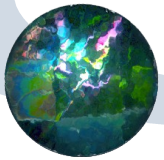
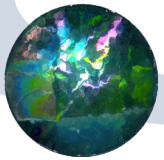
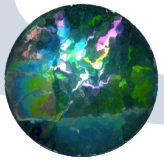


グラフの探索



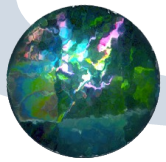
グラフ探索

- 単純な探索
  - ある頂点に到達できるか
  - 頂点を列挙する
- 単純でない探索：後述
  - 最小な木
  - 最短経路

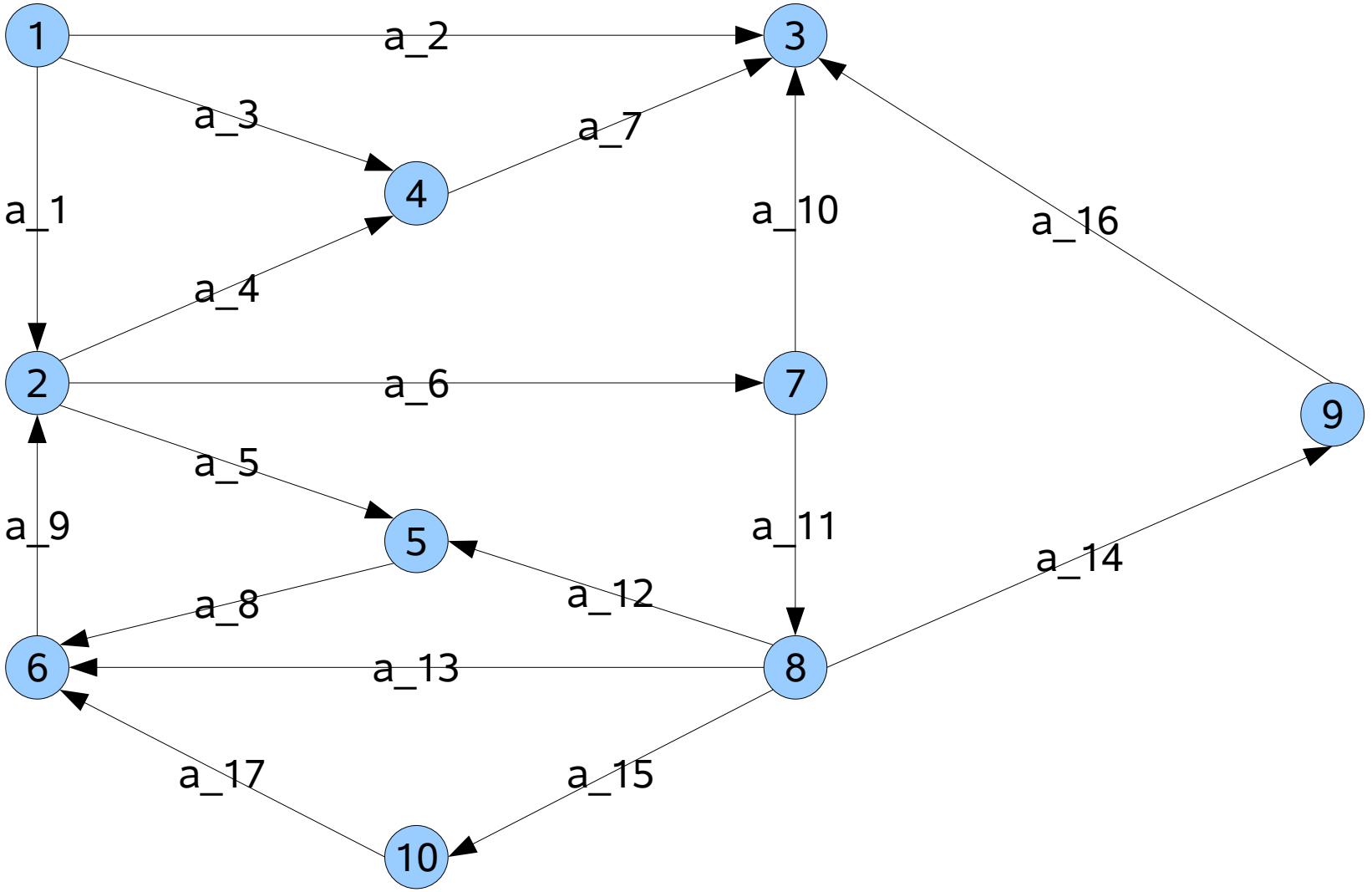


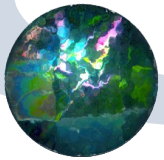
深さ優先探索
DFS (Depth-First Search)

