決定木複雑性における複数アドバーサリーの方法：有向アサイクリックグラフの場合

金山 寛奈

首都大学東京 理工学研究科 数理情報科学専攻

平成 27 年 1 月 9 日

概 要

有向グラフが有向 cycle を含まないとき，Directed Acyclic Graph(DAG) という．

隣接行列で表したグラフを入力とし，そのグラフが DAG であるかを判定するブール関数 f を考察する．ブール関数の決定木複雑性に関しては，deterministic complexity$D(f)$ と randomized complexity$R(f)$ の比較が重要である．$D(f)$ は乱数なしの指標の一種であり，$R(f)$ は決定性アルゴリズムの集合上の確率分布について，その確率分布が真理値割り当て全体に対して，どれだけコスト期待値を抑えることができるかという指標である．$R(f) < D(f)$ が成り立てば，乱数を利用することで計算コストの節約ができると言える．

本研究では以下の 3 つのことを示す．(1)(主結果) 二つの adversary を併用する adversary 論法を提案し，この手法が non-adaptive algorithm の場合に $D(f) = n^2 - n$ を証明する上で有効であること (2)non-adaptive，adaptive のうちどちらのタイプのアルゴリズムでも，頂点数が 2 のとき $R(f) = 2$，頂点数が 3 以上のとき $R(f) < n^2 - n$ が成り立つこと (3)adaptive algorithm を考えた場合，頂点数が 3 のとき，$D(f) = 3^2 - 3 = 6$ が成り立つこと．

Aanderaa-Karp-Rosenberg conjecture に関連した研究において，Best,Boas, Lenstra(1974) は多項式と数え上げを用いて $D(f) = n^2 - n$ であることを示した．これに対し，本研究は二つの adversary で挟み撃ちにする点に特色がある．この手法は non-adaptive algorithm に限定しているが，数え上げの議論を回避できるという特徴があり，研究する価値がある．

A directed graph without a cycle is called a Directed Acyclic Graph (DAG).

We consider a Boolean function f which receives a graph represented as an adjacency matrix and then determines if the graph is a DAG. On decision

tree complexity of a Boolean function, it is important to compare deterministic complexity $D(f)$ and randomized complexity $R(f)$. $D(f)$ and $R(f)$ are complexity measures; The former is on deterministic algorithms. The latter is on probability distributions on deterministic algorithms; $R(f)$ denotes the minimum expected cost against the worst inputs. If $R(f) < D(f)$, we can save the cost by using randomization.

In this study, we are going to show the following three results. (1)(Main result) We propose a new type of adversary argument consisting of two adversaries, and demonstrate that this method is effective to prove $D(f) = n^2 - n$ in the case of non-adaptive algorithms; (2) For non-adaptive or adaptive algorithms, $R(f) = 2$ if the number of vertices is two, and $R(f) < n^2 - n$ otherwise; (3) For adaptive algorithms, if the number of vertices is three, $D(f) = 3^2 - 3 = 6$.

In a preceding work on Aanderaa-Karp-Rosenberg conjecture, Best, Boas and Lenstra (1974) showed that $D(f) = n^2 - n$ by using polynomials and a counting argument. In contrast, our method features two different adversaries against algorithms; At least one of the two forces the given algorithm to probe all the components. At the moment, adaptive algorithms go beyond the scope of our method. However, our method has an advantage of avoiding counting arguments, thus is worth continuing the research.

目 次

1　序

1.1　背景

有向グラフが有向 cycle を含まないとき，Directed Acyclic Graph（DAG）という．

DAG は，ベイジアンネットワークに応用がある．ベイジアンネットワークとは，有限個の確率変数の集合に対して，それらの条件付き独立性を有向グラフの d-分離性で表現したものである．このときに使われるグラフが DAG である．ベイジアンネットワークを応用し，スパムメールフィルターが実用化されている（鈴木譲 [8]）．

さて，与えられた有向グラフ G が DAG であるか判定するための計算コストはどれだけ必要なのだろうか？より具体的に言い換えると，次の関数 f の計算コストはどれだけ必要なのだろうか？

f:頂点数 n の G を入力として受けとり，G が DAG のとき 1，
そうでないとき 0 を返す（G は連結なものに限る）．

ここで，グラフの表現には隣接行列を用いる．隣接行列の各成分をブール変数とみなす．1 が真，0 が偽である．すると，f はブール関数となる．ブール関数の計算コストとして活発に研究されているものに決定木複雑性という概念がある．我々の設定に即して説明しよう．

