グラフ理論

平成 28 年 5 月 28 日

1　基本用語

1.1　グラフの構成

枝 (edge)

頂点 (vertex)

自己ループ 一つ枝の両端が同じ頂点であるもの

並行枝 二つの枝が両端同じ頂点であるもの

単純グラフ 自己ループも並行枝も持たないグラフ

連結グラフ すべての頂点が繋がっているグラフ

非連結グラフ 連結グラフ以外のグラフ

有向グラフ 枝に向きがあるグラフ

無向グラフ 枝に向きがないグラフ

1.2　グラフの表現

グラフは次のように表現される

$$G = (V, E)$$

V : 頂点集合

E : 枝集合

$V = v_1, v_2, v_3, \ldots$, $E = e_1, e_2, e_3, \ldots$ と表現されるとする時 $e_1 = (v_1, v_2)$ のようにして枝は表される

$e_i = (1, 3)$ という枝がある時頂点 1 と 3 は隣接 (adjacent) している、頂点 1 と枝 e_i は接線 (incident) しているという

定義 1.1 (隣接行列 (adjacency matrix))
頂点数 n のグラフについて次のような $n \times n$ 行列で表される

頂点 i と頂点 j が隣接している時、行列の (i, j) 成分が 1
頂点 i と頂点 j が隣接していない時、行列の (i, j) 成分が 0

定義 1.2 (接続行列 (incidnet matrix))
頂点数 n、枝数 m のグラフは次の行列で表される

頂点 i と枝 j が接続している時、行列の (i, j) 成分が 1
頂点 i と枝 j が接続していない時、行列の (i, j) 成分が 0

定義 1.3 (隣接リスト (adjacency list))
各頂点について接続する枝をリスト構造で表現したもの

同型性 (isomorphism)

定義 1.4 (同形成)
グラフ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ について以下の条件を満たす V_1 から V_2 への *1* 対 *1* 写像 f が存在している時 G_1 と G_2 は同型 *(isomorphic)* であるという

任意の $u, v \in V_1$ について

$$(u, v) \in E_1 \Leftrightarrow (f(u), f(v)) \in E_2$$

f を同型写像という

1.3 グラフの次数

定義 1.5 (グラフの次数)
頂点 v につながる枝の数を v の次数 *(degree)* という $d(v)$ と書く

グラフの次数はそのグラフの頂点の中で最大の次数をグラフの次数とする
グラフ G の次数は $\Delta(G)$ と書く

1.4 様々なグラフ

木 (tree) 閉路のない連結なグラフ

森 (forest) 閉路のないグラフ (連結でなくても良い)

木の中の次数 1 の頂点を葉 (leaf) という
木のひとつの頂点を根 (root) として指定する場合もある

定理 1.1
n 頂点のグラフ G について以下の全ては同値である

1. G は木である
2. G には閉路ではなく枝が $n-1$ 本ある
3. G は連結で枝が $n-1$ 本ある
4. G の任意の 2 頂点を結ぶ道はちょうど 1 通りできる
5. G に閉路はないが、どこに枝を付け加えても閉路ができる

完全 2 分木

- 根からはに向かって常に 2 つの枝分かれをしている
- 根からどの葉までも距離が同じ
- 高さ h の完全 2 分グラフの葉の数は $2^{h+1}-1$

平面的グラフ

枝が交差しないように平面上に描画できるグラフ. 平面的グラフを枝が交差しないように描画したものを平面グラフと呼ぶ

定理 1.2 (オイラーの公式)
連結な平面グラフの頂点数を n, 枝数を m, 面数を h とすると

$$n+h=m+2$$

二部グラフ (bipartite graph)

頂点集合を各部分集合内で枝がないように 2 つの部分集合に分割できるグラフ
二部グラフであることを明示するために

$$G=(V_1,V_2,E)$$
$$V_1 \cup V_2 = V, V_1 \cap V_2 = \emptyset$$

完全二部グラフ (Complete bipartite graph)

二部グラフで異なる部分集合の全てに枝があるものを完全二部グラフという
n_1 頂点と n_2 頂点の完全二部グラフを $k_{n_1 n_2}$ と書く. 枝数は $n_1 n_2$

k 部グラフ (k-partite graph)

