3 最短経路（shortest path）

（有向）ネットワーク上のある点から他の点へ至る道（path）の中で，長さ（path length；道を構成する枝の長さの和）が最小のものを求める問題.（1対1，1対多，多対多）

3.1 行列演算による方法

- 距離行列 $D=(d_{ij})$：点 i から点 j へ向かう枝の長さを並べた行列.
  ・$d_{ii}=0$,　　・点 i から点 j へ向かう枝がない場合は $d_{ij}=+\infty$（無限大長の枝がある），
  ・無向枝の場合は，$d_{ij}=d_{ji}$.

  d_{ij} は，点 i から点 j へ1本以下の枝を経由する道（path）の長さを表す.
  （参考）　$(I+A)_{ij}$ は1本以下の枝を辿って点 i から点 j へ到達できる (1) か否 (0) かを表す.

3.1.1 べき乗法

- $D^{(\ell)}=(d_{ij}^{(\ell)})$ を定義する.
  $d_{ij}^{(\ell)}$ は，点 i から点 j へ ℓ 本以下の枝を経由して行くことを許した時の最短経路長を表す.
- $D^{(0)}=(d_{ij}^{(0)}):=\begin{cases}0 & (i=j)\\ \infty & (i\neq j)\end{cases}$,　　$D^{(1)}=D=D^1$.
- d_{ij} の中に負のものがある場合には，長さが負である閉路が存在するか否かを検知し，存在しなければ最短経路（長）を求める.
- つぎの漸化式で，$D^{(2)}$，$D^{(3)}$，…，$D^{(|V|-1)}$ を計算する.
  $d_{ij}^{(\ell)}:=\{\min_{k\in V}(d_{ik}^{(\ell-1)}+d_{kj})\}$，すなわち $D^{(\ell)}=D^{(\ell-1)}\times D$.
  行列乗算において，$\sum$ を min，$\cdot$ を $+$ で定義すると，$D^{(\ell)}$ は D^{ℓ} とみなせる.
- D^2，D^4，D^8，D^{16}，… と 指数が $|V|-1$ 以上になるまで計算する ⇒ べき乗法.

3.1.2 Warshall-Floyd（ウォーシャル・フロイド）法

- $W^{(\ell)}=(w_{ij}^{(\ell)})$ を定義する.
  $w_{ij}^{(\ell)}$ は，点 i から点 j へ，点集合 $\{1,2,\ldots,\ell\}$ に属する点を0個以上経由して行くことを許した時の最短経路長を表す.
- $W^{(0)}=D=(d_{ij})$.
- d_{ij} の中に負のものがある場合には，長さが負である閉路が存在するか否かを検知し，存在しなければ最短経路（長）を求める.
- つぎの漸化式で，$W^{(1)}$，$W^{(2)}$，…，$W^{(|V|)}$ を計算する.
  $w_{ij}^{(\ell)}:=\{\min\{w_{ij}^{(\ell-1)},w_{i\ell}^{(\ell-1)}+w_{\ell j}^{(\ell-1)}\}$.
  この演算は行列乗算ではないし，そうみなす必要もない.
- 計算量 $\mathcal{O}(m^3)$.　　$(m=|V|)$
  全点間の最短経路長を求めるアルゴリズムとしては，べき乗法（$\mathcal{O}(m^3\log m)$）より高速.

3.2 ダイクストラ法

- $d_{ij} \geq 0$ という条件の下で正しく働く，能率の良い（現役の）アルゴリズム．
  $d_{ij} \in \{R^+, 0, +\infty$ (枝なし) $\}$
- （任意の）ある 1 点から他の全点への最短経路長（と経路）を求める．
- ラベル修正法，Dynamic Programming の枠組み．
  出発点（例えば 1）から点 k へ至る最短経路長を L_k とすると，
  原理的に，「$L_k = \min_j L_j + d_{jk}$ 」が成立している．　　（最短経路の部分路は最短経路）

3.2.1 ダイクストラ法の流れ

与えられたネットワーク $\mathcal{N}(V, E, d)$ 上で，出発点 $s \in V$ から，他の全点に至る最短経路長（経路）を求める．

Dijkstra Method

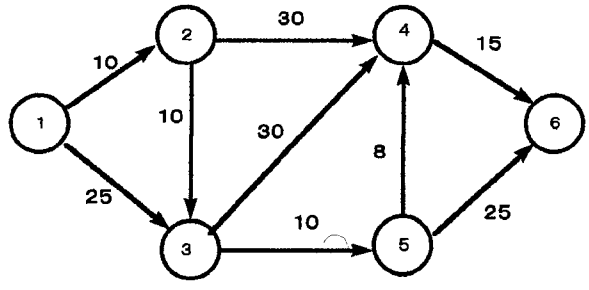
step0: $y_s := 0$; $x_j := \infty$ $(j \neq s)$; $T := V - \{s\}$; $P := \{s\}$; $i := s$;

step1: **for all** $j \in T$ s.t. $d_{ij} < \infty$ **do**
if $y_i + d_{ij} < x_j$ **then** [$x_j := y_i + d_{ij}$; $p_j := i$;]

step2: $i := \arg\min_{j \in T} x_j$; $y_i := x_i$; $(i \in T)$

step3: $T := T - \{i\}$; $P := P + \{i\}$;
if $T = \phi$ **then** 終了 **else goto step1**

例題



3.2.2 ダイクストラ法の正当性

命題：P に属する点のラベル値 y_h は，(点 s から点 h に至る）真の最短経路長を与える．
この命題を，$|P| = k$ に関する数学的帰納法で証明する．

