アルゴリズム B 第 4 回　　2022 年 10 月 31 日

4　グラフと木

グラフ

グラフ (graph) は，**頂点** (vertex) と頂点を結ぶ**辺** (edge) をまとめたものである．辺に向きを考える**有向グラフ** (directed graph) と向きを考えない**無向グラフ** (undirected graph) がある．本講義では無向グラフのみを考える．

形式的には，グラフは次のように定義される．V を集合として，$E \subset \{\{u, v\} \mid u, v \in V, u \neq v\}$ とする．$G = (V, E)$ を**無向グラフ**という．V の要素を**頂点**，E の要素を**辺**という．辺 $\{u, v\}$ を (u, v) と表す．$e = (u, v) \in E$ に対して，u, v を辺 e の**端点** (endvertex) という．また，辺 e は頂点 u と v に**接続** (incident) しているといい，頂点 u と v は**隣接** (adjacent) しているという．

この定義では，2 頂点 u, v に接続する辺は 1 本以下である．u, v に接続する辺が 2 本以上あることを許す場合もあり，このような辺を**多重辺** (multiple edge, parallel edge) という．また，辺の端点が一致することを許すこともあり，このような辺を**自己ループ** (self-loop)，**ループ** (loop) などという．多重辺も自己ループも持たないグラフを**単純グラフ** (simple graph) という．

本講義では単純無向グラフのみを考え，頂点集合 V，辺集合 E は有限集合であるとする．

$G = (V, E)$ をグラフとする．G の頂点の列 $p = (v_0, v_1, \ldots, v_k)$ が $(v_{i-1}, v_i) \in E$ $(1 \leq i \leq k)$ を満たすとき，p を頂点 v_0 から v_k への**長さ** (length) k の**道** (path) という．$v_0, v_1, \ldots, v_k$ がすべて異なるとき，p は**単純** (simple) であるという．

道 $p = (v_0, v_1, \ldots, v_k)$ が**閉路** (cycle) であるとは，$k > 0$, $v_0 = v_k$ であり，p に含まれる辺がすべて異なることをいう．さらに，$v_1, v_2, \ldots, v_k$ がすべて異なるとき，閉路 p は**単純** (simple) であるという[*1]．

グラフの表現

$G = (V, E)$ をグラフとして，$V = \{v_1, v_2, \ldots, v_n\}$, $E = \{e_1, e_2, \ldots, e_m\}$ とする．$n \times n$ 行列 $A = [a_{ij}]$ を

$$a_{ij} = \begin{cases} 1 & ((v_i, v_j) \in E \text{ のとき}) \\ 0 & ((v_i, v_j) \notin E \text{ のとき}) \end{cases}$$

で定義し，G の**隣接行列** (adjacency matrix) という．$n \times m$ 行列 $B = [b_{ij}]$ を

$$b_{ij} = \begin{cases} 1 & (e_j \text{ が } v_i \text{ に接続するとき}) \\ 0 & (e_j \text{ が } v_i \text{ に接続しないとき}) \end{cases}$$

で定義し，G の**接続行列** (incident matrix) という．

定義から明らかなように，無向グラフの隣接行列は対称行列である．

隣接行列，接続行列のどちらか一方からグラフの構造を復元することができる．よって，グラフを計算機上で表現するために隣接行列または接続行列を用いることができる．行列は多次元配列によって実装できる．この場合に必要な記憶領域は，隣接行列では $O(n^2)$，接続行列では $O(nm)$ である．

[*1] グラフに関する用語の定義は分野・著者によって大きく異なる．ここでは初回に挙げた参考書 [1, 2] におおよそ従った．岩波数学辞典 [3] では，本講義の道・単純道・閉路・単純閉路はそれぞれ歩道 (walk)・道・回路 (circuit)・閉路にほぼ対応する．

また，グラフを**隣接リスト** (adjacency list) で表現することもできる．隣接リストでは，頂点ごとにその頂点に隣接する頂点を格納した連結リストを構成することでグラフを表現する．隣接リストに必要な記憶領域は $O(n+m)$ である．

木と根付き木

$G=(V,E)$ をグラフとする．任意の頂点 $u,v\in V$ に対して，u から v への道が存在するとき，G は**連結** (connected) であるという．

単純閉路を持たないグラフを**森** (forest) といい，連結な森を**木** (tree) という．

命題 4.1. $T=(V,E)$ を木とする．このとき，任意の頂点 $u,v\in V$ に対して，u から v への単純道がただ一つ存在する．

特別な頂点が一つ定まっている木を**根付き木** (rooted tree) といい，この特別な頂点を**根** (root) という．隣接する 2 頂点 u, v に対し，u の方が根に近いとき，u を v の**親** (parent)，v を u の**子** (child) という．同じ親を持つ頂点を**兄弟** (sibling) という．また，頂点 v から根までの単純道に頂点 u が含まれるとき，u を v の**先祖** (ancestor)，v を u の**子孫** (descendant) という．子を持たない頂点を**葉** (leaf) という．頂点 v の子孫全体とそれらに接続する辺がなす木は v を根とする根付き木になる．この根付き木を v を根とする**部分木** (subtree) という．頂点 v から根への単純道の長さを v の**深さ** (depth) という．頂点の深さの最大値を根付き木の**高さ** (height)[*2]という．

各頂点の子の数が k 以下であるような根付き木を ***k* 分木** (k-ary tree) という．2 分木 (binary tree) においては，ある頂点の子がいくつあるかにかかわらず，子が左側か右側かを常に指定し，**左の子** (left child)，**右の子** (right child) という．また，2 分木 T に対して，T の根の左の子，右の子を根とする部分木をそれぞれ T の**左部分木** (left subtree)，**右部分木** (right subtree) という．

2 分木は，各頂点をデータと 2 個のポインタ *left*, *right* の組として表現できる．*left* は左の子を，*right* は右の子を指すポインタである．各頂点に親を指すポインタを加えることもある．k 分木も同様に表現できる．

参考文献

[1] 平田富夫『アルゴリズム設計とデータ構造』（サイエンス社，2015）

[2] T. コルメンほか著，浅野哲夫ほか訳『アルゴリズム・イントロダクション』第 3 版 第 1 巻（近代科学社，2012）

[3] 日本数学会編『岩波数学辞典』第 4 版（岩波書店，2007）

[*2] 頂点の深さの最大値に 1 加えたものを高さと呼ぶこともある．