



グラフの探索2

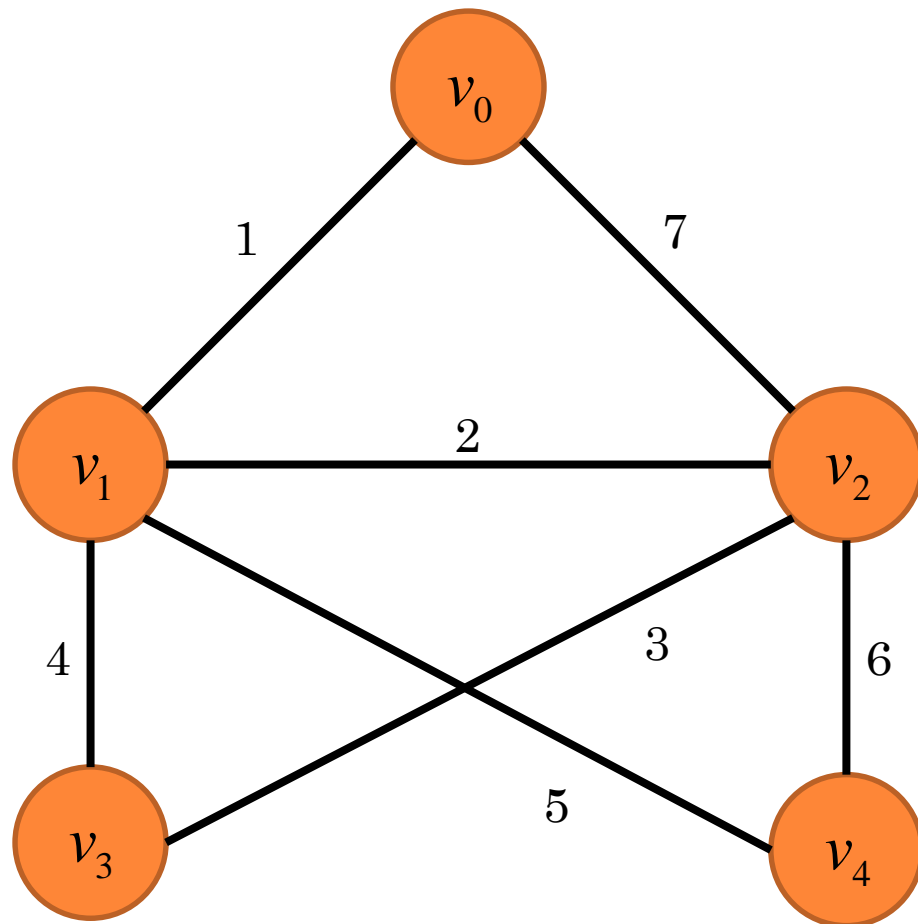
EULER閉路とHAMILTON閉路

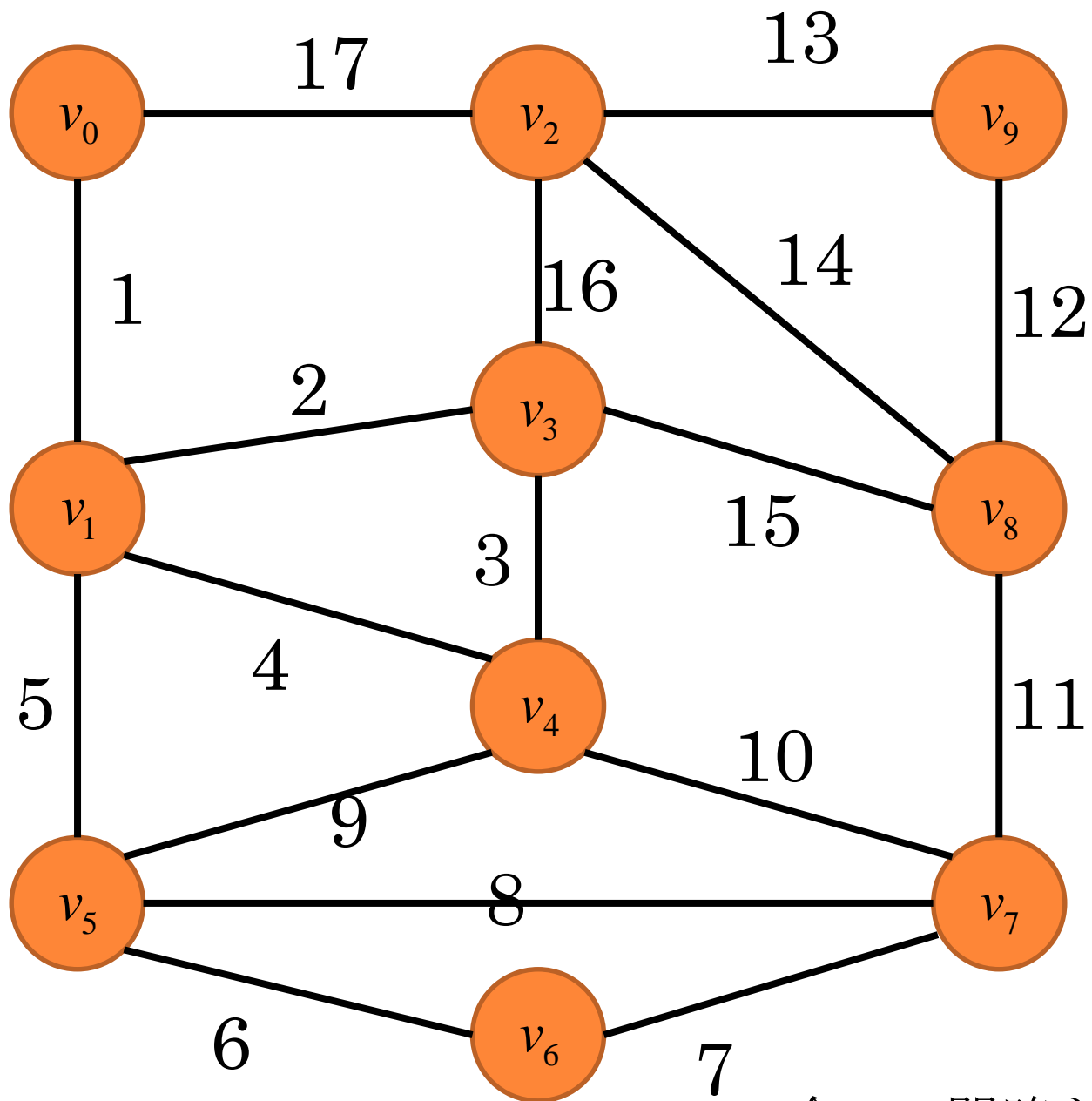
EULER閉路(EULER CIRCLE)

- 一筆書き
- 無向グラフが対象
- 全ての弧を一度ずつ経由して始点に戻る道
- 全ての点の次数が偶数
 - 次数が奇数だったらどうなる？



簡単なEULER閉路の例





全ての閉路を見つける方法は？

EULER閉路の列挙の方針

- 全てのEuler閉路を見つける
- 使用した弧の列を管理する
- 深さ優先で、一つの閉路を見つける
 - 使った弧の一覧を保持
- 分岐点まで戻って、他の閉路を見つける
 - 戻るたびに、対応する弧(一つの閉路の構成要素)を一覧から削除
- 上記を繰り返す



EULER閉路の列挙アルゴリズム

```
search(v){
  if ((v == r) ^ (|A| == |AEuler|)) {
    見つけた Euler 閉路を保存
  } else {
    forall (a ∈ δv) {
      if (a ∉ AEuler) {
        AEuler に a を追加
        w は a の反対の終端
        search(w)
        AEuler から a を削除
      }
    }
  }
}
```