- 出発点を定める
- たどれる限り、弧をたどる
  - それ以上進めなくなるまで
  - 新たな点が無くなる
- 戻って、別の弧をたどる
- 結果としてできる木(spanning tree)は、深いものができる



深さ優先探索
DFS (Depth-First Search)

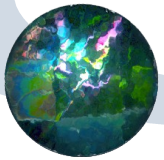




再帰的関数

- L : 既にチェックした点のリスト
- v : 現在の頂点

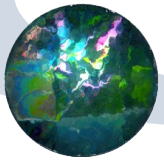
```
Search(v, L) {
  //vから出る全ての弧
  forall(a ∈ ∂⁺ v) {
    w = ∂⁻ a  //反対側の頂点
    if(w ∉ L){
      L = L ∪ {w}
      search(w, L)
    }
  }
  return
}
```



再帰的関数（終点指定）

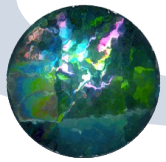
- L：既にチェックした点のリスト
- v：現在の頂点
- d：終点

```
Search(v, L,d) {
  forall(a ∈ ∂+ v) {
    w = ∂− a
    if(w ∉ L){
      L = L ∪ {w}
      if(w == d) return
      search(w, L,d)
    }
  }
  return
}
```

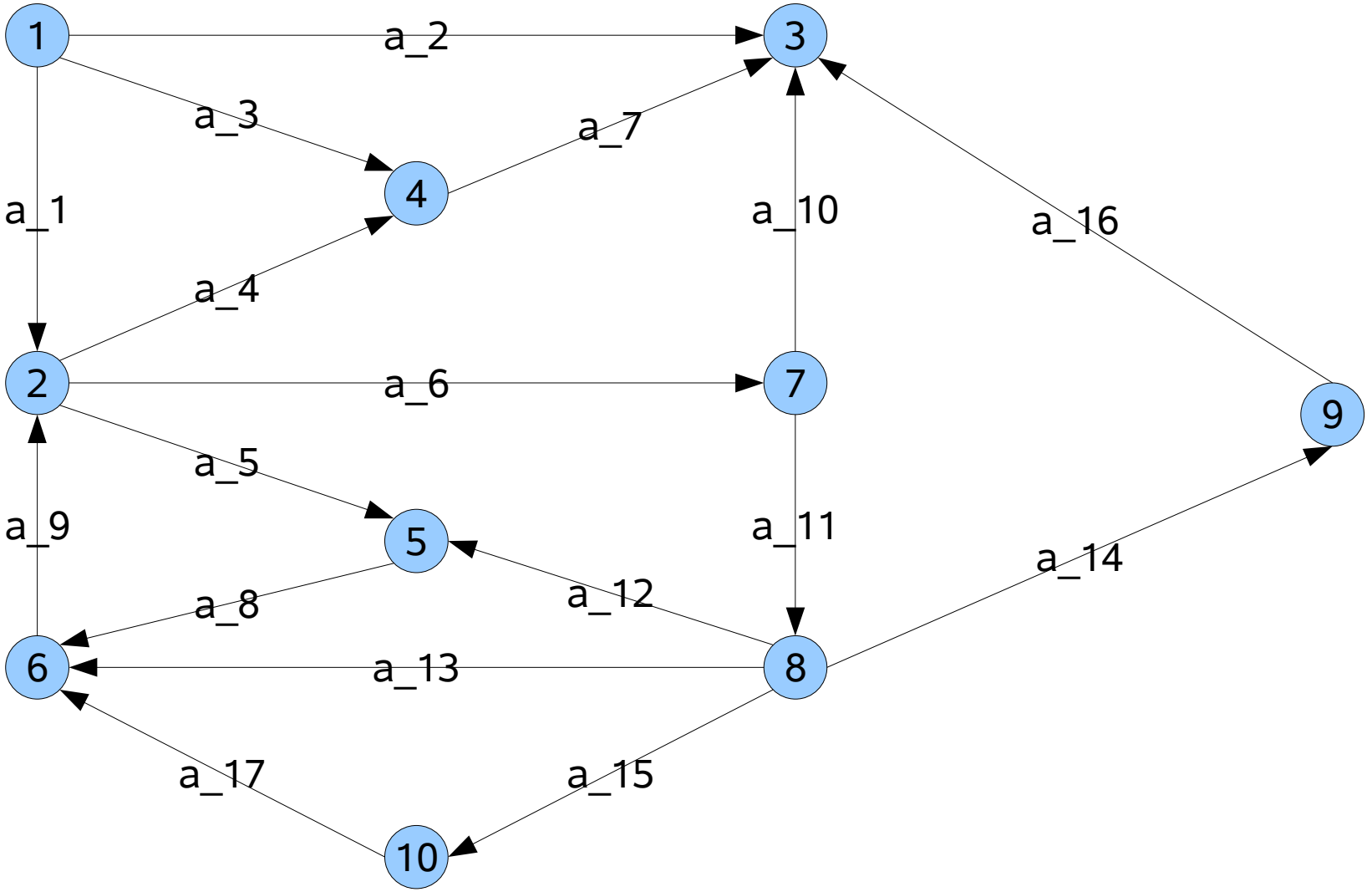


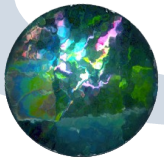
幅優先探索(Breadth-First Search)

- 出発点を定める
- 出発点に直接繋がっている点に印を付ける
- 印を付けた点に直接繋がっている点に印を付ける
- 結果としてできる木(spanning tree)は、幅の広いものができる



幅優先探索
BFS (Breadth-First Search)

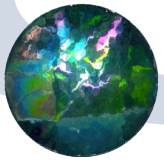




幅優先探索

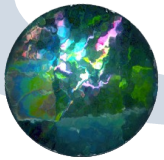
