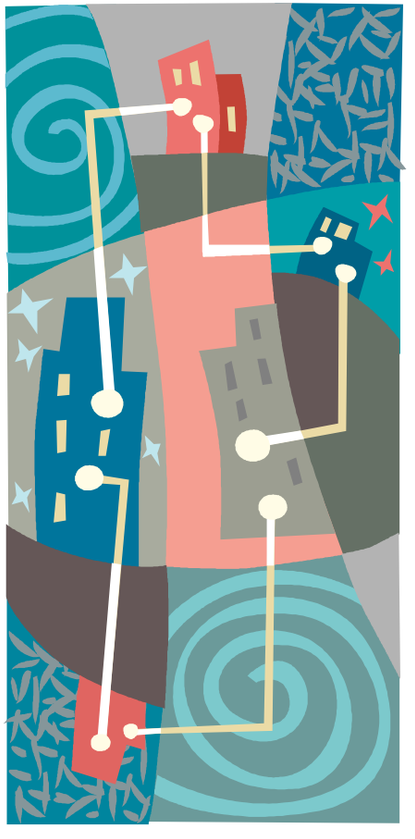
Graph Theory

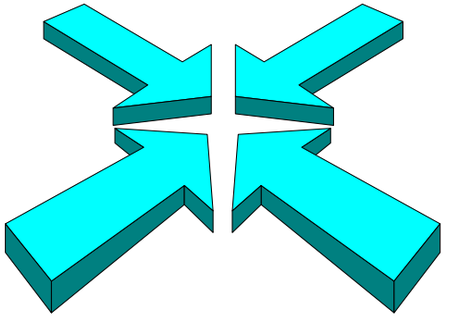
点と線で表現する

ここで学ぶこと

- 物事を表現する方法のひとつ：
  グラフ・ネットワーク
- グラフ・ネットワーク上の問題例
- グラフの扱い方



点と線で表現する

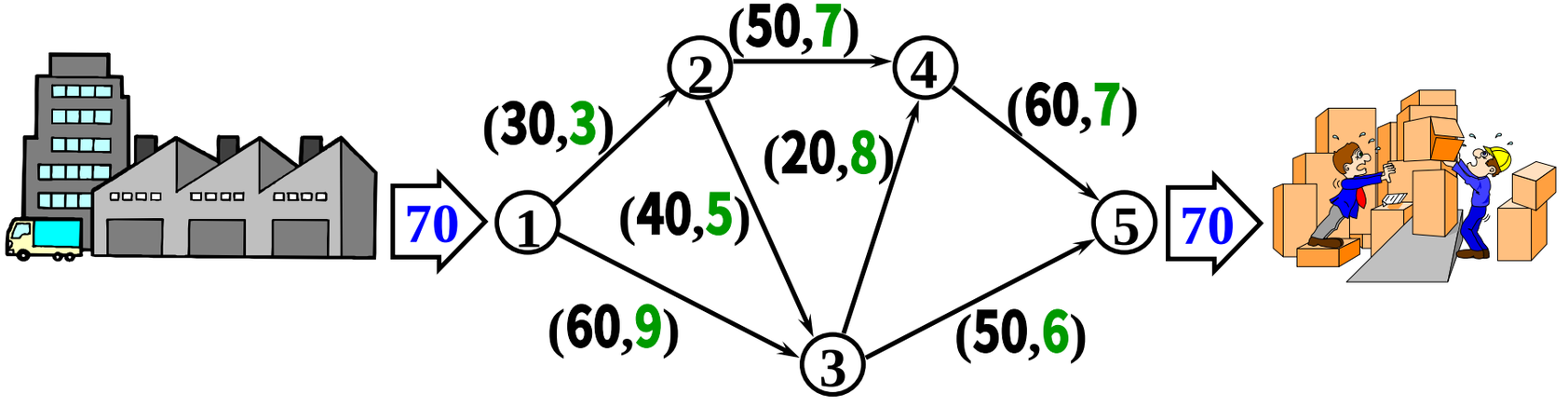


本質的に点と線で表現できるシステムは多い

- 例えば?

- 昔から考察の対象
  →グラフ理論の発生，ネットワーク計画
  →OR：モデル表現の大きな柱

ネットワークの例

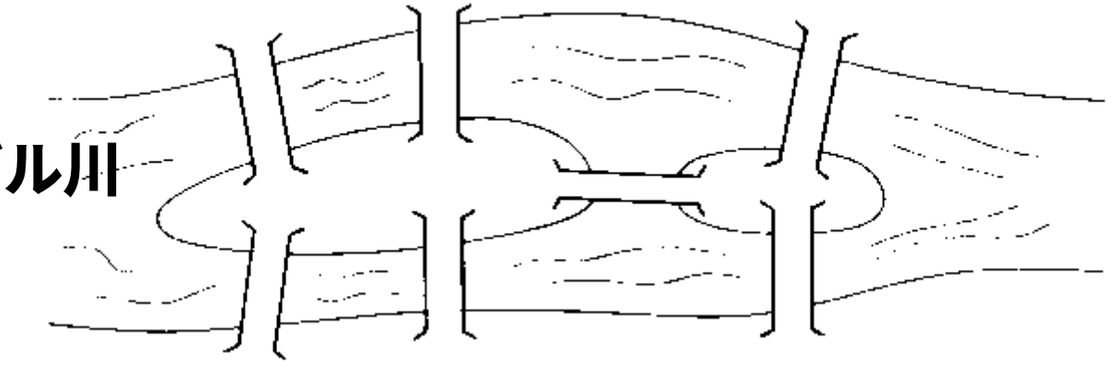


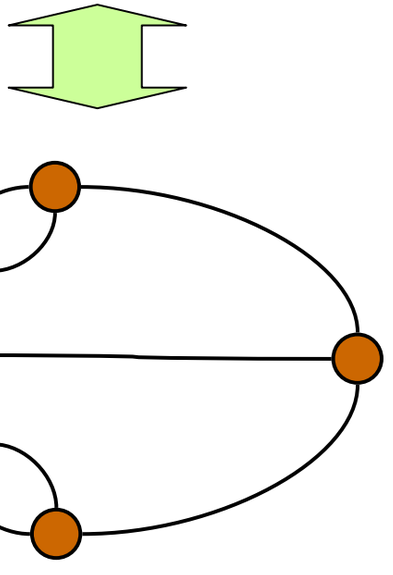
(道路の許容台数,1台当たりの通行料)

構造部分：グラフ

グラフ

プレーゲル川





Konigsbergの橋

ある地点から出発し
全ての橋を一度だけ渡り
出発地点に戻れるか？

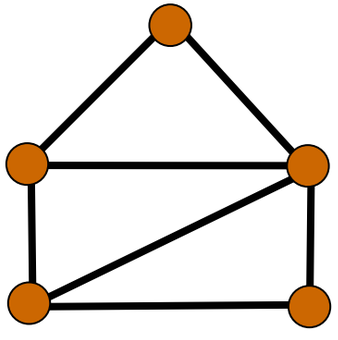
一筆書きは可能？

準備運動 一筆書き

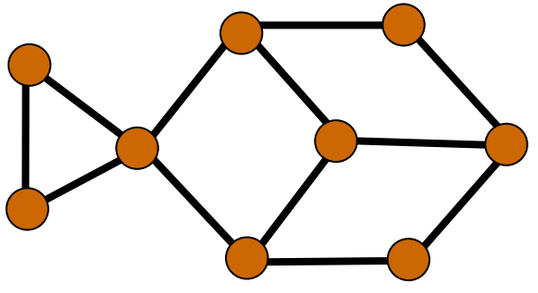
一筆書きは可能？

※2つのバージョン
◎出発点に戻ってくる必要があるバージョン
・戻ってこなくてもよいバージョン

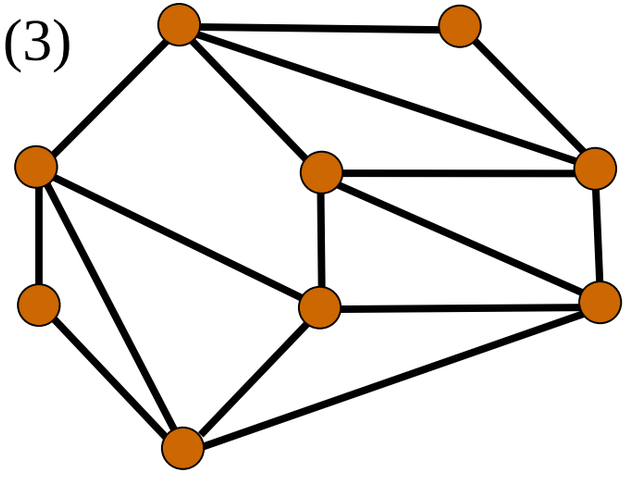
(1)



(2)



(3)

