数理解析研究所講究録
第1691巻 2010年 127-135

嘘を含む比較による最小値最大値発見アルゴリズム

Michael Hoffmann* Jiří Matoušek† 岡本 吉央‡ Philipp Zumstein§

概 要

Pohl は 1972 年に要素数 n の全順序集合における最小値と最大値を同時に発見するためには $\lceil 3n/2 \rceil - 2$ 回の比較が十分であり，また，最悪の場合には必要であることを証明した．ただし，全順序集合は一対比較を行うオラクルとして与えられる．最近，この問題は Rényi–Ulam の嘘つきゲームの文脈でも研究されるようになった．そこでは，オラクルが高々 k 回正しくない返答を与えることができる．Aigner は k が大きいときに $(k + \mathrm{O}(\sqrt{k}))n$ 回の比較が十分であることを証明した．本研究ではそれに対する改善として，ある定数 C に対して高々 $(k+1+C)n + \mathrm{O}(k^3)$ 回の比較を行うアルゴリズムを与える．知られている下界はある定数 D に対して $(k+1+c_k)n - D$ という形をしていて，$c_0 = 0.5$，$c_1 = \frac{23}{32} \approx 0.605$ であり，$k \to \infty$ に対して $c_k = \Omega(2^{-5k/4})$ である．

1 はじめに

未知の線形順序 $\leq$ を有する要素数 n の集合 X を考える．この順序に関する情報は与えられた 2 要素 $x, y \in X$ に対して $x < y$ であるか $x > y$ であるかを答えるオラクルを通して得られるものとする．オラクルへの質問を比較と呼ぶ．集合 X の最小要素を発見するためには $n-1$ 回の比較を用いれば十分であることが簡単に分かる．最小要素を発見するためには最悪の場合に $n-2$ 回の比較では十分ではなく，その意味でこれは最適である．最大要素についても同様である．

しかし，最小要素と最大要素を**同時**に発見したいとき，最小と最大を別々に発見するよりも**かなりよく**できるというのは，計算の理論におけるちょっとした**驚き**の 1 つである．Pohl [8] は $\lceil 3n/2 \rceil - 2$ がこの問題に対する最適な比較回数であることを証明した $(n \geq 2)$．そのアルゴリズムでは，はじめに X 全体を 2 つずつの対にして，その各対を比較する．その「負け組」の中に最小要素があり，「勝ち組」の中に最大要素があるので，それらを見つければよい．

ここでは，オラクルが完全には信用できない場合において最小と最大の両方を発見する問題を考える．すなわち，オラクルは正しくない答えを与えることもあるが，その回数は計算全体において高々 k に限られている (k は与えられたパラメータ)．

このモデルを **k 個の嘘に対抗する計算**と呼ぶことにする．強調しておきたいことは，オラクルに対して同じ問合せを複数回繰り返してもよく，間違った答えは 1 つずつ嘘であると数えられることである．これは最もふさわしい定義であると思う．というのは，同じ問合せを繰り返すことができなかったり，ある特定の問合せに対してオラクルが常に間違った答えを与えることができると，最小要素さえ決定することができないからである．

*スイス連邦工科大学チューリヒ校 理論情報科学科
†カレル大学 応用数学部および理論情報科学科／スイス連邦工科大学チューリヒ校 理論情報科学科
‡東京工業大学 大学院情報理工学研究科
§スイス連邦工科大学チューリヒ校 理論情報科学科

よって，例えば，1つの問合せを $2k+1$ 回繰り返すと，多数決によって正しい答えが必ず得られる．すなわち，嘘をつかないオラクルを用いる任意のアルゴリズムを各比較の $2k+1$ 回の繰返しによって模倣できる．しかし，ここで考える問題に対してこれは効率的な方法ではない．

最小と最大を k 個の嘘に対抗して発見する問題を Aigner [1] は研究し，$(k+\mathrm{O}(\sqrt{k}))n$ 回の比較が常に十分であることを証明した[1]．本研究はこれを次のように改善する．

