2.1 最小全域木

驚くべきことに，シンプルなアルゴリズムによって最小全域木を発見できる．ここでは貪欲法（greedy algorithm）に基づいた，各ステップで最もコストの低い選択をするアルゴリズムを 2 つ紹介する．

Kruskal 法の概要

G の全域森 $H=(V,F)$ を，最初は $F=\emptyset$ として保存する．
各ステップにおいて，森であることを保つようなコスト最小の枝 $e\notin F$ を加える．
H が全域木となったら止める．

この方法は 1956 年に Kruskal によって述べられた．次のアルゴリズムは Prim 法として知られている．

Prim 法の概要

木 $H=(V(H),T)$ を，最初は V の元として何らかの頂点 r を取って $V(H)=\{r\}, T=\emptyset$ として保存する．
各ステップにおいて，H が木であることを保つようなコスト最小の枝 $e\notin T$ を T へ加える．
H が全域木となったら止める．

まずはこれらのアルゴリズムが最小全域木を見つけることを示す．その後に，これらを効率化できることについて書く．

MST アルゴリズムの正当性

正当性を示す前に，いくつかの準備をする．

定義 2.1 連結性に関する基本的な記号

$G=(V,E)$ とし，A を V の部分集合とする．$\delta(A),\gamma(A)$ を

$$\delta(A) := \{vw\in E | v\in A, w\notin A\}$$

$$\gamma(A) := \{vw\in E | v,w\in A\}$$

と定義する．$\delta(A)$ を G のカット（*cut*）という．

定理 2.3

グラフ $G=(V,E)$ について，次が成立する．

$$G \text{ が連結である} \Leftrightarrow \emptyset \neq A \neq V \text{ かつ} \delta(A)=\emptyset \text{ となるような } V \text{ の部分集合 } A \text{ が存在しない}$$

証明

(⇒) について

対偶を示す．仮に $\emptyset\subsetneq A\subsetneq V$ かつ $\delta(A)=\emptyset$ となるような A が存在したとすると，$v\in A$ から $w\notin A$ への路が存在せず，G が連結でないとわかる．

(⇐) について

対偶を示す．G は非連結なので $u,v\in V$ を u から v への路が存在しないように取れる．$A:=\{w\in V|u$ から w への路が存在する $\}$ とすると，$u\in A, v\notin A$ より $\emptyset\subsetneq A\subsetneq V$ である．さらに $\delta(A)\neq\emptyset$ と仮定すると，$pq\in E$ を $p\in A, q\notin A$ となるように取れるが，p への路に pq を追加することで u から q への路が作れてしまい $q\in A$ となり矛盾する．よって $\delta(A)=\emptyset$ とわかる．以上より条件をみたす A が存在する．$\square$

定義 2.2

$G=(V,E)$，$A\subseteq E$ とする．A が G のある最小全域木の枝集合の部分集合であるとき，A を最小全域木へ拡張できる（*extendible*）という．

定義 2.3 連結成分（component）

グラフ $G=(V,E)$ と $v\in V$ について，$C_v := \{w\in V | w$ から v への路が存在する $\}$ と定義する．また，G の部分グラフ H がある $v\in V$ を使って $H=G[C_v]$ と表せるとき，H を G の連結成分という．

補題 2.7

$H=(V,T)$ を G の全域木とし，$e=vw\in G\setminus H$ とする．また，$f\in T$ を T 上の v から w への初等的な路 P に現れる枝とする．このとき，$H'=(V,(T\cup\{e\})\setminus\{f\})$ は G の全域木．

証明

$P=v,...,v_f,f,w_f,...,w$ とする．任意に $a,b\in V$ を取ると，T 上の a から b への路 W が存在する．ここで，W に現れた v_f,f,w_f の部分を $v_f,...,v,e,w,...,w_f$ に置き換えることで H' 上の路へと変形できる．よって，H' は連結であり，補題 2.2 より G の全域木とわかる．□

定理 2.4

$B\subseteq E$ は $G=(V,E)$ の最小全域木へ拡張できるとする．また，D を G のあるカットで $B\cap D=\emptyset$ をみたすものとし，e を D のコスト最小の枝とする．このとき，$B\cup\{e\}$ は最小全域木へ拡張できる．

証明

$H=(V,T)$ を $B\subseteq T$ をみたす G の最小全域木とする．$e\in T$ のときは明らかなので，$e\notin T$ のときを考える．$e=vw,(v,w\in V)$ とし，P を H 上の v から w への初等的な路とする．D がカットであることから $G\setminus D$ に含まれるような v から w への路は存在しないので，P はある $f\in D$ を含む．$c_f\geq c_e$ と補題 2.7 より，$(V,(T\cup\{e\})\setminus\{f\})$ もまた最小全域木である．$D\cap B=\emptyset$ から $f\notin B$ であり，$B\cup\{e\}$ は最小全域木へ拡張できるとわかる．□

