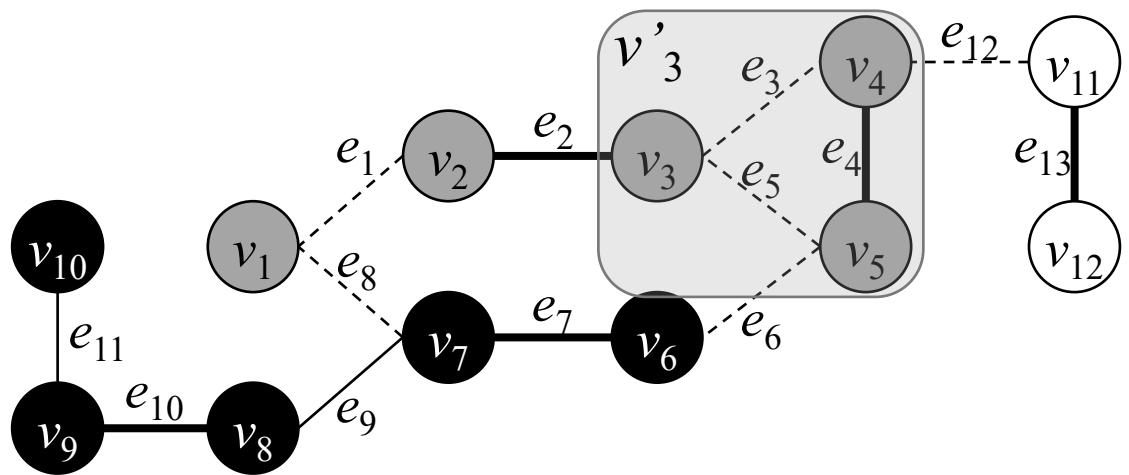
你理解花算法了吗？



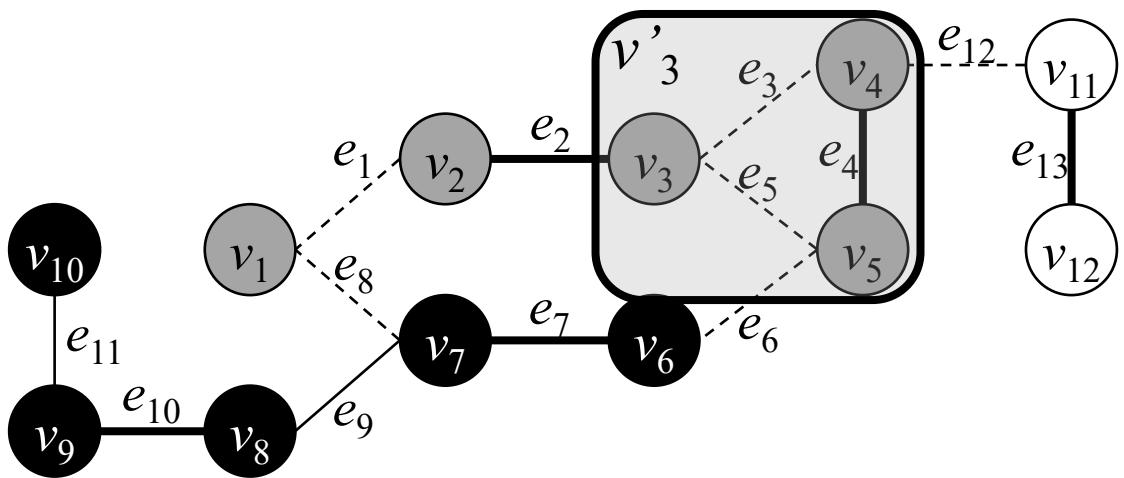
你理解花算法了吗？



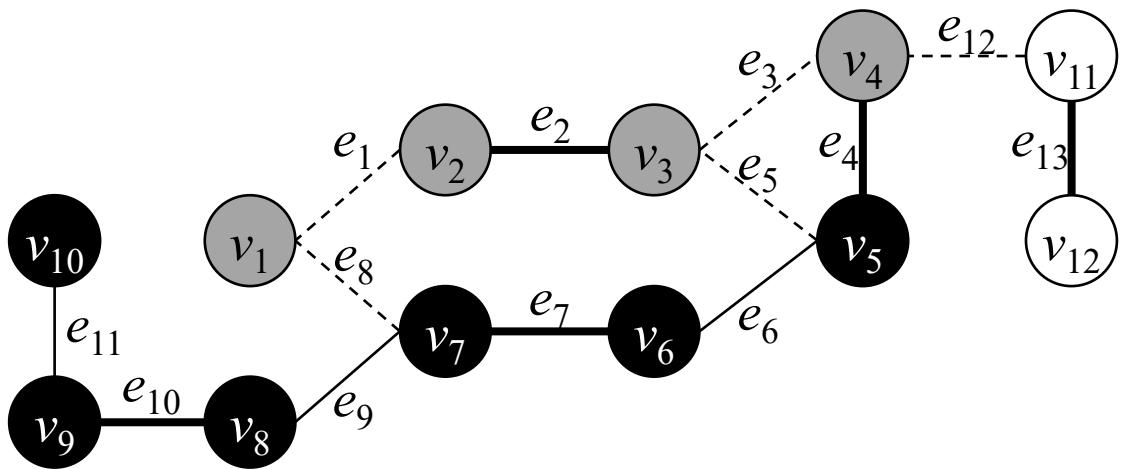