定理 1.1. 要素数 n の集合における最小要素と最大要素の両方を発見する問題に対して，k 個の嘘に対抗するアルゴリズムで比較回数が高々 $(k+1+C)n+\mathrm{O}(k^3)$ のものが存在する．ただし，C は定数である．

下で見る証明からは定数 C として割と小さい (k が十分大きければ例えば 10 以下の) 数になるが，それを最適化しようとはしていない．

下界. 最小と最大の両方を見つける問題に対して k 個の嘘に対抗するアルゴリズムが行う比較回数の下界として知られている中で一番よいものは $(k+1+c_k)n-D$ という形をしている．ここで，D は小さな定数であり，c_k は次の通りである．

- $c_0=0.5$ であり，これは最善である．これは上で述べた嘘をつかないオラクルに対する Pohl [8] の結果である．
- $c_1=\frac{23}{32}\approx 0.605$ であり，これもまた最善である．これは Gerbner, Pálvölgyi, Patkós, and Wiener による最近の結果 [5] であり，彼らは $k=1$ に対する最適比較回数を (小さな定数差を除いて) 定めた．すなわち，最適比較回数は $\lceil\frac{87}{32}n\rceil-3$ と $\lceil\frac{87}{32}n\rceil+12$ の間にある．これは Aigner [1] の予想を解決している．
- すべての k に対して $c_k=\Omega(2^{-5k/4})$．これを証明したのは Aigner [1] である．

最適な定数 $c_1=\frac{23}{32}$ を見ると，$k>1$ のときに正確な値を見つけることは難しそうである．

関連研究. 最小要素のみを k 個の嘘に対抗して定める問題は Ravikumar, Ganesan, and Lakshmanan [9] が解決し，$(k+1)n-1$ が最適比較回数であることを示した．

この論文で考えている問題は**嘘に対抗する探索問題**という領域に属し，より広い文脈では「誤りが存在する中での計算」の例である．この分野は豊かな歴史と美しい結果を有する．1960 年代に提起され未だに解かれていない **Rényi–Ulam の嘘つきゲーム**では，1 と n の間にある未知の整数 x を定める問題である．オラクルには x と他の数字の大小を尋ねることができ，オラクルは高々 k 回間違った答えを与えることができる．更なる情報については Pelc [7] や Deppe [2] によるサーベイ論文を見てもらいたい．

2 単純なアルゴリズム

定理 1.1 を証明する前に，主要なアイディアを説明するため単純なアルゴリズムから始める．得られる上界は定理 1.1 よりも弱い．そのために，詳細を指定しない一般的アルゴリズムを述べる．この節の単純なアルゴリズムと次節の改善アルゴリズムはどちらも一般的アルゴリズムの具体化である．

[1] 本研究では，O(.) と Ω(.) が隠すのは n と k の両方に独立な定数のみである．

一般的アルゴリズム

1. 適当な整数パラメータ $s = s(k)$ に対して，考えている要素数 n の集合 X を任意に要素数 s のグループ n/s 個 $X_1, \ldots, X_{n/s}$ に分割する．
2. 各グループ X_i において最小要素 m_i と最大要素 M_i を発見する．一般的アルゴリズムではこれを行うための方法を指定しない．
3. 集合 $\{m_1, \ldots, m_{n/s}\}$ の最小要素を k 個の嘘に対抗して発見し，独立に，集合 $\{M_1, M_2, \ldots, M_{n/s}\}$ の最大要素を k 個の嘘に対抗して発見する．

(整数 n が s で割り切れないときは，1つだけ要素数が s より小さいグループを作り，別に扱う．本質的でない詳細は省略する．)

第 2 段階が正しく実装されることを仮定すれば，一般的アルゴリズムの正当性は明らかである．

最終的に単純なアルゴリズムと改善アルゴリズムのどちらでも $s := k$ と置く．しかし，$s := k$ という選択は偶発的だったので，s をパラメータとして残しておく．