- L : すでにチェックした点のリスト: 初期 $L=\{r\}$
- Q : 調査すべき点のキュー：初期$Q=\{r\}$
-

$L=\emptyset$
$Q=\{v_0\}$
while ($Q\neq\emptyset$) {
  v = Q の先頭の要素取り出し
  forall($a\in\delta^+ v$) {
    $w=\partial^- a$
    if ($w\notin L$){
      $Q\leftarrow Q\cup\{W\}$
    }
  }
  $L\leftarrow L\cup\{V\}$
}



註：キュー(queue、待ち行列)

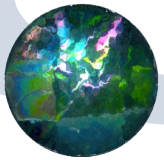
- 先入れ先出し (FIFO、First In First Out)のリスト構造
- java.util.concurrent.ConcurrentLinkedQueueクラス
  - add(E e) 要素eを追加
  - poll() 先頭の要素を取り出す



幅優先探索(終点指定)

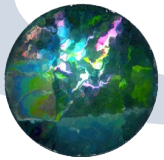
- L : すでにチェックした点のリスト: 初期 $L=\{r\}$
- Q : 調査すべき点のキュー：初期$Q=\{r\}$
- d : 終点

```
while (Q ≠∅ ) {
  v = Q の先頭の要素取り出し
  forall( a ∈ ∂⁺v ) {
    w = ∂⁻a
    if(w == d) {
       Qを空にする
      return}
    if (w ∉ L ){
        L = L ∪ {w}
        Q に w を追加
    } } }
```



グラフの連結性(connectedness)

- グラフ$G=(V,A)$が連結(connected)
  - $\forall u \in V$から$\forall v \in V$へ道が存在する
  - DFSまたはBFSで決定できる
- 連結でないグラフの極大(頂点数が極大)な連結部分(connected component)



2連結性(2-connectedness)