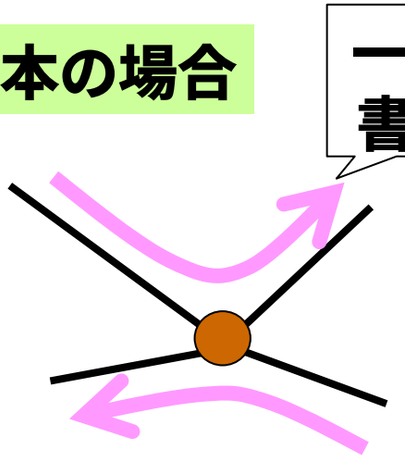


可能⇒一筆書きを具体的に示せばよい
不可能⇒？？？できないことを示すには？？？

一筆書きができる･できない

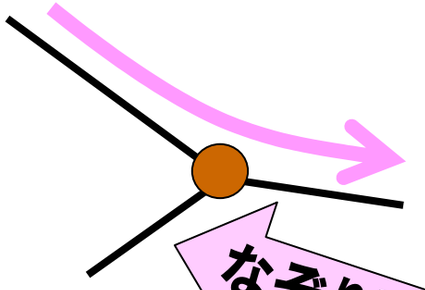
点に接続する枝数に注目

偶数本の場合



可能！

奇数本の場合



不可能！

示唆 奇数本の枝が接続の点があると一筆書きはできない

疑問 全ての点で偶数本の枝が接続なら一筆書き可能？

回答

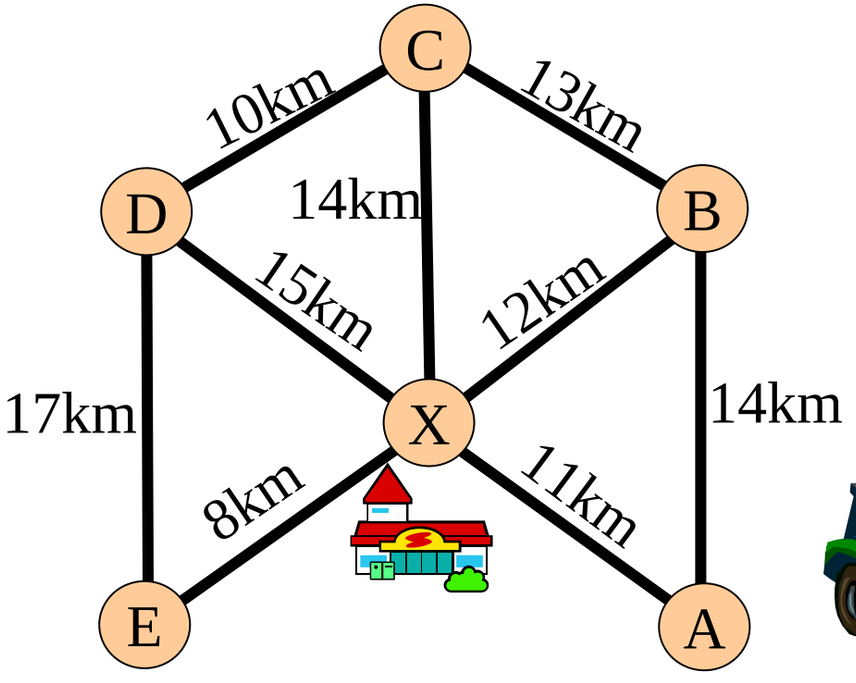
オイラー

できるよ！

オイラーの定理

**グラフ論の
はじまり**

例題1 道路の点検ルート

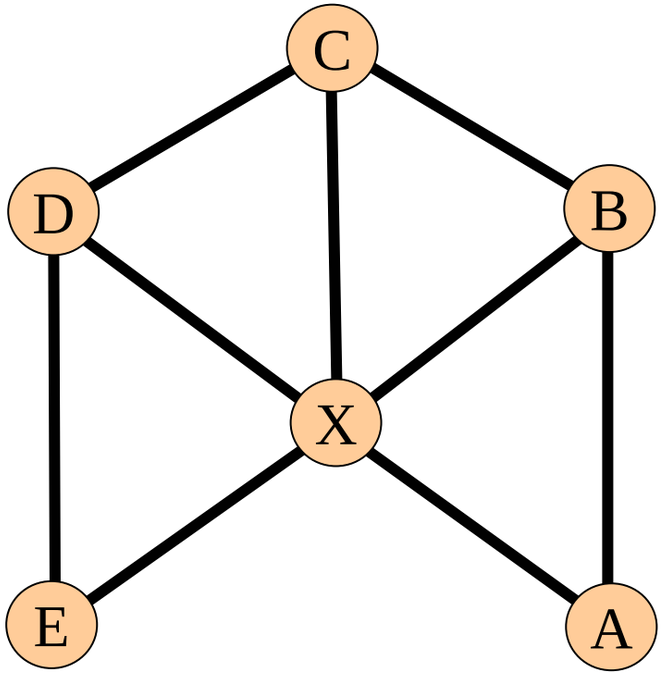


事務所(点X)から出発し，
全ての道を点検し
事務所に戻ってくる

質問

総移動距離最短の
点検ルートは？

「走行ルート」＝「一筆書き」



一筆書き可能＝無駄な走行無し

質問 一筆書きは可能？

回答 不可能

⇒無駄な走行が不可避

⇩

無駄な走行を最小にしよう

質問 どこを回送する？

一筆書き可能
＝
全点で偶数本の枝数

一筆書きができるように枝を増やす

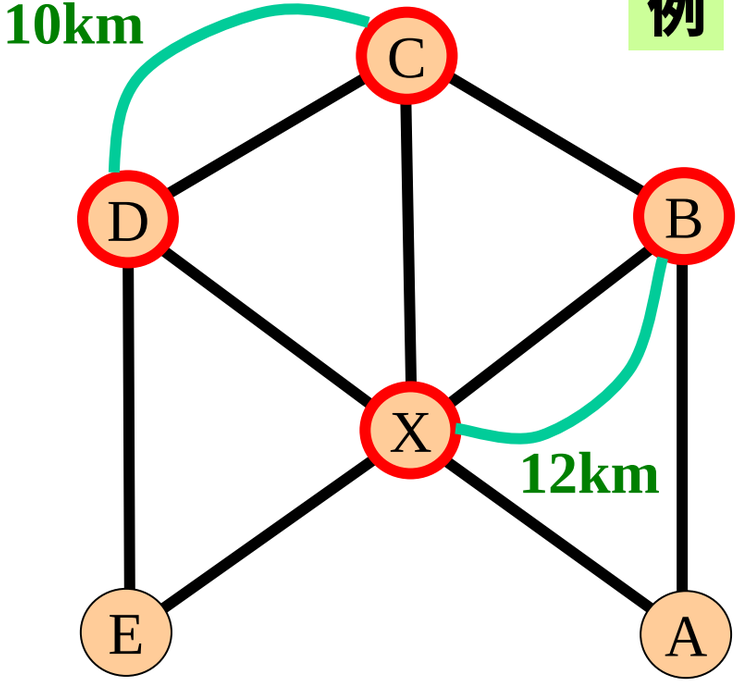
枝を増やして一筆書き可能状態を作る＝走行ルート完成

無駄な走行に対応

枝数が奇数本の点をなくす

少ない方が
好ましい

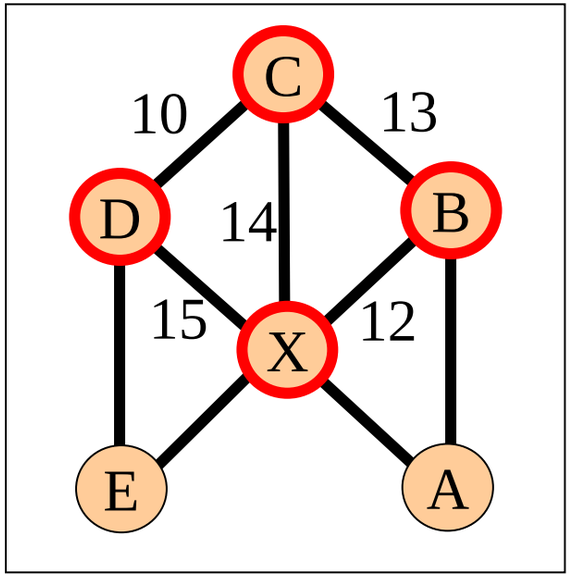
例



22kmの回送で
走行ルート作成可能

⇨ 他のパターンは？

