最大流問題3

1　定式化

入力 有向グラフ $D=(V,A)$ と，各枝の容量 $c:A\to\mathbb{R}_+$，入口と出口 $s,t\in V$

出力 $s-t$ フロー $f:A\to\mathbb{R}_+$ であって，流量 $\partial f(s)$ が最大のもの

2　プリフロー・プッシュ法

ディニッツ法は直感的にわかりやすい手法であったが，計算量が $O(mn^2)$ であり，密なグラフ $(m\propto n^2)$ では $O(n^4)$ となり多項式ではあるものの速いとは言いがたい．そこで，もう少し速い強多項式時間アルゴリズムを紹介しよう．

2.1　プリフロー

まず，**プリフロー**というものを定義する．プリフローは言葉の通り，フローの前段階にあたるものであり，条件を少し緩めたものとなっている．ネットワーク $\mathcal{N}$ 上の $s-t$ プリフローとは，枝上の関数 $f:A\to\mathbb{R}_+$ であって，以下の条件を満たすものをいう．

容量制約 各枝 $a\in A$ に対して，$0\le f(a)\le c(a)$.

弱境界条件 始点と終点を除いた各点 $v\in V\setminus\{s\}$ に対して，

$$\partial f(v) \;=\; \sum_{a\in\delta^+v} f(a) - \sum_{a\in\delta^-v} f(a) \;\le\; 0$$

境界は各点に対して出ていく量から入ってくる量を引いた値を定める関数であった．弱境界条件では，入口・出口以外の各点に関する境界の値が非正であることを求めており，これはすなわち入ってくる量より多くは出て行かないということである．イメージとしては，途中の点に溜まってしまうような状況も許すということである．

また，プリフロー f に関して流量保存則を満たしていない $(\partial f(v)<0)$ 点 $v\neq t$ を，f に関する**活性点**という．上のイメージに合わせれば，活性点は流れが滞って溜まっている点である．

2.2 プッシュとリラベル

プリフロー f に関する残余ネットワーク $\mathcal{N}_f$ をフローの時同様に定める．$h(s) = n,\ h(t) = 0$ かつ残余ネットワーク $\mathcal{N}_f$ 中の各枝 $a = (u, v) \in A_f$ に対して，

$$h(u) \ \leq \ h(v) + 1 \qquad (*)$$

を満たす関数 $h : V \to \mathbb{Z}_+$ を**高さ関数**と呼ぶ．

活性点 u から出る枝 $a = (u, v) \in A_f$ が $(*)$ を等号で満たすとき，a を**有資格な枝**と呼ぶ．

定義が連続して何のことだかさっぱりわからないと思うので，イメージで何を行うのかを書いておこう．まず高さが n の入口に目一杯 (出る枝の容量の総和分) の水を注ぎ込み，出る枝に流す．するとその先の各点に水が溜まっていく．水が溜まっている点 (活性点) を 1 つ選んで，その点よりちょうど一段だけ低い点に向かう枝 (有資格な枝) があれば，その枝に流せるだけ流す (**プッシュ**)．そのような枝がなければ，それができるように点を持ち上げて高くする (**リラベル**)．

これを繰り返して，水が溜まっている点がなくなれば，それはすなわちフローである．そして驚くべきことに，そうして得られるフローは最大フローとなっている．

2.3 アルゴリズム

ではイメージではなくちゃんとした表記でアルゴリズムを示そう．なお，このアルゴリズムはその特徴的な操作から一般には**プリフロー・プッシュ法**と呼ばれるが，考案者の名前を取って**ゴールドバーグ・タージャンのアルゴリズム**とも呼ばれる．

プリフロー・プッシュ法

1. $h(s) \leftarrow n,\ h(v) \leftarrow 0$ $(^\forall v \neq s)$ とし，各枝 $a \in \delta^+ s$ に対して $f(a) \leftarrow c(a)$，$a \in A \setminus \delta^+ s$ に対して $f(a) \leftarrow 0$ とする．
2. f に関する活性点がなくなるまで以下を繰り返す．
   (a) f に関する残余ネットワーク $\mathcal{N}_f$ を構成し，活性点 u を 1 つ選ぶ．
   (b) u から出る有資格な枝がなければ，$h(u) \leftarrow \min\{h(v) + 1 | (u, v) \in A_f\}$ とする．u から出る有資格な枝 $a \in A_f$ を 1 つ選び，$\alpha \leftarrow \min\{-\partial f(u),\ c_f(a)\}$ とし，$a \in A$ なら $f(a) \leftarrow f(a) + \alpha$，$a \notin A$ なら $f(\bar{a}) \leftarrow f(\bar{a}) - \alpha$ とする．
3. f を最大フロー，$\partial f(s)$ をその流量として出力する．