頂点集合を同じグループ内に枝がないように k 個に分割できるもの

完全グラフ (complete graph)

すべての 2 頂点間に枝があるグラフ
n 頂点完全グラフを k_n と書く
k_n の枝数は ${}_nC_2$

正則グラフ (regular graph)

すべての頂点の次数が同じグラフを正則グラフという. すべての頂点の次数が k のグラフを k 正則グラフという

1.5 歩道

歩道には以下のようなものがある

歩道 (walk) 頂点と枝が交互にあわられる列. 連続する頂点と枝は接続している. 頂点から始まって頂点で終わる列

小道 (trail) 同じ枝が 2 度現れない歩道

回路 (circuit) 最初と最後の頂点が同じ小道

道 (path) 同じ頂点が 2 回現れない小道

閉路 (cycle) 最初と最後の頂点が同じである道

定理 1.3
隣接行列の k 乗の (i,j) 成分は頂点 i から j へ至る長さ k の歩道の数である

1.5.1 2 頂点間の距離 (distance)

頂点 u と v を両端とする道の中で最短のものの長さを 2 頂点間の距離という.$d(u,v)$ と表す

1.6 部分グラフ (subgraph)

定義 1.6 (部分グラフ (subgraph))
グラフ $G=(V,E)$,$G'=(V',E')$ として以下を満たすとき G' は G の部分グラフという

$$V' \subseteq V, E' \subseteq E, \forall e=(u,v) \in E' \text{について}$$
$$u \in V' \text{かつ } v \in V'$$

定義 1.7 (誘導部分グラフ (induced subgraph))
$G=(V,E)$,$G'=(V',E')$,$V' \subseteq V$ に対して以下が成り立つとき G' を V' で誘導される G の誘導部分グラフといい $G[V']$ と書く

$$u \in V', v \in V' \text{について } (u,v) \in E \Leftrightarrow (u,v) \in E'$$

1.7 補グラフ (complement graph)

グラフ $G = (V, E)$ に対して $\overline{G} = (V, \overline{E})$,$\overline{E} = \{(u, v)|(u, v) \notin E\}$ を G の補グラフという

1.8 グラフの次数列

グラフの頂点の次数を降順に並べたものをグラフの次数列という. 非負整数の降順列があるグラフの次数列になっているときグラフか可能列という

定理 1.4

$(a_1, a_2, \cdots, a_n)$ がグラフ化可能であるための必要十分条件は $(a_2-1, a_3-1, \cdots, a_{a_1+1}-1, a_{a_1+2}, \cdots, a_n)$ を降順に並べ替えた列がグラフか可能であることである

2　最小全域木問題

全域木 与えられたグラフの部分グラフで木であるもの. ただしすべての頂点を含んでいる

最小全域木 与えられたグラフの全域木の中でコスト最小のもの

コスト その木に含まれる枝の重みの合計

2.1　クラスカルのアルゴリズム

Algorithm 1 クラスカルのアルゴリズム

1: **for all** 辺 e in E in 昇順 **do**
2:　**if** 採用した辺集合に e を加えても閉路を作らない **then**
3:　　e を採用
4:　**end if**
5: **end for**
6: **return**　採用した辺のコストの和

枝の重みを小さい順に処理して、現在見ている枝を選んで閉路ができなければ選ぶ、閉路ができるならば選ばないというアルゴリズム

2.2　プリムのアルゴリズム

Algorithm 2 プリムのアルゴリズム

1: $V_{new} = \{x\}$、$x \in V$
2: $E_{new} = \emptyset$
3: **while** $V_{new} \neq V$ **do**
4:　V_{new} に含まれる頂点 u と含まれない頂点 v を結ぶ重みが最小の辺 (u, v) を E から選択
5:　v を V_{new} に加える
6:　(u, v) を E_{new} に加える
7: **end while**

重みの小さな枝を使って連結成分を成長させていく

2.3　関連問題

最小シュナイター木問題 (NP 完全)

入力 枝重みつきグラフ $G(V, E)$, ターミナル集合 $T \subseteq V$

出力 G の部分木でターミナルをすべて含むもの. 木のコストが最小のもの

最短経路問題

入力 枝に重みがついたグラフ $G = (V, E)$,2 頂点 s, t.