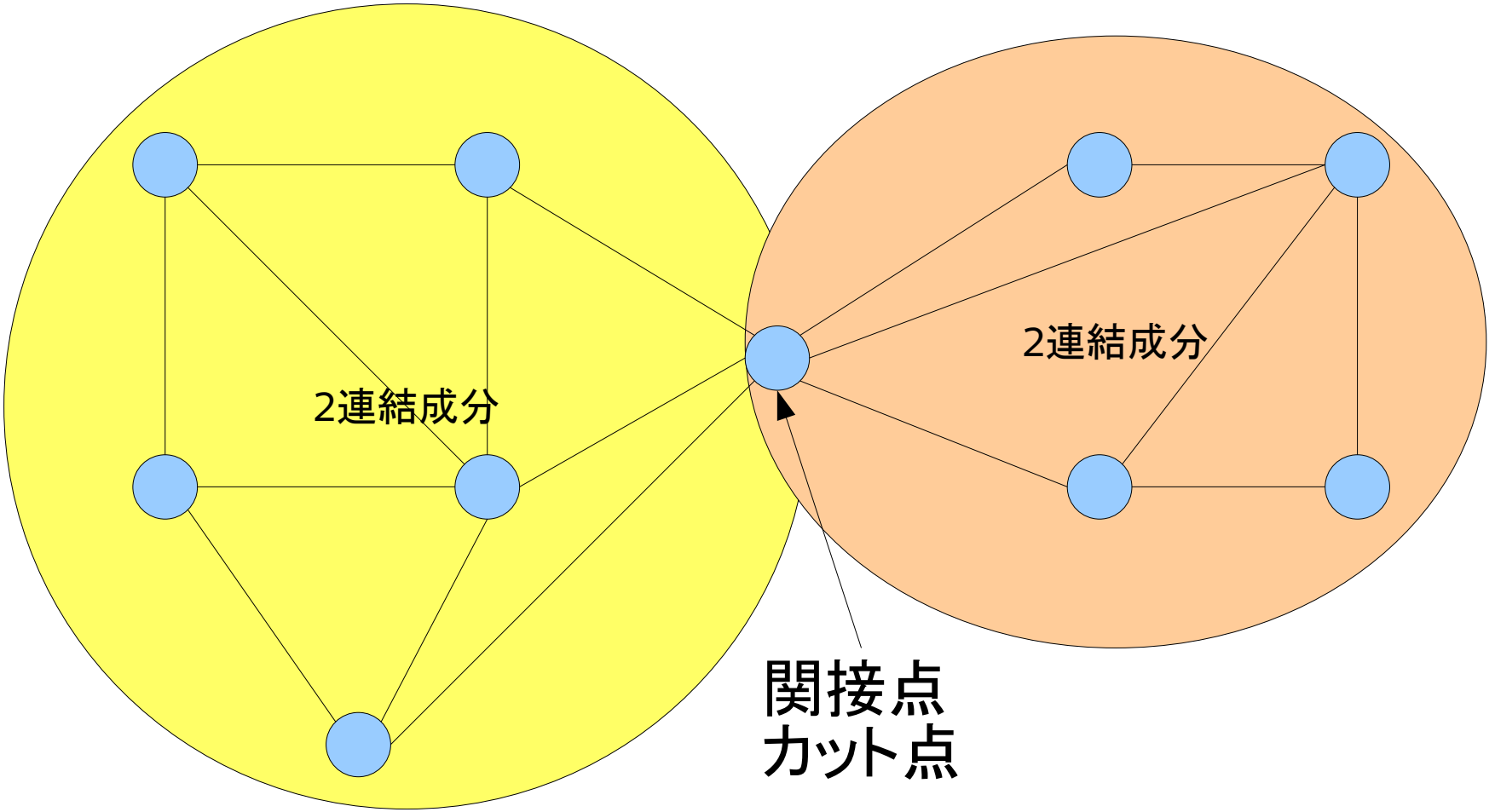
- 連結なグラフ$G=(V,A)$の部分グラフ$H_i=(V_i,A_i)$ $(i=1,2)$

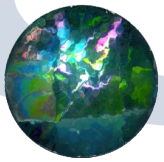
$$|A_i|\geq 1,\ A_1\cup A_2=A,\ A_1\cap A_2=\emptyset$$
$$V_1\cup V_2=V,\ |V_1\cap V_2|=1$$

- H_iを2連結成分と呼ぶ
- 関節点(articulation vertex)またはカット点(cut-vertex)