補題 6.1　アルゴリズム中の $h : V \to \mathbb{Z}_+$ は $\mathcal{N}_f$ に関する高さ関数．

証明　ステップ 2 の繰り返し回数に関する帰納法で示す．初期設定では s から出る枝には容量いっぱいのプリフローが流れているので，残余ネットワーク中に s から出る枝はない．すなわち $(*)$ の左辺 (枝の始点 u のラベル) として考えられるのは 0 のみで，右辺は 1 以上なので $(*)$ が成り立つ．

ステップ 2 を 1 回行う前に高さ関数であったとしよう．活性点として s, t が選ばれることはないので，$h(s) = n,\ h(t) = 0$ は不変．プッシュによって次の残余ネットワークに新たに現れる可能性がある枝については，その逆向き枝が有資格な枝 (u, v) であるので，$h(v) = h(u) - 1 < h(u) + 1$ より (v, u) に関しても $(*)$ が成り立つ．また，リラベルについては定義から明らかに $(*)$ を壊さない (左辺の値が右辺の最小値で定義されている)．$(q.e.d.)$

定理 6.2　プリフロー・プッシュ法の出力は最大フローである．

証明　まず，活性点が存在しないので f はフローである．補題 4.2 より，$\mathcal{N}_f$ 中に $s-t$ 有向道が存在しないことを示せば良い．

補題 6.1 より，アルゴリズム終了時でも h は $\mathcal{N}_f$ に関する高さ関数である．いま $s-t$ 有向道 P が取れたとすると，P に沿って $(*)$ を考えることにより

$$h(s) \ \leq \ h(v_1)+1 \ \leq \ h(v_2)+2 \ \leq \ \cdots \ \leq \ h(t)+|A_f(P)| \ < \ h(t)+n$$

が成り立つはずである．しかし，$h(s)=n,\ h(t)=0$ に矛盾．$(q.e.d.)$

3　計算量解析

ではこのアルゴリズムはどの程度速いのだろうか？まだ停止するかどうかも確認していないが，繰り返し回数の上界を求めることによって有限回の繰り返しの後に停止することを示す．

補題 6.3　任意の時点において $h(v) \leq 2n-1$ $(^\forall v \in V)$ が成り立つ．

証明　各点のラベルはリラベルによってしか変化しない．そこで，任意の活性点 v に対して，$\mathcal{N}_f$ 中に $v-s$ 有向道が存在することを示す．補題 6.1 より h は高さ関数であり，$h(s)=n$ を満たすので，そのような有向道 P に沿って $(*)$ を考えれば題意が示せる．

$$h(v) \ \leq \ h(v_1)+1 \ \leq \ h(v_2)+2 \ \leq \ \cdots \ \leq \ h(s)+|A_f(P)| \ < \ h(s)+n \ = \ 2n$$

活性点を任意に 1 つ選んで v とし，$\mathcal{N}_f$ 中で v から到達可能な点全体を U とする．A の枝で U から出る枝全体を $\delta^+(U)$，U に入る枝全体を $\delta^-(U)$ とすると，$\mathcal{N}_f$ 中に U から出る枝はない (あれば到達可能な点が増えてしまい矛盾する) ので，以下が成り立つ．

$$\sum_{u\in U}\partial f(u) \ = \ \sum_{a\in\delta^+(U)} f(a) - \sum_{a\in\delta^-(U)} f(a) \ = \ \sum_{a\in\delta^+(U)} c(a) - \sum_{a\in\delta^-(U)} 0 \ \geq \ 0$$

ここで活性点 $v\in U$ に関して $\partial f(v)<0$，その他の点 $u\in V\setminus\{s\}$ に関して $\partial f(v)\leq 0$ (弱境界条件) が成り立っているので，$s\in U$ である．$(q.e.d.)$

補題 6.4　リラベルは高々$2n^2$ 回しか行われない．

証明　リラベルが行われるときは必ずその点の高さは少なくとも 1 だけ増加する (減少するとすれば $h(u)>h(v)+1$ がある枝 $(u,v)\in A_f$ に関して成り立っていることになり，補題 6.1 に矛盾)．補題 6.3 より，各点の高さラベルは高々$2n-1$ であるので，リラベルの回数は高々$(2n-1)(n-2)$ 回である．$(q.e.d.)$

プッシュのうち，選んだ有資格な枝の容量いっぱいまで更新するものを**飽和プッシュ**，そうでないものを**非飽和プッシュ**と呼ぶ．すなわち，有資格な枝として $a=(u,v)$ を選んだ場合，$-\partial f(u)\leq c_f(a)$ ならば前者，$-\partial f(u)>c_f(a)$ ならば後者となる．