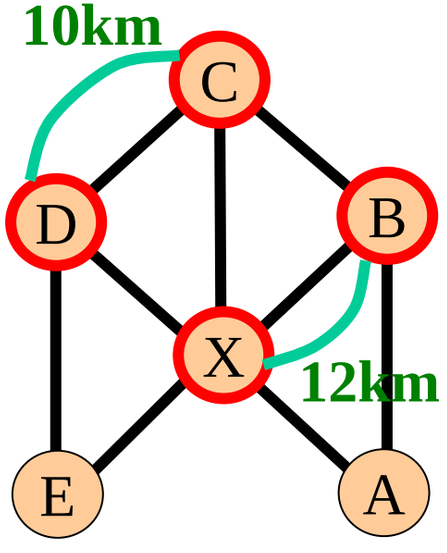
全てのパターン



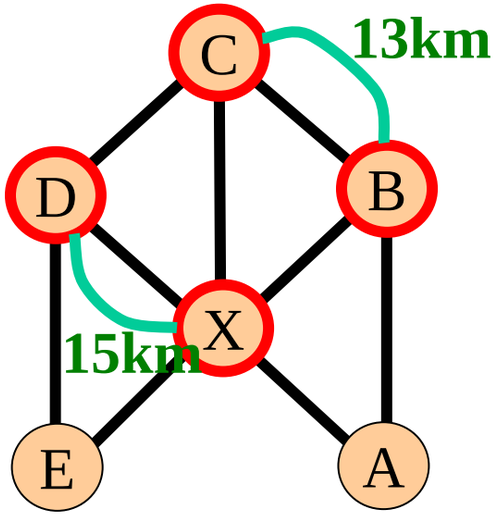
最適なパターンを見つける問題：
マッチング問題

全列挙は面倒

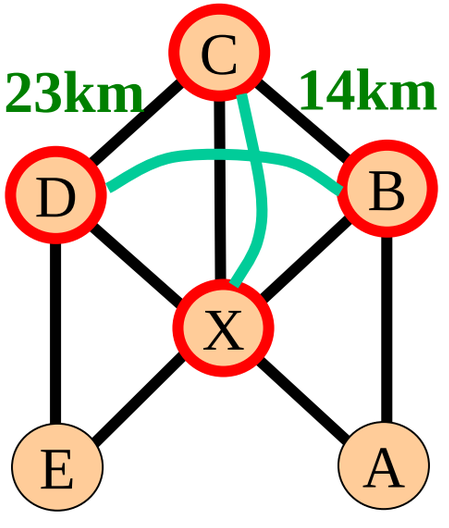
最適な回送プラン



回送：22km

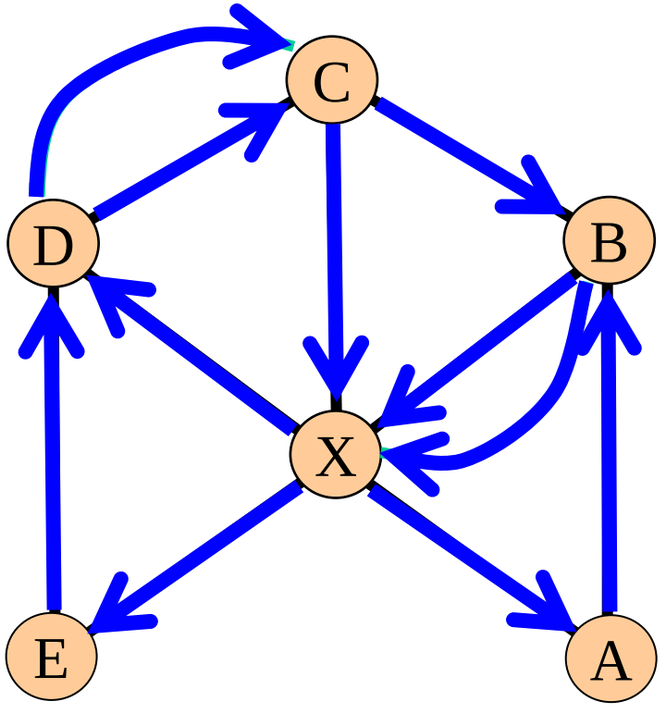


回送：28km



回送：37km

最適な走行ルート作成



最短走行ルート

走行距離：134km
点検走行：114km
回送：22km

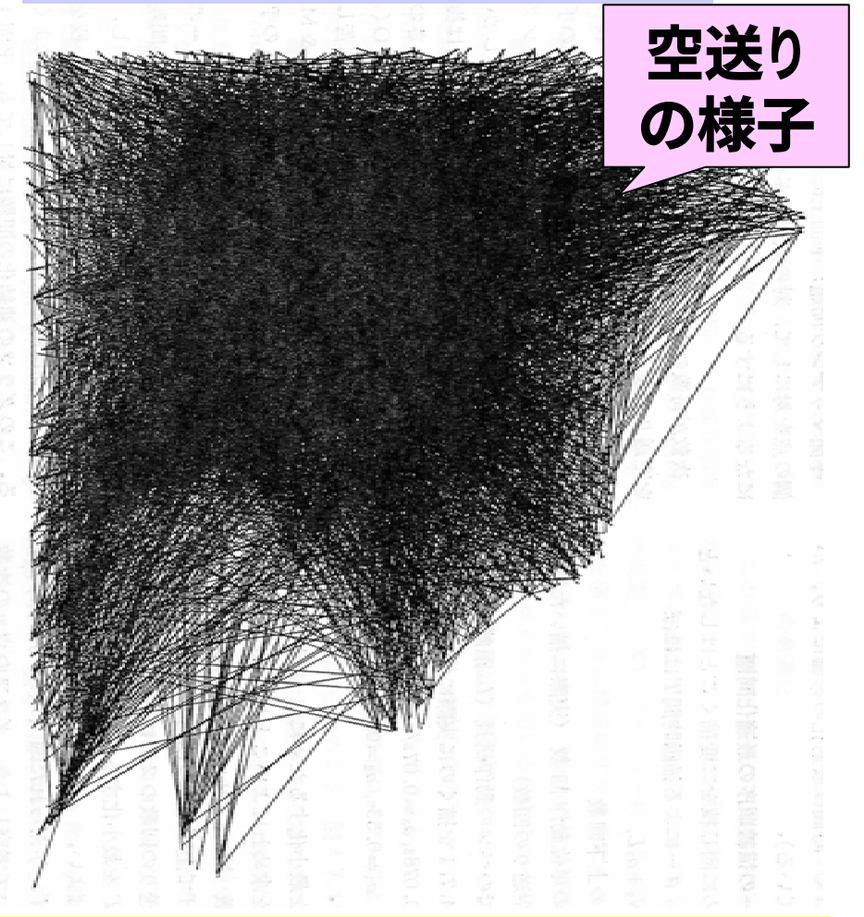
この部分を最小化

応用例 プロッター動作最適化

描きたい地図

20726点
25336枝

ランダムに枝を描画した時



空送り
の様子

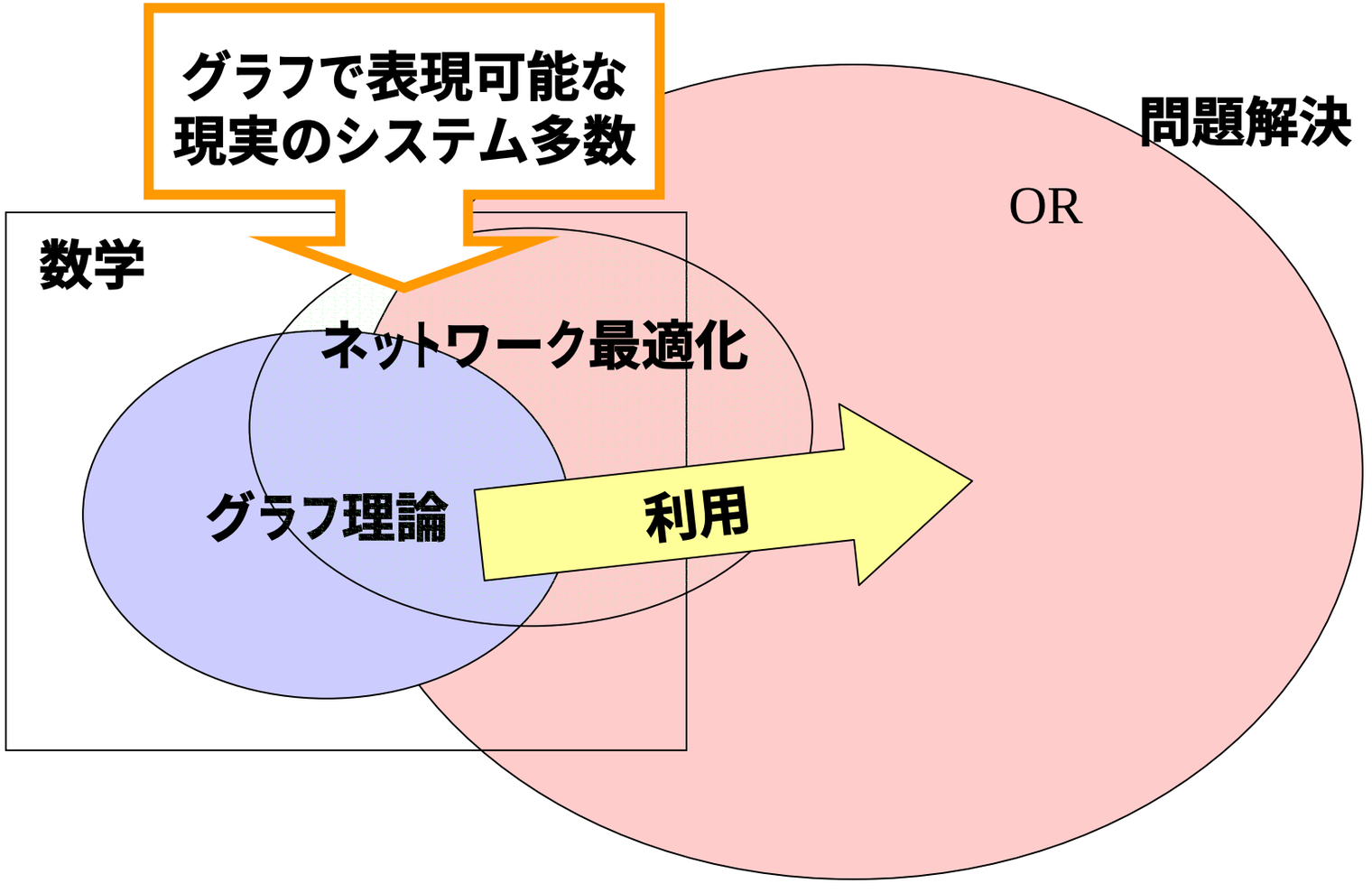
出典：計算機科学と地理情報処理，別冊Bit，共立出版

応用例 最適化の様子

人間による制御

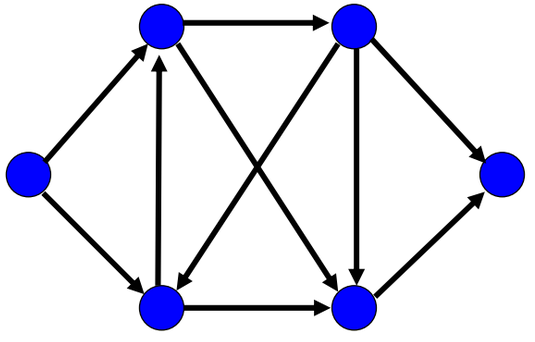
最適化処理

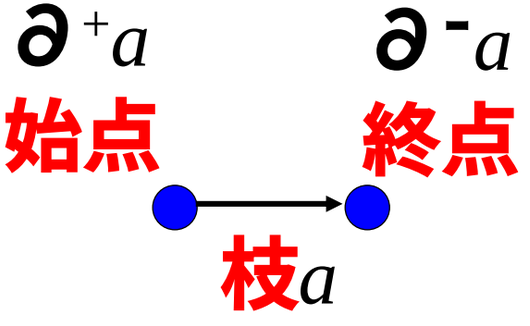
グラフ（ネットワーク）とOR



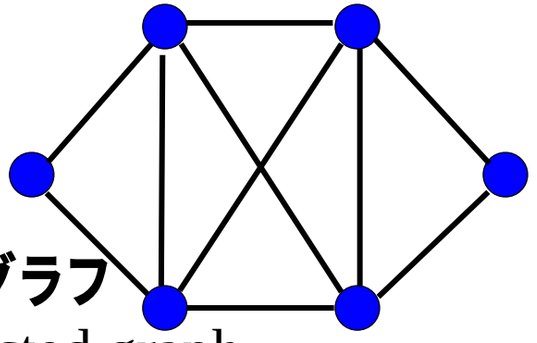
グラフとは

- **点集合** V
- **枝集合** A
- **写像** ∂^+ , ∂^- : $A \rightarrow V$
  とからなる複合概念
  $G=(V, A ; \partial^+, \partial^-)$