你理解花算法了吗？



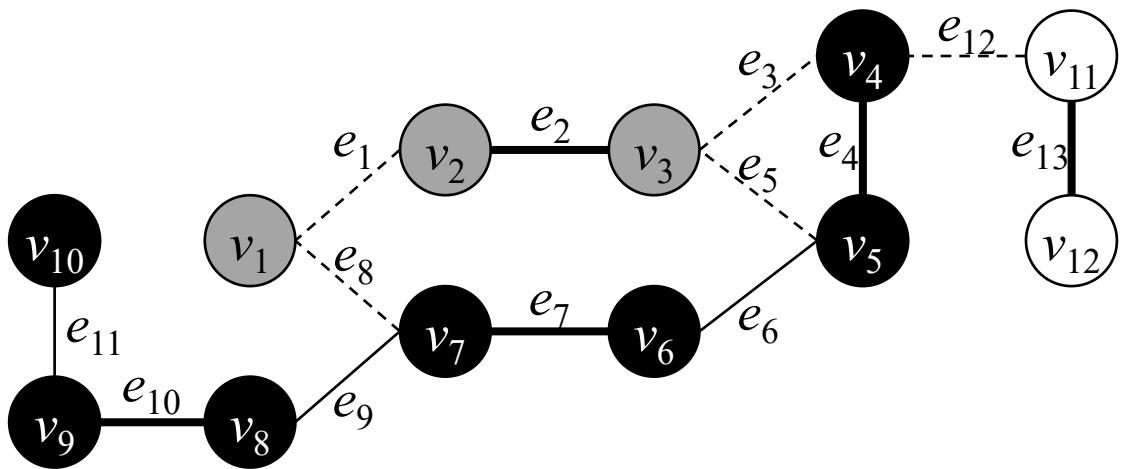
你理解花算法了吗？



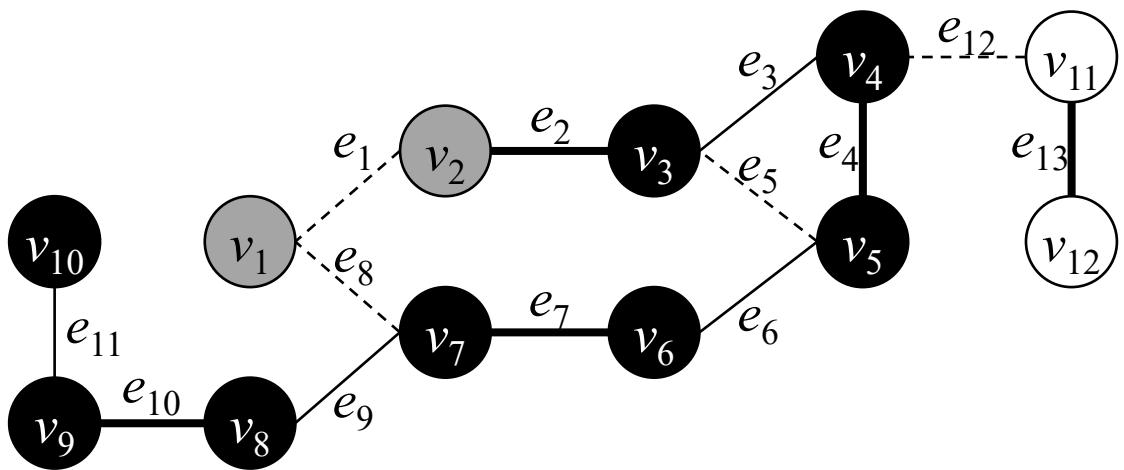
你理解花算法了吗？



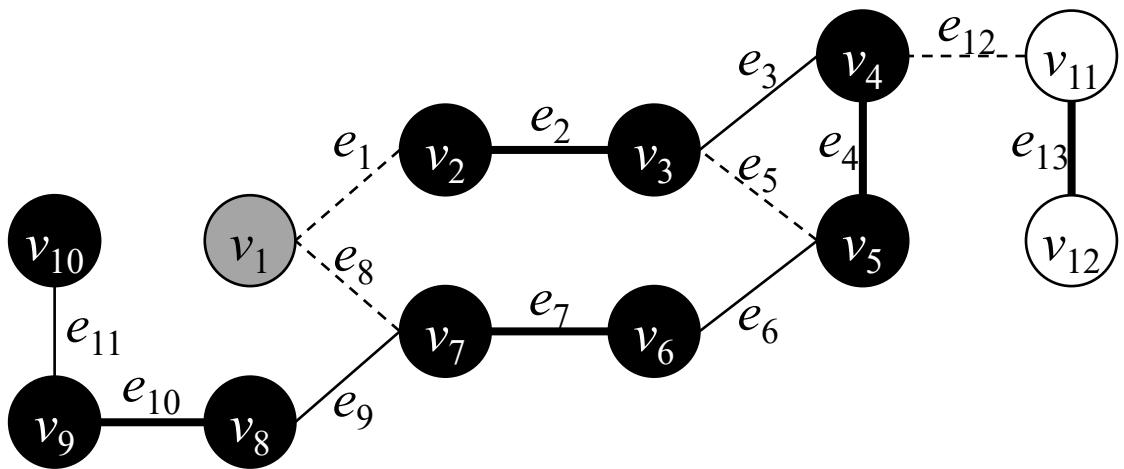
