第3章　ネットワーク計画法

この章では，ネットワークに関わる最適化の手法について学ぶ.

3.1　ネットワークの用語

グラフ： $G = (V, E)$. V は節点の集合，E は枝の集合.

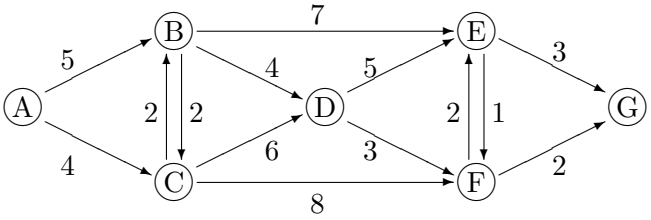
ネットワーク： 節点や枝に値や属性が与えられたグラフ

閉路： ある節点からまた元の節点へ到達する道順

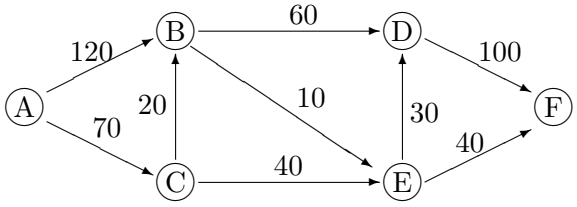
経路，路： 閉路を含まない道順

代表的なネットワークに関する最適化問題は以下の 3 つである.

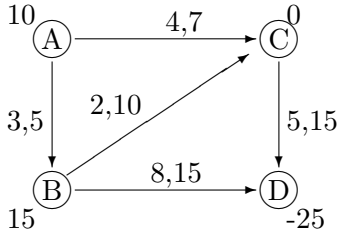
最短路問題：　下図に与えられたネットワークにおいて，節点 A から節点 G まで最短で行くための経路（路）を求めよ．ただし，枝に与えられた値はその枝の長さを表わす.



最大流問題:　下図で与えられたネットワークにおいて，節点 A から節点 F まで最大どれだけ流すことができるか．ただし，枝に与えられた値はその枝の容量を表わす



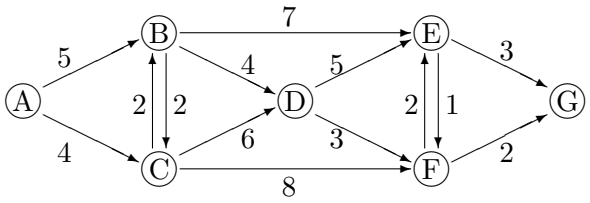
最小費用流問題:　下図に与えられたネットワークにおいて，全ての節点における需要量・供給量を満足しつつ，コストを最少にするにはどうしたらよいか.（枝に与えられた値＝(コスト，容量)，節点に与えられた値＝供給量（正），需要量（負））



3.2　最短路問題

3.2.1　概要

[**問題 3.1**]　下図に与えられたネットワークにおいて，節点 A から節点 G まで最短で行くための経路（路）を求めよ．ただし，枝に与えられた値はその枝の長さを表わす.



与えられたグラフ（ネットワーク）$G = (V, E)$ において，節点の数 $|V| = n$，枝の数 $|E| = m$ とする．枝 (i, j) の長さを $a_{i,j}$ とし，路の始まりの点（_____）を $s \in V$，終わりの点（_____）を $t \in V$ とすると最短路問題は線形計画問題として次のように定式化できる.

$$
\begin{aligned}
&\text{目的関数：} && \sum_{(i,j)\in E} a_{i,j}x_{i,j} \ \rightarrow \ \text{最小化} \\
&\text{制約条件：} && \sum_{(s,j)\in E} x_{s,j} - \sum_{(i,s)\in E} x_{i,s} = 1 \\
& && \sum_{(v,j)\in E} x_{v,j} - \sum_{(i,v)\in E} x_{i,v} = 0 \quad (v \neq s, t) \\
& && \sum_{(t,j)\in E} x_{t,j} - \sum_{(i,t)\in E} x_{i,t} = -1 \\
& && x_{i,j} \in \{0, 1\}
\end{aligned}
$$

$x_{i,j}$ は枝 (i, j) が路に含まれるならば 1，そうでないなら 0 をとる変数．目的関数は _____ だから，最小化．各制約条件の一つ目の和は注目してい

る節点から _____，各制約条件の二つ目の和は 注目している節点に _____．ソースではその差が 1，シンクでは −1，その他の節点では 0．

これをシンプレックス法を用いて解くこともできるが，もっと効率よく解く手法がある．

以後，全ての枝の長さは正の値とする．

3.2.2 ダイクストラのアルゴリズム

ダイクストラのアルゴリズム: 枝に長さが与えられたグラフ $G = (V, E)$ と，1 つのソース $s \in V$ があたえられたとき，ソースから任意の節点への最短路を求めるアルゴリズム．