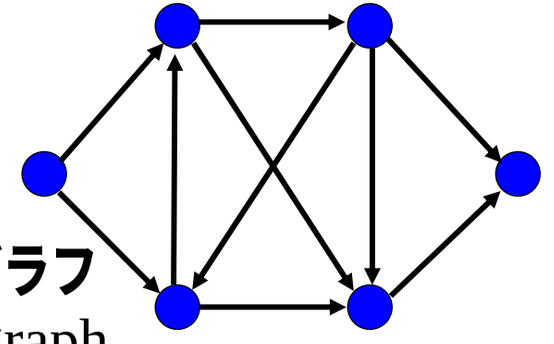




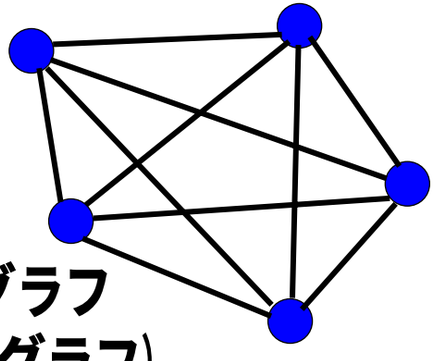
グラフの種類



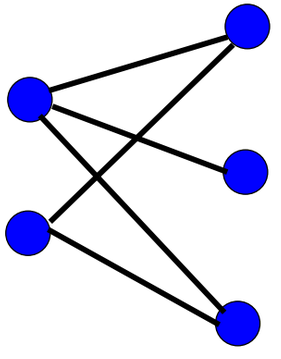
無向グラフ
undirected graph



(有向) グラフ
directed graph

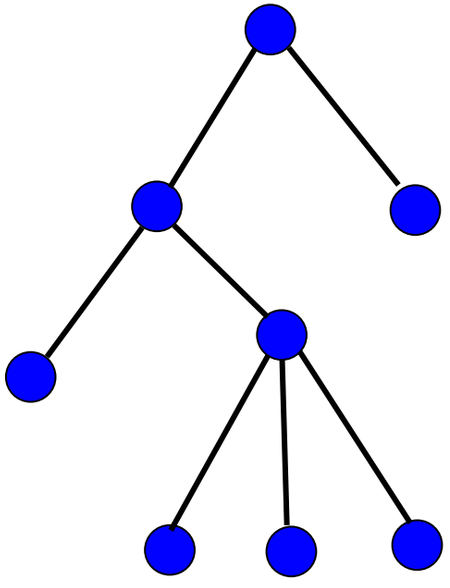


完備グラフ
(完全グラフ)
complete graph

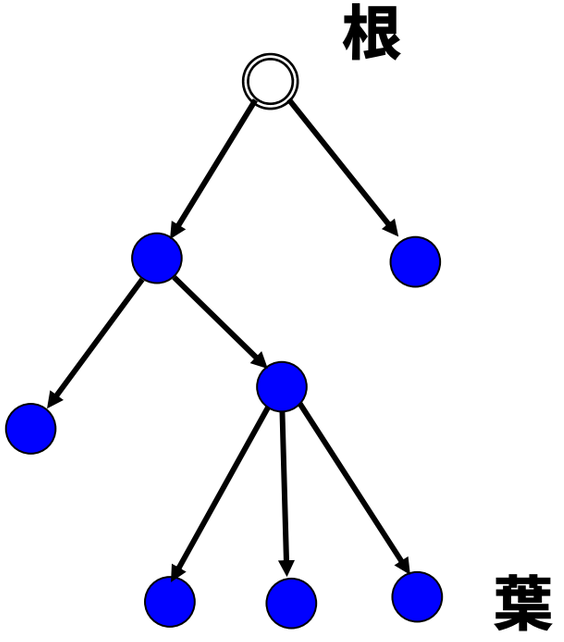


2部グラフ
bipartite graph

グラフの種類 (2)



木
tree



有向木
Directed tree

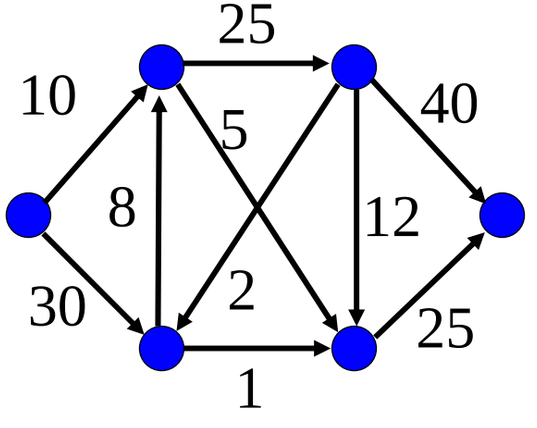
ネットワークとは

グラフの点や枝に情報が付与されたもの

(**例**)

各枝に距離*d*が与えられた
グラフはネットワーク

$N=(G=(V,A;\partial^+,\partial^-),d)$



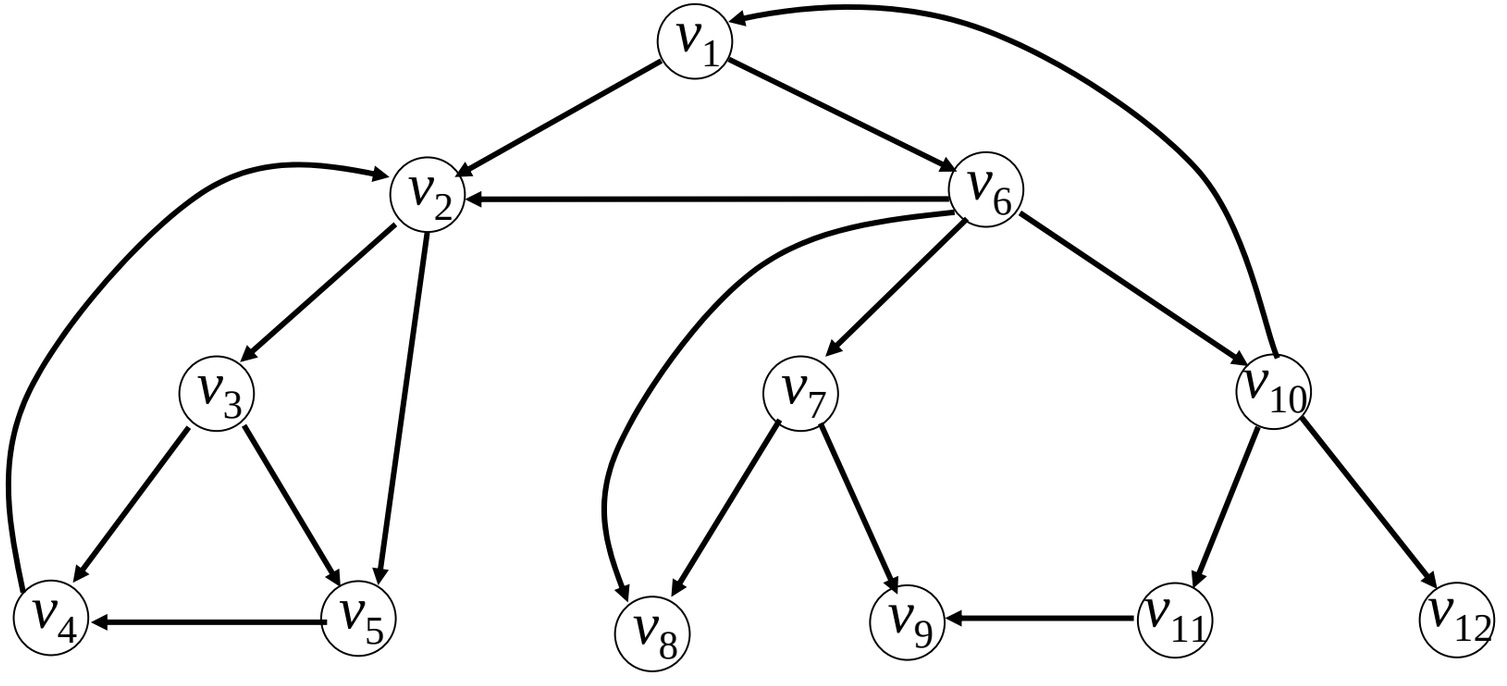
グラフ上の基本的な操作

グラフ，ネットワークで表現されたシステムを扱うために必要な基本的な技.