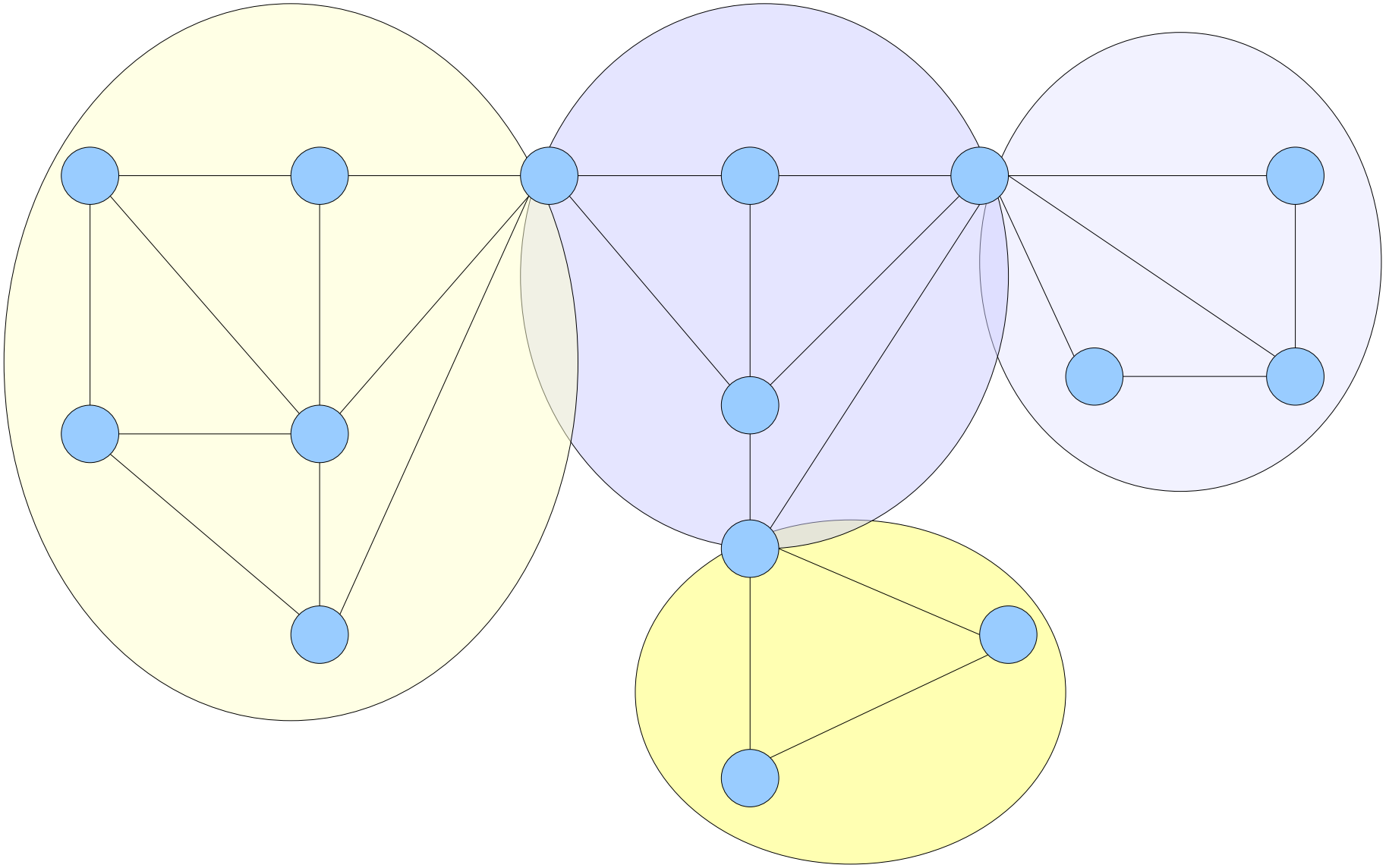
$$\{c\}=V_1\cap V_2$$

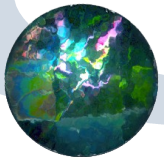
- 関節点を持たない連結グラフ
  - 2連結または非可分(non-separable)