単純なアルゴリズムでは第 2 段階を次のように具体化する．

単純なアルゴリズムにおける第 2 段階

2.1. (ソート．) 集合 X_i の要素を漸近的に最適なソート・アルゴリズム (例えばマージソート) によって $O(s \log s)$ 回の比較によって (嘘を無視して) ソートする．これによって，ソート・アルゴリズムで用いた問合せに対する答えがすべて**正しければ**，X_i の要素を $x_1 < x_2 < \ldots < x_s$ と並べることができる．間違った答えがあったとしても気にせず，ソート・アルゴリズムが異常停止をすることなくある並び順を出力することを仮定する．

2.2. (最小要素と最大要素の確認．) 各 $j = 2, 3, \ldots, s$ に対して，2 要素 x_{j-1}, x_j の大小をオラクルに $k+1$ 回問い合わせる．その中の 1 回でも $x_{j-1} > x_j$ であると答えたら，**やり直す**．すなわち，第 2.1 段階の最初へ戻り，グループ X_i に対する計算をはじめからやり直す．そのようなことがなければ，次の段階へ進む (すべての答えは $x_{j-1} < x_j$ であったことになる)．

2.3. $m_i := x_1$ および $M_i := x_s$ とする．

補題 2.1 (正当性)**.** 単純なアルゴリズムは k 個の嘘に対抗して最小要素と最大要素を正しく計算する．

証明. 上のアルゴリズムでグループ X_i を処理し，第 2.3 段階までたどり着いたら，$m_i = x_1$ は最小要素でなくてはならない．実際，他のどの要素 x_j $(j \geq 2)$ に対してもオラクルは $k+1$ 回 $x_j > x_{j-1}$ と答えているので，x_j は最小要素にならない．対称的に，M_i も最大要素でなくてはならず，したがって，アルゴリズムは常に正しい． □

実は，第 2.3 段階で $x_1, \ldots, x_s$ が X_i を大きさ順に並べたものになることが分かる．しかし，次節の改善アルゴリズムでは状況がもっと繊細である．次の補題が示すように，単純なアルゴリズムで既に Aigner の $(k + O(\sqrt{k}))n$ という上界が改善できる．

補題 2.2 (計算量)**.** 要素数 n の集合に対して，$s = k$ とした単純なアルゴリズムの比較回数は $(k + O(\log k))n + O(k^3)$ である．

証明. 第 2 段階でグループ X_i を処理するために，やり直しがなければ $O(s \log s) + (k+1)(s-1) = k^2 + O(k \log k)$ 回の比較で十分である．しかし，オラクルが嘘をつくときにしかやり直しを行わず，嘘の数は高々 k であるので，すべてのグループを通してやり直しの総数は高々 k である．よって，やり直し全体での比較回数は高々 $k(k^2 + O(k \log k)) = O(k^3)$ である．したがって，第 2 段階における比較総数は $\frac{n}{s}(k^2 + O(k \log k)) + O(k^3) = (k + O(\log k))n + O(k^3)$ である．

始めの節で述べたように，要素数 n の集合における最小要素 (または最大要素) は k 個の嘘に対抗して $(k+1)n - 1$ 回の比較で発見できるので，第 3 段階における比較回数は $2(k+1)(n/s) = O(n)$ 以下である．(最小要素発見のために最適アルゴリズムが必要であるというわけではなく，比較回数 $O((k+1)n)$ の任意のアルゴリズムでよい．) これで比較総数に対する所望の上界が得られる． □

3 アルゴリズムの改善：定理 1.1 の証明

集合 X_i の最小要素が本当に x_1 であることを確認するために，x_j (ただし $j \neq 1$) に対してオラクルは x_j が他の要素よりも大きいことを $k+1$ 回答えるようにしたい．(単純なアルゴリズムでは，これら $k+1$ 回の比較はすべて x_{j-1} と行われたが，x_j より小さな要素であればどれでもよい．) これだけで 1 つのグループにおいて $(k+1)(s-1)$ 回の比較を必要として，すなわち，全体で $(k+1)(n - n/s)$ 回の比較を必要とする．これは既に定理 1.1 で標的としている上界に近い．一般的アルゴリズムの第 3 段階を効率的に行うためには s のオーダーが k 以上である必要があることにも注意する．

同様に，最大要素の確認のために全体で $(k+1)(n - n/s)$ 回の比較を必要とする．よって，その中の $O(n)$ 以外の比較はすべて**両方**を発見するために同時に用いられなくてはならない．

