Internet Week 2013

【S8】SDN 時代を生き抜く為のグラフ理論とネットワークのアルゴリズム入門

導入

浅間 正和

有限会社 銀座堂

このプログラムの概要

- 導入
  - 背景、用語の説明、グラフの表現方法と探索

浅間（３０分）

- グラフ理論とネットワークのアルゴリズムの基礎
  - 最短路問題、最小木問題、アルゴリズムと計算量

伊波さん（５０分）

- ネットワークフローとその代表的な問題
  - 最大流問題、多品種流問題

金子さん（５０分）

- まとめ
  - システム最適化流問題、参考情報、まとめ

浅間（２０分）

背景

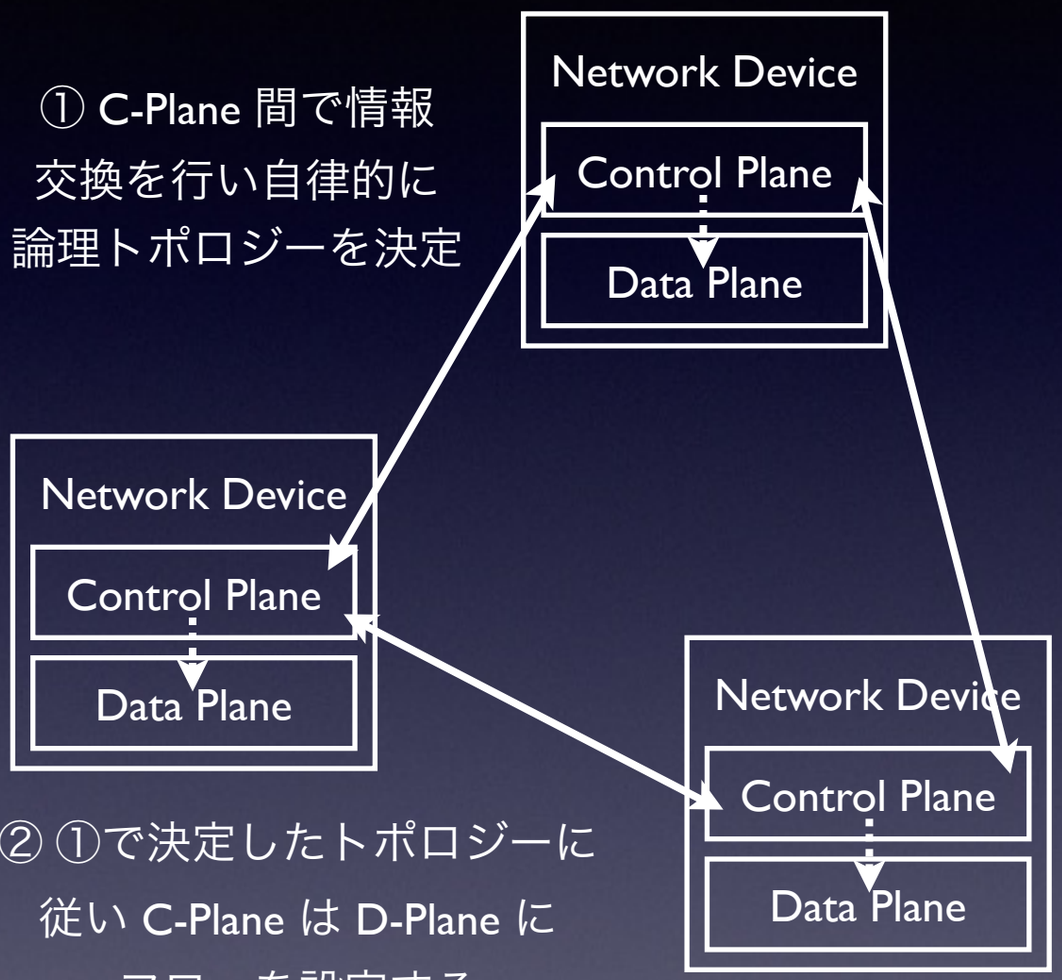
- OpenFlow の登場により集中制御型（⇔自律分散型）のコントロール・プレーンの自作が可能となった
- たしかに可能となったがやってみるとかなりめんどい
- グラフ理論（離散数学）のなかのネットワークのアルゴリズムの分野ではそれに役立ちそうな様々な研究があるっぽい

- 役に立ちそうなアルゴリズムを体系的に学べるようなプログラムをやりたい（つうか勉強したい）！！！

OpenFlow のおさらい（極端な説明）

従来

① C-Plane 間で情報交換を行い自律的に論理トポロジーを決定

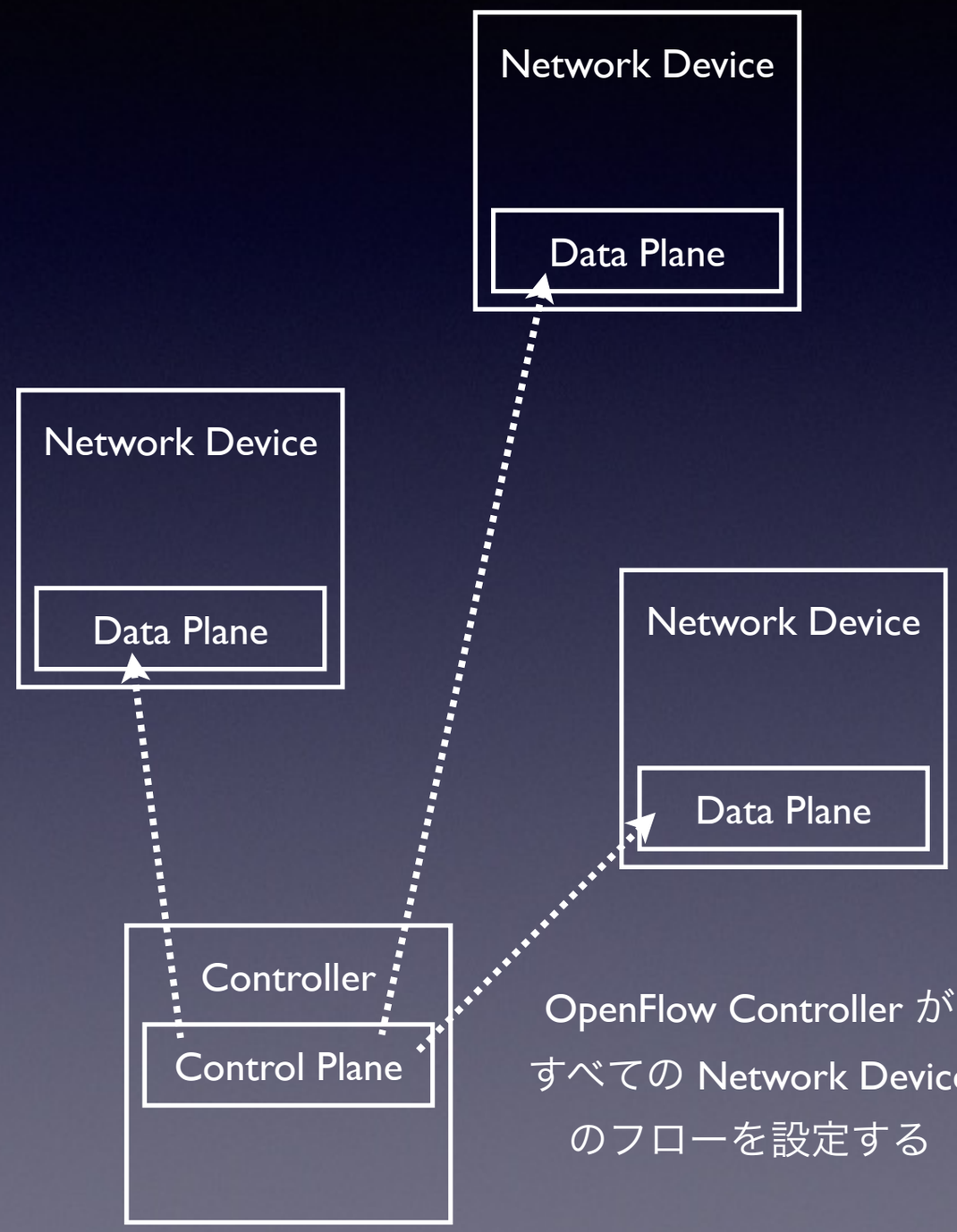


② ①で決定したトポロジーに従い C-Plane は D-Plane にフローを設定する

STP（L2）の場合：
ループが生じないよう特定ポートをブロックする

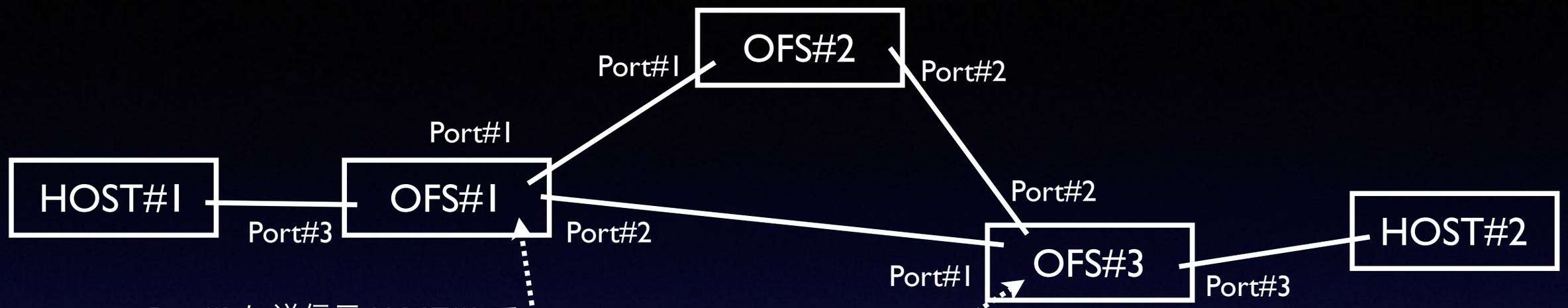
OSPF（L3）の場合：
宛先への最短経路を計算しフローを設定する

OpenFlow の場合



OpenFlow Controller がすべての Network Device のフローを設定する

OpenFlow Controller の処理内容（想像）



Port#3 に送信元 HOST#1 で宛先 HOST#2 のフレームが届いたときは Port#2 から出すフローを設定

Port#1 に送信元 HOST#1 で宛先 HOST#2 のフレームが届いたときは Port#3 から出すフローを設定

OFC

HOST#1 から HOST#2 への **最適な**経路を計算

= HOST#1→OFS#1→OFS#3→POST#2
（ということにしておく）

OpenFlow Switch 隣接テーブル

OFS	ポート	隣接
OFS#1	Port#1	OFS#2
OFS#1	Port#2	OFS#3
OFS#2	Port#1	OFS#1
OFS#2	Port#2	OFS#3
OFS#3	Port#1	OFS#1
OFS#3	Port#2	OFS#2

端末接続テーブル

端末	OFS	ポート
HOST#1	OFS#1	Port#3
HOST#2	OFS#3	Port#3

※ ほんとうはここにコストや帯域幅の情報もいるはず

最適な経路？

- 導入
  - 背景、用語の説明、グラフの表現方法、探索
    浅間（３０分）

他の通信状況を考慮せずに
最適な経路を求める方法

- グラフ理論とネットワークのアルゴリズムの基礎
  - 最短路問題、最小木問題、アルゴリズムと計算量
    伊波さん（５０分）
- ネットワークフローとその代表的な問題
  - 最大流問題、多品種流問題
    金子さん（５０分）
- まとめ
  - 他の通信状況も考慮にいれ
    最適な経路を求める方法
  - 参考情報、まとめ
    浅間（２０分）

他の通信状況も考慮にいれ
最適な経路を求める方法

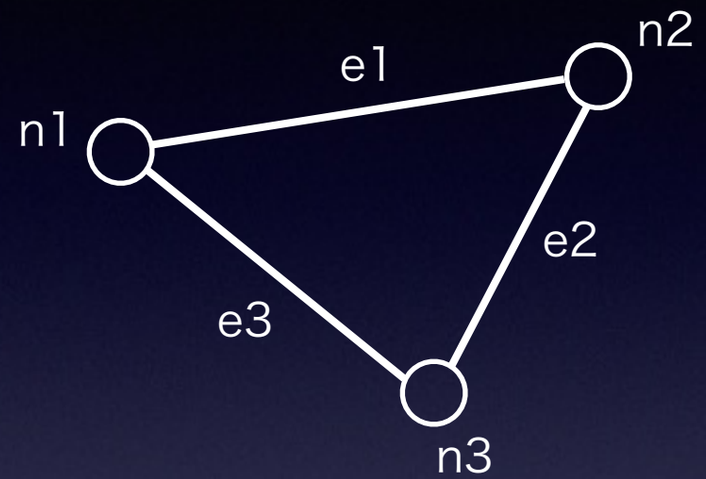
グラフとネットワーク

- グラフ（Graph）
  - ノード（Node）とエッジ（Edge）の集合
  - ノードはほかに接点や点、頂点（Vertex）等と呼ばれることもある
  - エッジはほかに辺や弧、線、リンク、ブランチ、枝（Arc）等と呼ばれることもある
    - エッジに向きがある場合は特に有向グラフと呼ぶ
- ネットワーク（Network）
  - グラフのノードやエッジに数量が与えられたもの
  - 例）ノードに需要と供給が与えられたグラフ
  - 例）エッジに距離や費用や時間が与えられたグラフ

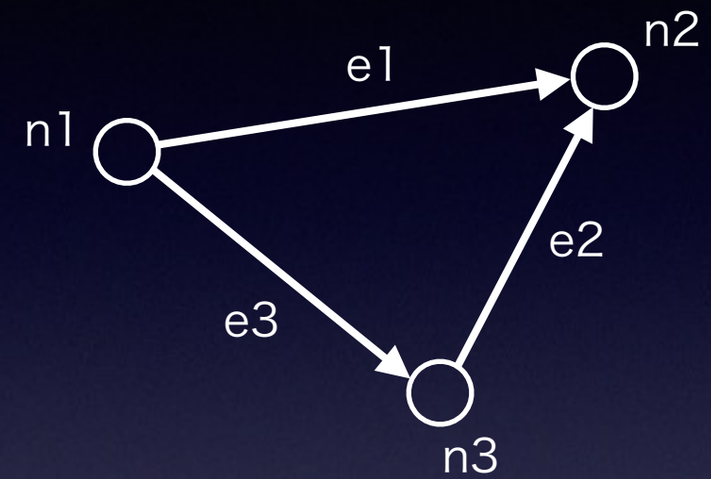
グラフとネットワーク

グラフ

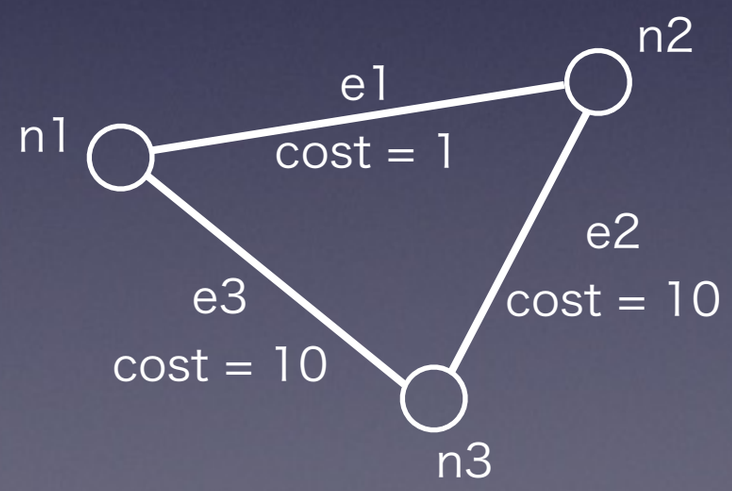
無向グラフ

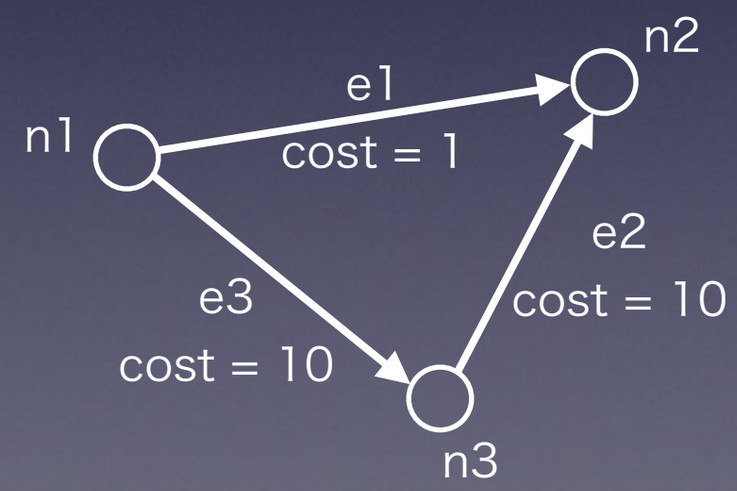


有向グラフ



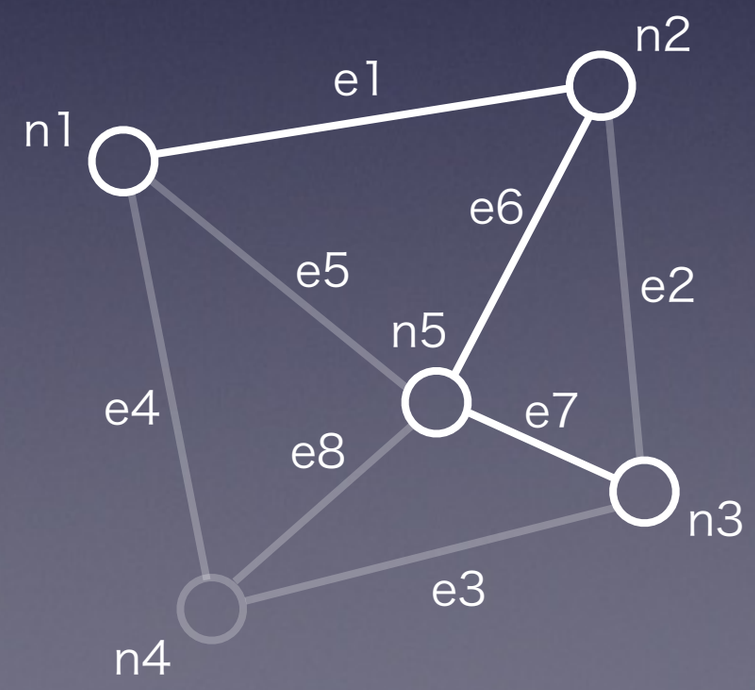
ネットワーク





路

- 路（Path）
  - 隣接するノード同士を辿ることでできる交互系列
    (n0, e1, n1, …, e(k-1), n(k-1), ek, nk) を路と呼ぶ
  - n0 を始点と呼ぶ
  - nk を終点と呼ぶ