本研究では，入力したグラフが DAG であるか判定する計算を次のように考える．関数 f は，隣接行列で表現したグラフを入力とする．決定性アルゴリズムは，各辺へ問い合わせを行い，1 あるいは 0 の値を応答として受け取る．1 はそこに辺があることを表し，0 は辺がないことを表す．このような問い合わせと応答のやりとりを何回か行い，そのグラフが DAG であると判定できた時点で 1，そうでないとき 0 を出力する．そのとき，決定性アルゴリズム A に対するグラフの計算コストを，「A に対して，DAG かどうかが判明するまでにかかった辺への問い合わせ回数」と定義する．決定性アルゴリズムは，2 種類のものについて考える．探索の履歴によって次に探索する辺を変えないものを non-adaptive algorithm といい，そうでないものを adaptive algorithm という．

ブール関数の決定木複雑性に関しては，乱数なしの指標の一種である deterministic complexity $D(f)$ と randomized complexity $R(f)$ の比較が重要である．deterministic complexity とは，最も効率のよい決定性アルゴリズムに最悪な入力を入れて計算した時にかかるコストを表す．randomized complexity とは，決定性アルゴリズムの集

合上の確率分布について，その確率分布が真理値割り当て全体に対して，どれだけコスト期待値を抑えることができるかという指標である．deterministic complexity は，randomized complexity で考える確率分布の特殊な場合なので，明らかに $R(f) \leq D(f)$ である．$R(f) < D(f)$ が成り立てば，乱数を利用することで計算コストの節約ができるといえる（Arora,Barak[1]）．

ここで先行研究が明らかにした計算複雑性の下界を紹介し，我々の結果がどのように改良になっているかを説明しよう．

Holt,Reingold [5] では，頂点数 n のグラフに cycle が含まれるか，すなわち DAG であるか判断するためには，最悪の場合において少なくとも $n(n-1)/2$ 回クエリしなければならないことが示されている．すなわち，どんなアルゴリズムに対しても，$n(n-1)/2$ 回はかかってしまう入力が存在するということである．これは，$D(f) \geq n(n-1)/2$ を表している．

これに対し，Aanderaa-Karp-Rosenberg conjecture に関連した研究で，Best,Boas, Lenstra[2] は $D(f) = n^2 - n$ であると示した．Holt,Reingold の結果に対し，1/2 を外したのである．必ず $D(f) \leq n^2 - n$ となるから，これは最適な結果である．なお，Aanderaa-Karp-Rosenberg conjecture は決定木複雑性についての有名な予想である．後の 1.3 項でその概要を述べる．

1.2 本研究の取り組み

アルゴリズムと決定木複雑性の組み合わせとしては，$2 \times 2 = 4$ 通りある．本研究では，以下のことを明らかにする．

	non-adaptive	adaptive
$D(f)$	$= n^2 - n$	$= n^2 - n \quad (n = 3)$
$R(f)$	$= 2 \quad (n = 2)$ $< n^2 - n \quad (n \geq 3)$	$= 2 \quad (n = 2)$ $< n^2 - n \quad (n \geq 3)$

non-adaptive algorithm の場合に $D(f) = n^2 - n$ を示すため用いた手法が本研究の主結果である．$D(f) = n^2 - n$ であることを示すには，どんなアルゴリズムに対しても，そのアルゴリズムに応じてうまく入力を選べば $n^2 - n$ 手かかることをいえばよい．我々は adversary argument を用いてこれを示す．例えば n 変数の OR 関数

の場合，$n-1$ 回目のクエリまで 0 と返し，n 回目に 1 と返すアドバーサリーによって，deterministic complexity が n であることがわかる（Arora,Barak[1]）．Best et al. は多項式と数え上げを用いているのに対し，本研究の特色は，二つの adversary で挟み撃ちにすることにある．一つ目の adversary でコストを最大化できないアルゴリズムは，二つ目の adversary を適用することでコストを最大化できる．

現時点で，この手法によって Best et al.[2] と完全に同じ結果を出すには至っていない．というのは，本研究ではアルゴリズムが non-adaptive であるという制約を付けているからだ．しかし，二つの adversary を併用する手法は以前からあまり知られていない手法であり，応用範囲を広げていくことが期待できる．数え上げの議論を回避できるという特徴もあり，研究する価値がある．