- グラフの探索
  - 幅優先探索
  - 奥優先探索
- グラフを表現する
  - 隣接行列
  - 接続行列
  - リスト表現
- グラフを分解する
  - 連結成分分解 (無向グラフ)
  - 強連結成分分解 (有向グラフ)

グラフの探索

グラフ上のすべての点と枝を走査すること



グラフの効率の良い探索方法

•奥優先探索
Depth-first search

•幅優先探索
Breadth-first search

Step1: **出発点にラベルを付ける**

以下のStep2，3**を未探索辺がなくなるまで繰り返す．**

Step2: **最新ラベルを持つ点から未探索辺を走査する．**

Step2: **最も早いラベルを持つ点から未探索辺を走査する．**

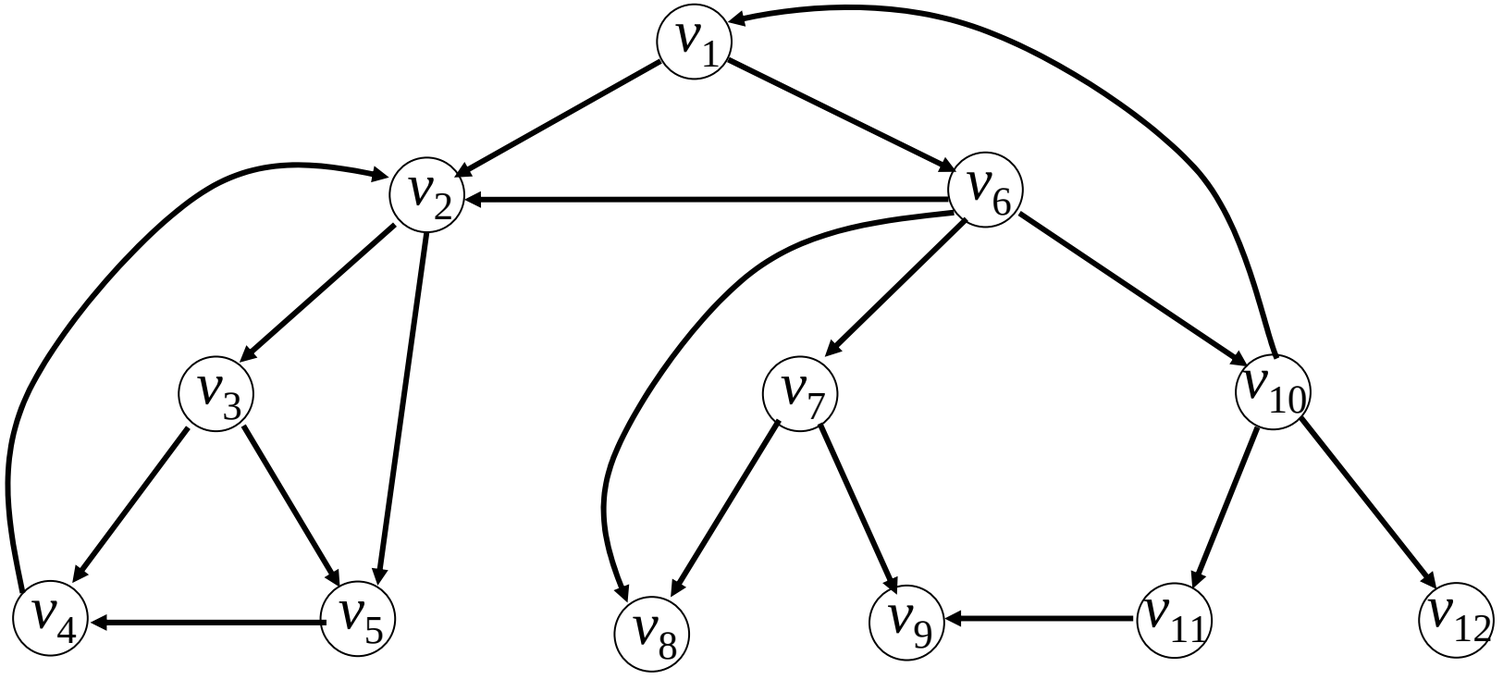
Step3: **走査した未探索辺の終点にラベルが付いていなかったらラベルを付ける．**

探索木

- **探索木**：探索中に点にラベルを付ける原因となった枝の集まり
  - 出発点を根とする全張有向木になっている．（どうして？）
- 探索木は有用な情報を与えてくれる

練習3-1 奥優先探索をしてみよう

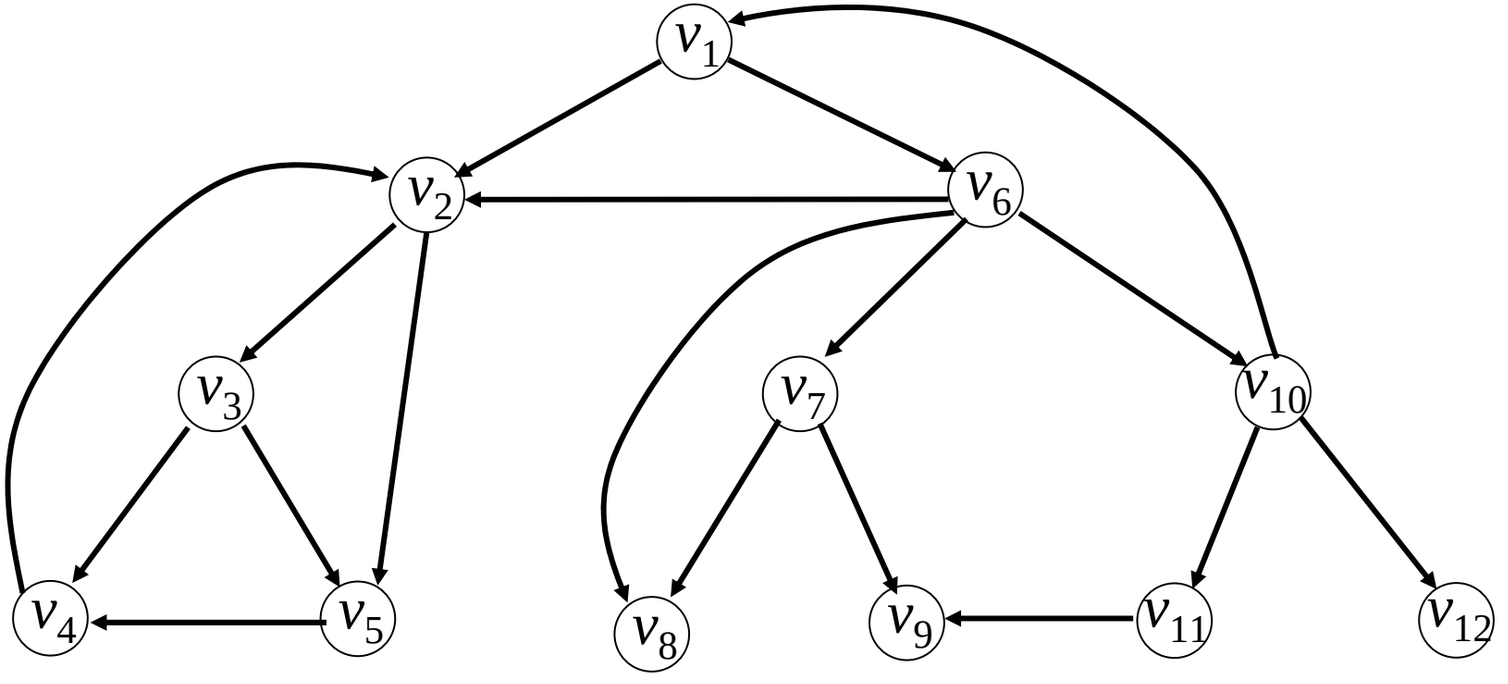
出発点：v_1



探索木は?

練習3-2 幅優先探索をしてみよう

出発点：v_1



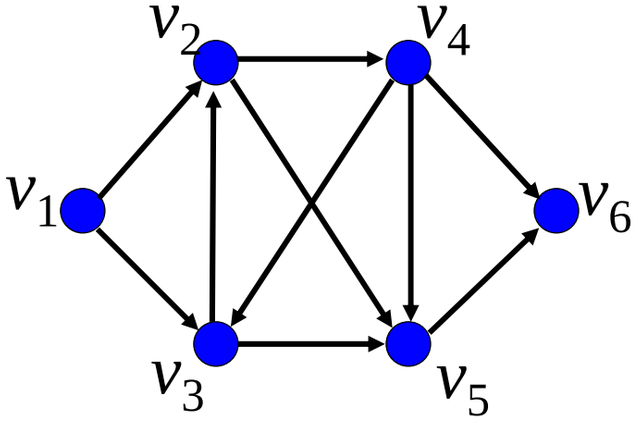
探索木は?

探査木利用 点への順序付け

- **前順**(先行順：pre order)
  - ラベルと同時に番号付け
- **後順**(後行順：post order)
  - 親の点に戻るときに番号付け