2連結成分
2連結成分
関接点
カット点



2連結成分への分割

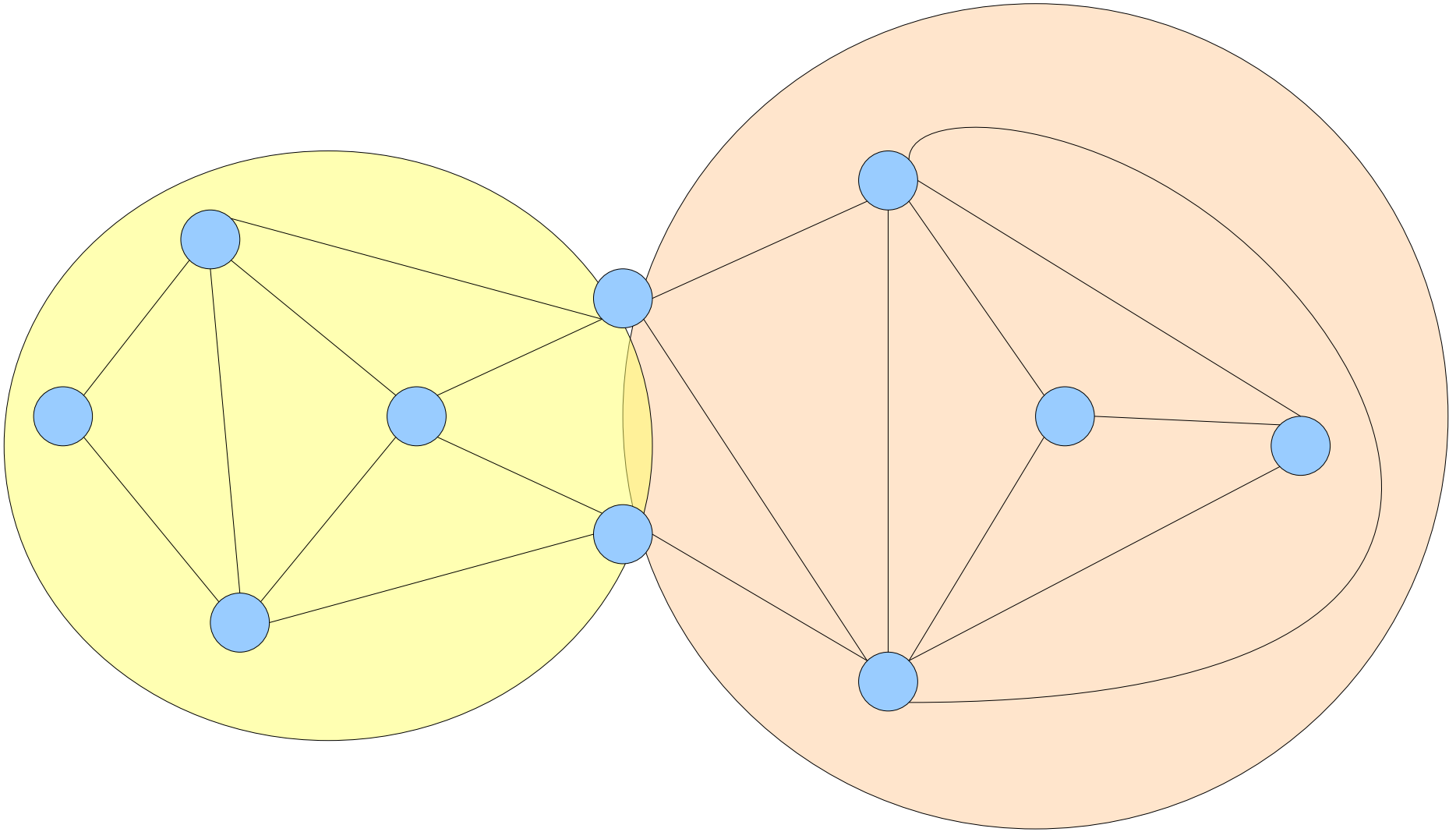


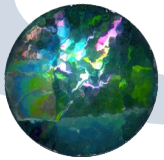


3連結性(3-connectedness)

- 連結なグラフ$G=(V,A)$の部分グラフ$H_i=(V_i,A_i)$ $(i=1,2)$
- $$|A_i|\geq 1,\ A_1\cup A_2=A,\ A_1\cap A_2=\emptyset$$
  $$V_1\cup V_2=V,\ |V_1\cap V_2|=2$$
- H_iを2端子部分グラフと呼ぶ
- 2部分グラフを持たないグラフ
  - 3連結グラフ