また，乱択アルゴリズムによって計算した際，頂点数が 3 以上の場合は $R(f) < n^2 - n$ となることを示す．先に述べた $D(f)$ の結果と合わせて，$R(f) < D(f)$ が成り立つことがいえる．すなわち，頂点数が 3 以上の場合，乱数を利用することで計算コストの節約ができることがわかる．ただし，$R(f)$ のオーダーが n^2 未満になることを示せたわけではないので，オーダーの意味で計算コストが大きく節約できるということではない (Karp は $R(f) = \Theta(n^2)$ であると予想している (Saks,Wigderson[7]))．これら $R(f)$ についての結果は指導教員である鈴木登志雄氏によって示された．

adaptive algorithm を考えた場合，$R(f)$ については non-adaptive algorithm の結果から容易に同様のことが示せる．$D(f)$ については頂点数が 3 のとき，non-adaptive algorithm と同様のことが示せる．頂点数が 4 以上の場合に我々の方法で証明できるかは，明らかにできていない．

本論文の構成は以下の通りである．

第 2 節では，有向グラフや決定木計算量に関する定義と解説を述べる．第 3 節から第 5 節において，本研究の結果を述べる．第 6 節において，本研究から考えられる今後の課題について述べる．

1.3 Aanderaa-Karp-Rosenberg conjecture

グラフの計算量について，Aanderaa-Karp-Rosenberg(A.K.R) conjecture がある．「すべての nontrivial(恒真でも恒偽でもない) で monotone(辺を加えても成り立つ) な graph property(頂点のラベル付けに依存せず成り立つ性質) は evasive である」という予想である (Khan,Saks,Sturtevant[6])．evasive とは，隣接行列の全成分を調べな

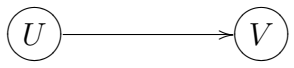
いと，property を満たすか判断できないということである．グラフに loop は含まれないとしているので，$n \times n$ 行列の場合，deterministic complexity が $n^2 - n$ であるだろうということを示している．本研究で扱う DAG 判定も，この conjecture の条件を満たす property である．

2 定義

2.1 有向グラフと経路

定義 2.1 (鈴木譲 [8]) グラフとは，いくつかの頂点を辺で結んだものである．グラフには大きく分けて 2 種類あり，辺に向きがついていないものを**無向グラフ**，向きのついたものを**有向グラフ**という．本稿では有向グラフのみを扱うため，ここでは無向グラフの定義は省く．

$\mathbf{U}$ を有限集合，$\vec{\mathbf{E}} \subseteq \vec{\epsilon} = \{(U, V) \mid U, V \in \mathbf{U}, U \neq V\}$ とする．$\vec{G} = (\mathbf{U}, \vec{\mathbf{E}})$ を**有向グラフ**とよぶ．$\mathbf{U}$ を $\vec{G}$ の**頂点集合**，その要素を $\vec{G}$ の**頂点**とよび，$\vec{\mathbf{E}}$ を $\vec{G}$ の**辺集合**，その要素を $\vec{G}$ の（**有向**）**辺**とよぶ．$U, V \in \mathbf{U}$ について、$(U, V) \in \vec{\mathbf{E}}$ を U から V に向かう矢印で表す．$(U, V) \in \vec{\mathbf{E}}$ または $(V, U) \in \vec{\mathbf{E}}$ のとき，U と V は**隣接**しているという．



$U_0, \cdots, U_k \in \mathbf{U}$ とする．$i = 1, \cdots, k$ のそれぞれで $(U_{i-1}, U_i) \in \vec{\mathbf{E}}$ または $(U_i, U_{i-1}) \in \vec{\mathbf{E}}$ を満たすとき，$\{U_i\}_{i=0}^{k}$ を U_0, U_K を結ぶ，長さ k の**無向経路**とよぶ（簡単に経路とよぶこともある）．

$$U_0 = U_k, U_j \neq U_i, j = 0, \cdots, i-1; i = 1, \cdots, k-1$$

となる無向経路 $\{U_i\}_{i=0}^{k}$ を長さ $k(\geq 3)$ の**無向巡回経路**とよぶ．

他方，$i = 1, \cdots, k$ のそれぞれで $(U_{i-1}, U_i) \in \vec{\mathbf{E}}$ を満たすとき，$\{U_i\}_{i=0}^{k}$ を U_0 から U_k への長さ k の**有向経路**とよぶ．