你理解花算法了吗？



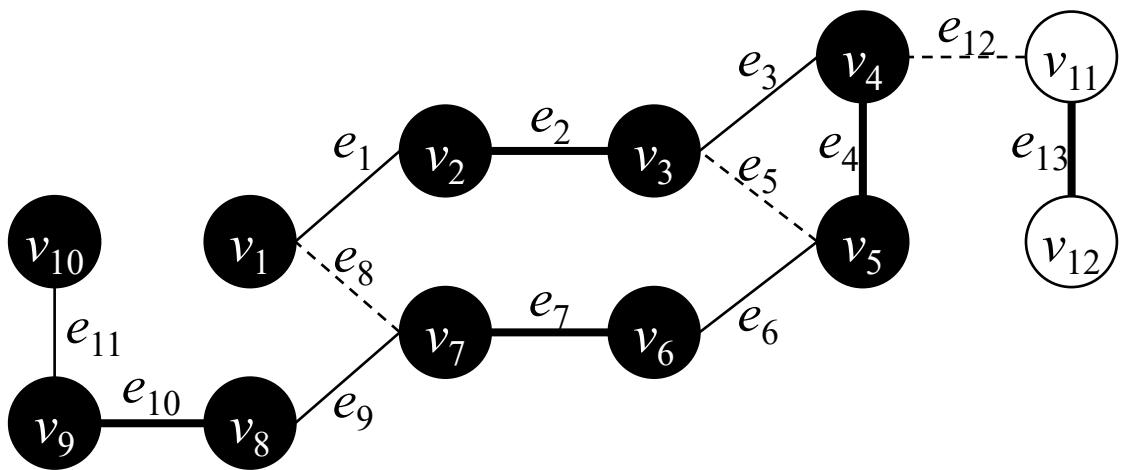
你理解花算法了吗？



你理解花算法了吗？

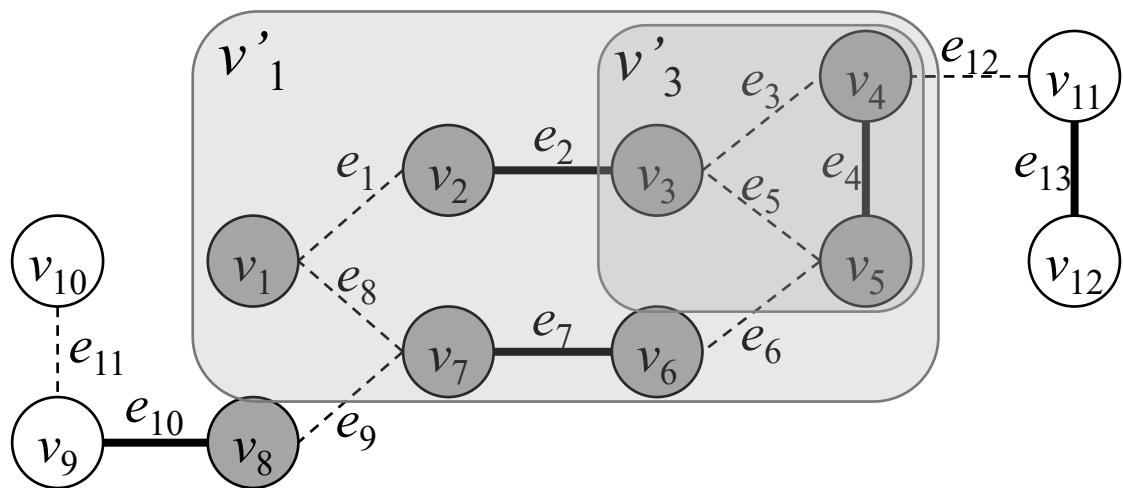


你理解花算法了吗？