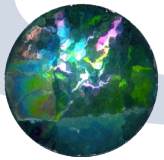
2端子部分グラフの例





強連結(strongly connected)

- グラフ$G=(V,A)$が強連結
  - $\forall u \in V$から$\forall v \in V$へ有向道が存在する
- グラフ中の強連結成分$H_i=(V_i,A_i)$
  - $\forall u \in V_i$から$\forall v \in V_i$へ有向道が存在する



連結性の簡単な確認

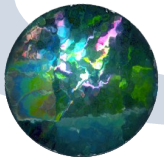
- 隣接行列Γを使う
- 点iから点jへの弧がある $\Gamma_{ij}=1$
- 「かけ算」を定義

$$(A \otimes B)_{ij} = \begin{cases} 1 \text{ if } \sum_k A_{ik} B_{kj} > 0 \\ 0 \text{ otherwise} \end{cases}$$

- 点iから点jへ長さ2の道がある
- 拡張

$$\Gamma^n = (\Gamma^{n-1}) \otimes \Gamma$$

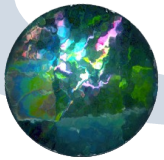
-
- 点の数をmとするとm-1乗で連結成分に分解できる



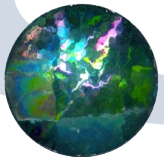
グラフとマトロイド(matroid)

- 無向グラフ $G=(V, A)$ $|A|\neq\emptyset$ を考える
- その極大木 (spanning tree) T の集合族
  - 木をグラフの弧の集合と同一視する

$$T=\{T_i \mid T_i \subseteq A\}$$

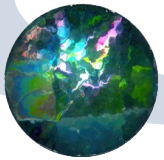


- (B0) $T \neq \emptyset$
- (B1) 木の組$\forall T_1, T_2 \in T$と弧 $\forall a_1 \in T_1 \backslash T_2$ に対して
  - 弧 $\exists a_2 \in T_2 \backslash T_1$ が存在し
  - 別の木を構成できる:$(T_1 \backslash \{a_1\}) \cup \{a_2\} \in T$



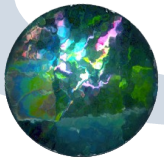
- (B0)と(B1)を満たす $M=(A, T)$
  - 基族(family of bases) T によって定められるマトロイド(matroid)
  - T_i :マトロイドの基(base)
  - 基は全て同じ大きさ:枝の数は同数

- (問)頂点数が与えられたとき、その極大木の枝の数



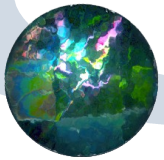
独立集合(independent set)

- $I_i \subseteq A$, I_i を含む基があるとき、I_iを独立集合と呼ぶ
  - I_iは点の部分集合のspanning tree
- 独立集合族$I=\{I_i \subseteq A\}$
- 極大な独立集合族が基に対応



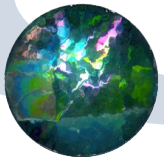
独立集合の性質

- (I0) $\emptyset \in I$
- (I1) $I_1 \subseteq I_2 \in I$ ならば $I_1 \in I$
- (I2) $I_1, I_2 \in I$ かつ $|I_1| < |I_2|$ ならば
  - $\exists e \in I_2 \setminus I_1$ が存在し、$I_1 \cup \{e\} \in I$



サーキット(circuits)

- 従属集合(dependent set):独立集合でないAの部分集合
  - つまり、木ではない枝の集合
- サーキット:極小な従属集合
- サーキット族



サーキットCの性質

- (C0) $C \neq \{\emptyset\}$
- (C1) $C_1, C_2 \in C$ かつ $C_1 \subseteq C_2$ ならば $C_1 = C_2$
  - $C_1 \subset C_2$ であれば、C_2 が極小であることに反する
- (C2) $C_1, C_2 \in C$ かつ $C_1 \neq C_2$ ならば
  - $e \subseteq (C_1 \cap C_2)$ に対して
  - $C_3 \subseteq (C_1 \cup C_2) \setminus \{e\}$ なる $C_3 \in C$ が存在する