他にも
中間順 (in order)
幅優先順 (breadth-first order)
などがある

演習3-1 グラフの探索



出発点：v_1として右のグラフを
以下の方法で探索せよ．

⑴ 奥優先探索
⑵ 幅優先探索

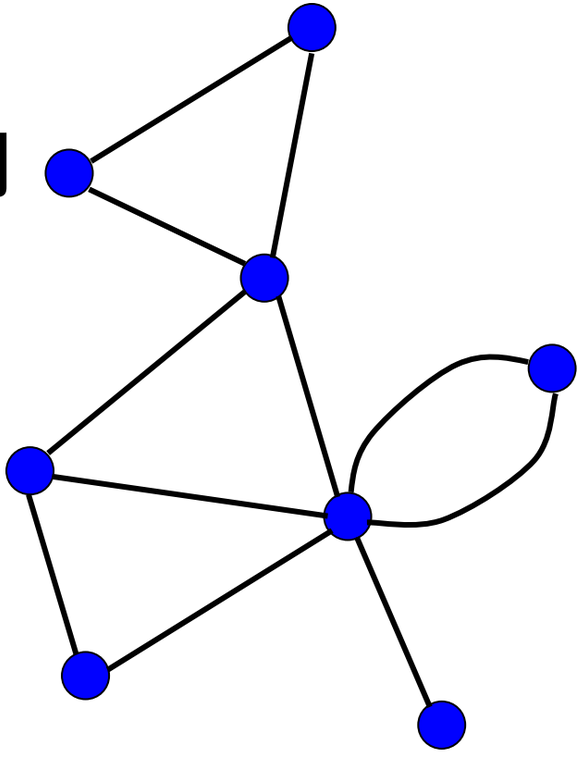
また，各々の探索木を示せ．

⑶ 奥優先探索での探索木を利用し，
各点に前順・後順を各々付けてみよう。

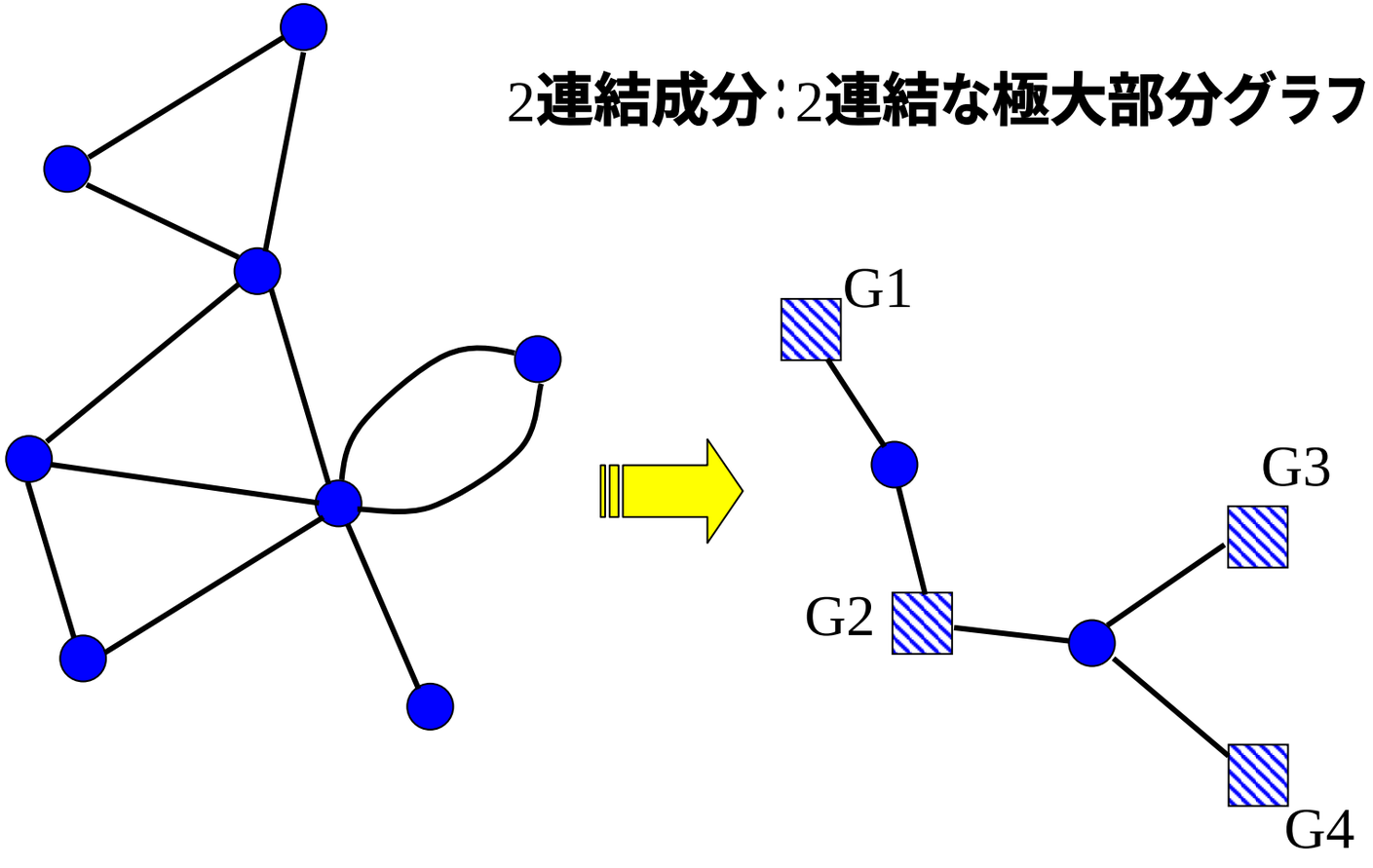
グラフの分解

表現されているシステムの解析に利用

- **道** (path) :接続する点と枝の交互列
- 無向グラフが**連結**
  ⇔任意の2点間に道が存在
- 点vは**関節点**
  ⇔連結なグラフから点vを除くと
  非連結になる
- 無向グラフが**2連結**
  ⇔関節点がないグラフ

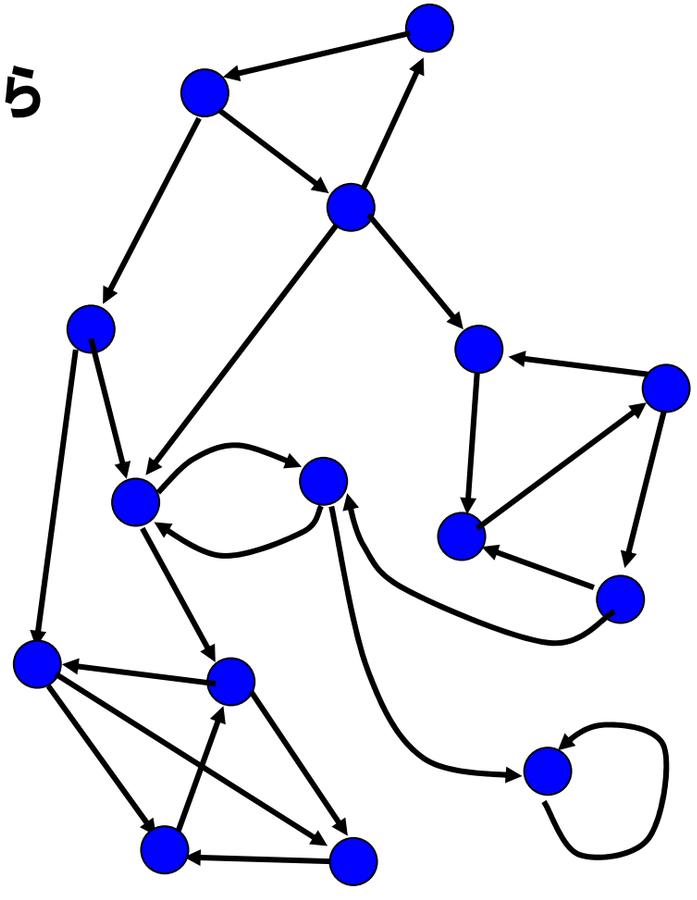


無向グラフの2連結成分分解

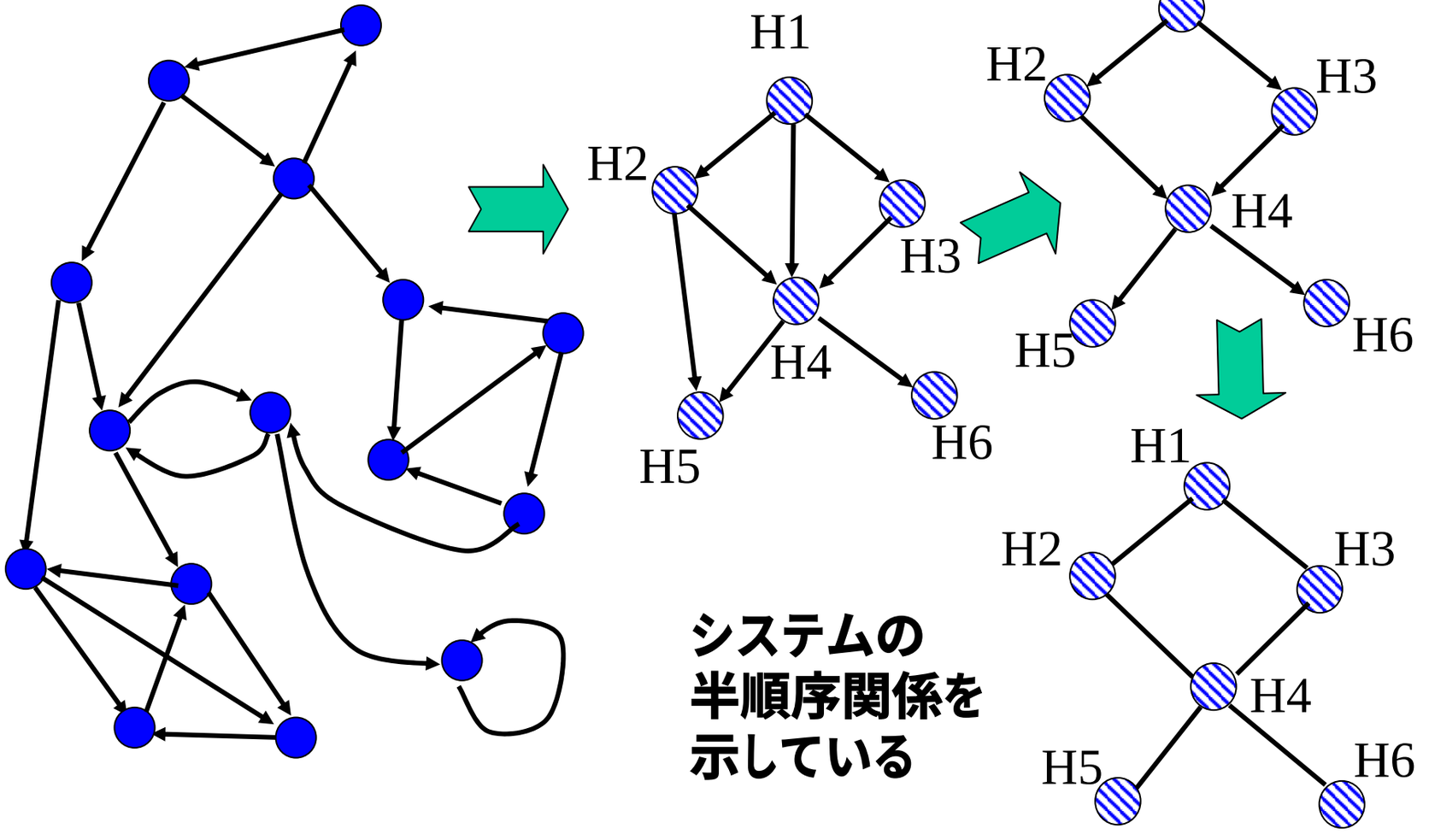


有向グラフの強連結成分分解

- **有向道**：すべて同じ向きの枝からできている道
- 有向グラフが**強連結**
  ⇔任意の2点間に両方向の有向道が存在
- **強連結成分**
  ⇔強連結な極大部分グラフ