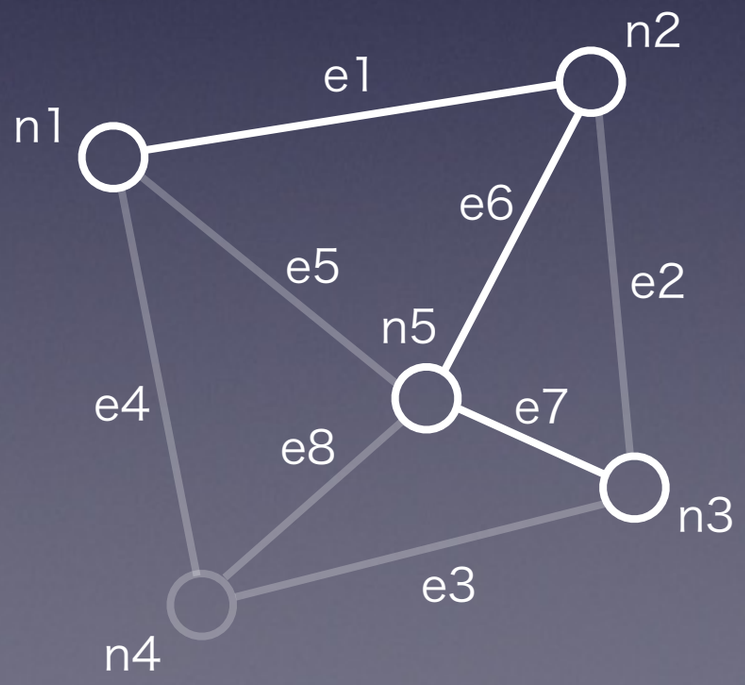
(n1, e1, n2, e6, n5, e7, n3)
… 路

始点は n1

終点は n3

路

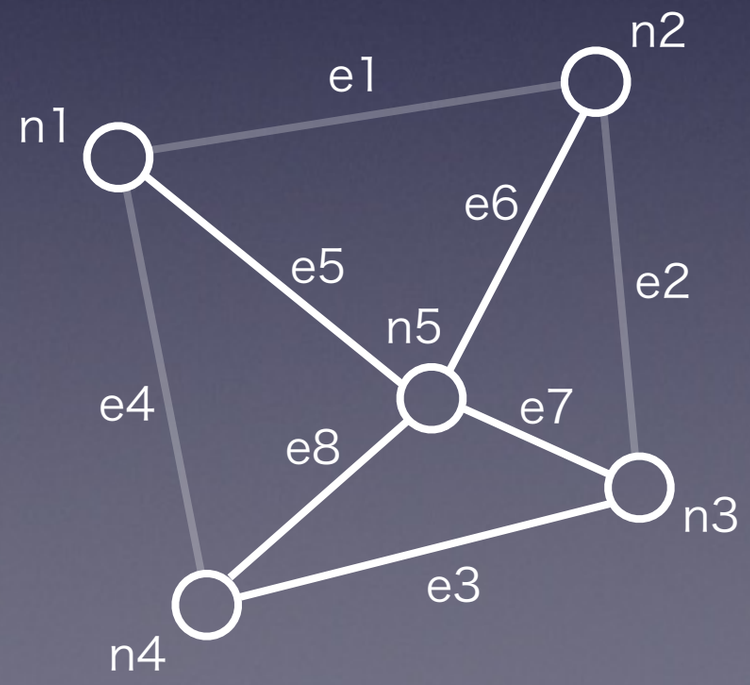
- 単純な路（Simple Path）
  - 同じエッジを 2 回以上通らない路
- 初等的な路（Elementary Path、道）
  - 同じノードを 2 回以上通らない路
- 閉路（Closed Path、Cycle、Circuit）
  - n0 と nk が一致し少なくとも 1 つのエッジを辿る路



(n1, e1, n2, e6, n5, e7, n3)
… 単純な路で且つ初等的な路

路

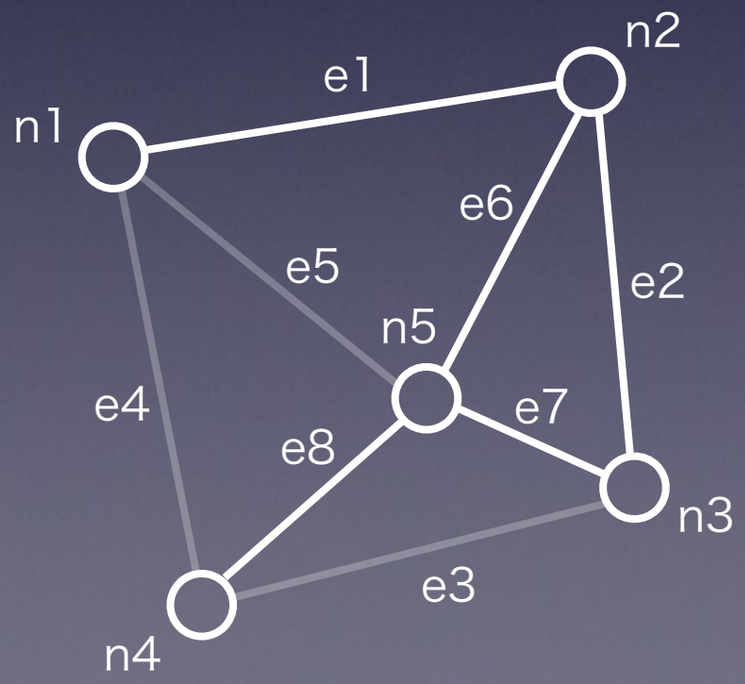
- 単純な路（Simple Path）
  - 同じエッジを 2 回以上通らない路
- 初等的な路（Elementary Path、道）
  - 同じノードを 2 回以上通らない路
- 閉路（Closed Path、Cycle、Circuit）
  - n0 と nk が一致し少なくとも 1 つのエッジを辿る路



(n1, e5, n5, e8, n4, e3, n3, e7, n5, e6, n2)
… 単純な路だが初等的な路ではない路

路

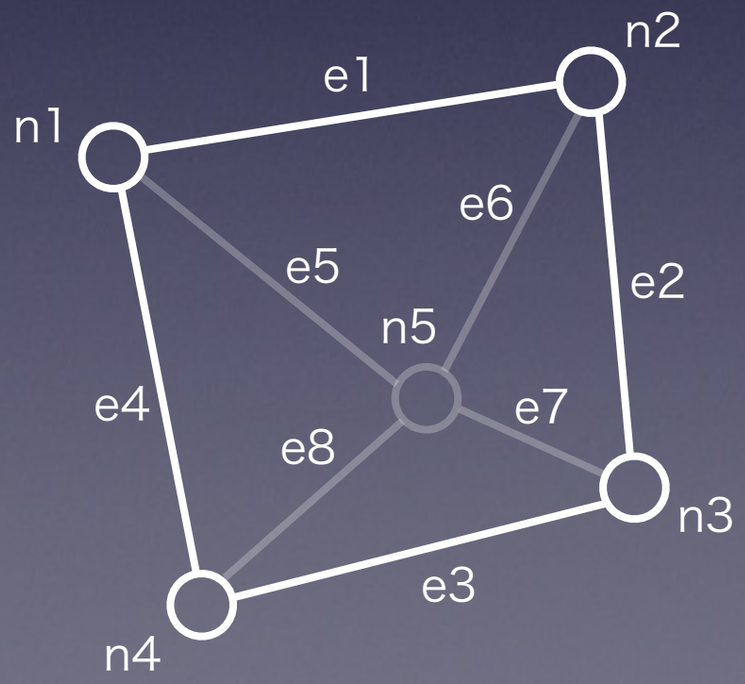
- 単純な路（Simple Path）
  - 同じエッジを 2 回以上通らない路
- 初等的な路（Elementary Path、道）
  - 同じノードを 2 回以上通らない路
- 閉路（Closed Path、Cycle、Circuit）
  - n0 と nk が一致し少なくとも 1 つのエッジを辿る路



(n1, e1, n2, e6, n5, e7, n3, e2, n2, e6, n5, e8, n4)
… 単純な路でもなく初等的な路でもない路

路

- 単純な路（Simple Path）
  - 同じエッジを 2 回以上通らない路
- 初等的な路（Elementary Path、道）
  - 同じノードを 2 回以上通らない路
- 閉路（Closed Path、Cycle、Circuit）
  - n0 と nk が一致し少なくとも 1 つのエッジを辿る路

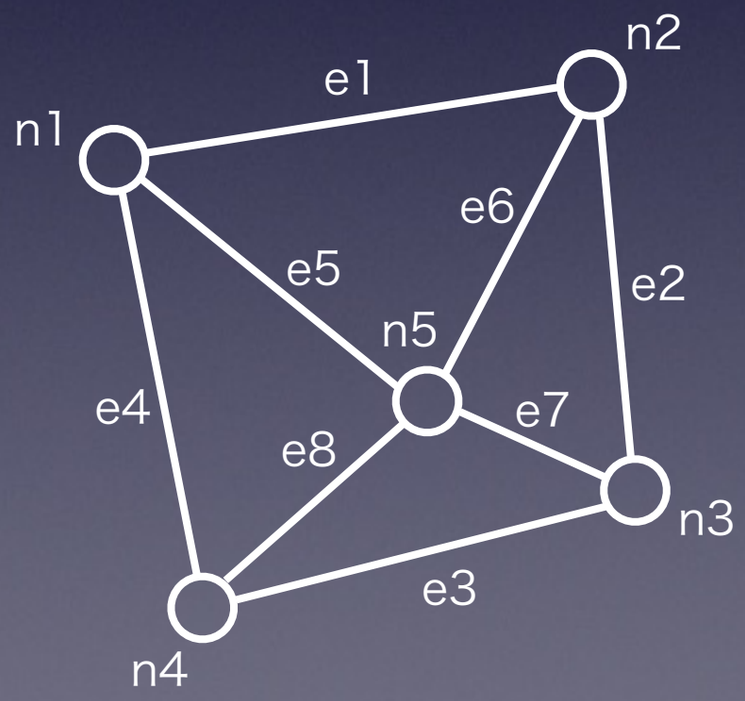


(n1, e1, n2, e2, n3, e3, n4, e4, n1)
… 閉路

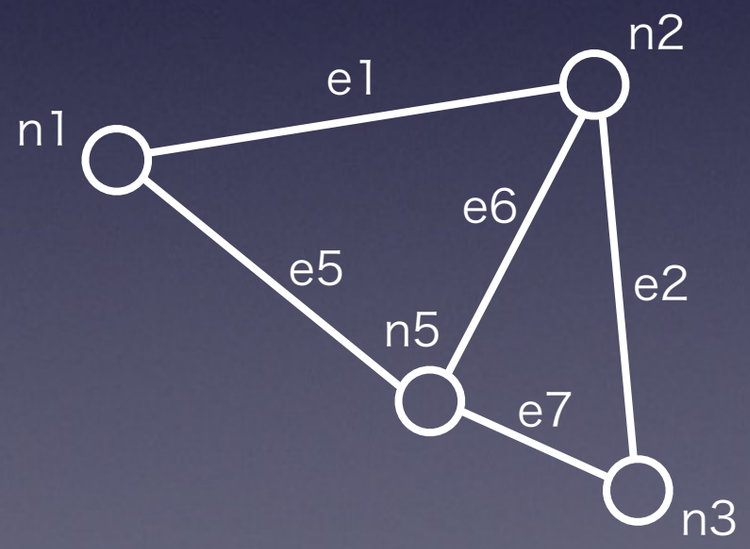
部分グラフと連結

- 部分グラフ（Subgraph）
  - あるグラフのノード集合の部分集合とエッジ集合の部分集合からなるグラフ

グラフ G



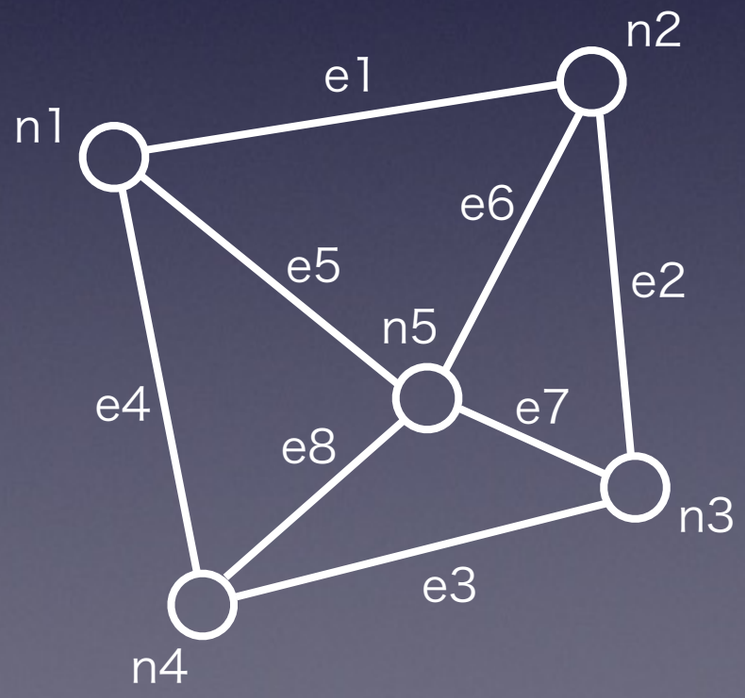
グラフ G の部分グラフ



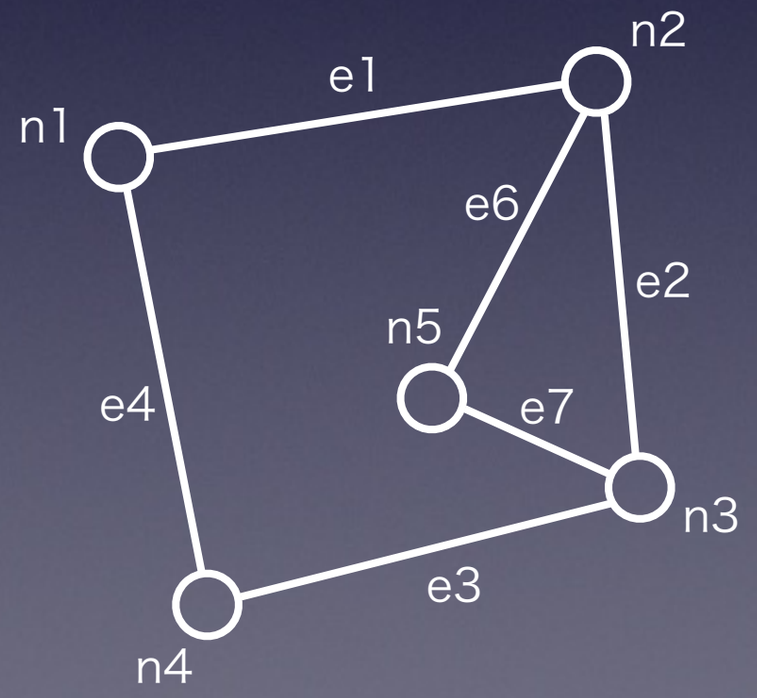
部分グラフと連結

- 全域部分グラフ（Spanning Subgraph）
  - 元のグラフとノード集合が等しい部分グラフ
  - あるグラフからエッジのみを差し引くことで出来る部分グラフ