$$U_0 = U_k, U_j \neq U_i, j = 0, \cdots, i-1; i = 1, \cdots, k-1$$

となる有向経路 $\{U_i\}_{i=0}^{k}$ を長さ $k(\geq 2)$ の**有向巡回経路**とよぶ．$\vec{G} = (\mathbf{U}, \vec{\mathbf{E}})$ が有向巡回経路を含まないとき，**有向非巡回グラフ** (Directed Acyclic Graph) とよぶ．簡単のため有向巡回経路を **cycle**，有向非巡回グラフを **DAG** とよぶ．

$\vec{G}$ の任意の異なる 2 つの頂点 X，Y に無向経路が存在するとき，$\vec{G}$ は**連結されている**という．

本研究では，グラフ上の任意の頂点 $U_i, U_j \in \mathbf{U}, (i \neq j)$ について，U_i と U_j を結ぶ辺は $U_i \to U_j, U_i \leftarrow U_j$ の 2 本のみとし，3 本以上は存在しないこととする．

定義 2.2 2 つの頂点 U_i, U_j を結び得る $U_i \to U_j, U_i \leftarrow U_j$ の 2 本の辺を合わせて，**組辺**とよぶ．

2.2 DAG 判定の計算

グラフを表現するために用いる行列を，**隣接行列**という．頂点数 n であるグラフの隣接行列は $n \times n$ 行列 G である．ただし，

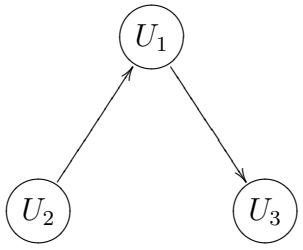
$$G_{ij} = \begin{cases} 1 & ((U_i, U_j) \in \vec{\mathbf{E}}) \\ 0 & (\text{それ以外}) \end{cases}$$

である．(Cormen,Leiserson,Rivest,Stein[4]) $\vec{\mathbf{E}}$ の定義から各対角成分 G_{ii} は 0 である．本論文では，G_{ij} を $G(U_i, U_j)$ と表すこともある．

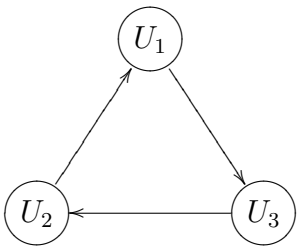
ここで，$\vec{G}$ が DAG か判定する関数 f を定義する．

f:頂点数 n の G を入力として受けとり，G が DAG のとき 1，
そうでないとき 0 を返す（G は連結なものに限る）．

例 2.3 • $G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ のとき，$f(G) = 1$



- $G = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ のとき，$f(G) = 0$



2.3 決定木複雑性

本研究で言う「計算量」とは時間計算量ではなく，何個のブール変数に問い合わせたかで計る計算コストのことである．本稿では，グラフを隣接行列で表し，各成分に何回問い合わせたかで計算量を計る．

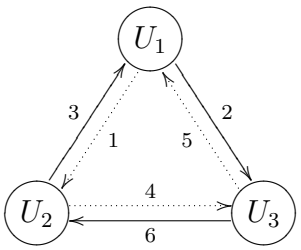
定義 2.4 頂点数 n のグラフの辺の真理値割り当て全体を $\mathcal{G}$ と表記する．ただし，グラフは連結されているものに限る．また，定義から隣接行列の対角成分は *0* なので，$\mathcal{G}$ の各要素は，対角成分以外の成分を表す $n^2 - n$ ビット列とみなせる．

f の計算コストを以下のように定義する．グラフの各辺に真理値を割り当て，その値は隠されているものとする．次に，決定性アルゴリズムにより辺を探索させ，隠された真理値を明らかにする．ここで，決定性アルゴリズムとは,「隠された辺の真理値を知るために，どの辺をどの順番で探索するかの手順」のことである．探索の履歴（クエリの応答の履歴）に依存せず，探索に先立って固定した順序にしたがって次に探索する辺を選ぶアルゴリズムを non-adaptive algorithm といい，そうでないものを adaptive algorithm という．(Buhrman,Spaan[3])