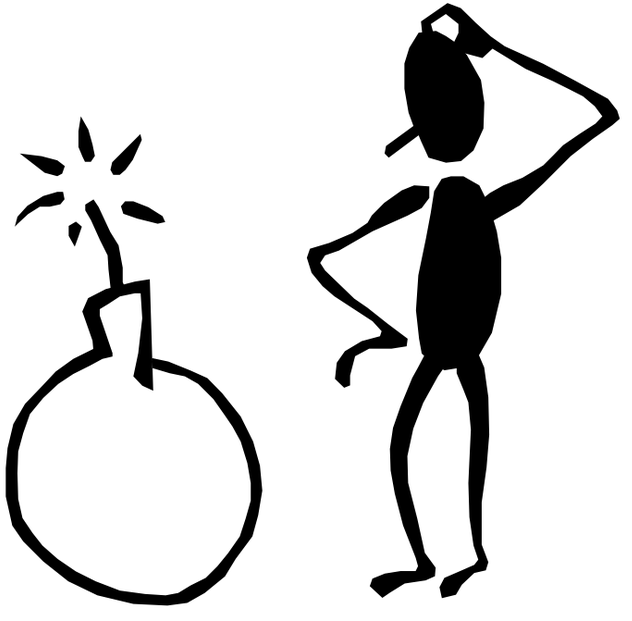
Hasse図



システムの
半順序関係を
示している

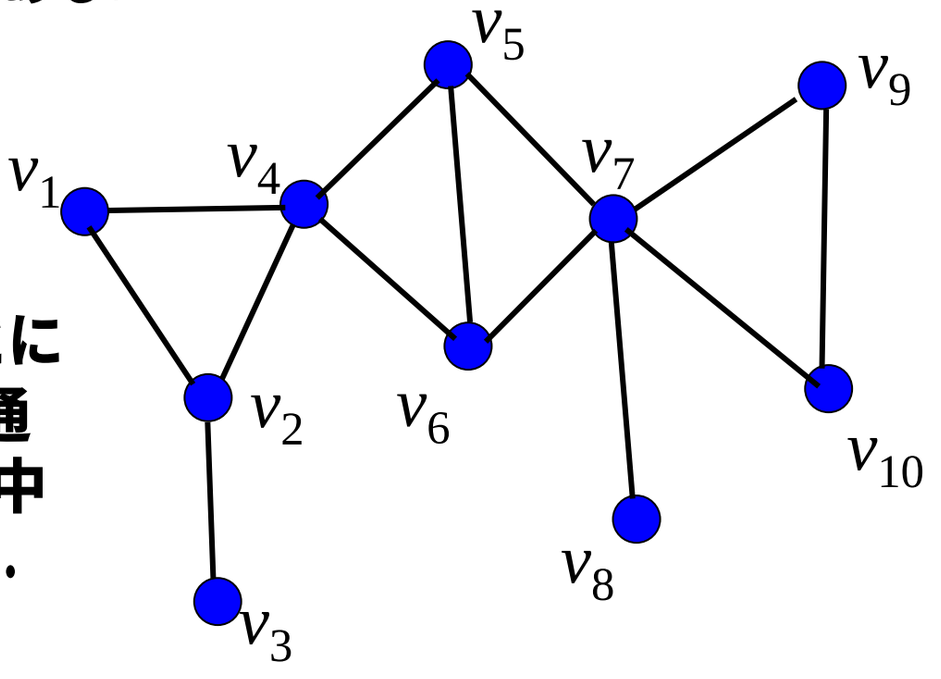
2連結成分，強連結成分の求め方

- 2連結成分分解，強連結成分分解共に，深さ優先探索を利用し可能
  - より効率的な解法を作ってみよう!!



演習3-2　2連結成分分解

右図のような通信網がある．
点は中継局を示す．

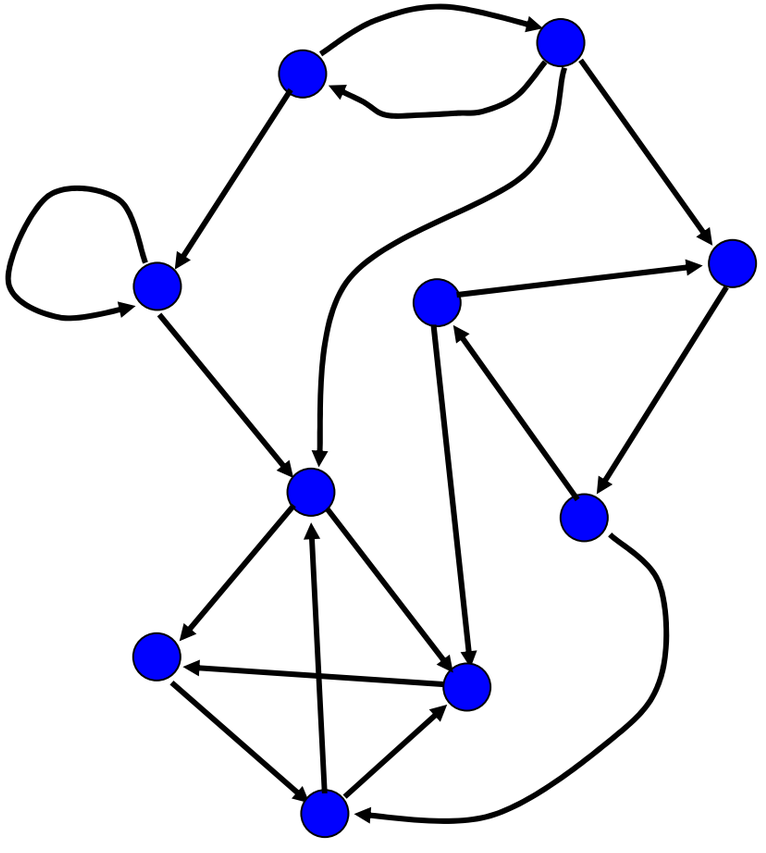


⑴ 機能が停止することにより，他の中継局間の通信にまで影響を及ぼす中継局はどれか指摘せよ．

⑵ 通信の信頼性を高めるためには新たにどのような線をどこに引けばよいか．適切な設置計画を提案せよ．

演習3-3 強連結成分分解

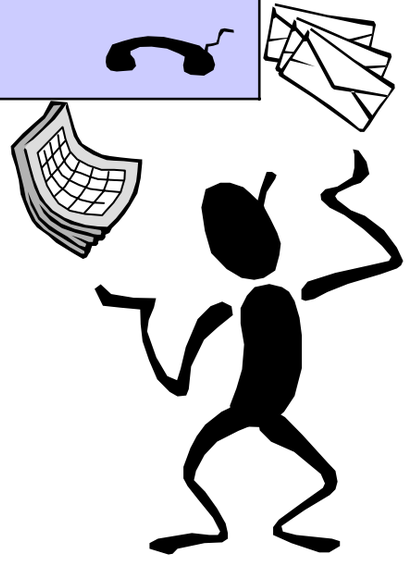
右の有向グラフの
Hasse**図を作成しなさい**



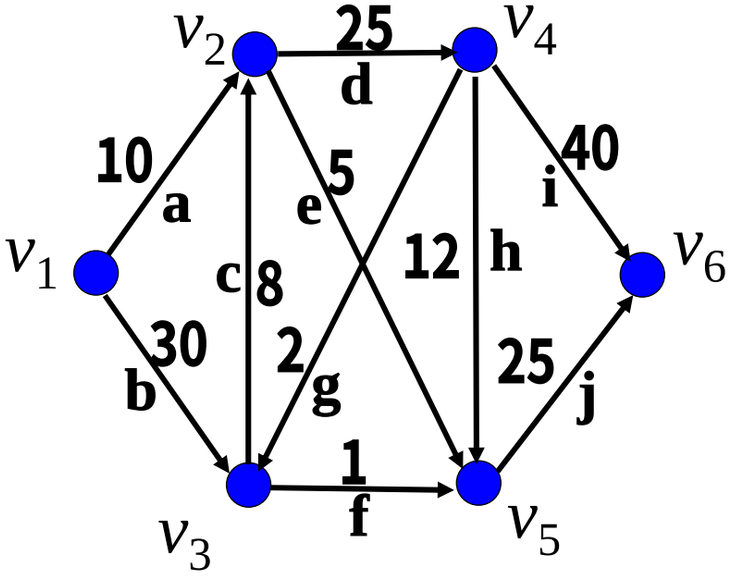
グラフ，ネットワークの表現

表現方法	利点	欠点
描画表現	目での理解容易	計算機不向き 情報伝達曖昧
数値表現	計算機向き 情報伝達確実	目での認識不向き

- ネットワークのデータ
- 隣接行列での表現
- 接続行列での表現
- リスト表現



ネットワークのデータ



欠点：扱いにくい

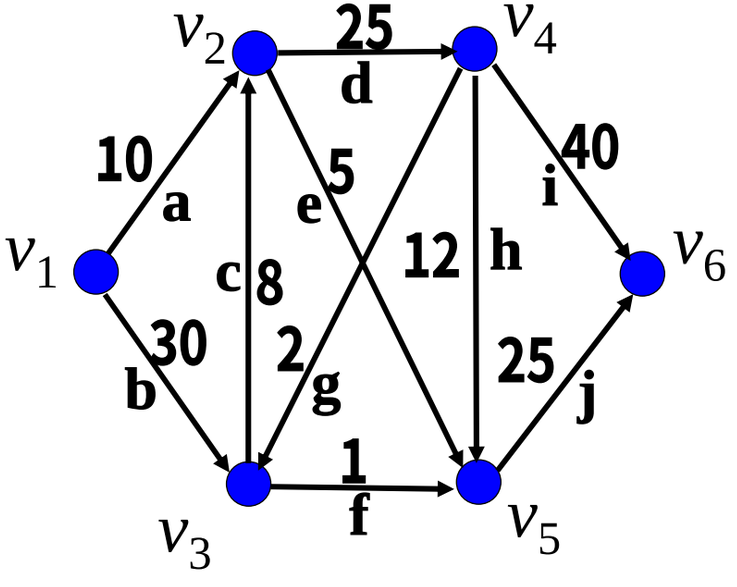
始点終点

枝のインデクス　付随情報

a	1	2	10
b	1	3	30
c	3	2	8
d	2	4	25
e	2	5	5
f	3	5	1
g	4	3	2
h	4	5	12
i	4	6	40
j	5	6	25