出力 s から t への道で, 長さ最小のもの

ダイクストラのアルゴリズム

Algorithm 3 ダイクストラのアルゴリズム

1: 各頂点 $v \in V$,$d(v) \leftarrow (v = s$ ならば $0, otherwize\ \infty)$
2: 各頂点 $v \in V$,$prev(v) \leftarrow$「無し」
3: Q に V の頂点をすべて追加
4: **while** $Q \neq \emptyset$ **do**
5: $u \leftarrow Q$ から $d(u)$ が最小である頂点を取り除く
6: **for all** u からの辺がある頂点 $v \in V$ **do**
7: **if** $d(u) > d(u) + length(u, v)$ **then**
8: $d(v) \leftarrow d(u) + length(u, v)$
9: $prev(v) \leftarrow u$
10: **end if**
11: **end for**
12: **end while**

最初の頂点から一つ枝を渡った先の頂点の値を枝の重み分増やしそのあと頂点の中で一番重みの小さい頂点について同様のことを繰り返すアルゴリズム

3　オイラー回路とハミルトン閉路

オイラー回路　グラフの各枝をちょうど 1 回ずつ通る回路

ハミルトン閉路　グラフの各頂点をちょうど 1 回ずつ通る回路

定理 3.1
連結グラフがオイラー回路を持つ必要十分条件は奇数字数の頂点を持たないことである

定理 3.2 (ディラック (Dirac) の定理)
$G = (V, E), |v| \geq 3$ とする.
$\forall v \in V$ について

$$d(v) \geq \frac{|v|}{2} \Rightarrow G \text{ はハミルトン閉路を持つ}$$

定理 3.3 (オーレ (Ore) の定理)
$G = (V, E), |v| \geq 3$ とする.
隣接していないすべての頂点ペア v, w について

$$d(v) + d(w) \geq |v| \Rightarrow G \text{ はハミルトン閉路を持つ}$$

4　グラフの彩色問題

4.1　頂点彩色

グラフの各頂点に色をつける. ただし隣接する頂点は違う色

定義 4.1 (k-頂点色彩問題)
グラフの k-頂点色彩問題とは写像 $f : v \to \{1, 2, 3, \cdots, k\}$ で任意の枝 $(u, v) \in E$ に対して $f(u) \neq f(v)$ を満たすものである

G に k-頂点彩色が存在するとき,G は k-頂点彩色可能という
G が k-頂点彩色可能な最小の k を G の頂点彩色数といい $\chi(G)$ と表す

定理 4.1
グラフ G は $(\Delta(G) + 1)$-彩色可能

定理 4.2 (ブルックス (Brooks) の定理)
連結グラフ G が完全グラフでも奇数長閉路でもなければ G は $\Delta(G)$ 彩色可能である

定理 4.3 (先の定理より弱い定理)
正則でないグラフは $\Delta(G)$-彩色可能

4.1.1　2-彩色問題

定理 4.4

$$G \text{ が } 2 \text{ 彩色可能} \Leftrightarrow G \text{ が二部グラフ}$$

定理 4.5

$$G \text{ が } 2 \text{ 彩色可能} \Leftrightarrow G \text{ は奇数長閉路を持たない}$$

Algorithm 4 2 彩色可能の判定

```
ある頂点 v とし S_1 ← {v}
n ← 1
while V ≠ ∪_{i=1}^{n} S_i do
    S_{n+1} ← S_n の頂点のいずれかに隣接しかつまだ S_1, S_2, ..., S_n に入っていない
    if S_{n+1} 内に枝がある then
        return false
    end if
    n ← n + 1
end while
return true
```

定理 4.6
平面グラフは 4 彩色可能である. またこれは四色定理 (隣接領域が異なる色になるように塗るには 4 色あれば十分であるという定理) と同値である

4.2 辺彩色

同じ頂点に接続する枝が異なる色になるように色をつける.
k 色で辺彩色できるとき k-辺彩色可能という.
G が辺彩色可能な最小の k を G の辺彩色数といい $\chi'(G)$ と書く