グラフ G



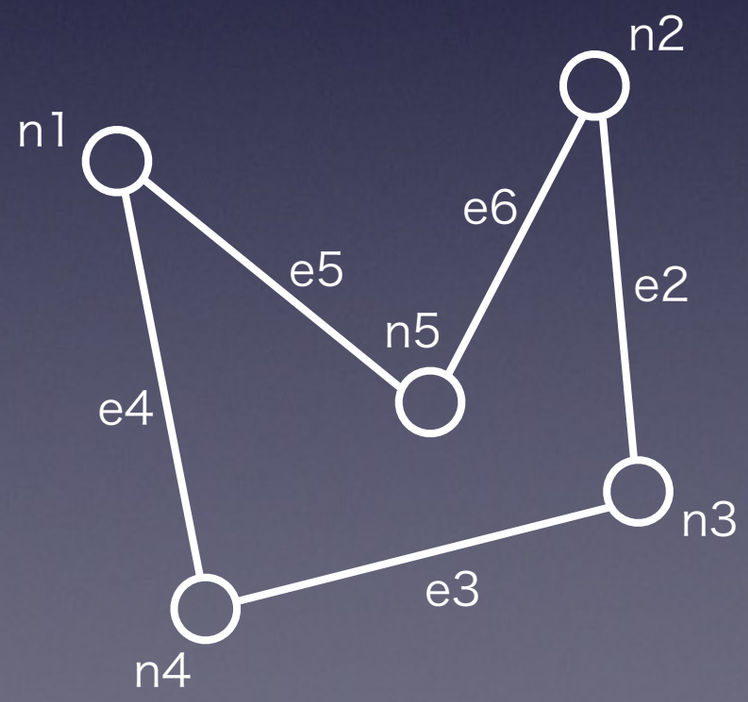
グラフ G の全域部分グラフ



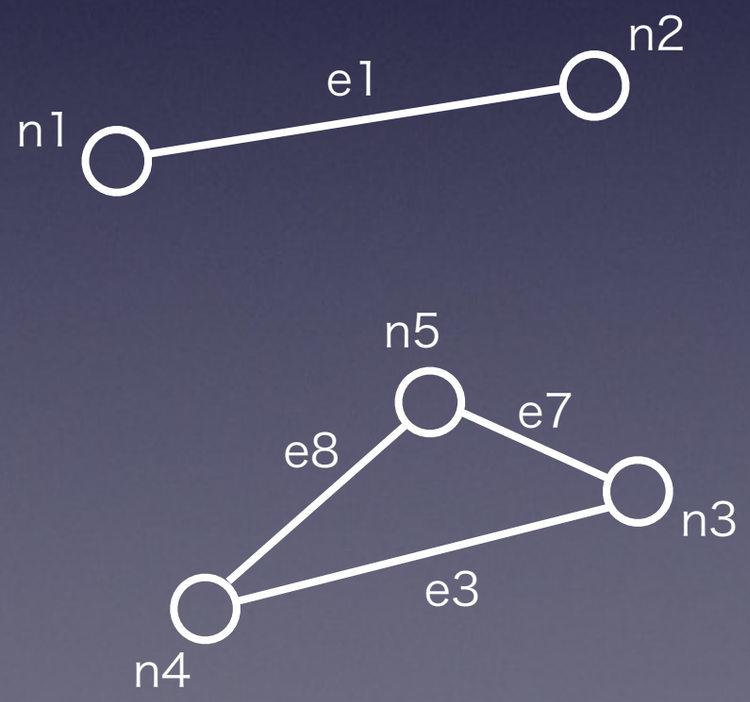
部分グラフと連結

- 連結（Connected）
  - 任意の 2 つのノード間に道が存在するグラフを “連結である” と言う

連結なグラフ



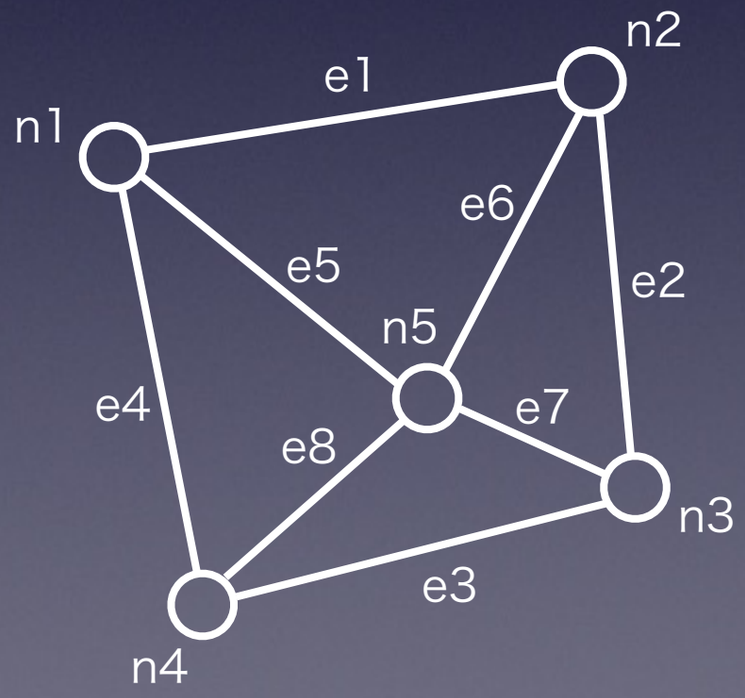
連結でないグラフ



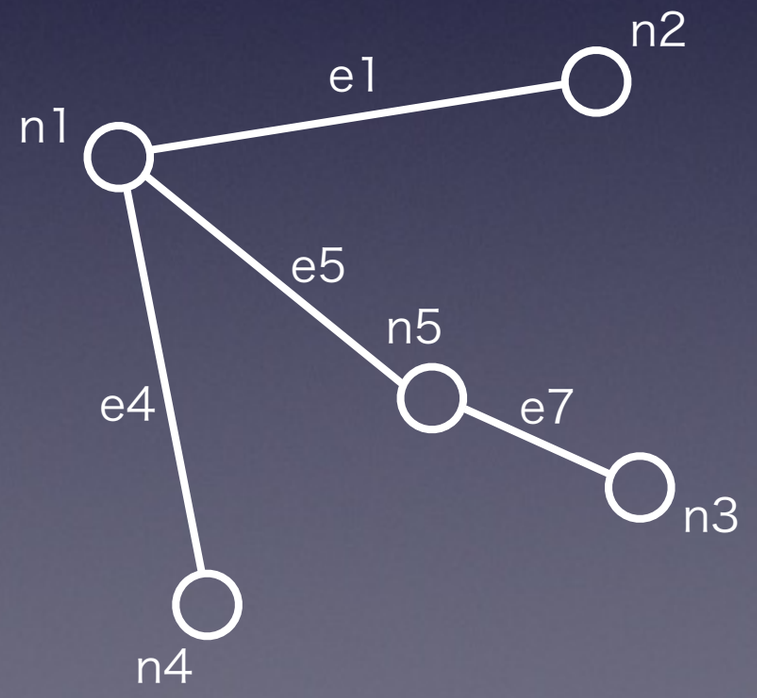
木

- 木（Tree）
  - 連結であり且つ閉路を持たないグラフ
- 全域木（Spanning Tree）
  - 全域部分グラフで且つ木のグラフ

グラフ G

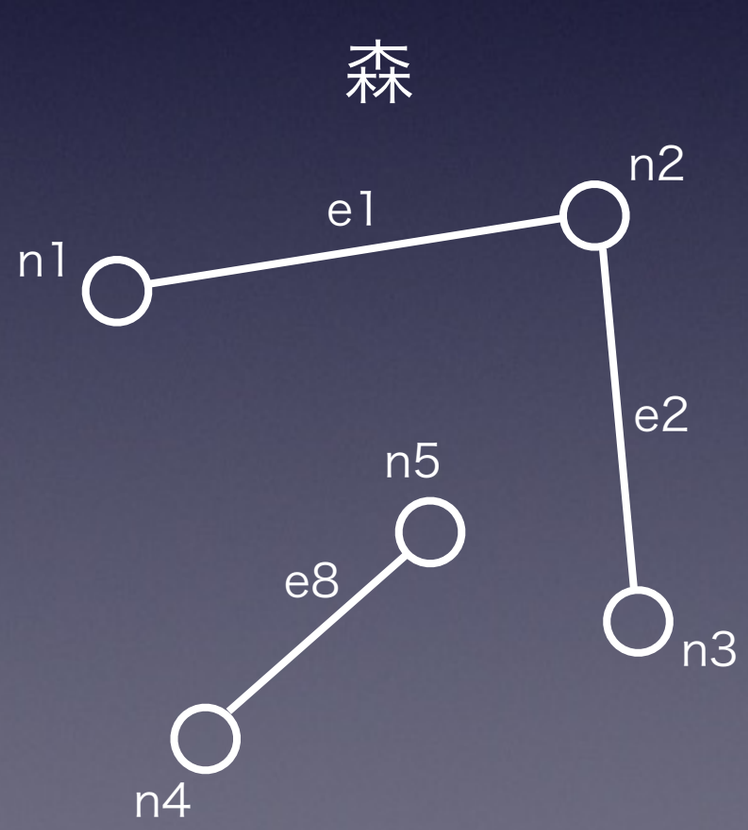


グラフ G の全域木



木

- 森（Forest）
  - 単に閉路をもたない部分グラフ
  - 森は連結でなくても良くもりの各連結成分は木となっている

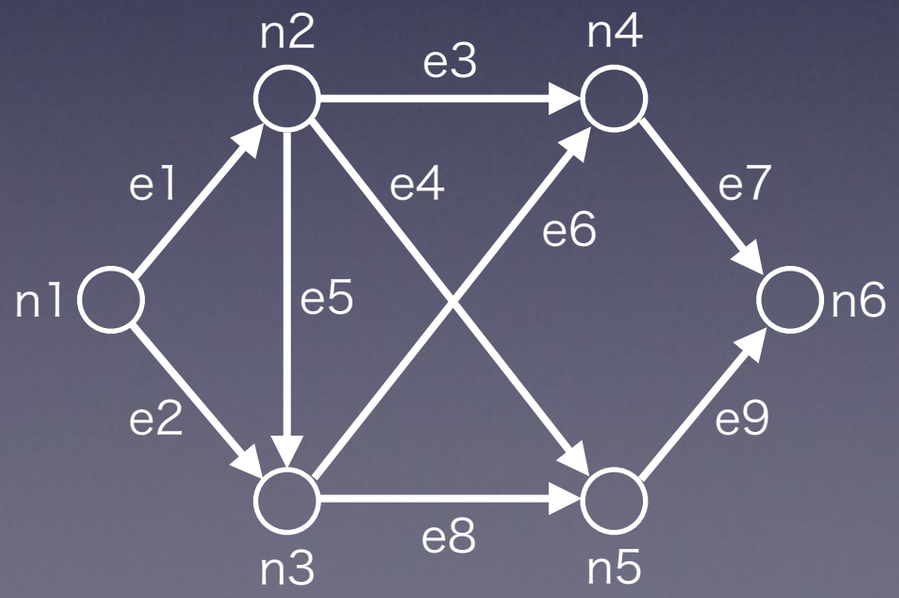


グラフの表現方法

- 行列による表現
  - 接続行列
  - 隣接行列
- オブジェクトによる表現

グラフの表現方法（接続行列）

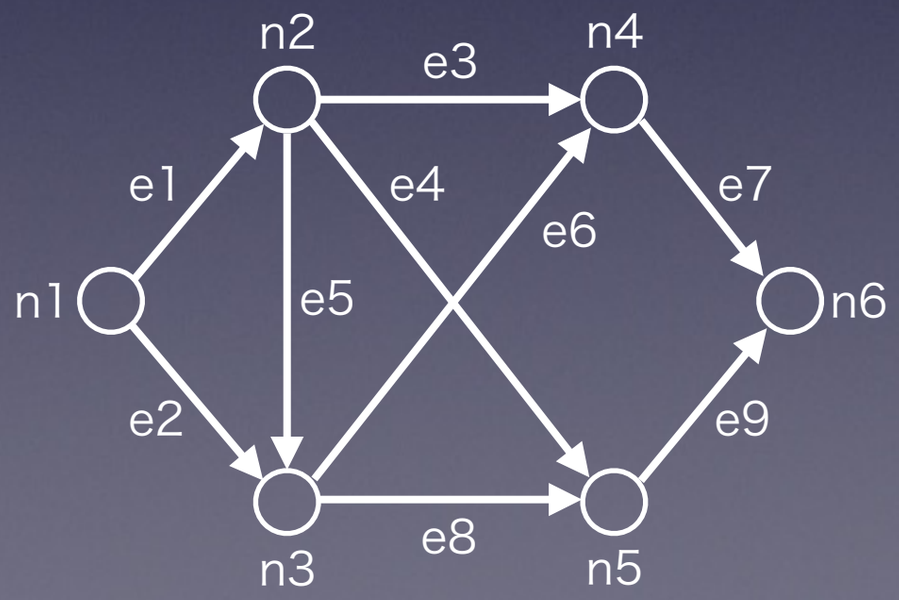
- ノードとエッジを行と列に対応させた行列で表現
- 要素は以下のようにする
  - 1 … その行に対応するノードがその列に対応するエッジの始点
  - -1 … その行に対応するノードがその列に対応するエッジの終点
  - 0 … その行に対応するノードがその列に対応するエッジと未接続
- ループを表現することが出来ない
- 線形計画問題に落とし込む場合によく用いられる？



	e1	e2	e3	e4	e5	e6	e7	e8	e9
n1	1	1	0	0	0	0	0	0	0
n2	-1	0	1	1	-1	0	0	0	0
n3	0	-1	0	0	1	-1	0	-1	0
n4	0	0	-1	0	0	1	1	0	0
n5	0	0	0	-1	0	0	0	1	1
n6	0	0	v6	0	0	0	-1	0	-1

グラフの表現方法（隣接行列）

- エッジの始点と終点を行と列に対応させた行列で表現
  - 1 ... その行に対応するノードを始点としその列に対応するノードを終点とするエッジが存在する
  - 0 ... その行に対応するノードを始点としその列に対応するノードを終点とするエッジが存在しない
- 多重枝を表現できない
- 無向グラフの場合はかならず対称行列となる



	v1	v2	v3	v4	v5	v6
n1	0	1	1	0	0	0
n2	0	0	0	1	1	0
n3	0	1	0	0	0	0
n4	0	0	1	0	0	1
n5	0	0	1	0	0	1
n6	0	0	0	0	0	0

グラフの表現方法（オブジェクト）

- ノードとエッジを表現するクラス（構造体）をそれぞれ定義し接続関係をそのメンバ変数で保持させる方法
- 接続関係以外の情報を保持させることも出来る
- 注意すれば多重エッジやループも表現することが可能

☞ 今回取り上げる例はほとんどがこの方法を用います

オブジェクト表現の例

```
class Node
    def name
        @name
    end
    def edges
    ...
end
```

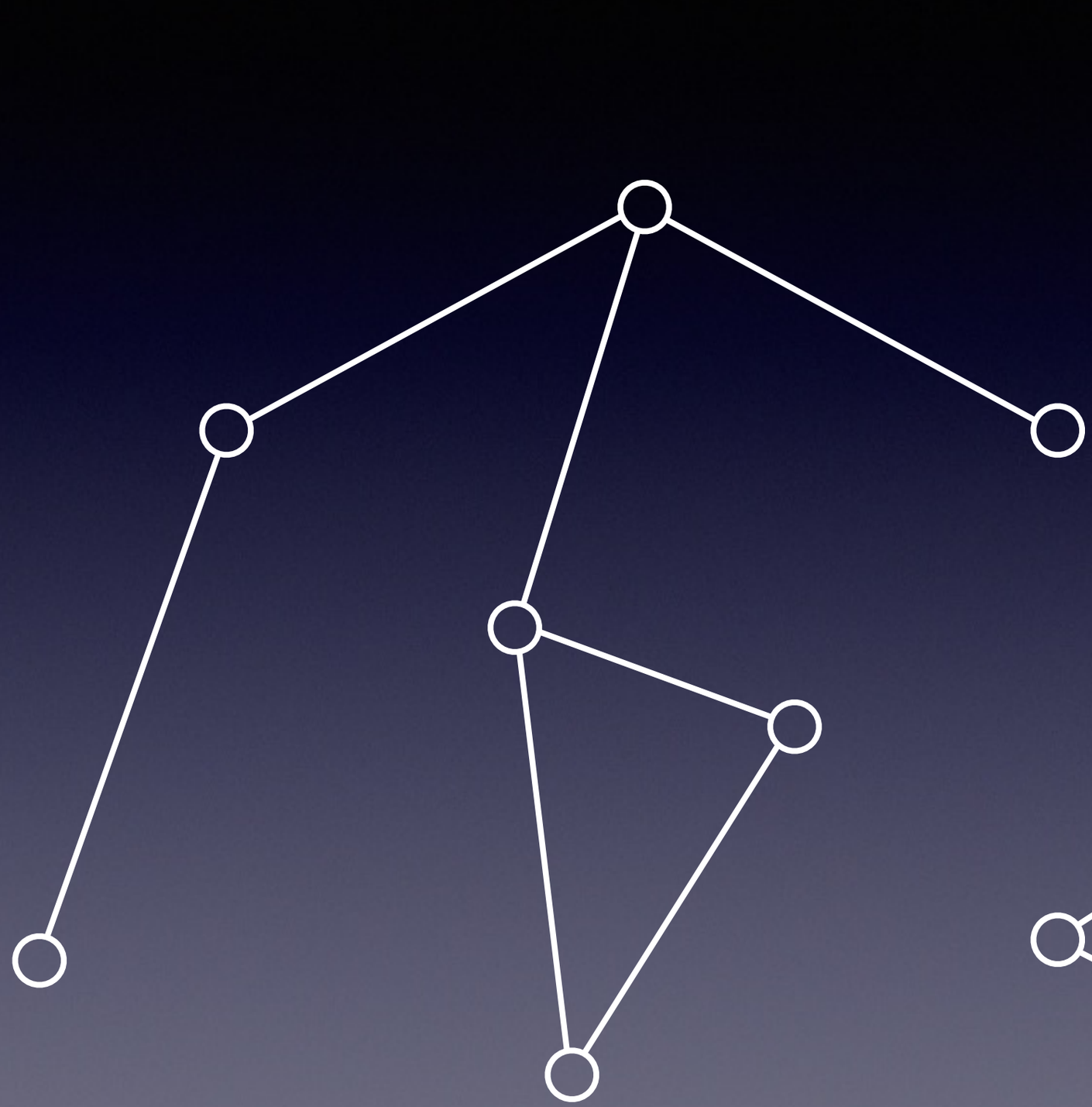
```
class Edge
    def name
        @name
    end
    def origin
    ...
end
```

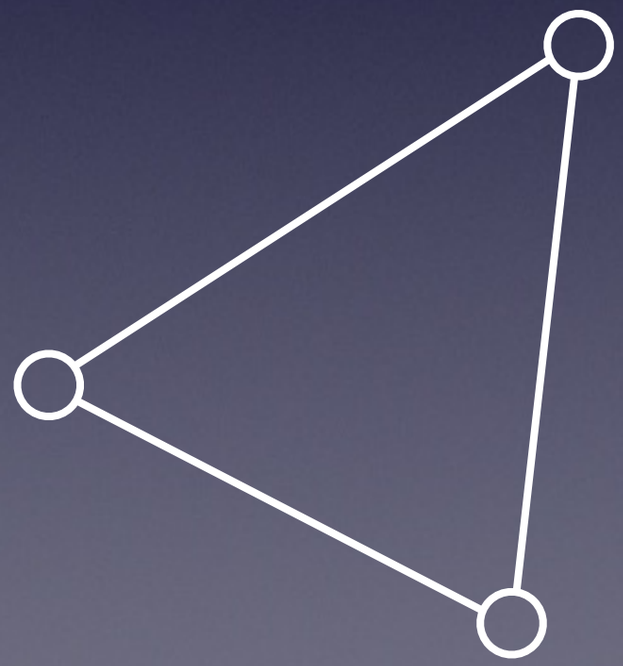
```
v.edges.each do |e|
    puts e.name
    puts e.origin.name
    puts e.destination.name
    ...
end
```

グラフの探索（深さ優先探索）

- 深さ優先探索（Depth-First Search: DFS）は与えられたグラフに “全域木が存在するか” と “閉路が存在するか” を効率的に判定することができるアルゴリズム
- あるノードを出発しエッジに沿って未探索のノードへ移動していくことで探索を続ける
- 探索中のノードから移動できる未探索のノードがなくなったら一つ前のノードに戻る
- 最初に出発したノードに戻ったとき未探索のノードが存在する場合はそのうちのひとつをあらたに出発点としておなじ処理を繰り返し未探索のノードがなくなるまで続ける