この意味で，単純なアルゴリズムの第 2.1 段階でグループをソートするために使われた比較には無駄がある．無駄をなくすため，その中のほとんどを第 2.2 段階で最小要素と最大要素を確認するためにも用いる．例えば，x_{17} がそれよりも大きな要素と既に 23 回比較をされている場合，確認の段階では x_{17} とそれよりも大きな要素を比較すべき回数は $k + 1 - 23$ になる．

ここですぐ問題になることは，ソート・アルゴリズムが x_{17} と比較する x_{17} よりも大きな要素の数が $b > k+1$ であるとき，$b - (k+1)$ 回の比較は無駄になってしまうことである．しかし，$s = k$ と置き，各要素が他の要素と比較される回数は高々 $k-1$ なので (つまり，ソーティング・アルゴリズムは重複した比較を行わないので)，これは問題にならない．

別の問題はもっと繊細である．それを説明するために，ソート・アルゴリズムが行った比較全体をグラフの辺として表現しよう．頂点集合は $1, 2, \ldots, s$ であり，これは X_i の要素全体をソート順に並べた $x_1, \ldots, x_s$ を表現している．辺集合はソート・アルゴリズムが行った 2 要素に対応する頂点間に引かれる．図 1 (左) を見てもらいたい．

確認の段階では，各 x_j (ただし $j \neq 1$) がそれより小さな要素と $k+1$ 回以上比較され，各 x_j (ただし $j \neq s$) がそれより大きな要素と $k+1$ 回以上比較されるように，追加の比較を行う必要が

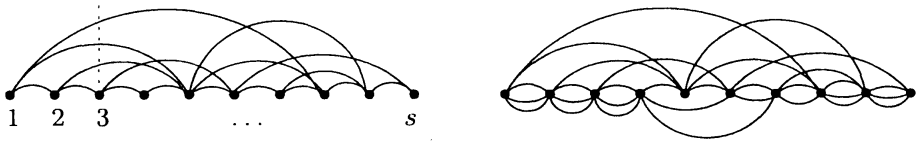


図 1: 比較の再利用.

ある．これは，グラフに適切な辺を加えることに対応する．図 1 (右) を見てもらいたい (そこでは $k=2$ としていて，追加した辺は下側に描かれている).

図が示しているように，ときにはある要素をそれより大きな要素かそれより小さな要素と $k+1$ 回**より多く**比較しなくてはならなくなる (よって，そのような比較は「半分無駄」になる)．例えば，どのように新しい辺を追加しても，x_1, x_2, x_3 という 3 要素が関わる比較の中の少なくとも 3 つは半分無駄になる．実際，x_2 と x_3 に対して合わせて 6 個の比較が左側 (すなわち小さな要素) に必要である．これらの比較には x_1 か x_2 が関わらなければならないが，x_1 と x_2 が右側に望む比較は 6 回だけであり，その中の 3 個は既に x_3 よりも大きな要素で費やされている (これらは図 1 (左) の縦点線と交わる辺として表現されている).

次の補題は，この種の議論からしか無駄な比較が生じないことを示している．頂点集合が横一列に並んでいる多重グラフ H (頂点集合は $\{1,2,\ldots,s\}$) に対して，H の**厚み** $t(H)$ を $\max\{t(j): j=2,3,\ldots,s-1\}$ で定義する．ただし，$t(j) := |\{\{a,b\} \in E(H) : a < j < b\}|$ は頂点 j の「上」を通る辺の数であるとする.

補題 3.1. 頂点集合を $\{1,2,\ldots,s\}$ とする (ループは含まない) 無向多重グラフを H とし，各頂点 $j=1,2,\ldots,s$ に対して

$$d_H^{\mathrm{left}}(j) := |\{\{i,j\} \in E(H) : i < j\}| \le k+1,$$
$$d_H^{\mathrm{right}}(j) := |\{\{i,j\} \in E(H) : i > j\}| \le k+1$$

とする．このとき，H に辺を追加して次の条件を満たす無向多重グラフ $\overline{H}$ を構成できる.