A_{Euler} : 既に経由した弧の列

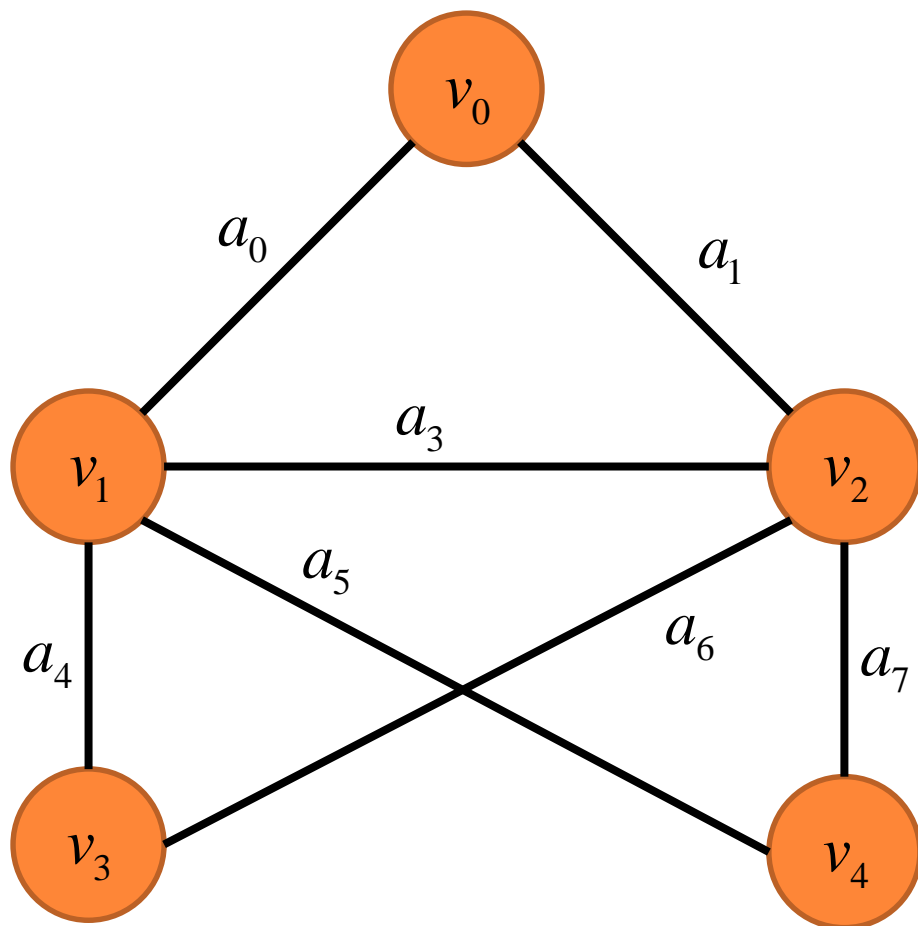
初期値は $A_{\text{Euler}} = \emptyset$

r : 始点

$|A|$: 弧の数



列挙の例

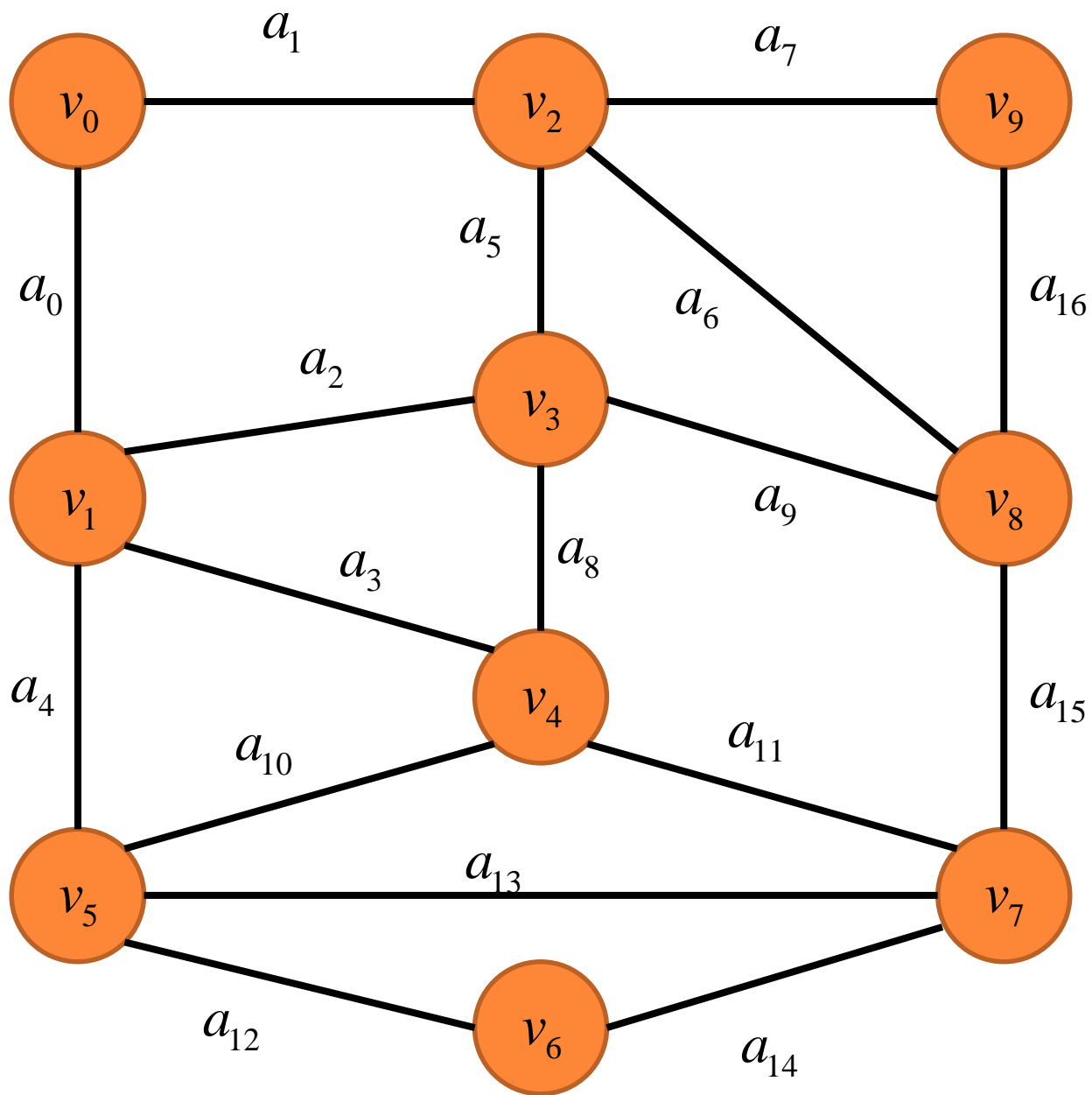


探索の経過

a_0	a_3	a_1	×				
		a_6	a_4	a_5	a_7	a_1	○
		a_7	a_5	a_4	a_6	a_1	○
	a_4	a_6	a_1	×			
			a_3	a_5	a_7	a_1	○
			a_7	a_5	a_3	a_1	○
	a_5	a_7	a_3	a_4	a_6	a_1	○
			a_6	a_4	a_3	a_1	○

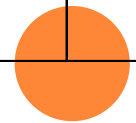
以下省略





a_0	a_2	a_5	a_1	×												
			a_6	a_9	a_8	a_3	a_4	a_{10}	a_{11}	a_{13}	a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1
								a_{12}	a_{14}	a_{13}	a_{10}	a_{11}	a_{15}	a_{16}	a_7	a_1
						a_{10}	a_4	a_3	a_{11}	a_{13}	a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1
							a_{12}	a_{14}	a_{11}	a_3	a_4	a_{13}	a_{15}	a_{16}	a_7	a_1
								a_{13}	a_4	a_3	a_{11}	a_{15}	a_{16}	a_7	a_1	
								a_{15}	a_{16}	a_7	a_1	×				
							a_{13}	a_{11}	a_3	a_4	a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1
								a_{14}	a_{12}	a_4	a_3	a_{11}	a_{15}	a_{16}	a_7	a_1
						a_{11}	a_{13}	a_4	a_3	a_{10}	a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1
								a_{10}	a_3	a_4	a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1
								a_{12}	a_{14}	a_{15}	a_{16}	a_7	a_1	×		
							a_{14}	a_{12}	a_4	a_3	a_{10}	a_{13}	a_{15}	a_{14}	a_{12}	a_4
								a_{10}	a_3	a_4	a_{13}	a_{15}	a_{14}	a_{12}	a_4	
								a_{13}	a_{15}	a_{16}	a_7	a_1	×			
							a_{15}	a_{16}	a_7	a_1	×					