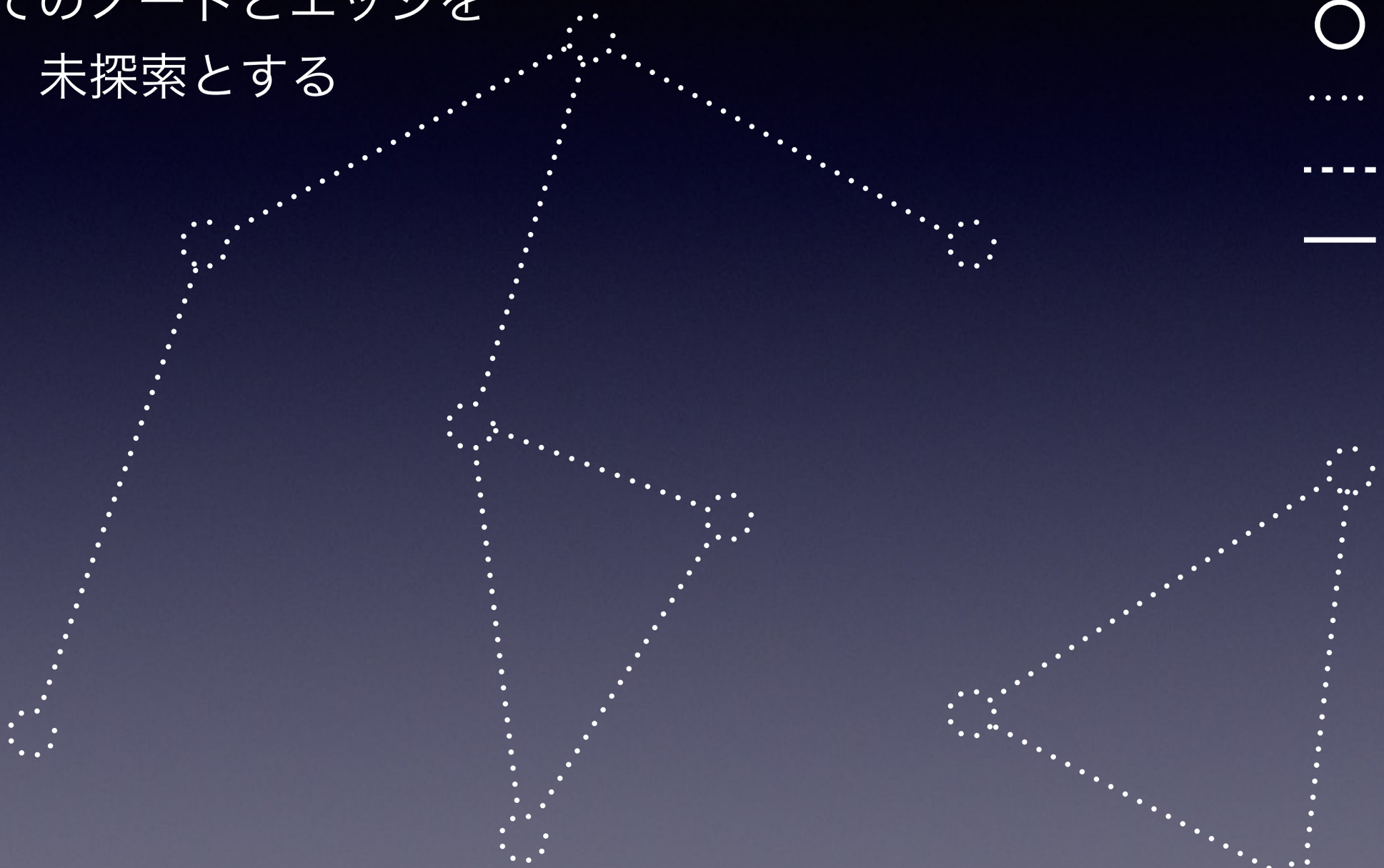
グラフの探索（深さ優先探索）





グラフの探索（深さ優先探索）

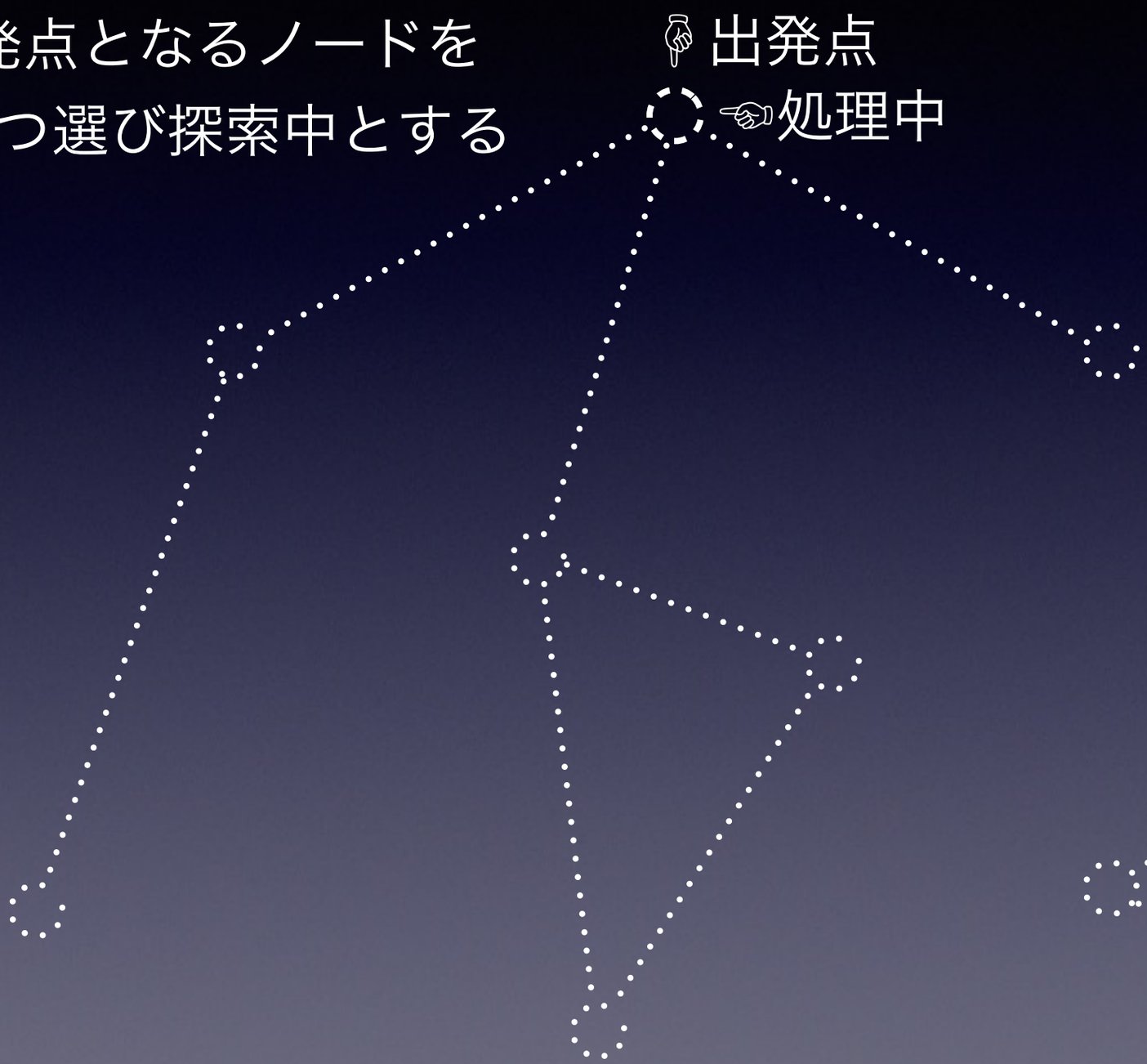
すべてのノードとエッジを
未探索とする



未探索
探索中
探索済
未探索
探索済
探索枝

グラフの探索（深さ優先探索）

出発点となるノードを
ひとつ選び探索中とする



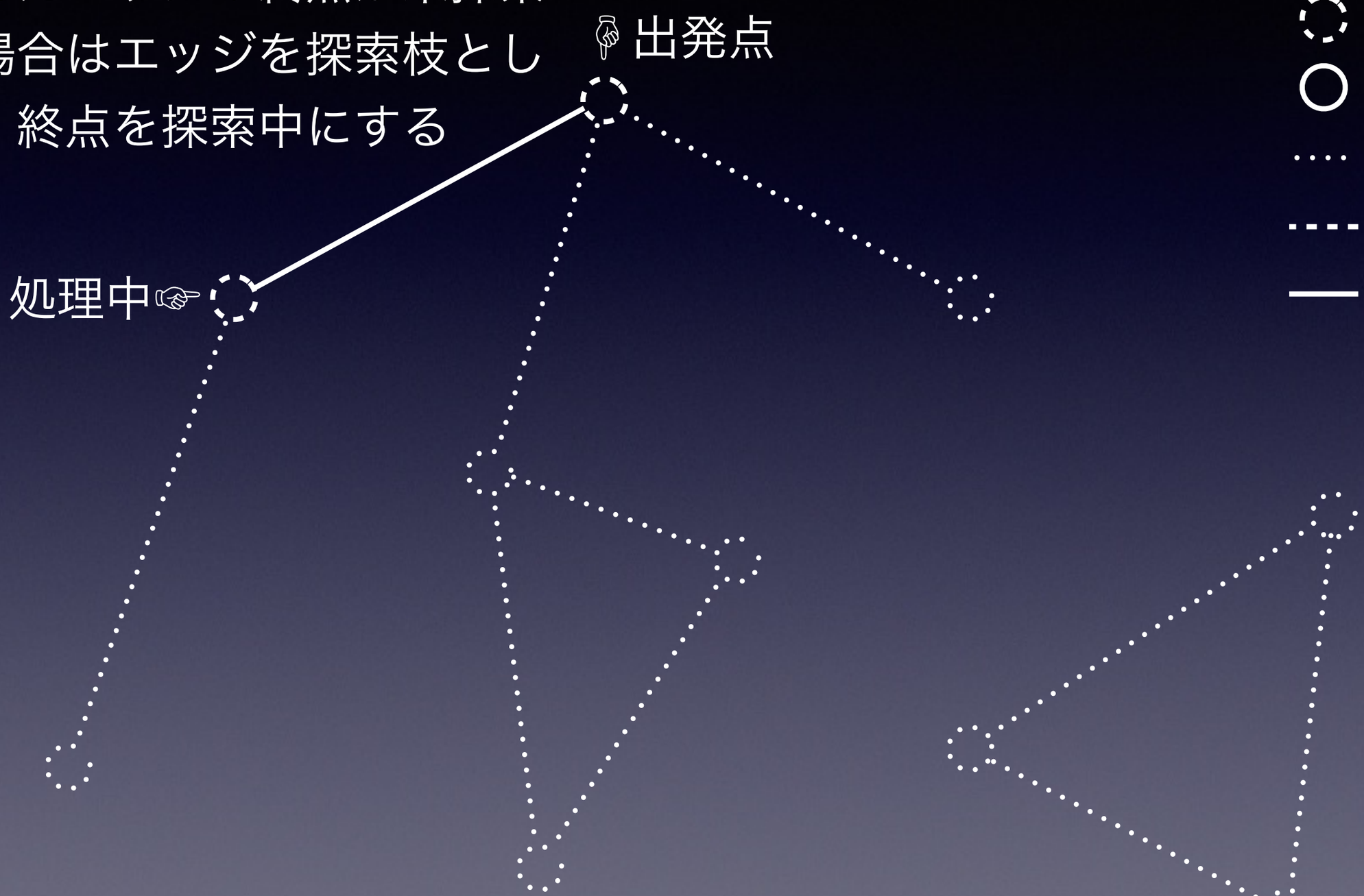
未探索
探索中
探索済
未探索
探索済
探索枝

グラフの探索（深さ優先探索）

未探索のエッジをひとつ選び
探索済みに変更

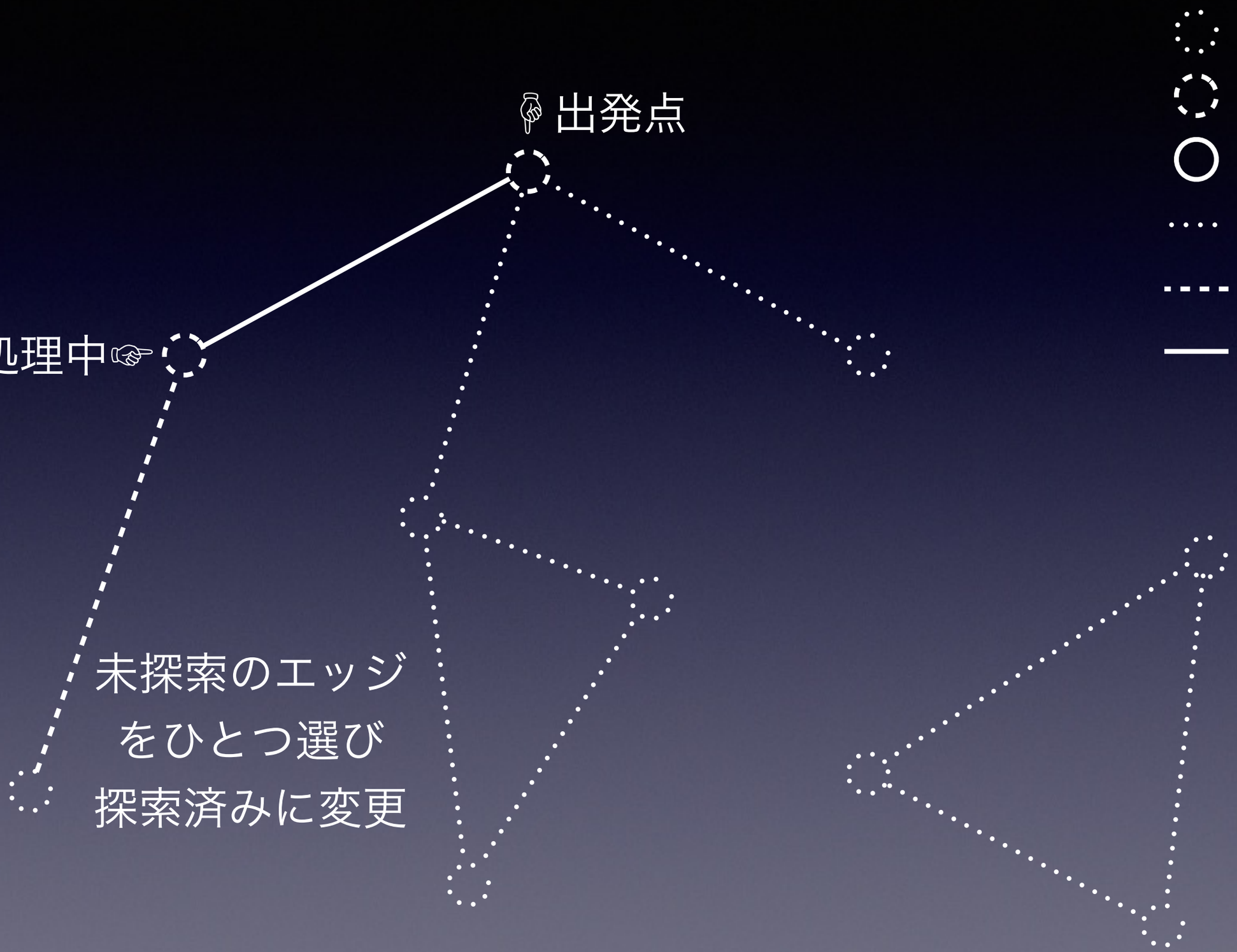
グラフの探索（深さ優先探索）

選んだエッジの終点が未探索
の場合はエッジを探索枝とし
終点を探索中にする