定理 4.7 (ビジング (vizing) の定理)

$$\forall G, \Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$$

定理 4.8 (ケーニッヒ (knig) の定理)

$$\forall G \in BinaryGraph, \chi'(G) = \Delta(G)$$

5　最大流問題

定義 5.1 (最大流問題)
入力 有向グラフ,2 頂点 s, t, 枝の重み $w(e)$

出力 s から t までに最大どれだけ流せるか
　フロー関数を f とした時

$$\forall e, f(e) \leq w(e)$$
$$\sum_{v_{in}} f(e) = \sum_{v_{out}} f(e)$$

を満たす

5.1　最大フローを求めるアルゴリズム

残余ネットワーク

　現在のフローに対してあとどれだけの量を流すことができるかを表したグラフ. 各枝 e に対して正方向に $w(e) - f(e)$ の重み, 逆方向に $f(e)$ の重みをつける

増大道 (augmenting path)

　残余ネットワーク上で s から t に至る道. 増大道の値はその道上で最も小さい重み. 増大道にその値を増大道にその値を流してグラフを書き換える

フォード・ファルカーソン (Ford-Fulkerson) 法

1. 現在のフローを決める
2. 残余ネットワークを作る
3. if 増大道がある then フローを更新し 1. へ
   else 現在のフローを出力

定理 5.1
フロー f が最大フローであるための必要十分条件は f の残余ネットワークに増大道が存在しないことである

5.2　グラフのカット

入力 最大フロー問題と同じ

出力 頂点集合の 2 分割, ただし 2 頂点 s, t は異なる側に

カットのサイズ Y 側から $V - Y$ 側に向かう枝の重みの総和

定理 5.2 (最大フロー最小カットの定理)
任意のグラフについて

$$\text{最大フローの値} = \text{最小カットのサイズ}$$

6　マッチング問題

頂点を共有しない枝の集合を探す.
マッチング M に含まれる枝に接続している頂点を「マッチしている」という. マチングのサイズは M に含まれている枝の数.

極大マッチング (maximal matching)

そのマッチングに (マッチングであることを保ちつつ) 枝を追加できないもの

最大マッチング (maximum matching)

サイズが最大のマッチング

定理 6.1
任意のグラフにおいて

$$(\text{マッチングのサイズ}) \geq (\text{最大マッチングのサイズ})$$

完全マッチング (perfect matching)

すべての頂点がマッチしているマッチング

6.1　二部グラフのマッチング

頂点集合 $S \subseteq U$ に対して $\delta(S)$ を S のいずれか少なくとも一つに隣接する頂点集合とする

定理 6.2 (ホールの定理)
$G(U, V, E)$ が完全マッチングを持つための必要十分条件は任意の $S \subseteq U$ に対して

$$|\delta(S)| \geq |S| (Hall \text{ 条件})$$

となることである

6.2　ハンガリー法

二部グラフの最大マッチングを求めるアルゴリズム

交互道 (alternating path)　M の枝と $E \backslash M$ の枝を交互に使う道

増大道 (augmenting path)　交互道でかつ最初と最後の頂点がマッチしていないもの. 最初と最後の枝は $E \backslash M$. 増大道上でマッチングに使われている枝と使われていない枝を交換するとサイズが一つ大きなッチングが得られる

アルゴリズム

1. 現在のマッチングを求める
2. if そこに増大道がある then それに沿ってサイズの一つ大きなマッチングを求め,1. へ
   else return マッチング M

増大道を見つける手続き

1. M でマッチングしていない頂点の一つを u とする
2. u に隣接している V の頂点集合を T_1 とする
3. T_1 の中でマッチしていない頂点があれば増大道が見つかった
4. T_1 の頂点が全てマッチしているならそれらにマッチしている U の頂点集合を S_1 とする
5. S_1 のいずれかの頂点に隣接する V の頂点で T_1 に入っていないものを T_2 とする
6. T_2 の中にマッチしていない頂点があれば増大道が見つかった
7. T_2 の頂点が全てマッチしていればそれらとマッチしている U の頂点集合を S_2 とする. 以上を繰り返す

これを全ての u の選び方について行う