(i) 任意の頂点 $j \neq 1$ の右側近傍の頂点数は $k+1$ 以上であり，任意の頂点 $j \neq s$ の左側近傍の頂点数は $k+1$ 以上である.

(ii) $\overline{H}$ の総辺数は $(k+1)(s-1)+t(H)$ 以下である.

証明にはネットワーク流の論法を用いるが，本節の最後にまわす.

比較に基づくソート・アルゴリズム $\mathcal{A}$ に対して，その**厚み** $t_{\mathcal{A}}(s)$ を自然に定義する．すなわち，要素数 s のすべての入力列に対して $\mathcal{A}$ を適用したときに作られるグラフ H の厚み $t(H)$ の最大値として定義する．補題 3.1 が示すように，ソートに使われるが確認に使われない比較の回数はソート・アルゴリズムの厚みで上から抑えられる.

補題 3.2. (決定性の) ソート・アルゴリズム $\mathcal{A}$ で厚みが $t_{\mathcal{A}}(s) = \mathrm{O}(s)$ のものが存在する.

証明. アルゴリズムはクイックソートであるが，厚みを制御するため各再帰段階において 2 つの子問題の大きさが等しくなるように分割したい.

よって，与えられた入力の中央要素 (メディアン) を計算することから始める．そのための比較回数は $\mathrm{O}(s)$ である (例えば，Knuth [6] を参照のこと．知られている中で最善の決定性アルゴリズムは Dor and Zwick [3] によるもので，その比較回数は $2.95s + \mathrm{o}(s)$ 以下である)．そして，計

算した中央要素よりも小さい要素全体のグループと大きい要素全体のグループに残りを分割する．そして，各グループを再帰的に処理することでソートが完了する．

このアルゴリズムの厚みは $t_{\mathcal{A}}(s) \leq \mathrm{O}(s) + t_{\mathcal{A}}(\lfloor s/2 \rfloor))$ という再帰式に従い，よって $\mathrm{O}(s)$ で上から抑えられる． □

補題 3.2 にあるアルゴリズム $\mathcal{A}$ をオラクルが間違った答えを与える場合に用いるので，中央要素の計算が正しく行われる保証がなく，2 つのグループの大きさが等しくなるとは言えなくなる (そして，補題が保証する厚みの上界が成り立たなくなる)．しかし，2 つのグループの大きさが等しいかどうかは比較を行うことなく確認することができるので，等しくなければ (単純なアルゴリズムと同様に) 計算をやり直せばよい．

それでは，一般的アルゴリズムの第 2 段階を具体化することで改善アルゴリズムを記述する．

改善アルゴリズムにおける第 2 段階

2.1′. (ソート．) 集合 X_i を補題 3.2 の厚み $\mathrm{O}(s)$ のアルゴリズムでソートする．(上で議論したような) 不整合が発見されたら，グループ X_i に対する計算をはじめからやり直す．

2.2′. (最小要素と最大要素の確認．) アルゴリズム $\mathcal{A}$ が行った比較に対応するグラフ H を作成し，補題 3.1 に従って多重グラフ $\overline{H}$ へ拡大する．追加された辺に対応する比較を行う．不整合が発見されたら，やり直す．すなわち，第 2.1′ 段階の最初へ戻り，グループ X_i に対する計算をはじめからやり直す．そのようなことがなければ，次の段階へ進む．

2.3′. $m_i := x_1$ および $M_i := x_s$ とする．

定理 1.1 の証明. 改善アルゴリズムの正当性は単純なアルゴリズムに対する議論と同様に従う．第 2.2′ 段階において，オラクルは各要素 x_j (ただし $j \neq 1$) が他の要素よりも大きいと $k+1$ 回答えるので，x_j は最小要素にはなれない．対称な論法は最大要素にも適用できる．