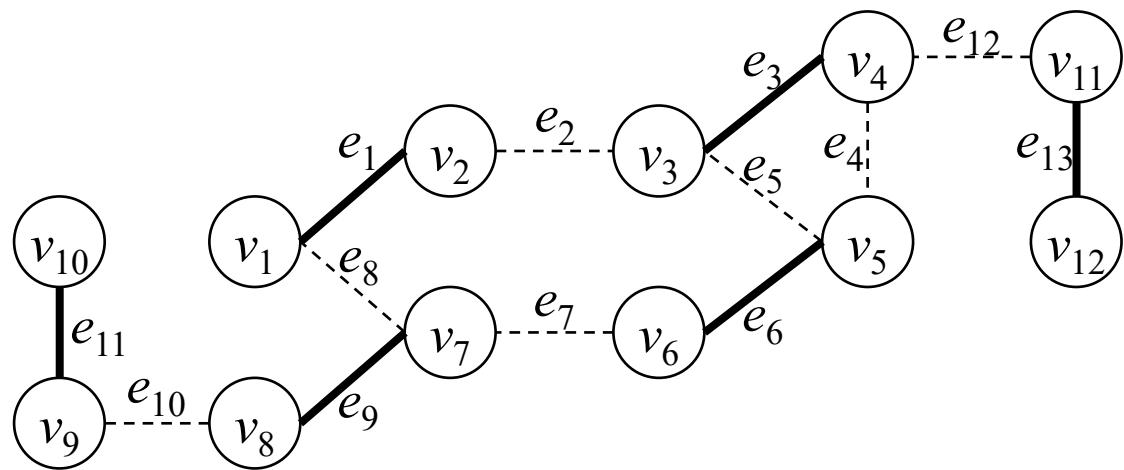
你理解花算法的时间复杂度了吗？

- 对于阶为 n 、边数为 m 的图，花算法的时间复杂度 $O(n^2(n + m))$
- 若采用合适的数据结构来处理花的收缩，则时间复杂度可降为 $O(n^3)$