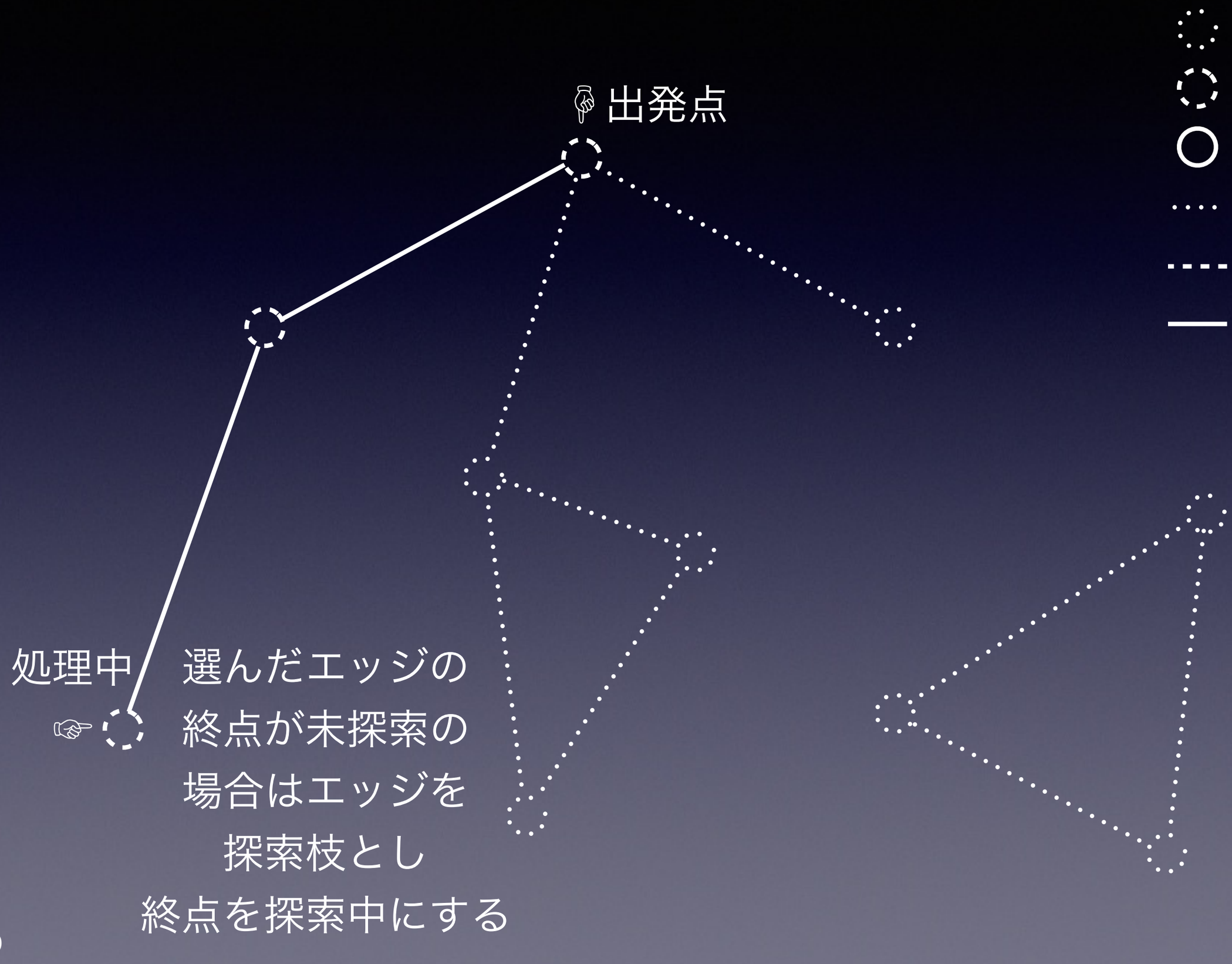


未探索
探索中
探索済
未探索
探索済
探索枝

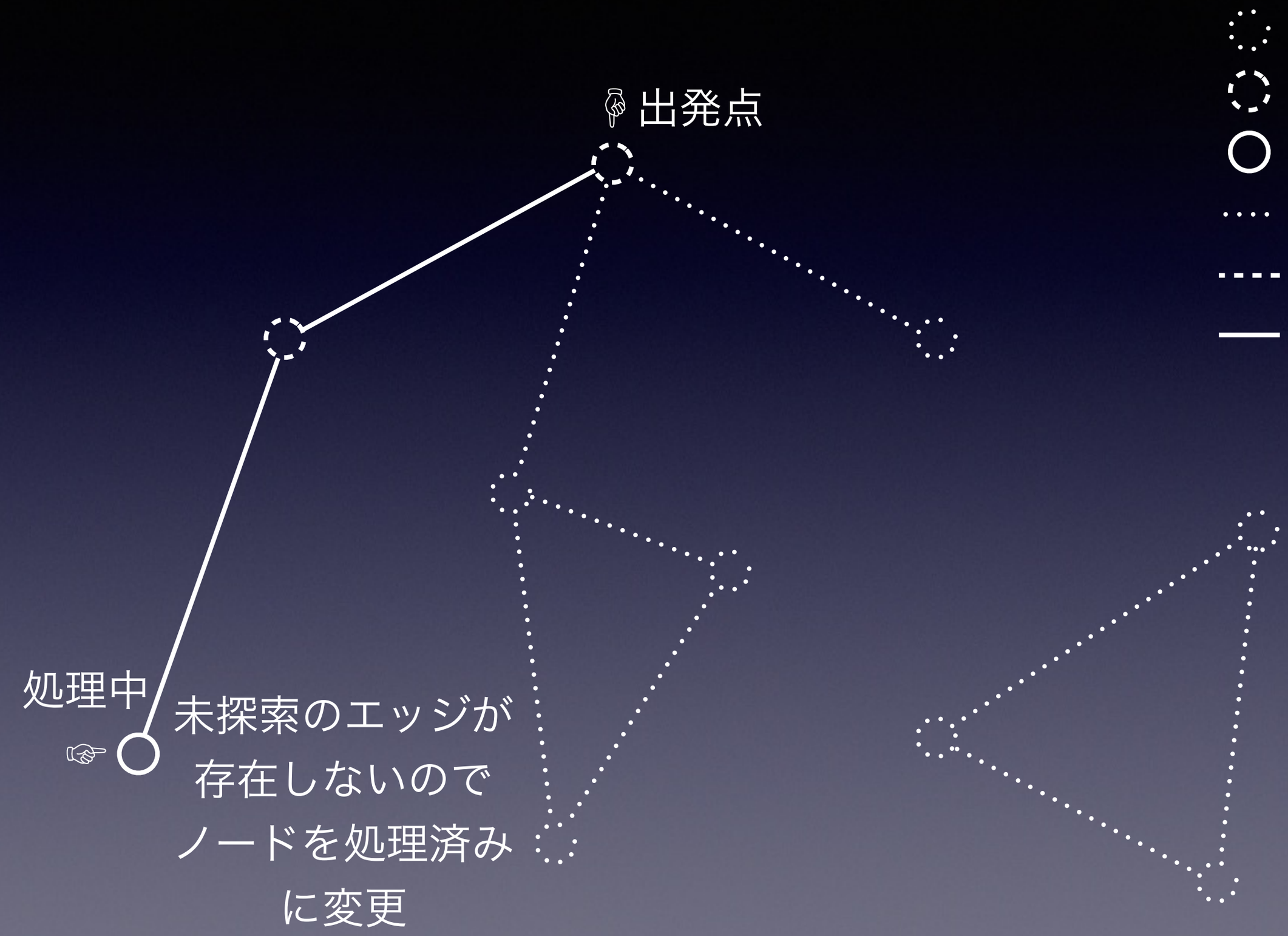
グラフの探索（深さ優先探索）



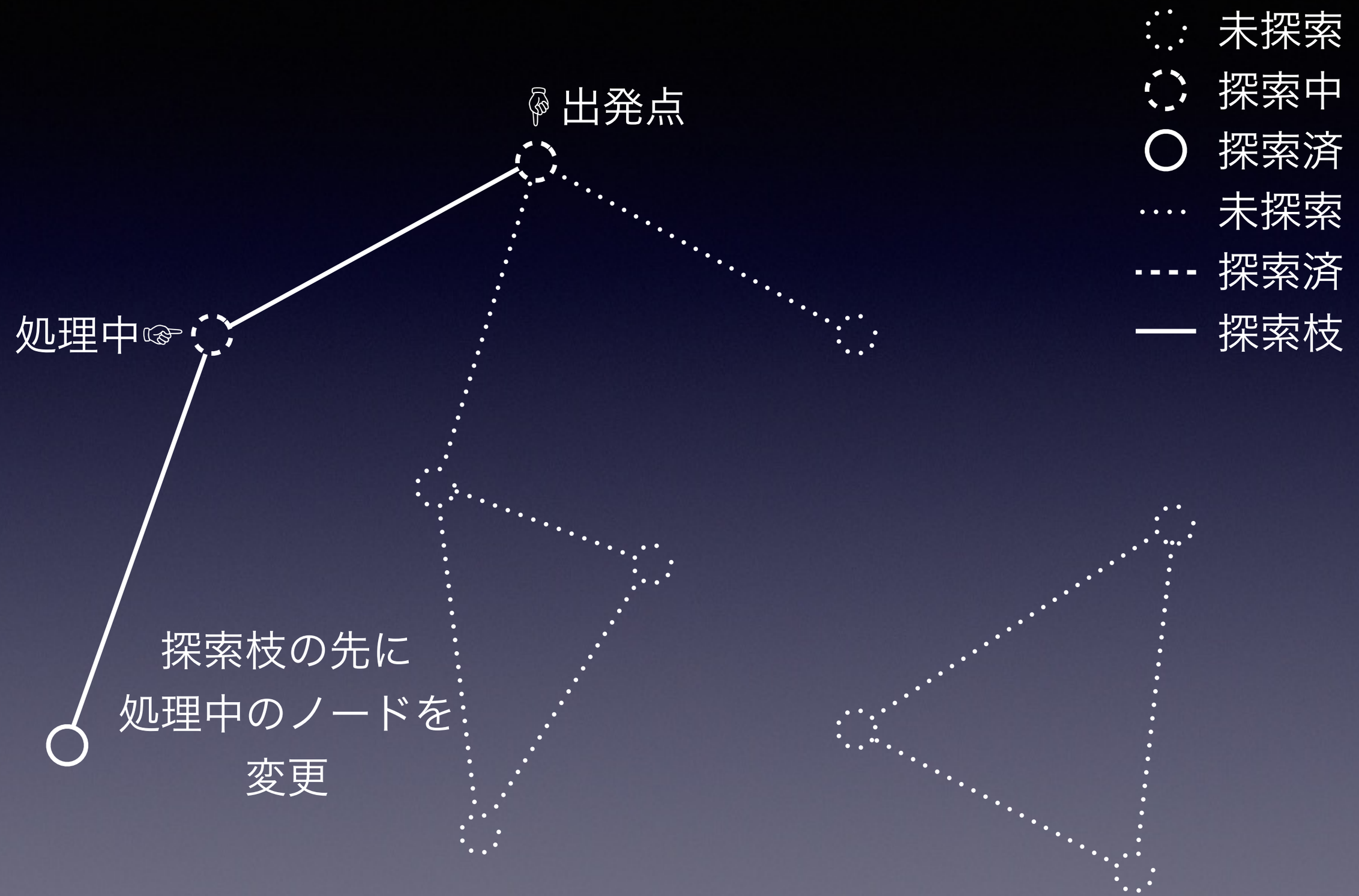
グラフの探索（深さ優先探索）



グラフの探索（深さ優先探索）

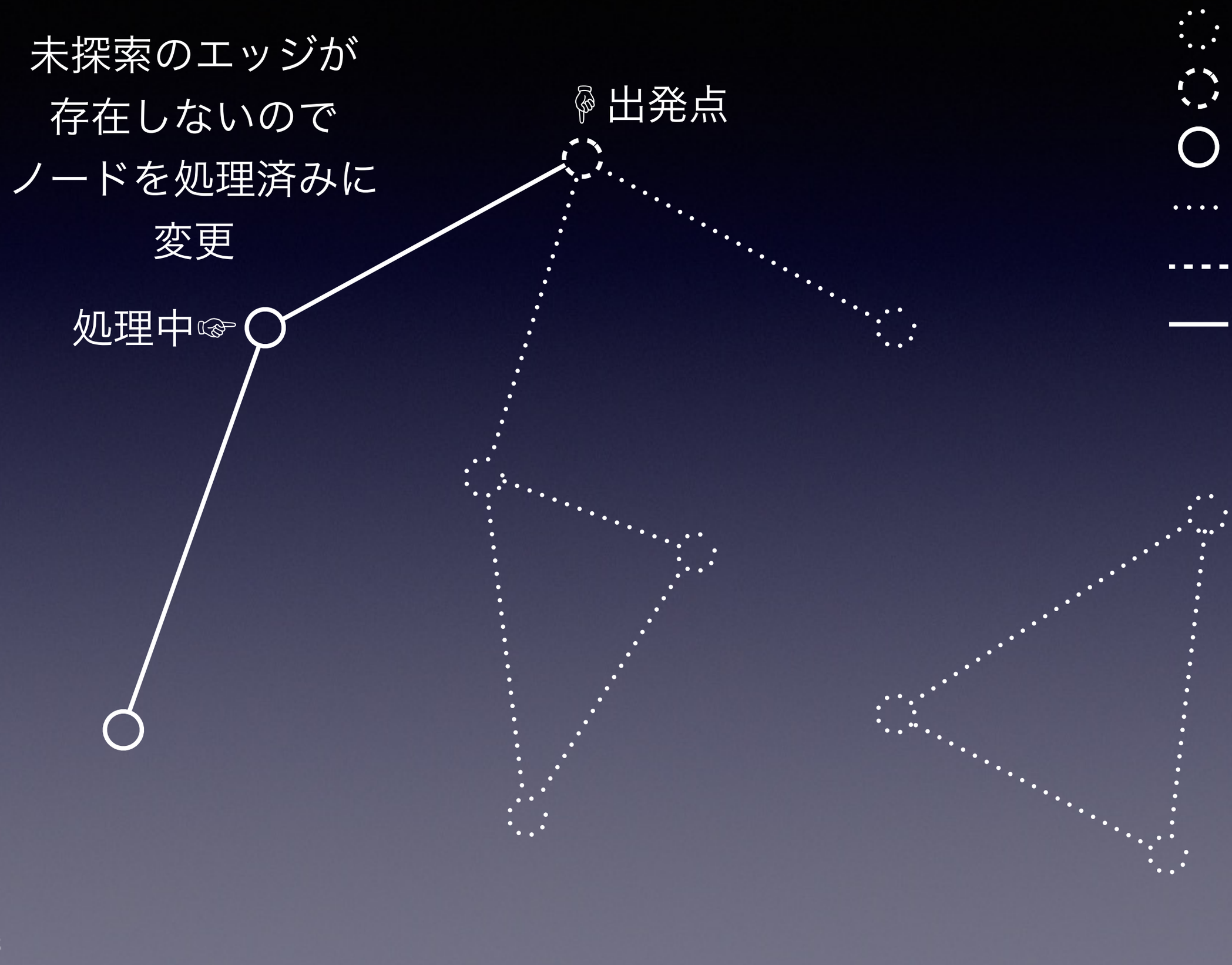


グラフの探索（深さ優先探索）

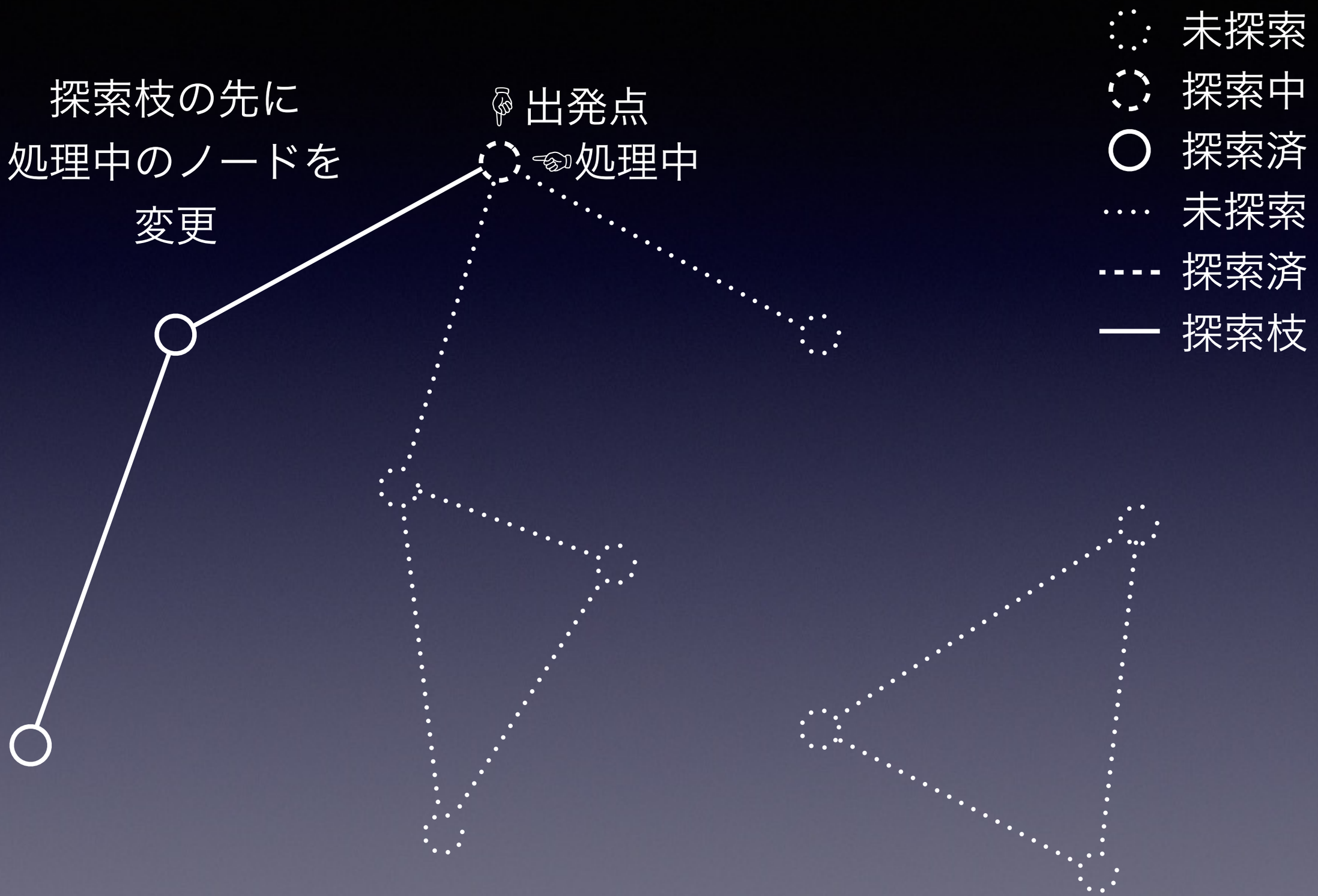


グラフの探索（深さ優先探索）

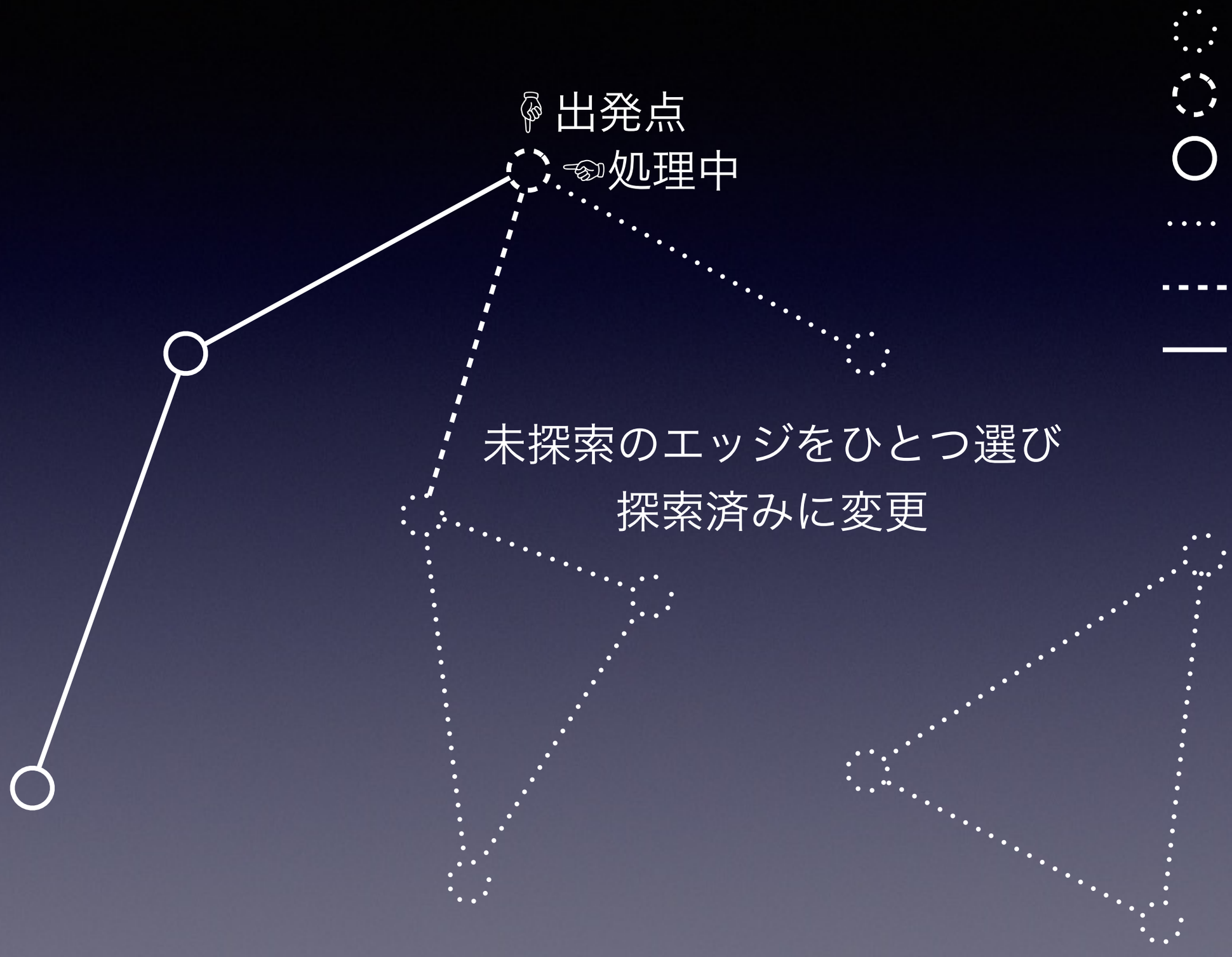
未探索のエッジが
存在しないので
ノードを処理済みに
変更