木 (tree)： 閉路を持たないグラフ．特に，あるグラフ $G = (V, E)$ に対して，全ての節点を含む木をスパニングツリーという．

最短路木： 1 つの節点から各節点への最短路からなる木（スパニングツリー）．

ダイクストラのアルゴリズムでは，ソースから順に，到達可能な節点にラベル（ソースからの距離，直前の点）を付けながら，最短路木をグラフ全体に張っていく．具体的な手順は以下のとおり．

ダイクストラのアルゴリズム

(Step 0) 初期状態として，$S = \{\}$, $d(s) = 0$, $d(i) = \infty$ for all $i \in V \setminus \{s\}$ とする．

(Step 1) $i \notin S$ である節点のうち、$d(i)$ が最も小さい i を選ぶ。これを $\hat{i}$ とする。すなわち、

$$d(\hat{i}) = \min_{i \notin S}(d(i)).$$

(Step 2) $S \leftarrow S \cup \{\hat{i}\}$ とする．$(\hat{i}, j) \in E$ である全ての $j \notin S$ に対して，

$$d(j) > d(\hat{i}) + a_{\hat{i},j} \text{ ならば } d(j) \leftarrow d(\hat{i}) + a_{\hat{i},j},\ p(j) \leftarrow \hat{i}$$

とする．

(Step 3) $S = V$ ならば終わり．そうでないなら，Step 1 に戻る．

- S は，_____ 節点の集合，
- 節点 j に対して，$d(j)$ は現在の繰り返しにおける _____，
- $p(j)$ は節点 j の一つ前（ソース側）にある節点番号．

例として，図 3.1 を考えよう．$V = \{A, B, C, D, E\}$, ソースは A.

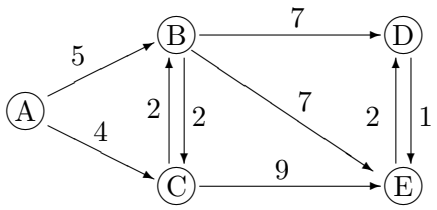


図 3.1: 枝に値が与えられているネットワーク

(0) 初期状態は $S = \{\}, d(A) = 0, d(B) = d(C) = d(D) = d(E) = \infty$.

(1) $\min(0, \infty, \infty, \infty, \infty) = 0$ より，$\hat{i} = A$ を選択．$S = \{A\}$.
$(A, j) \in E$ である j は B と C.
$d(B) > d(A) + 5$ より，$d(B) = 5, p(B) = A$,
$d(C) > d(A) + 4$ より，$d(C) =$____, $p(C) =$____.

(2) $\min(5, 4, \infty, \infty) = 4$ より，$\hat{i} = C$ を選択．$S = \{A, C\}$.
$(C, j) \in E$ である j は B と E.
$d(B)$__$d(C) + 2$ より，$d(B) =$____, $p(B) =$____,
$d(E)$__$d(C) + 9$ より，$d(E) =$____, $p(E) =$____.

(3) $\min(5, \infty, 13) = 5$ より，$\hat{i} =$__ を選択．$S = \{$_____$\}$.
$d(D)$__$d(B) + 7$ より $d(D) =$____, $p(D) =$____,
$d(E)$__$d(B) + 7$ より，$d(E) =$____, $p(E) =$____.

(4) $\min(12, 12) = 12$ より，$\hat{i} =$__ または __．ここでは D を選択すると，$S = \{$_____$\}$.
$d(E)$__$d(D) + 1$ より $d(E) =$____, $p(E) =$____.

(5) $\hat{i} =$__．$S = \{$_____$\}$．S に入っていない節点がないので何もしない．$S = V$ となったので終わり．

これを表で書き表すと，

	A	B	C	D	E
(0)	$\underline{0}$	∞	∞	∞	∞
(1)		5(A)	$\underline{4}$(A)	∞	∞
(2)		$\underline{5}$(A)		∞	13(C)
(3)				$\underline{12}$(B)	12(B)
(4)					$\underline{12}$(B)
(5)					

となる．下線の引いてあるのは，次の繰り返しにおいて選ばれた節点と，ソース A からの最短距離を示している．また，最短路木は図 3.2 のようになる．

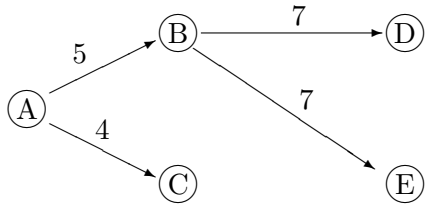


図 3.2: 図 3.1 に示したネットワークの最短路木

[**課題 3.1**] 図 3.3 に示されたネットワークに対して，ソースを節点 3 としてダイクストラのアルゴリズムを使うことにより，他の節点への最短路木を求めよ．