你能解释这些术语吗？

- 完美匹配



你能解释这些术语吗？

■ 完美匹配

思考题 5.21 完美匹配一定是最小匹配吗？最小匹配一定是完美匹配吗？

思考题 5.22 每个图都有完美匹配吗？完美匹配存在的必要条件有哪些？

思考题 5.24 阶为 n 的图的完美匹配有多少条边？

你能解释这些术语吗？

■ 完美匹配

思考题 5.25 偶数阶完全图 K_{2n} 一定有完美匹配吗？若有，则最多有多少个两两不相交的完美匹配？

思考题 5.26 偶数阶非空正则图一定有完美匹配吗？

思考题 5.27 偶数阶欧拉图和偶数阶哈密尔顿图一定有完美匹配吗？

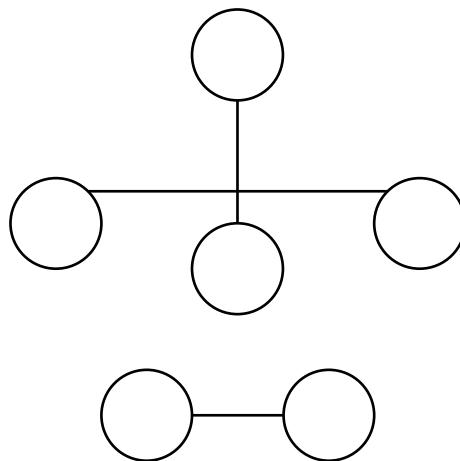
你能解释这些术语吗？

■ 完美匹配

思考题 5.25 偶数阶完全图 K_{2n} 一定有完美匹配吗？若有，则最多有多少个两两不相交的完美匹配？

思考题 5.26 偶数阶非空正则图一定有完美匹配吗？

思考题 5.27 偶数阶欧拉图和偶数阶哈密尔顿图一定有完美匹配吗？



你能解释这些术语吗？

■ 完美匹配

思考题 5.28 图的两个完美匹配的对称差的边导出子图的每个连通分支的结构有什么特征？

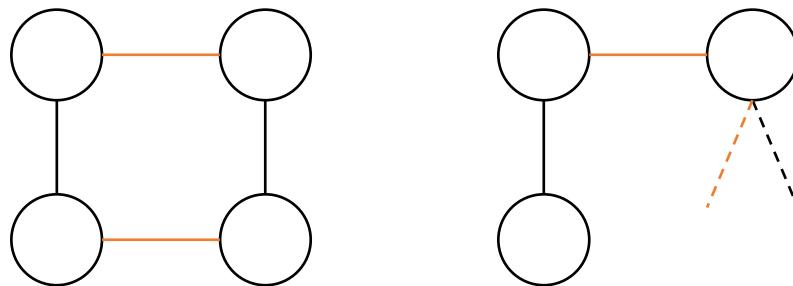
思考题 5.29 树一定有完美匹配吗？若有，则最多有多少个完美匹配？

你能解释这些术语吗？

■ 完美匹配

思考题 5.28 图的两个完美匹配的对称差的边导出子图的每个连通分支的结构有什么特征？

思考题 5.29 树一定有完美匹配吗？若有，则最多有多少个完美匹配？



请证明下述定理

定理 5.5 (霍尔定理) 对于二分图 $G = \langle X \cup Y, E \rangle$, G 有饱和顶点子集 X 中所有顶点的匹配当且仅当对于任意顶点子集 $S \subseteq X$, $|N(S)| \geq |S|$ 。

证明. 先证必要性。根据饱和顶点子集 X 中所有顶点的匹配, 顶点子集 S 中的顶点与邻点集 $N(S)$ 中的某些顶点一一对应, 因此, $|N(S)| \geq |S|$ 。

再证充分性。采用反证法, 假设顶点 $u \in X$ 未被图 G 的最大匹配 M 饱和, 则以 u 为起点的所有 M 交错路经过的顶点的集合记作 R , 如图 B.11 所示。

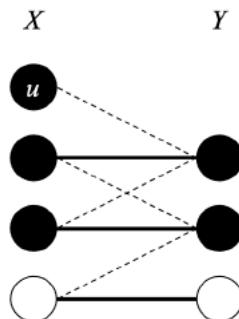


图 B.11 思考题 5.30 证明的示意图

粗实线—最大匹配 M 黑色顶点—以顶点 u 为起点的所有 M 交错路经过的顶点的集合 R

请证明下述定理

定理 5.5 (霍尔定理) 对于二分图 $G = \langle X \cup Y, E \rangle$, G 有饱和顶点子集 X 中所有顶点的匹配当且仅当对于任意顶点子集 $S \subseteq X$, $|N(S)| \geq |S|$ 。

接下来, 首先证明: $|(R \cap X) \setminus \{u\}| = |R \cap Y|$ 。一方面, 上述 M 交错路经过匹配 M 中不同的边从集合 $R \cap Y$ 中的顶点到集合 $(R \cap X) \setminus \{u\}$ 中的每个顶点, 这些边的集合记作匹配 $M' \subseteq M$, 则 $|M'| = |(R \cap X) \setminus \{u\}|$ 。另一方面, 上述 M 交错路经过不在 M 中的边从集合 $R \cap X$ 中的顶点到集合 $R \cap Y$ 中的每个顶点, 且 M 是最大匹配, 即没有 M 增广路, 因此, $R \cap Y$ 中的每个顶点被 M' 中不同的边饱和, $|M'| = |R \cap Y|$ 。综上, $|(R \cap X) \setminus \{u\}| = |R \cap Y|$ 。