注意すべきこととして，飽和プッシュを行えばその枝は $\mathcal{N}_f$ からなくなり，非飽和プッシュを行えばその点は活性点ではなくなる．

補題 6.5　飽和プッシュは高々$2mn$ 回しか行われない．

証明　$\mathcal{N}_f$ に現れる可能性のある枝は元の逆向き枝を含めて $2m$ 本で，各枝に対して飽和プッシュを行うのは以下より高々n 回である．以下では飽和プッシュを 1 回行った後に，同じ枝に対して飽和プッシュを行う際には h の値が少なくとも 2 増加していることを示す．(補題 6.3 よりこれで十分)

枝 $a = (u, v) \in A_f$ に対して飽和プッシュを行ったとすると，この時点では $h(u) = h(v) + 1$ が成り立っている．この枝に再び飽和プッシュを行うためには，逆向き枝 $\bar{a} = (v, u)$ に対してプッシュを行う必要があり，その時点では $h(v) = h(u) + 1$ が成り立っている．h の値はアルゴリズムを通して減少しないので，再び飽和プッシュを行うときの h の値はこの時点の値を下回らない．すなわち，2 式を満たす時の h をそれぞれ h_1, h_2 とすれば，

$$h(u) \;=\; h(v)+1 \;\geq\; h_2(v)+1 \;=\; h_2(u)+2 \;\geq\; h_1(u)+2 \qquad (q.e.d.)$$

補題 6.6　非飽和プッシュは高々$O(mn^2)$ 回しか行われない．

証明　活性点全体の高さの総和を $\Phi(f) := \sum_{v:\text{活性点}} h(v)$ (≥ 0) とする．この関数の挙動を追うことで非飽和プッシュの回数を見積もる．Φ の値が変化するのは，(1) リラベル，(2) 飽和プッシュ，(3) 非飽和プッシュのいずれかが行われるときのみである．

まず (1) は補題 6.4 より高々$2n^2$，(2) は補題 6.3, 6.5 より高々$4mn^2$ だけ Φ の値を増加させる．一方 (3) の後には必ず枝の始点 u が活性点でなくなり，終点 v が活性点になったとしても $h(u) = h(v)+1$ より Φ の値は少なくとも 1 だけ減少することになる．増分が高々$O(mn^2)$ なので，非飽和プッシュの回数も高々$O(mn^2)$ 回．$(q.e.d.)$

1 回の繰り返しは，残余ネットワークをいちいち構成し直さなくても (高々1 本追加 1 本削除で) 良いので，活性点を選び，有資格な枝を選ぶ作業と合わせても $O(1)$ 時間で実現可能である．ただしそのためには，活性点のリストと，各点に関して有資格な枝のリストを持っておく必要がある．

これらの更新作業はリラベルの際に新たな有資格の枝を探すのに $O(n)$ 時間，プッシュで活性点または有資格な枝が消える際に $O(1)$ 時間かかる．よって全体で $O(n^3 + mn^2)$ 時間でできる．繰り返し自体が $O(1)$ 時間で，その回数は補題 6.4〜6.6 より高々$O(mn^2)$ 回なので，全体の計算量も $O(n^3 + mn^2)$ と言える．

しかし mn^2 の項があるので，これではディニッツ法と同じである．そこで非飽和プッシュに関してもう少し厳しい評価を与えることを考える．

補題 6.7　常に活性点の中で最も高い (ラベル値が大きい) ものを選べば，非飽和プッシュは高々$O(n^3)$ 回しか行われない．

証明　非飽和プッシュを行うと，活性点が 1 つ活性点でなくなる．そこで，$h^* := \max\{h(v) | v : \text{活性点}\}$ の値の変化を考える．増加するのはリラベルの際で，これは補題 6.4 より高々$O(n^2)$ 回である．

減少する可能性があるのは非飽和プッシュ(または等号成立の飽和プッシュ) の際で，リラベルとリラベルの間に同じ点に関して非飽和プッシュは行われない (行われるとすれば常に最も高い活性点を選んでいることに矛盾する) ので，高々$O(n)$ 回の非飽和プッシュしかない．よって全体では高々$O(n^3)$ 回．$(q.e.d.)$

補題 6.7 により全体の計算量を $O(n^3)$ にできそうである．ただし，データ構造に関して少し工夫しないと，常に最高の活性点を効率良く選ぶことは難しい．各高さ毎に属する点のリストを持ち，$\max\{h(v) | v \in V \setminus \{s, t\},\ h(v) \leq n\}$ を保持しておけば良いのだが，詳細は割愛する．

4　おまけ

プリフロー	preflow	ゴールドバーグ	Goldberg
活性点	active vertex	タージャン	Tarjan
有資格な	eligible	飽和	saturated
プッシュ	push	リスト	list
リラベル	relabel	データ構造	data structure

執筆日：2012 年 4 月 3 日
執筆者：com_math_arith