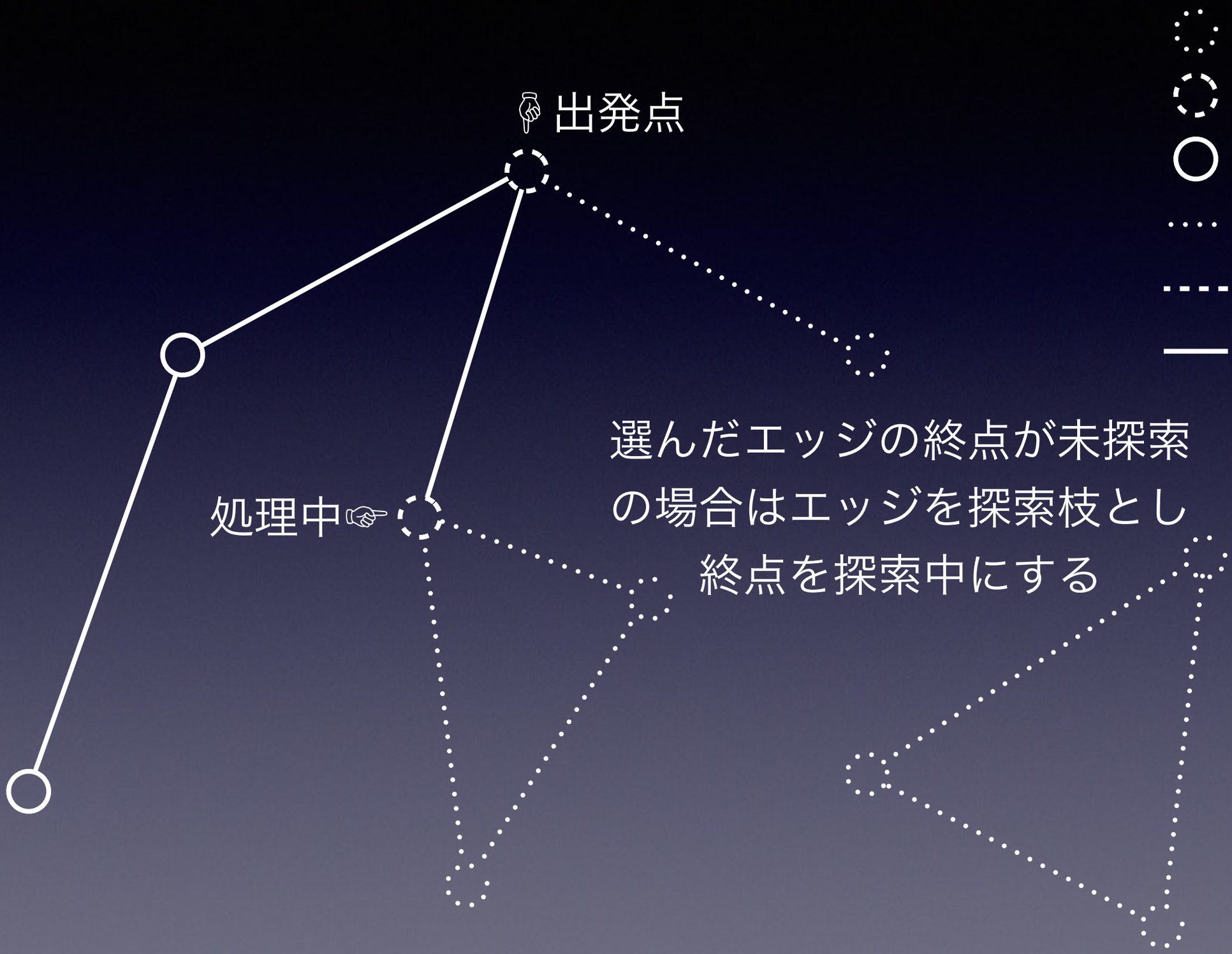
グラフの探索（深さ優先探索）



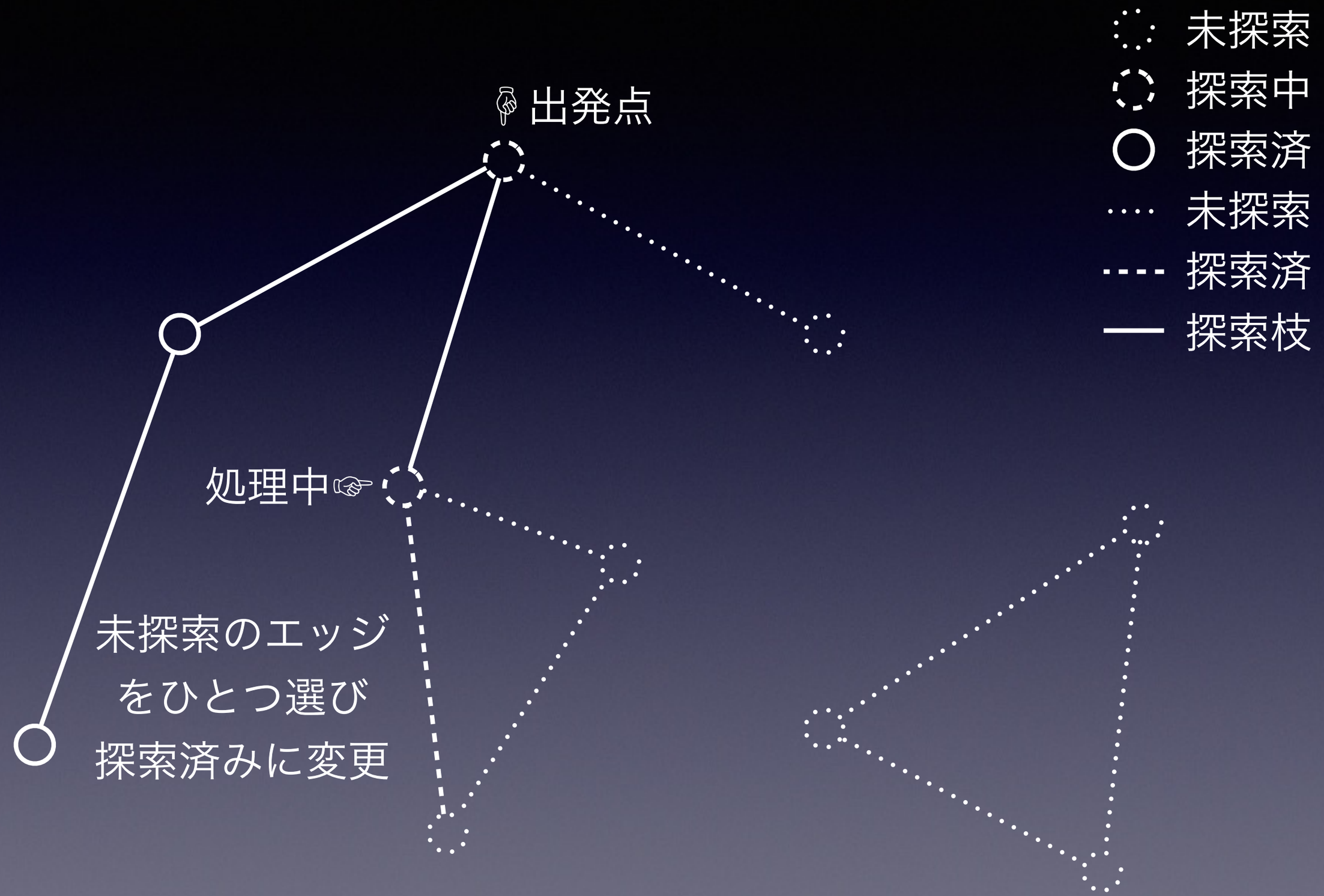
グラフの探索（深さ優先探索）



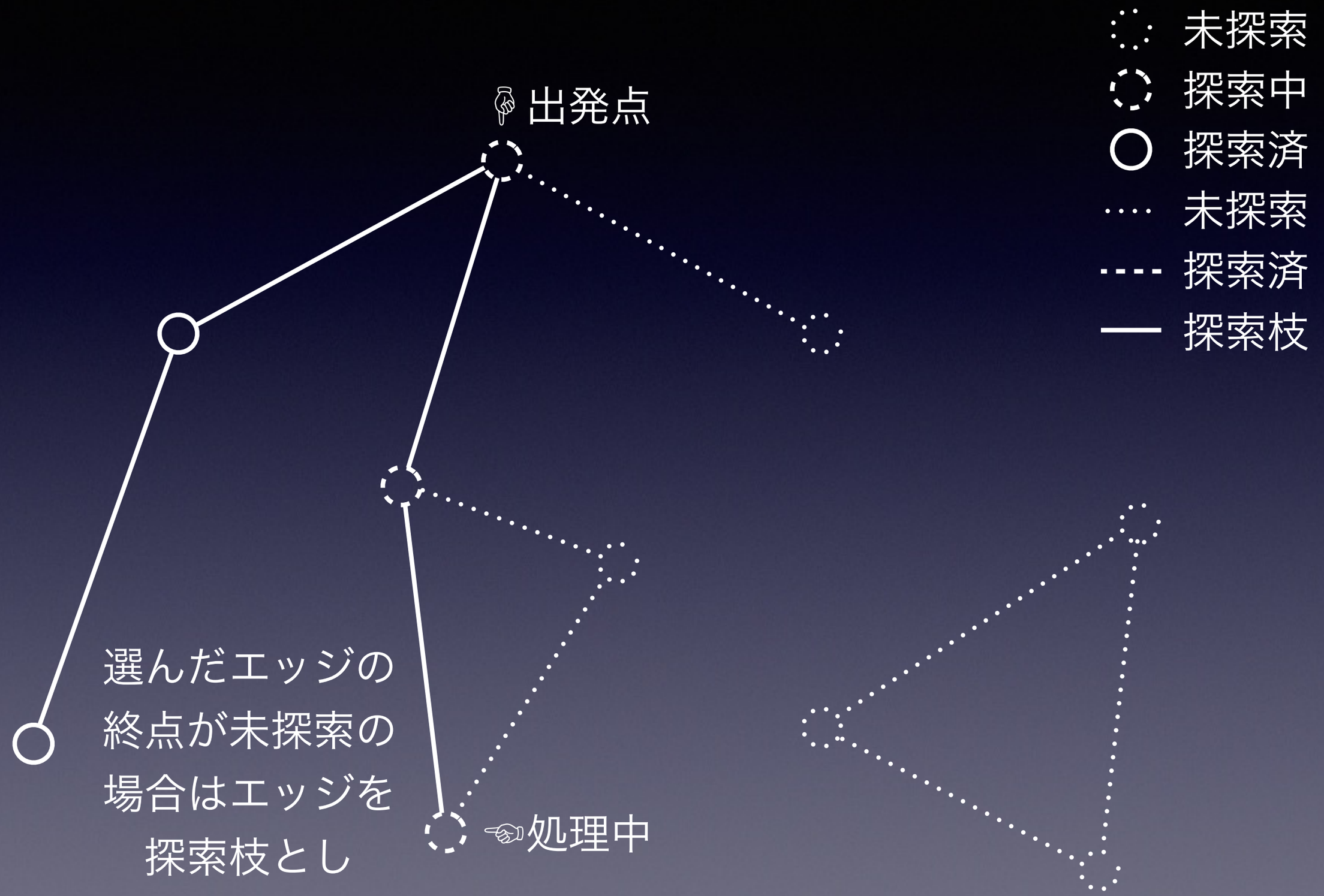
グラフの探索（深さ優先探索）



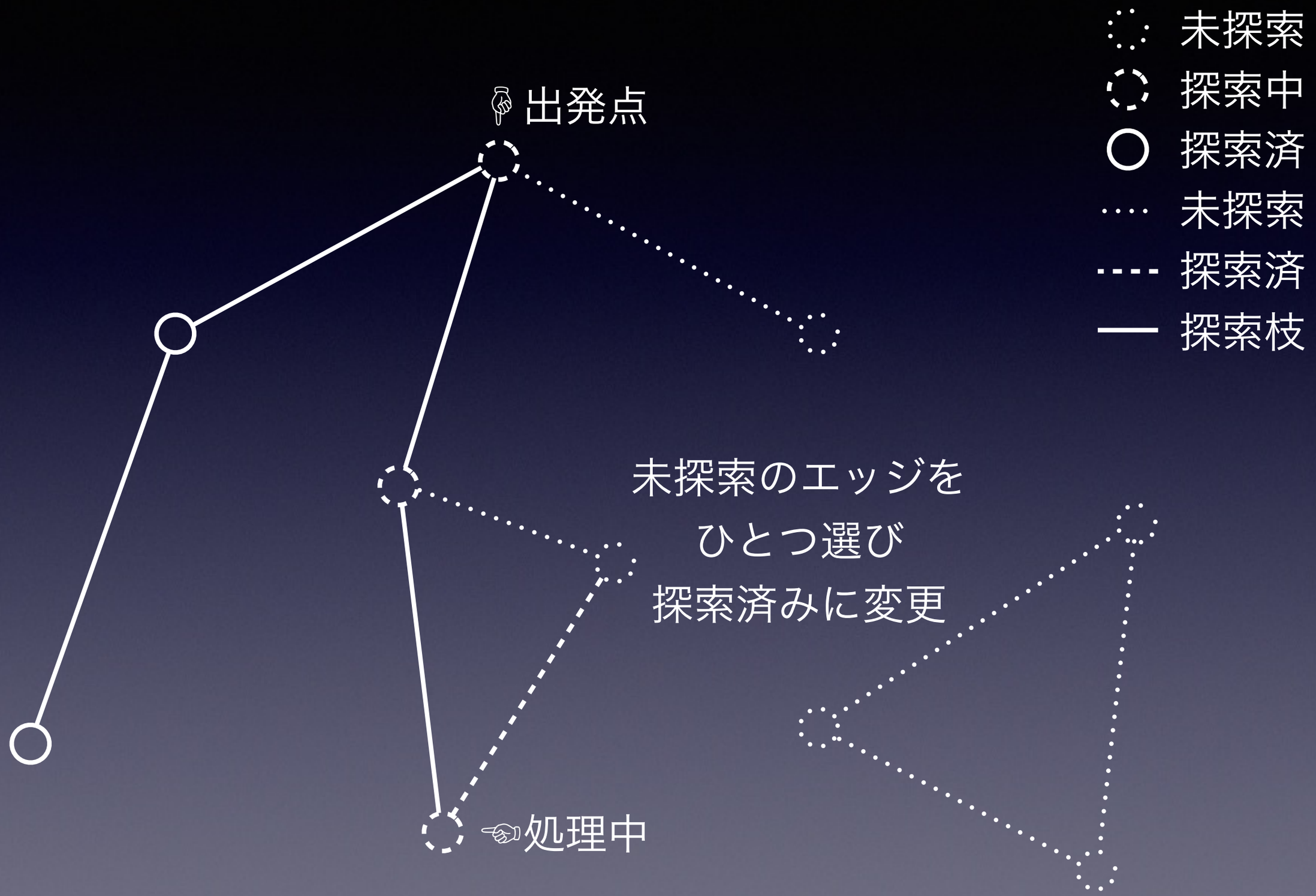
グラフの探索（深さ優先探索）



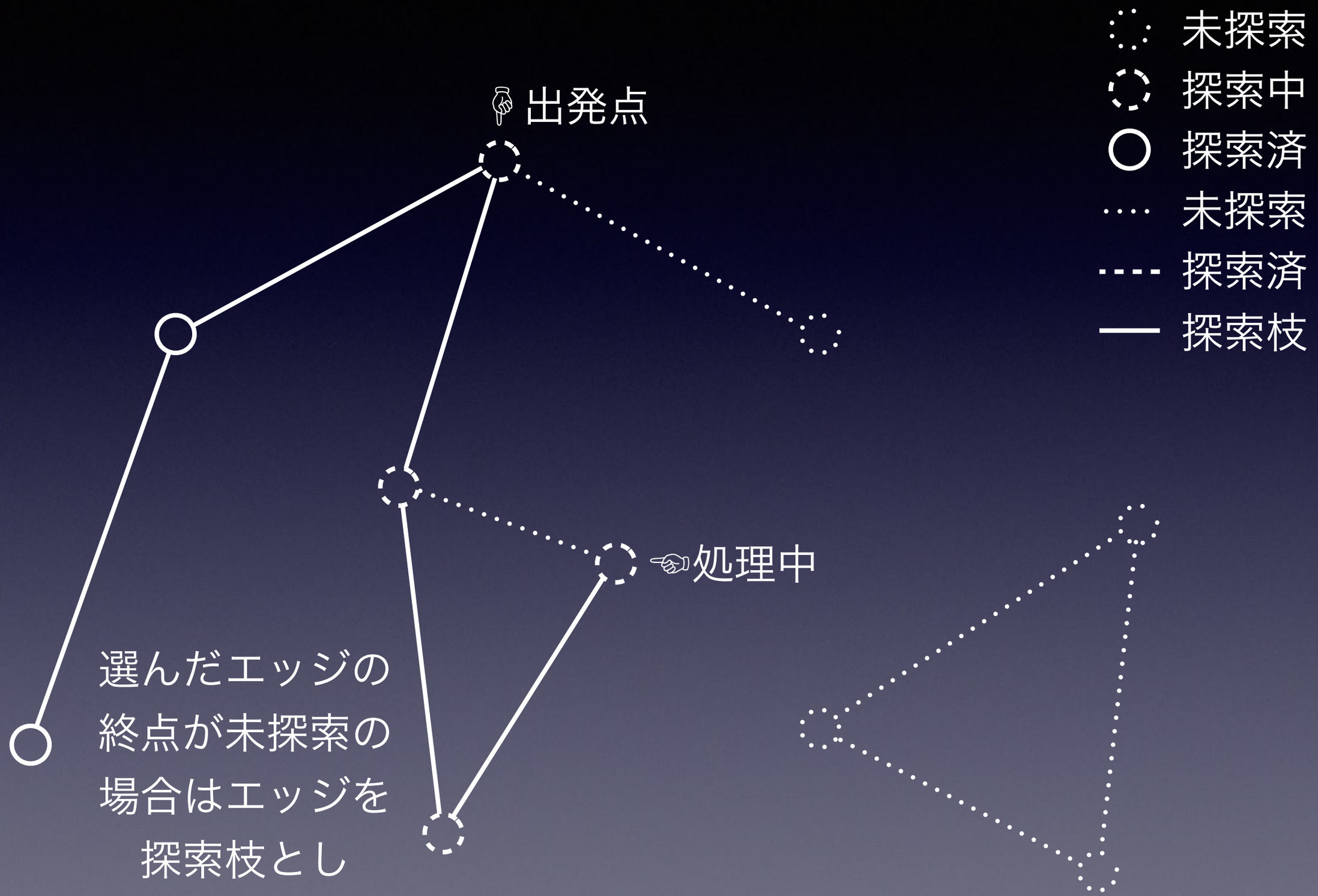
グラフの探索（深さ優先探索）



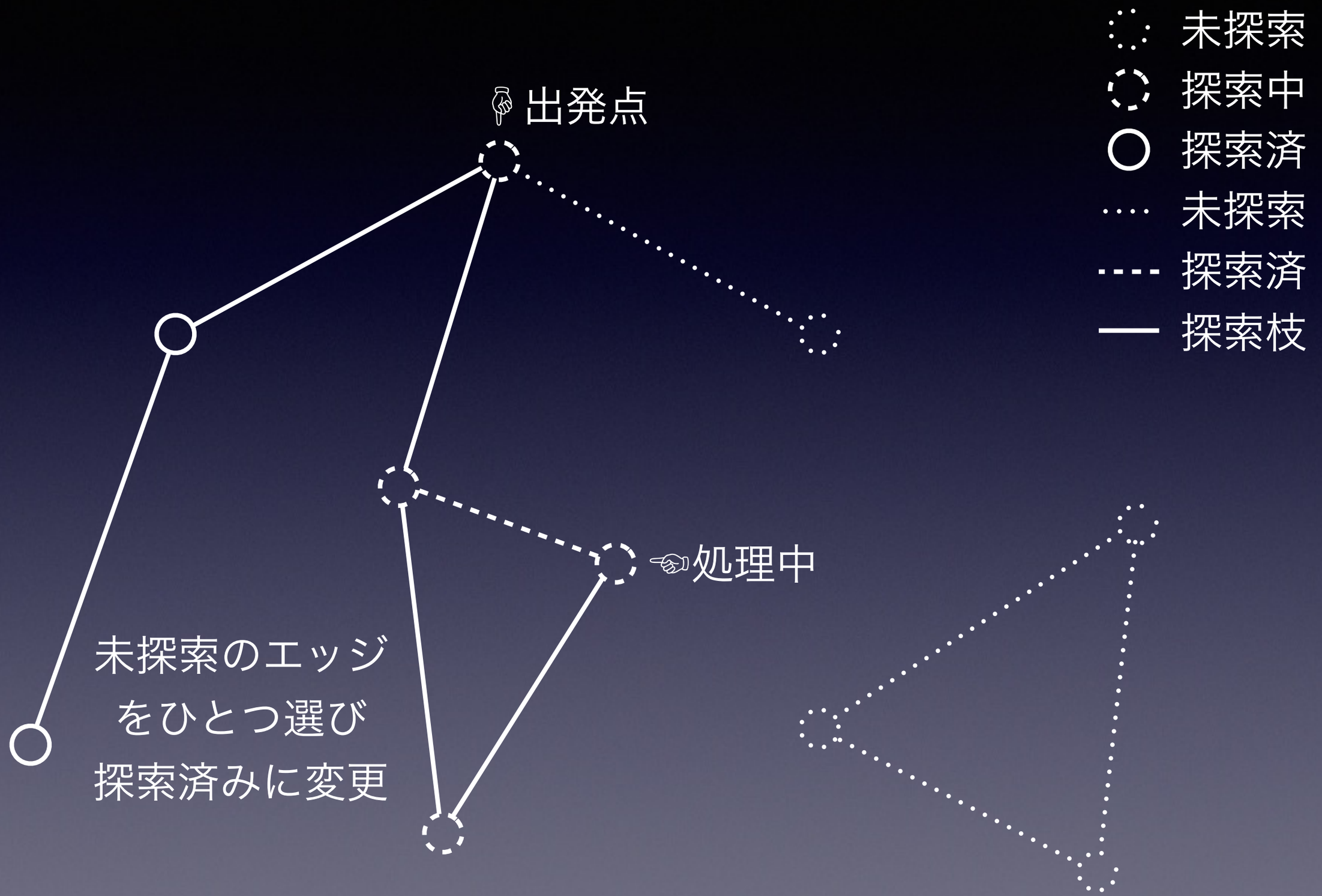
グラフの探索（深さ優先探索）



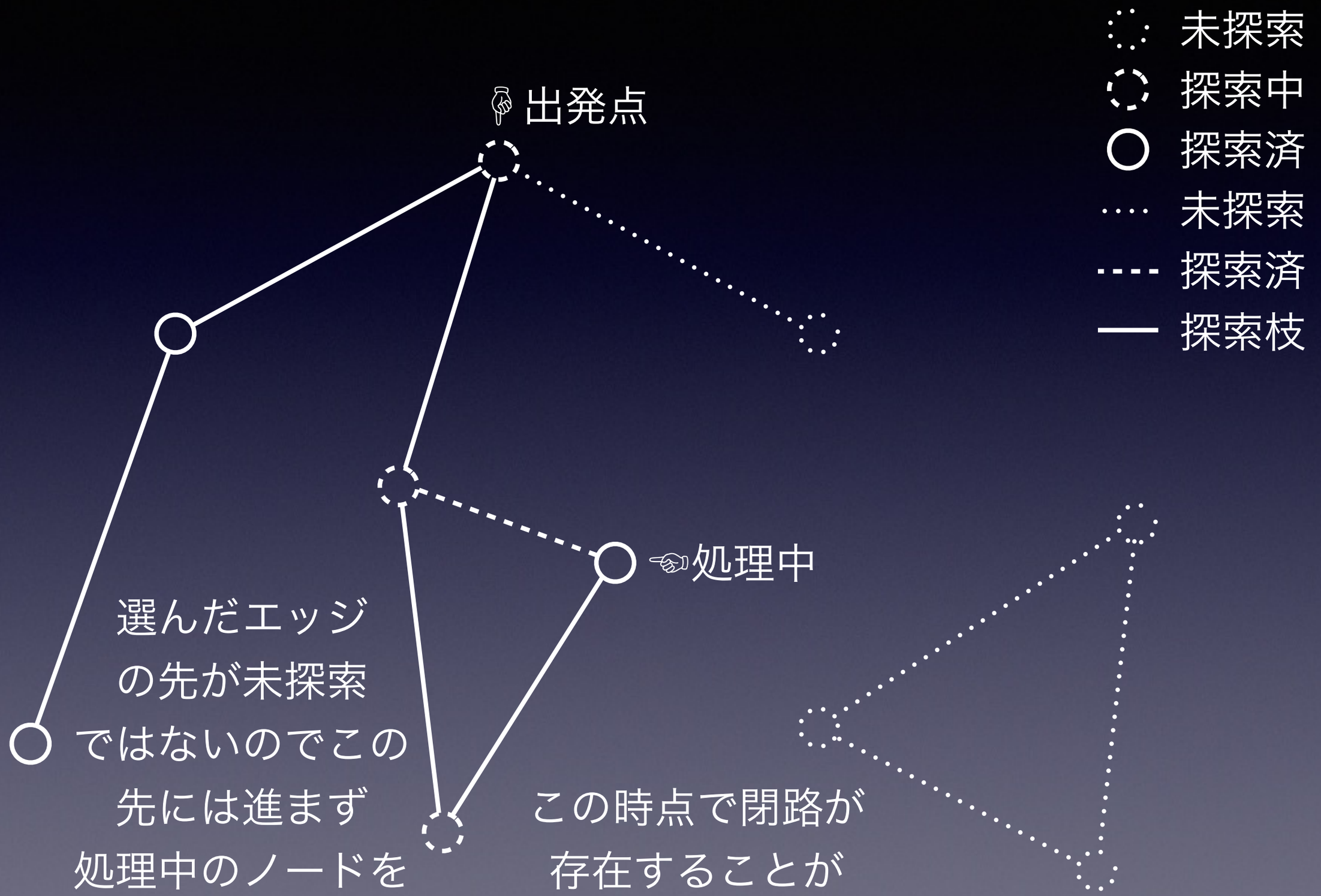
グラフの探索（深さ優先探索）