X Y

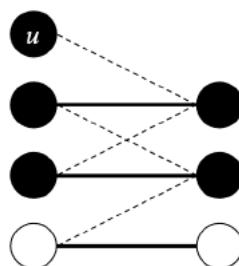


图 B.11 思考题 5.30 证明的示意图

粗实线—最大匹配 M 黑色顶点—以顶点 u 为起点的所有 M 交错路经过的顶点的集合 R

请证明下述定理

定理 5.5 (霍尔定理) 对于二分图 $G = \langle X \cup Y, E \rangle$, G 有饱和顶点子集 X 中所有顶点的匹配当且仅当对于任意顶点子集 $S \subseteq X$, $|N(S)| \geq |S|$ 。

其次证明: $N(R \cap X) = R \cap Y$ 。由上述证明可知 $R \cap Y \subseteq N(R \cap X)$, 只需证明 $N(R \cap X) \subseteq R \cap Y$ 。采用反证法, 假设存在顶点 $y \in N(R \cap X)$ 满足 $y \notin R \cap Y$, 则 y 在集合 $R \cap X$ 中的任意一个邻点记作 x 。接下来证明: 边 $(x, y) \notin M$ 。

若 $x = u$, 则由于顶点 u 未被匹配 M 饱和, 因此, $(x, y) \notin M$ 。

若 $x \in (R \cap X) \setminus \{u\}$, 则由于顶点 x 已通过匹配 M' 中的一条边与集合 $R \cap Y$ 中的某个顶点相邻, 因此, $(x, y) \notin M$ 。

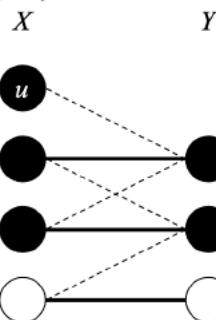


图 B.11 思考题 5.30 证明的示意图

粗实线—最大匹配 M 黑色顶点—以顶点 u 为起点的所有 M 交错路经过的顶点的集合 R

请证明下述定理

定理 5.5 (霍尔定理) 对于二分图 $G = \langle X \cup Y, E \rangle$, G 有饱和顶点子集 X 中所有顶点的匹配当且仅当对于任意顶点子集 $S \subseteq X$, $|N(S)| \geq |S|$ 。

既然边 $(x, y) \notin M$, 从顶点 u 到 x 的 M 交错路和边 (x, y) 拼接形成一条 M 交错路, 因此, $y \in R \cap Y$, 与 $y \notin R \cap Y$ 矛盾。

综上,

$$\begin{aligned} |N(R \cap X)| &= |R \cap Y| \\ &= |(R \cap X) \setminus \{u\}| \\ &= |R \cap X| - 1 \\ &< |R \cap X|. \end{aligned} \tag{B.12}$$

与 $|N(R \cap X)| \geq |R \cap X|$ 矛盾。 \square

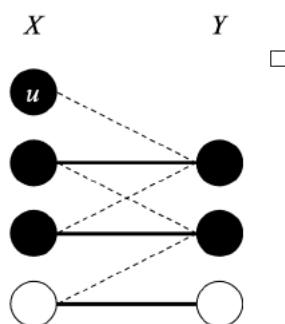


图 B.11 思考题 5.30 证明的示意图

粗实线—最大匹配 M 黑色顶点—以顶点 u 为起点的所有 M 交错路经过的顶点的集合 R

霍尔定理的应用

定理 5.5 (霍尔定理) 对于二分图 $G = \langle X \cup Y, E \rangle$, G 有饱和顶点子集 X 中所有顶点的匹配当且仅当对于任意顶点子集 $S \subseteq X$, $|N(S)| \geq |S|$ 。

思考题 5.31 非空 r 正则二分图一定有完美匹配吗? 若有, 则最多有多少个两两不相交的完美匹配?

思考题 5.32 完全二分图 $K_{n,n}$ 一定有完美匹配吗? 若有, 则最多有多少个两两不相交的完美匹配?

塔特定理的应用

定理 5.6 (塔特定理) 对于图 $G = \langle V, E \rangle$, G 有完美匹配当且仅当对于任意顶点子集 $S \subseteq V$, $o(G - S) \leq |S|$ 。

思考题 5.33 (◊) 2 边连通的 3 正则图一定有完美匹配吗?

思考题 5.34 偶数阶 $(k - 1)$ 边连通的 k 正则图一定有完美匹配吗?