隣接行列での表現

- (点数) × (点数) の行列
- 付随情報は別な配列で保持



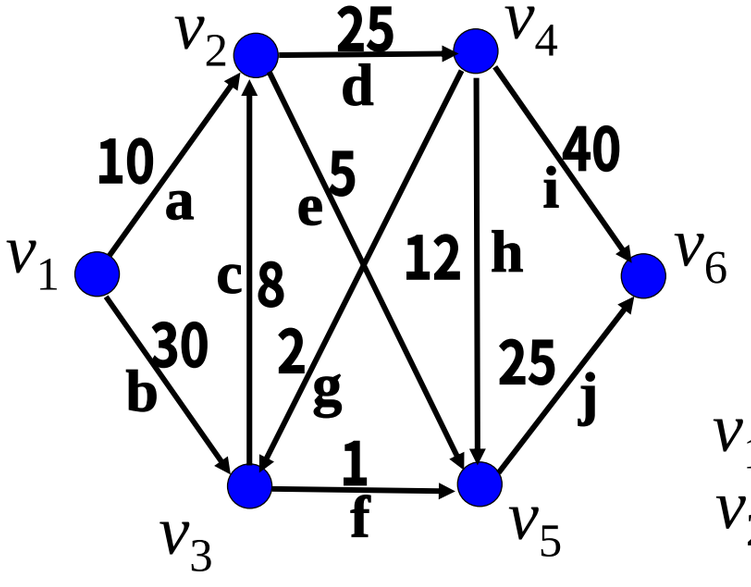
終点

始点

$$\begin{array}{c} \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{array} \begin{array}{c} \begin{array}{cccccc} v_1 & v_2 & v_3 & v_4 & v_5 & v_6 \end{array} \\ \begin{pmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \end{array}$$

欠点 並列枝の情報喪失

接続行列での表現



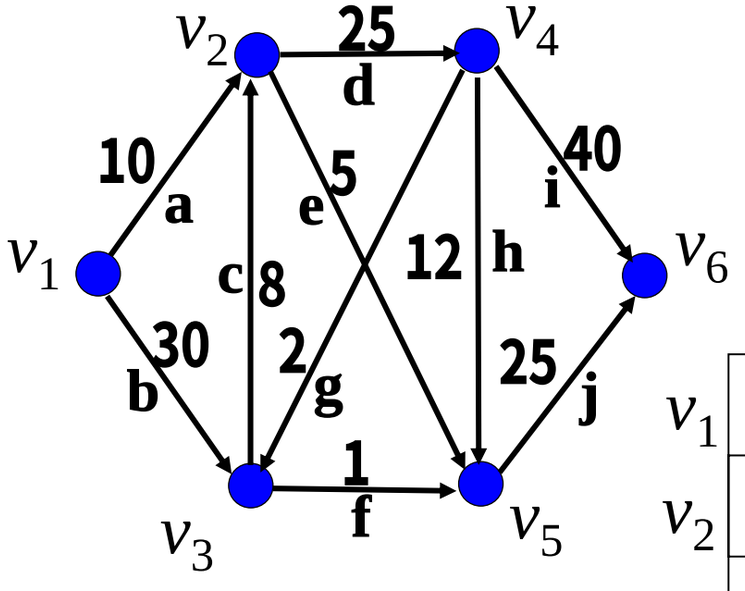
- (点数) × (枝数) の行列
- 付随情報は別な配列で保持

枝

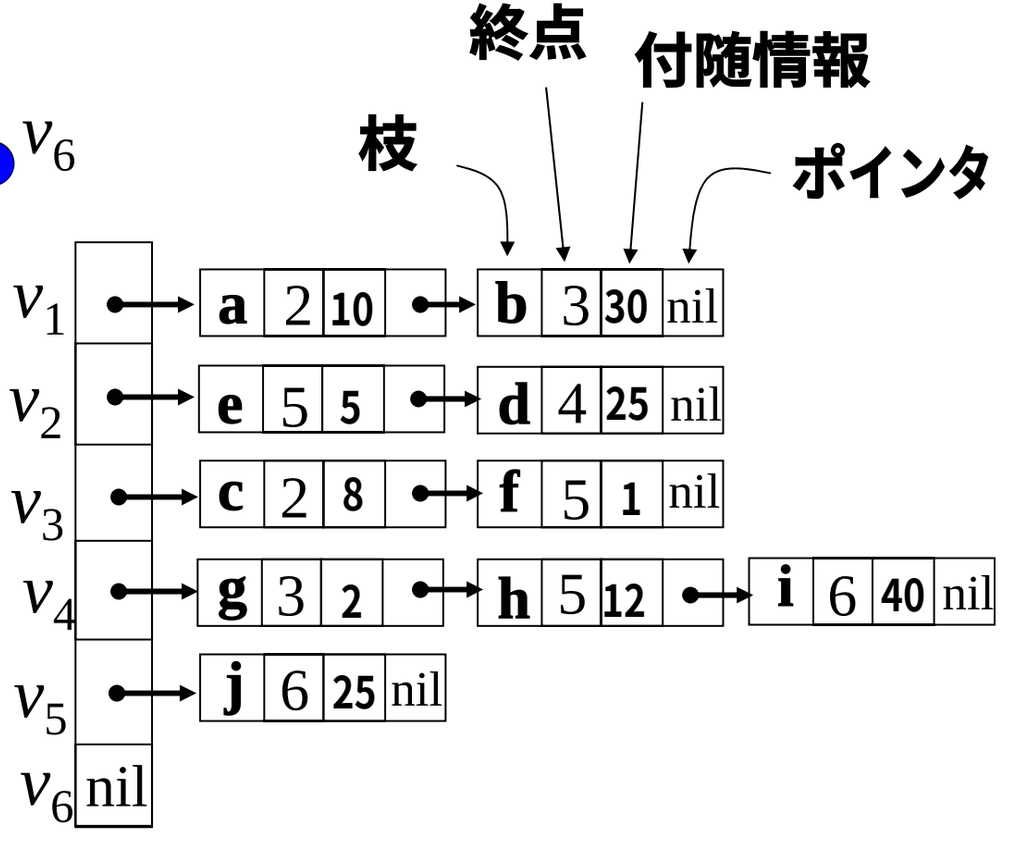
点	a	b	c	d	e	f	g	h	i	j
v_1	1	1	0	0	0	0	0	0	0	0
v_2	–1	0	–1	1	1	0	0	0	0	0
v_3	0	–1	1	0	0	1	–1	0	0	0
v_4	0	0	0	–1	0	0	1	1	1	0
v_5	0	0	0	0	–1	–1	0	–1	0	1
v_6	0	0	0	0	0	0	0	0	–1	–1

欠点 自己閉路の情報喪失

リスト表現



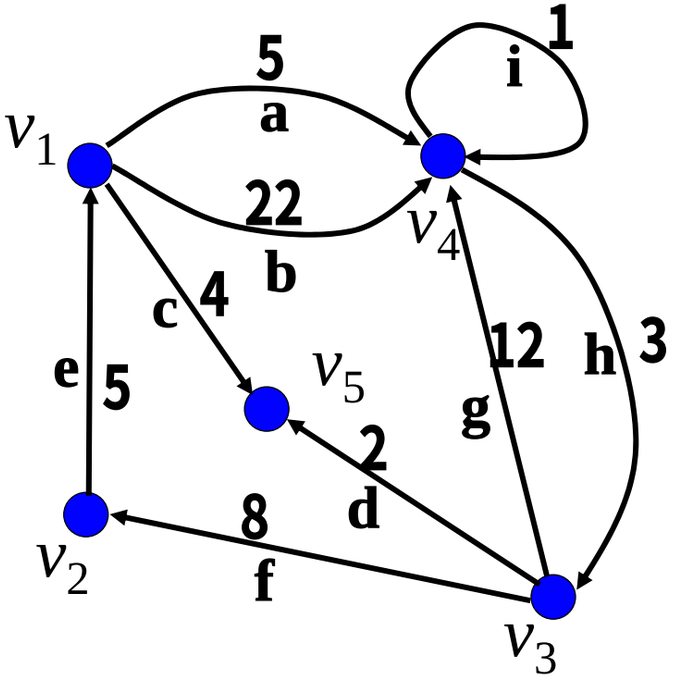
長所 メモリの節約
扱いやすい



演習3-4

右のグラフを以下の方法で表現せよ.

⑴ 隣接行列
⑵ 接続行列
⑶ リスト表現



なお, 各表現において不都合が生じる場合は, それがどのような状況でおきるのか考察せよ.