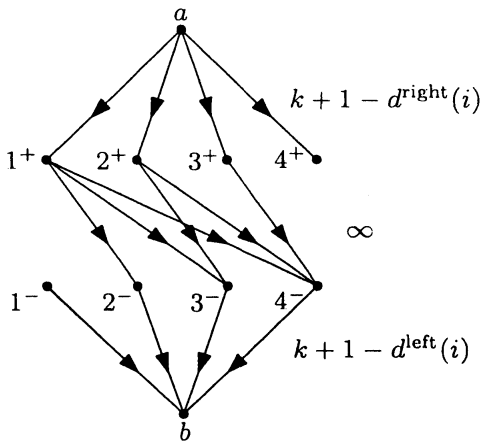
後は比較回数を抑えればよい．上の議論から，比較回数は高々 $((k+1)(s-1) + t_{\mathcal{A}}(s))(\frac{n}{s} + k) + 2(k+1)\frac{n}{s}$ であり，補題 3.2 より $t_{\mathcal{A}}(s) = \mathrm{O}(s)$ である．ここで $s = k$ と置くと，ある定数 C を用いて比較回数が $(k+1+C)n + \mathrm{O}(k^3)$ 以下になることが分かる． □

補題 3.1 の証明. 頂点集合を $\{1, 2, \ldots, s\}$ とする多重グラフ H' において，各頂点の左側次数と右側次数がともに $k+1$ 以下であるとする．このとき，H' の**不足度** $\Delta(H')$ を

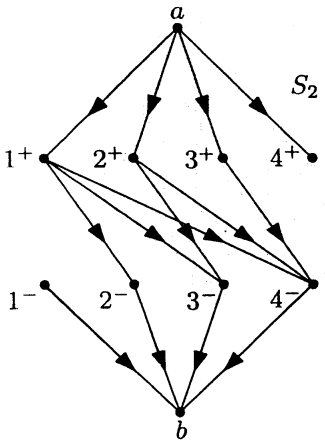
$$\Delta(H') := \sum_{j=1}^{s-1} (k+1 - d^{\mathrm{right}}_{H'}(j)) + \sum_{j=2}^{s} (k+1 - d^{\mathrm{left}}_{H'}(j))$$

で定義する．考えている多重グラフ H に対しては $\Delta(H) = 2(k+1)(s-1) - 2e(H)$ となる．ただし，$e(H)$ は H の辺数である．

ネットワーク流の論法を使って，H に $m^* := (k+1)(s-1) - e(H) - t(H)$ 個の辺を上手に追加することで，各頂点の左側次数と右側次数がともに $k+1$ 以下の多重グラフ H^* で，$\Delta(H^*) \leq 2t(H)$ を満たすものが構成できることを示す．補題の言明にあるようなグラフ $\overline{H}$ はそこから $\Delta(H^*)$ 個の辺を追加すれば得られる．例えば，H^* の各頂点 $j \geq 2$ に対して，それが $d^{\mathrm{left}}_{H^*}(j) < k+1$ を満



(a) グラフ G と辺容量.



(b) G におけるカット S_2.

図 2: 補題 3.1 の証明において構成される有向グラフ G.

たすときは，j と 1 を結ぶ辺を $k+1-d^{\mathrm{left}}_{H^*}(j)$ 個追加する．同様に，右側次数が不足している頂点と s を結ぶ辺も追加する．

後は，上で述べたような H^* を構成すればよい．そのために，補助有向グラフ G を次のように定義する．そのときに，各有向辺 e には整数容量 $c(e)$ も割り当てる．図 2(a) も参照．

グラフ G の頂点集合は各 $j \in \{1,2,\ldots,s\}$ に対する頂点 j^-，各 $j \in \{1,2,\ldots,s\}$ に対する頂点 j^+，そして，a，b という 2 つの特別な頂点から成る．グラフ G において，a から各 j^+ へ向かう有向辺が存在し，その容量は $k+1-d^{\mathrm{right}}_H(j)$ であるとする．同様に，G において，各 j^- から b へ向かう有向辺が存在し，その容量は $k+1-d^{\mathrm{left}}_H(j)$ であるとする．さらに，各 i,j (ただし，$1 \le i < j \le s$) に対して，G において有向辺 (i^+,j^-) が存在し，その容量は ∞ (すなわち，十分大きな数) であるとする．これらの他に G の辺は存在しない．