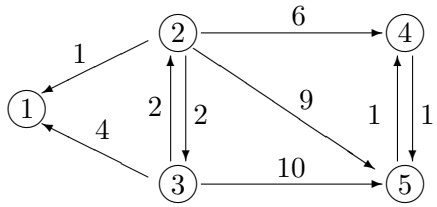


図 3.3: 枝に値が与えられているネットワークその 2

3.2.3　ダイクストラのアルゴリズムの正当性と効率

ダイクストラのアルゴリズムでは，各繰り返しの終わりの状態において，

$$d(i) = [\ S \text{ に含まれる節点のみを経由して } i \text{ に到達する最短路の長さ}] \qquad (3.1)$$

となっている．また，節点 i が S に含まれる場合，

$$d(i) = [\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}] \qquad (3.2)$$

となっており，S に含まれている節点の範囲内では最短路木が得られている．

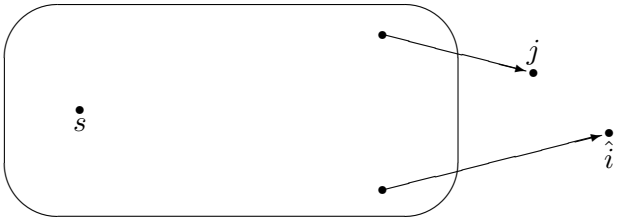
式 (3.1)(3.2) を証明しよう.（帰納法を使う）

まず，一回目の繰り返しの終わりでは，S は必ずソースのみを含み，ソースから直接到達可能な節点 i（ソースを含む）に対してのみ，$d(i)$ にはソースからの距離が入っている．したがって，式 (3.1)(3.2) は満たされている．

次に，ある繰り返しの始めで式 (3.1)(3.2) が満たされていると仮定しよう．その繰り返しで選ばれた接点を $\hat{i}$ とする．繰り返しの始めでは $\hat{i}$ は S に入っていないので、仮定より，$d(\hat{i})$ は「S に含まれる節点のみを経由して $\hat{i}$ に到達する最短路の長さ」である．もし $d(\hat{i})$ が「本当の $\hat{i}$ までの最短路の長さ」ではないとすると，S に含まれない節点を経由して $\hat{i}$ に到達する最短路が存在することになる．その路を

$$(s \to S \text{ 内の節点} \to S \text{ 外の節点 } j \to \cdots \to \hat{i})$$

と書くと，$d(j)$＿＿$d(\hat{i})$ であるはずである.（下図参照.）



これは，この繰り返しで $\hat{i}$ が選ばれたことに矛盾する．したがって，$d(\hat{i})$ は「本当の $\hat{i}$ までの最短路の長さ」であり，この繰り返しの終わりで式 (3.2) は満たされている．また，$d(i)$ の更新の方法より，$i \notin S$ について式 (3.1) が成り立つことは明らか.

以上より，全ての繰り返しにおいて式 (3.1)(3.2) が成り立っている．つまり，$S = V$ となった時点で，全ての節点について最小距離が求まっている．

次に，アルゴリズムの計算量について考えてみよう.

計算の複雑さ： 入力にしたがって計算して出力を出すアルゴリズムを考える．入力が n ビットのときに出力が終わるまでに必要なステップ数が高々$T(n)$ であるとき，そのアルゴリズムの計算量を $T(\cdot)$ で表す．ビット数が等しい入力でも，計算が早く終わる場合と長くかかる場合があるから，$T(\cdot)$ は，最悪の場合を考える．

オーダー記法： $T(\cdot)$ が，おおむね $f(\cdot)$ に比例するとき，アルゴリズムの計算の複雑さを $O(f(\cdot))$ で示す.（オーダーは $f(\cdot)$.）
例）$T(x) = 5x^3 + x + \log x$ であるアルゴリズムの計算の複雑さは $O(x^3)$.
$\log x$ は x よりも低次，2^x や $x!$ は全ての x^d より高次.
足し算の計算の複雑さは $O(x)$，掛け算は $O(x^2)$.

ここで，計算量を表す単位の 1 ステップは，_____ である．計算の複雑さによって，アルゴリズムは次の二つに大別できる．

- アルゴリズムの計算の複雑さがある定数 d を使って $O(x^d)$ と書けるとき，このアルゴリズムを _____ という．
- 逆に，計算の複雑さが $O(2^x)$ や $x!$ のように x のべき乗で表せないとき，そのアルゴリズムを _____ という．

ダイクストラのアルゴリズムでは，入力のビット数は節点の数 $|V| = n$, 枝の数 $|E| = m$ に依存する．しかし，$m < n^2$ と考えると，アルゴリズムの計算量を n のみで表せばよい．アルゴリズムの各ステップごとに考えると，

- (Step 0) は $2n$ 回の代入
- (Step 1) は繰り返し一回あたり最大 n 回の比較
- (Step 2) は全ての繰り返し全体で，m 回の比較と高々$2m$ 回の代入
- 繰り返しの回数は n 回

より，_____（$m < n^2$ に注意）．つまりこれは_____ アルゴリズム．