3.3 線形計画として見た最短経路問題

- 「点 s から他の全点へ至る最短経路（長）を求める問題（最短経路問題）」を，
  「点 s から他の全点へ『1 単位ずつ』のモノを総費用最小で運ぶ問題（輸送問題）」
  と考え，線形計画問題として扱うことを考える．

  ただし，各枝上を運ぶことができるモノの量には制限はない．好きなだけ運べる．

  また，各枝上を運ぶ（通過する）とき，枝ごとに 1 単位当たりで決まっている費用がかかる．例えば，枝 ij（点 i から点 j へ向かう枝）上では，1 単位当たり d_{ij} 円かかるとする．
  x 単位運ぶ（通過する）とすると，$d_{ij}x$ 円の費用が発生する（x は実数）．
- ここで，単位当たり費用 d_{ij} を，枝 ij の長さ（点 i，j 間の距離）で与える．

  このとき，
  「点 s からの総費用最小の輸送計画を求める」ことは
  「点 s から各点へ至る最短経路（長）を求める」こと」と同値である．

  ⋆ 点 s から他の全点への総費用最小の輸送経路 $\Longleftrightarrow$ 点 s から他の全点への最短経路

  ⋆ 点 s から点 j へ 1 単位のものを運ぶ最小費用 $\Longleftrightarrow$ 点 s から点 j への最短経路長

3.3.1 線形計画としての定式化

3.2.2 の例題において，点 1 から他の全点へ 1 単位ずつのモノを総費用最小で輸送する最適化問題を線形計画として定式化する．

- 枝 ij 上を点 i から点 j へ輸送されるモノの量を x_{ij} 単位とする．
- 点 1 から運び出されるモノは 5（$|V|-1$）単位．
- 点 j（$\neq 1$）から運び出されるモノ量と運び込まれるモノの量の差は -1．
- x_{ij} は非負の実数．

(S-PATH)

$$
\left\{
\begin{array}{lrrrrrrrrrrcr}
\min & z = & 10x_{12} & +25x_{13} & +10x_{23} & +30x_{24} & +30x_{34} & +10x_{35} & +15x_{46} & +8x_{54} & +25x_{56} & & \\
\text{s. t.} & & x_{12} & +x_{13} & & & & & & & & = & 5 \\
& & -x_{12} & & +x_{23} & +x_{24} & & & & & & = & -1 \\
& & & -x_{13} & -x_{23} & & +x_{34} & +x_{35} & & & & = & -1 \\
& & & & & -x_{24} & -x_{34} & & +x_{46} & -x_{54} & & = & -1 \\
& & & & & & & -x_{35} & & +x_{54} & +x_{56} & = & -1 \\
& & & & & & & & -x_{46} & & -x_{56} & = & -1 \\
& & x_{12}\,, & x_{13}\,, & x_{23}\,, & x_{24}\,, & x_{34}\,, & x_{35}\,, & x_{46}\,, & x_{54}\,, & x_{56}\,, & \geq & 0
\end{array}
\right.
$$

独立な制約式（除 符号制約）は 5 本．従って基底変数の数は 5 個．($5 = |V| - 1$)

- （実行可能）基底解は非負整数ベクトル．　x_{ij} が基底変数 $\Leftrightarrow x_{ij} = 1, 2, \cdots$.
- （任意の実行可能基底解の）基底変数に対応する枝を基底枝と呼ぶ．
- 基底枝とその両端の点からなる部分グラフを基底グラフと呼ぶ．

- 基底グラフは，枝の方向を無視すると，元グラフ（ネットワーク）の全域木であり，方向を考慮すると，点 1 から他の全点へ至るユニークな道（path）を与える．
- 元グラフ（ネットワーク）の全域木（方向無視）で，点 1 から他の全点へ到達可能なものは基底グラフである．

♣ 基底木を利用した改訂単体法（ネットワーク単体法）が考えられる．

♣ ここでは，シンプレックス乗数 π_i を，制約式の第 i 行（点 i に対応）に乗じて目的行に**加える**数とする．

♣ $|V|$ 個の制約式は冗長（独立なものは $|V|-1$ 本）であるが，シンプレックス乗数は記述用に 1 個多く使い（$i=1,2,\ldots,|V|$），$\pi_1=0$ とする．

♣ 変数 x_{ij}（枝 ij に対応）の費用係数（目的関数）は d_{ij}，被約費用は $d'_{ij}=d_{ij}+\pi_i-\pi_j$．

♣ x_{ij} が基底変数（基底枝）であれば，$d'_{ij}=0$．

♣ π_i は点 1（出発点）から（現在の）基底木を辿って点 i へ至る道（path）の長さに等しい．

♣ 全ての非基底変数 x_{ij}（非基底枝 ij）について $d'_{ij}\geq 0$ であれば，現在の基底木は最適解を与える．（「全変数について $d'_{ij}\geq 0$」といっても同じこと）

最短経路を求めるネットワーク単体法

♠$_1$ 初期基底木を用意する．

♠$_2$ 基底枝に対する $d'_{ij}=0$ から π_i を求めて非基底枝の d'_{ij} を計算する．
（π_i は現在の基底木における path 長から直ちに求まる）

♠$_3$ 非基底枝の d'_{ij} から最適性を判定する．

♠$_4$ 最適なら終了．そうでなければ，取入れ枝と追出し枝を選んで，基底木を更新する．

♠$_5$ ♠$_2$ へ戻り上記を繰り返す．