以下省略



HAMILTON 閉路

- 無向グラフが対象
- 全ての点を一度ずつ経由して始点に戻る道
- 巡回セールスマン問題の厳密解を得る際に必要

- 経由した頂点の列の管理が必要



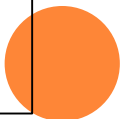
HAMILTON閉路の列挙アルゴリズム

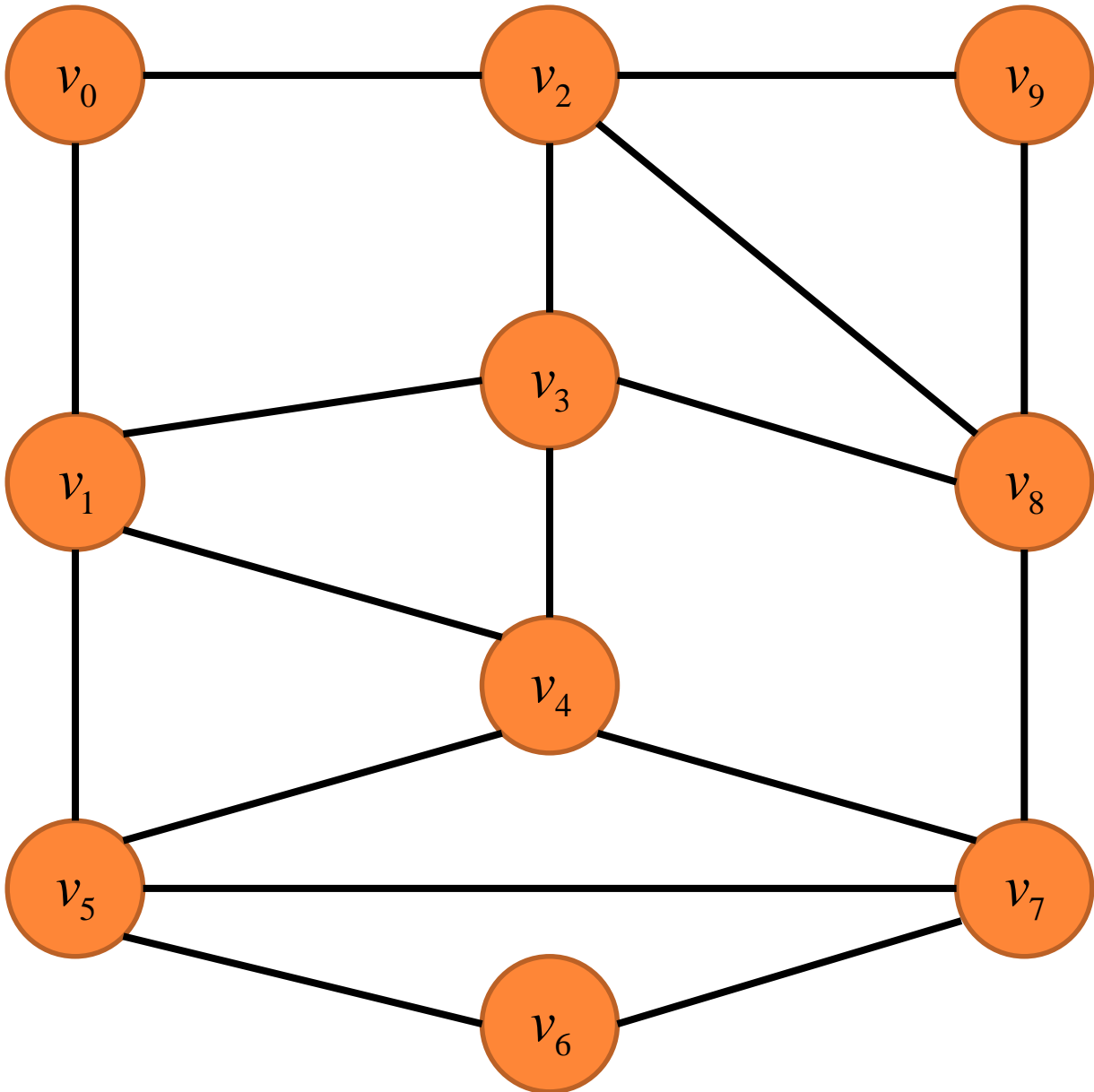
```
search (v) {  
  forall (a ∈ δv) {  
    wはaの反対の終端  
  
    if ((w == r) ∧ (|L| == |V|)) { 見つけた Hamilton 閉路を保存 }  
  
    else {  
      if (w ∉ L) {  
        Lにwを追加  
        search (w)  
        Lからwを削除  
      }  
    }  
  }  
}
```