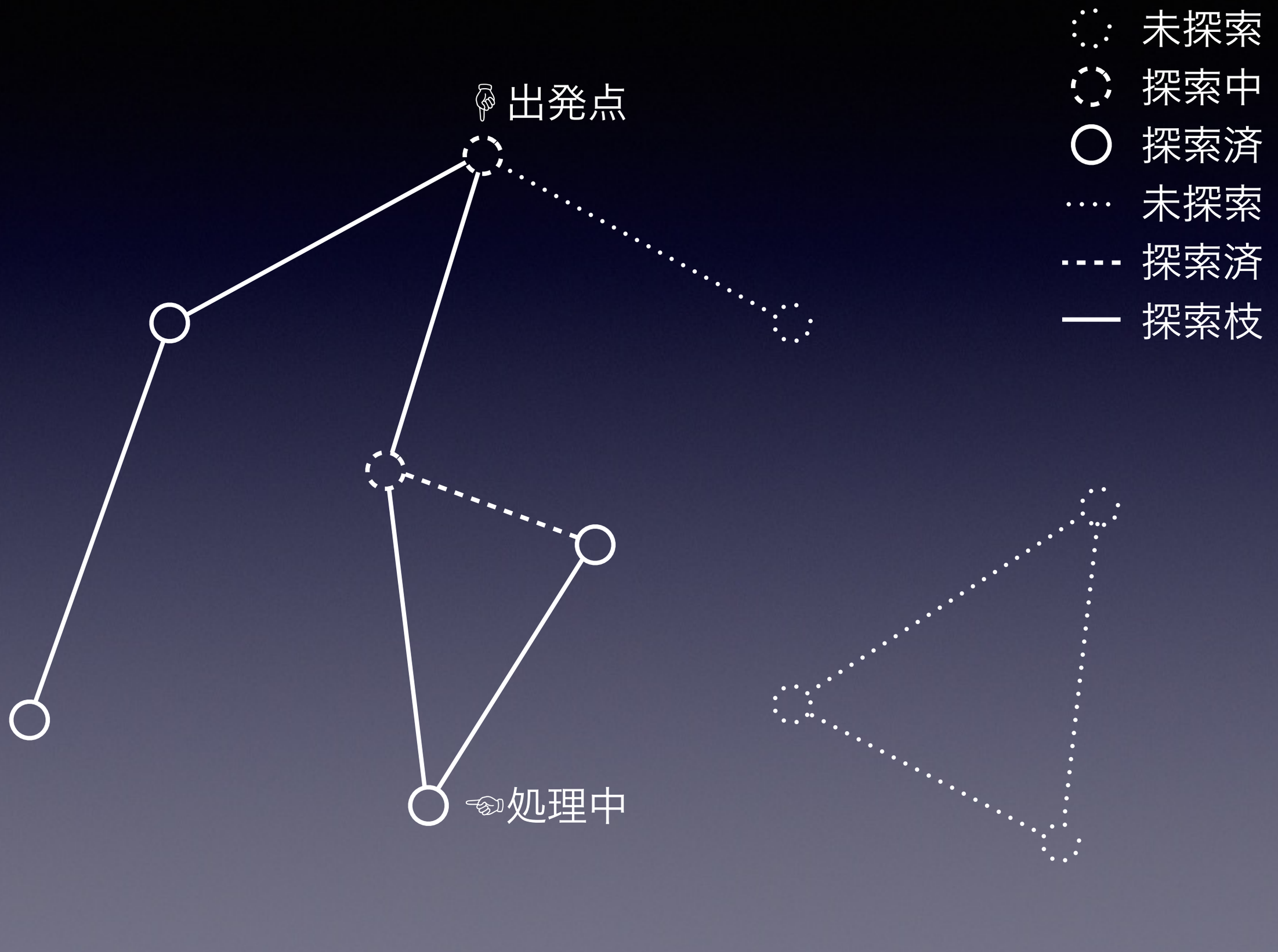
グラフの探索（深さ優先探索）



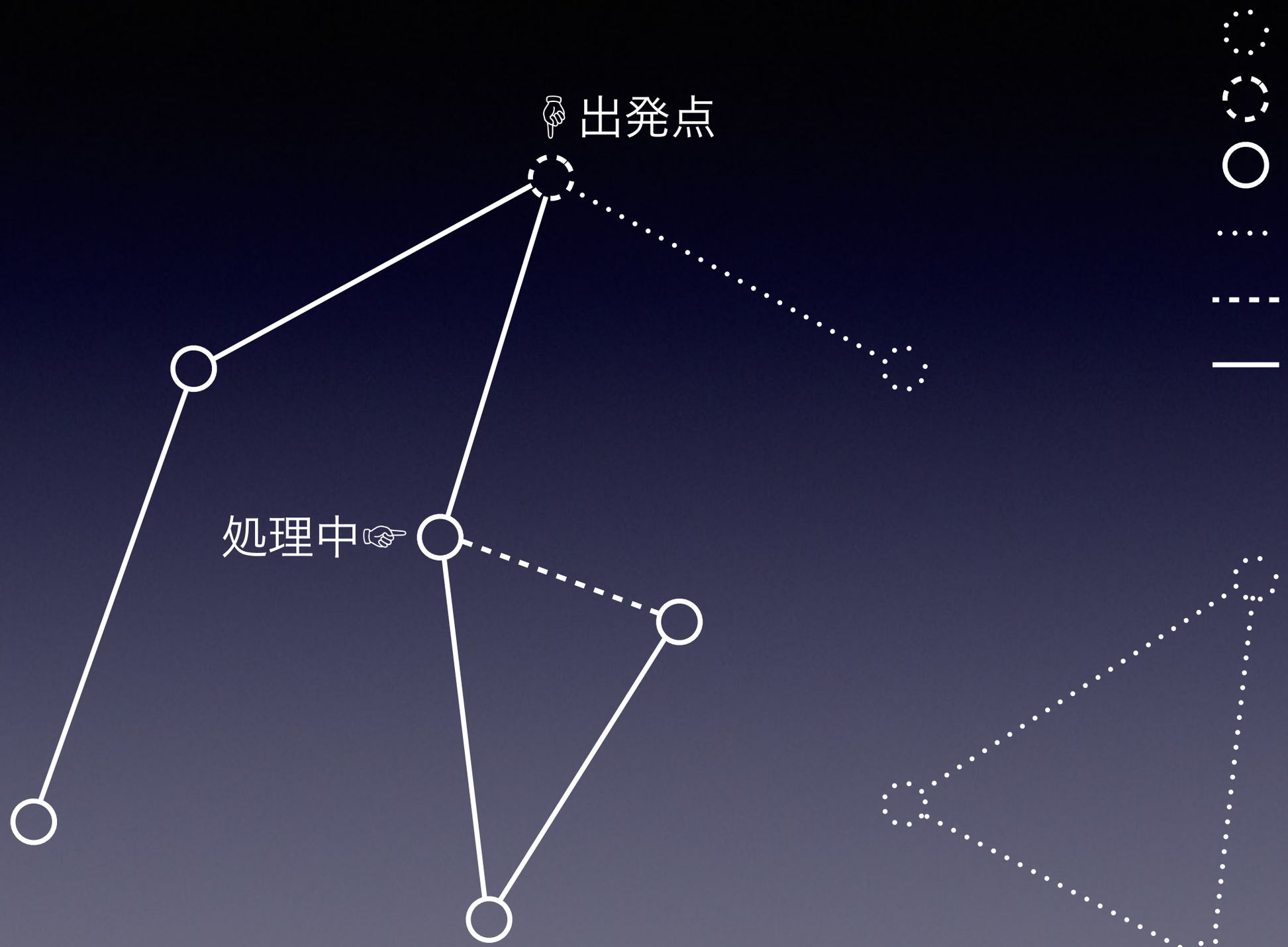
グラフの探索（深さ優先探索）



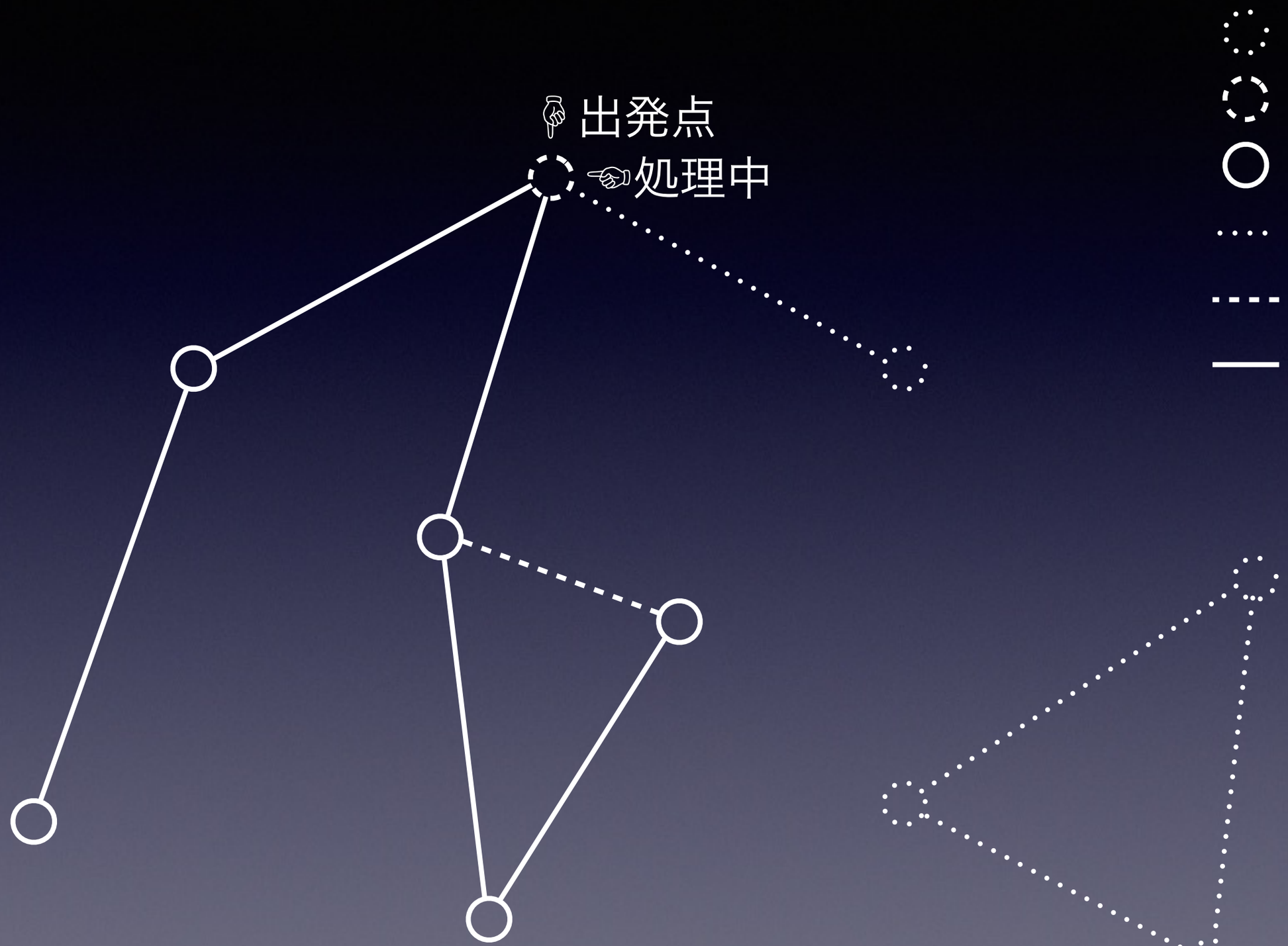
グラフの探索（深さ優先探索）



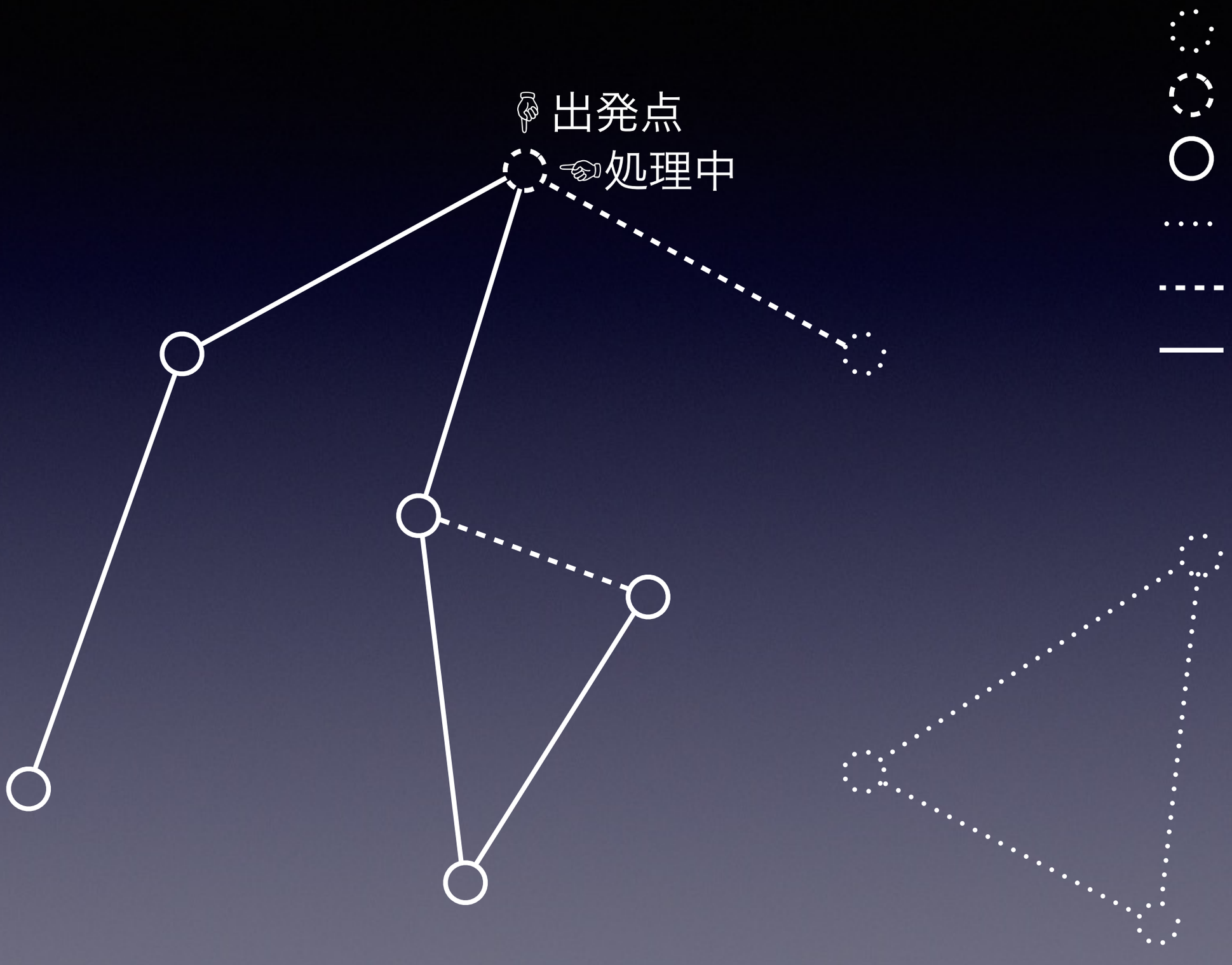
グラフの探索（深さ優先探索）



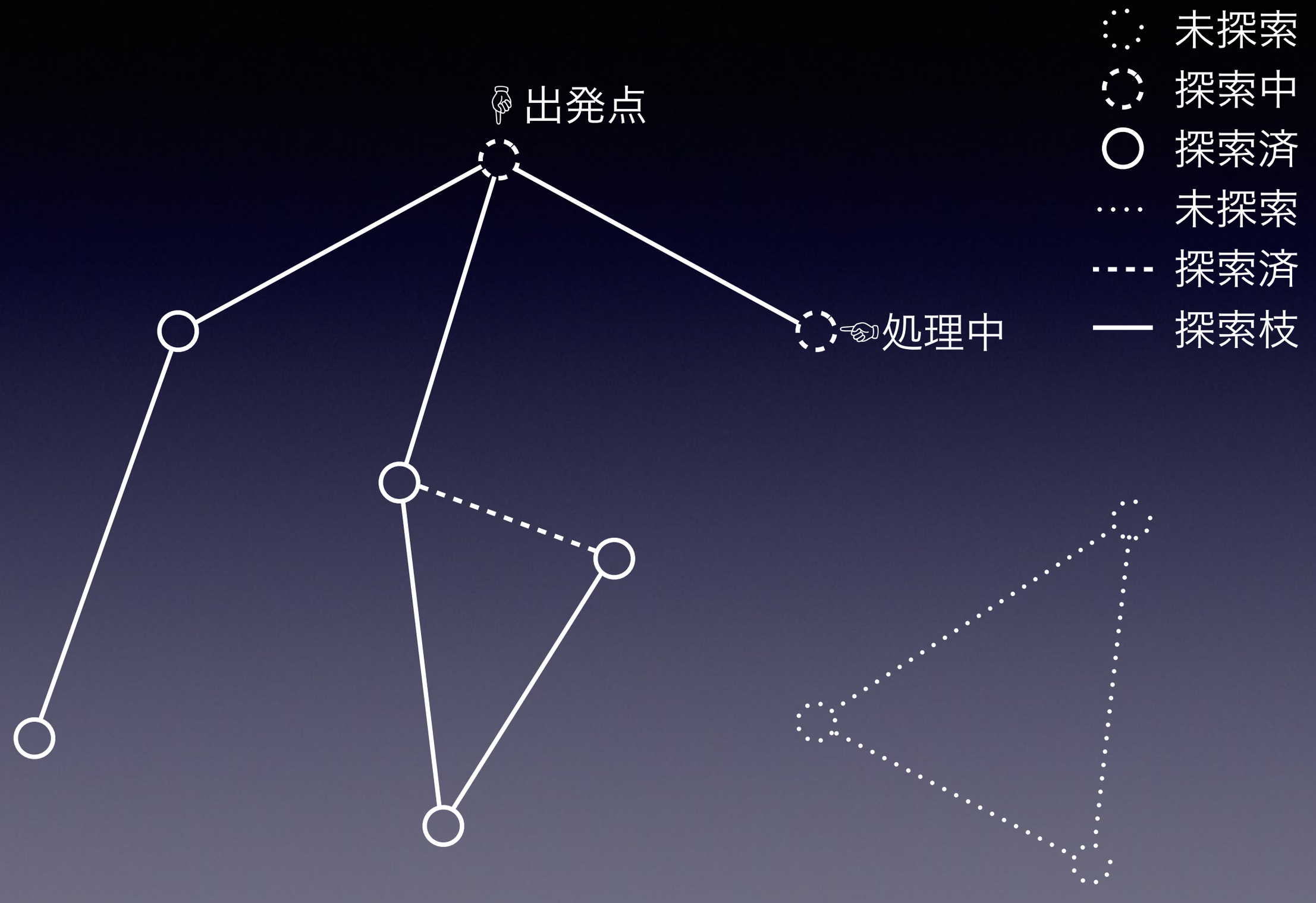
グラフの探索（深さ優先探索）



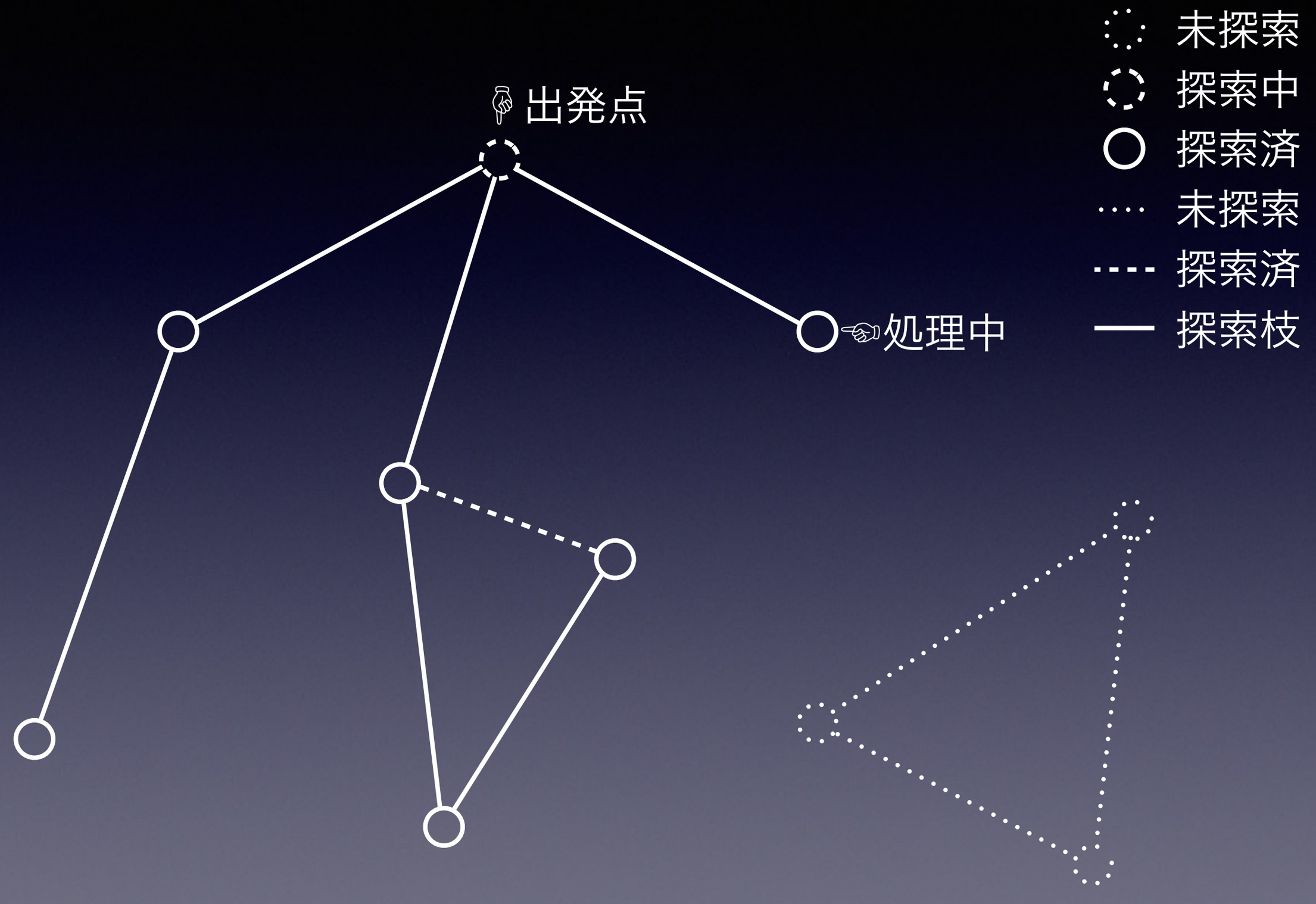
グラフの探索（深さ優先探索）



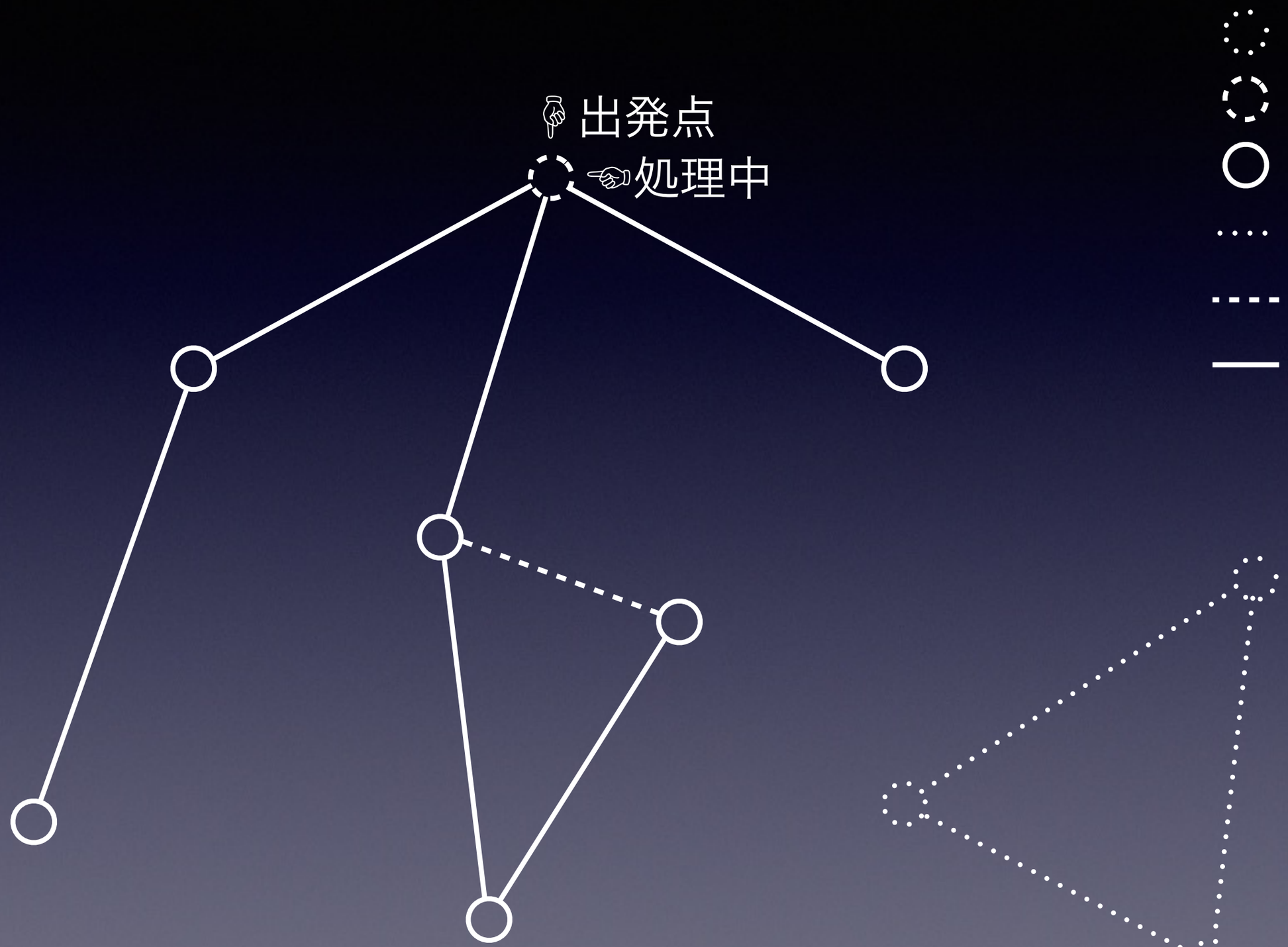
グラフの探索（深さ優先探索）



グラフの探索（深さ優先探索）