定理 2.5

任意の連結グラフ G と枝のコスト c に対して，*Prim* 法は最小全域木を見つける．

証明

まず，各ステップで $\delta(V(H))=\{f\in E\mid f$ を H に追加したときに木であることを保つ $\}$ が成立する．これは，補題 2.1 より H へ追加して閉路ができる枝は両方の端点が H に含まれる枝であり，さらに定理 2.3 から，H へ追加して非連結となる枝は両方の端点が H に含まれない枝であることから従う．よって，各ステップでは $\delta(V(H))$ の中でコスト最小の枝が選ばれる．そして G が連結であることと定理 2.3 より $\delta(V(H))$ は H が G の全域木となるまで空集合にならないので，H は全域木となってこのアルゴリズムが停止するとわかる．さらに，空集合は最小全域木へと拡張できることと，各ステップにおいて $B=T,D=\delta(V(H))$ として定理 2.4 を適用した結果から，数学的帰納法により $E(H)$ が常に最小全域木へ拡張できるとわかる．よって，*Prim* 法は最小全域木を見つける．□

定理 2.6

任意の連結グラフ G と枝のコスト c に対して，*Kruskal* 法は最小全域木を見つける．

証明

まず，$S_1,...,S_k$ をあるステップでの H の各連結成分の頂点集合とする．補題 2.1 より，追加される枝としては $\displaystyle\bigcup_{i=1}^{k}\delta(S_i)$ の中でコストが最小の枝

が選ばれると分かる．H が全域木でなくて，なおかつ $\bigcup_{i=1}^{k} \delta(S_i) = \emptyset$ となると仮定すると，$\delta(S_i) = \emptyset, \emptyset \subsetneq S_i \subsetneq V$ となり，定理 2.3 より G が連結であることに矛盾する．よって，H は全域木となってこのアルゴリズムは停止する．また，空集合は最小全域木へ拡張できる．そして，あるステップで $E(H)$ に追加される枝 e の端点が $\delta(S_i)$ に含まれているとすると，$B = E(H), D = \delta(S_i)$ として定理 2.5 を適用できて，$E(H) \cup \{e\}$ も最小全域木へ拡張できるとわかる．以上から，数学的帰納法より *Kruskal* 法は最小全域木を見つける． □

MST アルゴリズムの効率性

まずはグラフ $G = (V, E)$ を保存するためのデータ構造から考えよう．各頂点 v に対して，v を端点にもつ枝のリスト L_v をもっておくとする．そして，各枝に対してコストが定まっているとき，コストのデータを枝と一緒に保存する．以下では複雑さを見積もる上で $n = O(m), m = O(n^2)$ と仮定する．（$n = |V|, m = |E|$）定理 2.5 の証明における観察から Prim 法の擬似コードを次のように書ける．

Prim 法の擬似コード

$H = (V(H), T)$ を $(\{r\}, \emptyset)$ として初期化する
while H が全域木でない **do**
　　T に $\delta(V(H))$ の最小コストの枝を加える

主張 2.1
$\delta(V(H))$ を特性ベクトルとして保存することで，計算量を $O(nm)$ にできる．

証明
$\boldsymbol{x}$ を $V(H)$ の特性ベクトルとして保存する．すると，$\delta(V(H))$ の最小コストの枝を探索するときは，$x_u \neq x_v$ なる枝 uv について最小のものをとってくればよい．そして新たに枝を加えたとき，加えた頂点について $\boldsymbol{x}$ を更新すればよい． □

主張 2.2
各 $v \notin V(H)$ について，$V(H)$ と v をつなぐ辺の中でコストが最小のもの $h(v)$ を保存しておくことで，*Prim* 法の計算量を $O(n^2)$ にできる．

証明
まず，$\delta(V(H))$ の最小コストの枝を見つけるには，各 $v \notin V(H)$ の $c_{h(v)}$ を比較すれば良く，この比較の回数は $O(n)$ である．また，$V(H)$ へ頂点 w を加えたとき，各 $v \notin H(V)$ について $h(v)$ を更新する必要がある．更新するときは，各 $v \notin H(V)$ について c_{vw} と $c_{h(v)}$ を比較するだけでよく，1 つの頂点を加えたときの更新に必要なステップは $O(n)$ で抑えられる． □

Kruskal 法は次のように書き換えられる．

Kruskal 法の擬似コード

$E = \{e_1, e_2, ..., e_m\}(c_{e_1} \leq c_{e_2} \leq ... \leq c_{e_m})$ とソートする
$H = (V, F)$ を $F = \emptyset$ として初期化する．
for $i = 1$ to m **do**
　　if e_i の端点が H の異なる連結成分に属している **then**
　　　　辺 e_i を F へ加える

主張 2.3

$Kruskal$ 法は $O(m \log m)$ で実行できる．

証明

まず，最初のソートは $O(m \log m)$ でできる．（ソートについては補足のヒープソートを参照）

次に，辺を F へ加えるステップを見よう． □