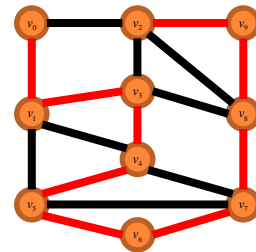
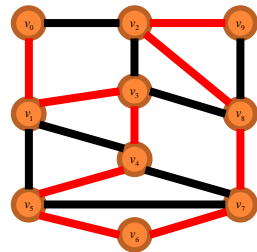
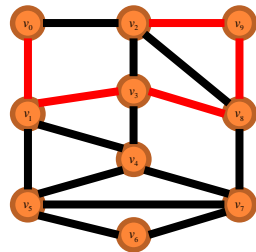
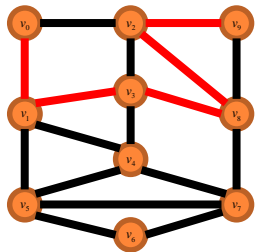
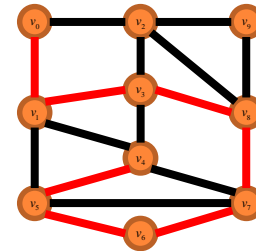
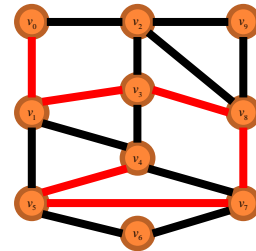
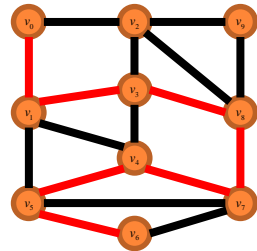
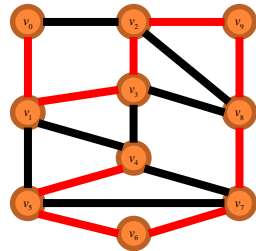
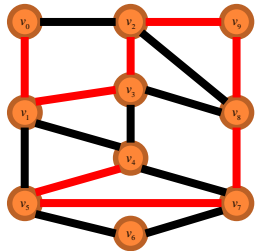
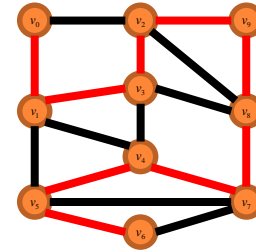
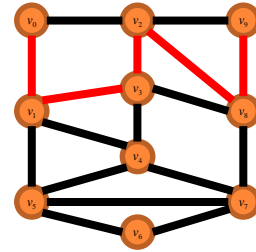
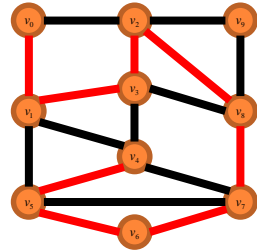
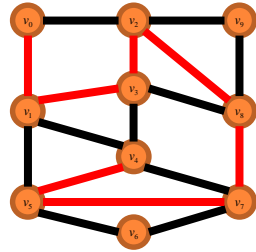
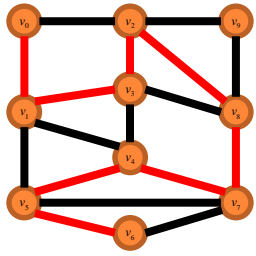
L : 既に経由した点の集合