グラフ G に整数 a-b 流で，その値が m^* であるものが存在することを示す．最大流最小カット定理 [4] より，G の任意の a-b カットの容量が m^* 以上であることを示せば十分である．

最小 a-b カット $S \subseteq V(G)$ に対して，i を $i^+ \in S$ となる最小のものとする．最小カットは容量無限大の辺を用いることができないので，任意の $j > i$ に対して $j^- \in S$ が成り立つ．

一般性を失わず，任意の $j > i$ に対して $j^+ \in S$ であり，任意の $j \le i$ に対して $j^- \notin S$ であることを仮定してよい (そうでないとしても，カットの容量は減少しない)．したがって，考える a-b カットの形は $i = 1,\ldots,s$ に対して

$$S_i := \{a\} \cup \{x^+ : x \ge i\} \cup \{x^- : x > i\}$$

となる．カット S_i の容量は

$$\sum_{j<i} c(a,j^+) + \sum_{j>i} c(j^-,b) \quad = \quad (s-1)(k+1) - \sum_{j<i} d^{\mathrm{right}}_H(j) - \sum_{j>i} d^{\mathrm{left}}_H(j)$$

に等しい．図 2(b) も参照のこと．

では，$\sum_{j<i} d^{\mathrm{right}}(j) + \sum_{j>i} d^{\mathrm{left}}(j)$ という量を見て，H の辺 $\{j,j'\}$ (ただし $j < j'$) がそれにどれだけ貢献するか調べよう．添え字 $j < i < j'$ に対して貢献は 2 であるが，他の辺は 1 だけ貢

献する．よって，カット S_i の容量は $(k+1)(s-1)-e(H)-t(i)$ であり，a-b カットの最小容量は示したかった量 $(k+1)(s-1)-e(H)-t(H)=m^*$ となる．

よって，上で述べたとおり，値が m^* となる整数流 f が存在する．そして H へ追加する辺を次のように選ぶ．グラフ G の有向辺 (i^+,j^-) に対して，辺 $\{i,j\}$ を $f(i^+,j^-)$ 個だけ H に加える．これによって多重グラフ H^* が得られる．追加した辺の数はフロー f の値 m^* であり，容量制約から H^* における頂点の左側次数と右側次数がともに $k+1$ 以下であることも分かる．さらに，H^* の不足度は $2t(H)$ 以下である． □

4 最後に

最初の節で述べた $k=0$ のときのアルゴリズムは本研究の一般的アルゴリズムの枠組に当てはめることができる．すなわち，$s=2$ と置き，第 2 段階において各グループの 2 要素を比較するのである．本研究のアルゴリズムの重要な特徴は，やり直しの影響が局所的な 1 つのグループにしか及ばないことである．

本論の手法によって定理 1.1 の上界を改善するためには，厚みが $o(s)$ のソート・アルゴリズムが必要となる．しかし，次の命題が示すようにそのようなアルゴリズムは存在しない．定理 1.1 を改善するためには異なるアイディアが必要となるだろう．

命題 4.1. 要素数 s の集合をソートする任意の (乱択) アルゴリズムの厚みの期待値は $\Omega(s)$ である．

証明. Yao の原理 [10] より，任意の決定性ソート・アルゴリズム $\mathcal{A}$ のランダム入力に対する厚みの期待値が $\Omega(s)$ となることを示せばよい．ここでは，X 上の未知の線形順序が $s!$ 個の可能性より一様に選ばれると仮定する．

各段階で，アルゴリズム $\mathcal{A}$ はある 2 要素 $x,y\in X$ を比較する．要素 $x\in X$ がある段階の開始時点で**純粋**であるとは，それが以前のどの比較にも現れないこととして，純粋でない要素は**不純**であるとする．比較が**新鮮**であるとは，そこに 1 つ以上純粋な要素が関わっていることとする．

記法の簡便さのため，s は 8 で割り切れるものとする．(ランダムな) 入力順序における最初の $s/2$ 個の要素からなる集合を $L\subset X$ として，残りを $R:=X\setminus L$ とする．事象 E_i を，i 番目の新鮮な比較が LR-**比較**であること，すなわち，それに関わる 2 つの要素 x,y の一方が L に属し，もう一方が R に属すような比較であることとする．このとき，任意の $i=1,2,\ldots,s/8$ に対して E_i の確率が $\frac{1}{3}$ であることを主張する．