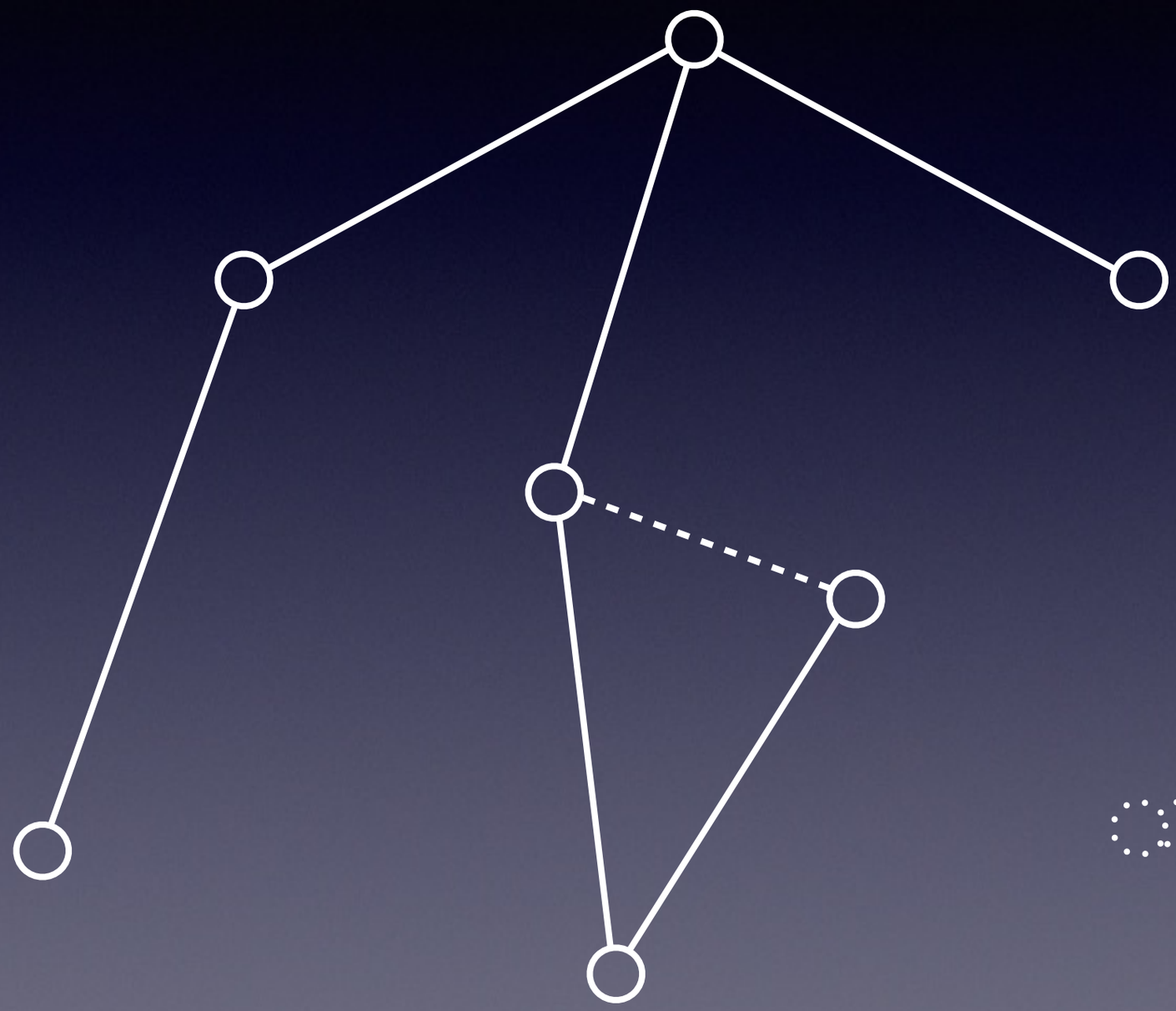
グラフの探索（深さ優先探索）



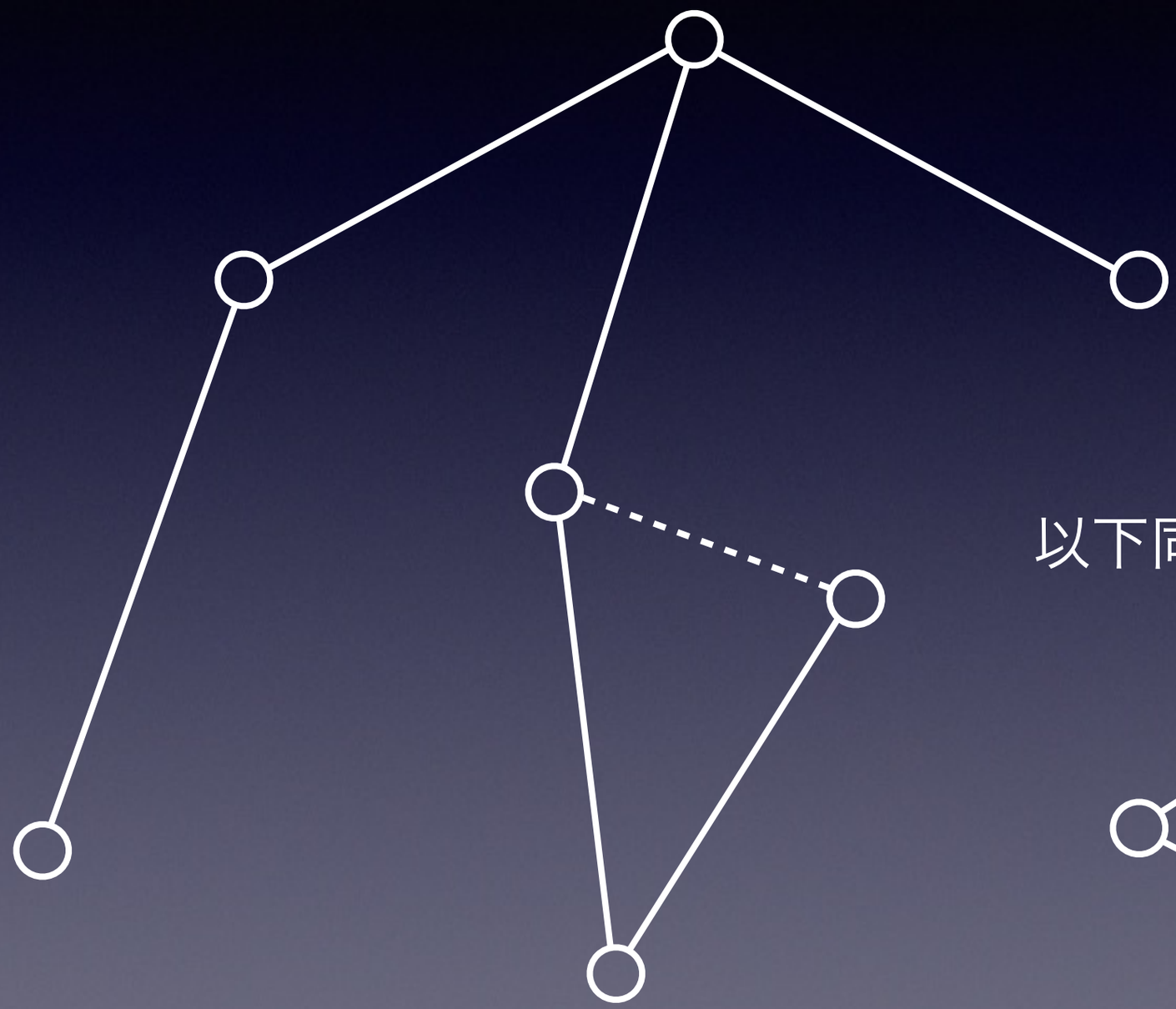
グラフの探索（深さ優先探索）

この時点で
連結でないことが
判断できる！

グラフの探索（深さ優先探索）



グラフの探索（深さ優先探索）



以下同様…

引き続きプログラムをお楽しみください...

- 導入
  - 背景、用語の説明、グラフの表現方法と探索

    浅間（３０分）
- グラフ理論とネットワークのアルゴリズムの基礎
  - 最短路問題、最小木問題、アルゴリズムと計算量

    伊波さん（５０分）
- ネットワークフローとその代表的な問題
  - 最大流問題、多品種流問題

    金子さん（５０分）
- まとめ
  - システム最適化流問題、参考情報、まとめ

    浅間（２０分）