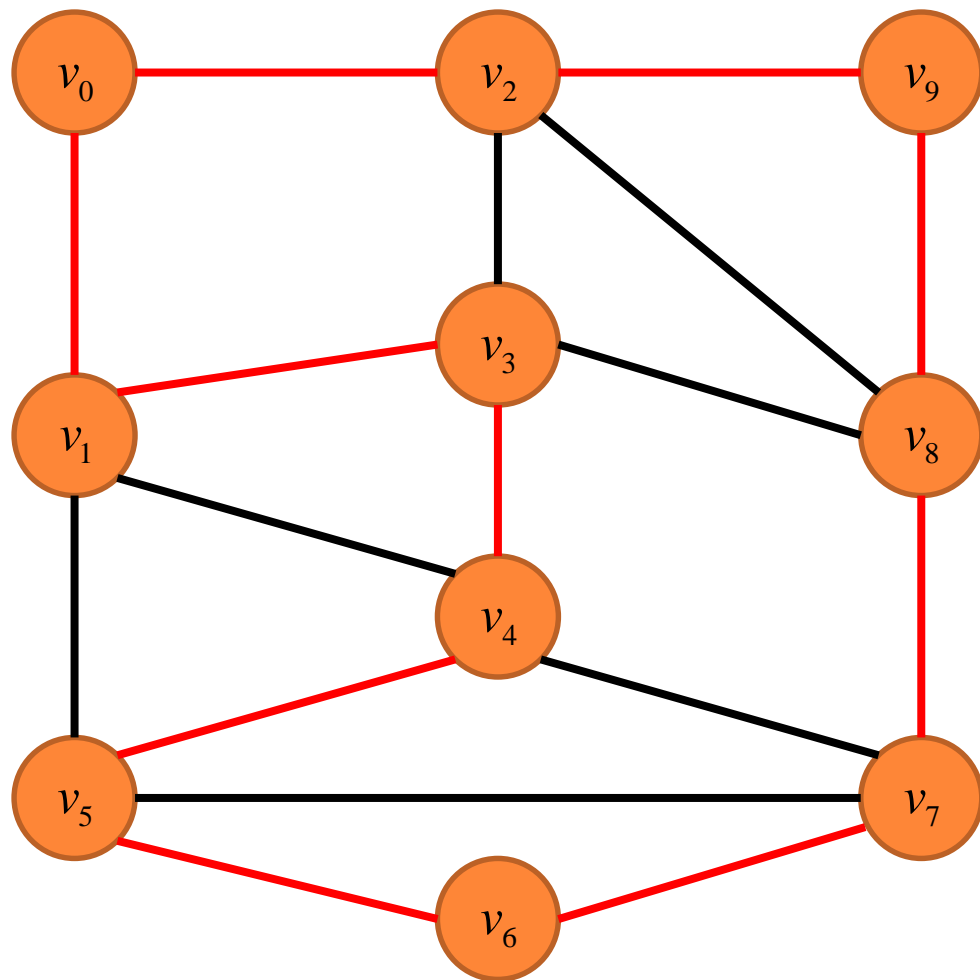
初期値 : $L = \emptyset$

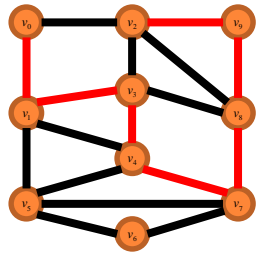
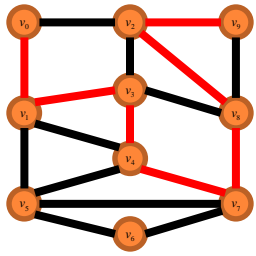
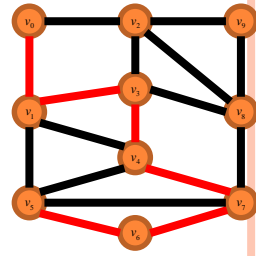
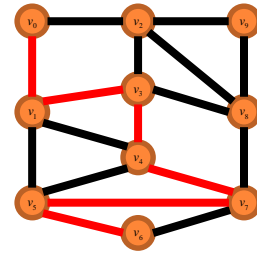
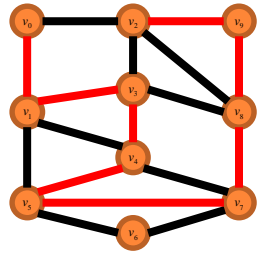
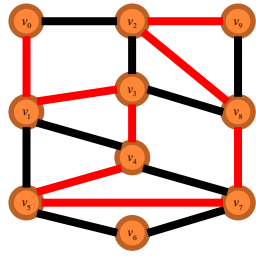
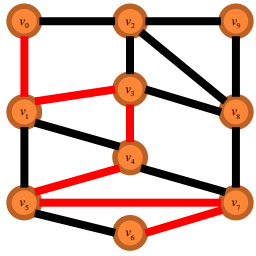
r : 始点











...