そのために，i 番目の新鮮な比較の前までに $\mathcal{A}$ が行ったすべての比較の結果を (任意に) 固定する．それらは不純な要素全体の集合を定めるが，それら不純な要素が入力順序で占める場所も固定する．では，これらの選択がされたという**条件の下で** E_i の確率を考えよう．鍵となる観察は，入力順序において純粋な要素は (不純な要素で占められていない) 残りの場所の間で一様ランダムに分布しているということである．

L における純粋な要素の数を ℓ として，R における純粋な要素の数を r とすると，$s/4\le\ell,r\le s/2$ が成り立つ．

2 つの場合を分けて考える．まず，i 番目の新鮮な比較に現れる要素 x,y のちょうど 1 つが純粋であるとする．一般性を失わずに，x が不純であり，L に属するとする．このとき，E_i の確率は $r/(\ell+r)\ge\frac{1}{3}$ となる．

2 番目の場合として，x と y がともに純粋であるとする．このとき，E_i の確率は $2\ell r/((\ell+r)(\ell+r-1))$ であり，$s/4\le\ell,r\le s/2$ であることから，この確率は $\frac{4}{9}$ 以上である．

したがって，上で述べた E_i の条件付き確率は $\frac{1}{3}$ 以上であり，ゆえに，ランダム入力に対する E_i の確率は $\frac{1}{3}$ 以上となる．したがって，$\mathcal{A}$ が行う LR-比較の期待値は $\Omega(s)$ である．

L の最大要素，すなわち X の $s/2$ 番目の要素を a として，R の最小要素，すなわち X の $s/2+1$ 番目の要素を b とする．$\mathcal{A}$ が同一の比較を繰り返さないことを仮定してよいので，a と b が比較されることは高々1回しかない．その他の LR-比較は a か b (またはその両方) を間に含んでいる．よって，$\mathcal{A}$ の厚みの期待値は少なくとも LR-比較の期待回数であり，それは $\Omega(s)$ である． □

上の命題で必要としたことは，対応する順序付きグラフが単純であり，その最小次数が 1 以上である，ということのみであるので，注意する．

謝辞

この論文で扱った問題を紹介してくれた Döm Pálvölgyi に感謝する．本研究は 7th Gremo Workshop on Open Problems, Hof de Planis, Stels in Switzerland, July 6–10, 2009 で始まった．このワークショップの参加者が作ってくれた刺激的な環境にも感謝する．

参考文献

[1] M. Aigner. Finding the maximum and the minimum. Discrete Applied Mathematics **74** (1997) 1–12.

[2] C. Deppe. Coding with feedback and searching with lies. In *Entropy, Search, Complexity*, Bolyai Society Mathematical Studies 16 (2007), pages 27–70.

[3] D. Dor and U. Zwick. Selecting the median. SIAM Journal on Computing **28** (1999) 1722–1758.

[4] L. R. Ford Jr. and D. R. Fulkerson. Maximal flow through a network. Canadian Journal of Mathematics **8** (1956) 399–404.

[5] D. Gerbner, D. Pálvölgyi, B. Patkós, and G. Wiener. Finding the maximum and minimum elements with one lie. Discrete Applied Mathematics, to appear.

[6] D. E. Knuth. The Art of Computer Programming. Volume 3. Sorting and Searching. Addison-Wesley Publishing Co. (1973)

[7] A. Pelc. Searching games with errors—fifty years of coping with liars. Theoretical Computer Science **270** (2002) 71–109.

[8] I. Pohl. A sorting problem and its complexity. Communications of the ACM **15** (1972) 462–464.

[9] B. Ravikumar, K. Ganesan, and K. B. Lakshmanan. On selecting the largest element in spite of erroneous information. Proceedings of 4th STACS, Lecture Notes in Computer Science **247** (1987) 88–99.

[10] A. C.-C. Yao. Probabilistic computations: Towards a unified measure of complexity. Proceedings of 18th FOCS (1977) 222–227.