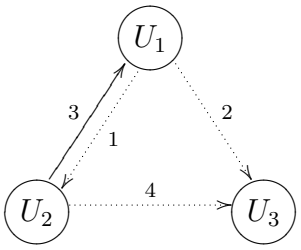
このとき，入力されたグラフが DAG かどうかがわかるまでに探索する辺の数は，入力するグラフや決定性アルゴリズムによって変わる．そこで，G に対しアルゴリズム A によって探索していくときの**コスト** $\mathrm{cost}(A, G)$ を,「G が DAG かどうかがわかるまでに探索した辺の数」と定義する．

例 2.5 A を $G(U_1, U_2), G(U_1, U_3), G(U_2, U_1), G(U_2, U_3), G(U_3, U_1), G(U_3, U_2)$ の順番に探索していくアルゴリズムとする．

- $G_1 = \begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ のとき，$\mathrm{cost}(A_1, G_1) = 6$



- $G_2 = \begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$ のとき，$\mathrm{cost}(A_2, G_2) = 4$



定義 2.6 (Arora,Barak [1]) 頂点数 n のグラフの辺を探索する決定性アルゴリズム全体を $\mathcal{A}_D$ と表記する．

$$D(f) = \min_{A \in \mathcal{A}_D} \max_{G \in \mathcal{G}} \mathrm{cost}(A, G)$$

を f の **deterministic complexity** という．

定義 2.7 (Arora,Barak[1]) 決定性アルゴリズム全体の集合上の確率分布を考える．このような分布全体の集合を $\mathcal{A}_R$ とする．このとき，

$$R(f) = \min_{A_R \in \mathcal{A}_R} \max_{G \in \mathcal{G}} \mathrm{cost}(A_R, G) \tag{1}$$

を f の **randomized complexity** という．
ここで，$\mathrm{cost}(A_R, G)$ はコスト期待値を表し，

$$\mathrm{cost}(A_R, G) = \sum_{A \in \mathcal{A}_D} \mathrm{prob}[A_R \text{ is } A] \times \mathrm{cost}(A, G) \tag{2}$$

である．

2.4 Adversary

Deterministic complexity の下界を調べる手法の一つに adversary がある．Adversary とは，(1)k 手目より前のクエリとアンサーの履歴と (2)k 手目のクエリが指定したブール変数 x の二つを受け取って，0 または 1 を返す関数である．

特に，与えられたアルゴリズムに対して，出力が判明するまでの手数をなるべく多くするようなものを考える．直観的な解釈として，次のように説明できる．アルゴリズムがプレイヤーI，隣接行列の各成分をセットする側がプレイヤーIIとなり，成分を1つ知るたびプレイヤーIがプレイヤーIIに1ドル払う状況を考える．このとき，プレイヤーIIにとっての戦略がadversaryであり，プレイヤーIから見て敵(adversary)なのである．

例 2.8 (Arora,Barak [1])n変数のOR関数について，deterministic complexityを考える．この場合のadversaryを，「$n-1$手目までの各クエリに対して0と答える」とする．すると，どのようなアルゴリズムであろうとn手目をクエリするまでOR関数の値はわからない．このことから，$D(f)=n$が導ける．

3 DAG判定のdeterministic complexity

本節では，アルゴリズムとしてnon-adaptive algorithmを考える．まず，DAGの判定をする関数fについて，$D(f)=n^2-n$になることを示す．そのためにadversary argumentを用いる．adversaryが1つではうまくいかないが（例3.2），2つ併用することで証明できる．

定義 3.1 決定性アルゴリズムによって辺を1つずつ探索するとき，二つのadversary，アド1とアド2を次のように定める．

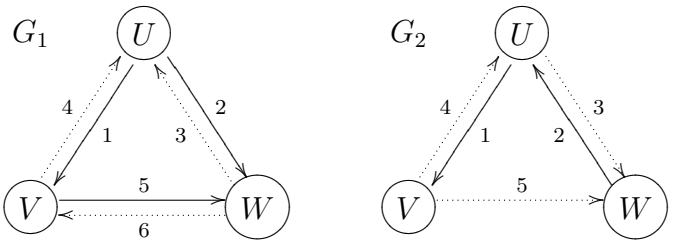
- アド1：各クエリに対して，基本的に1を返す．ただし，1を返すことでcycleができてしまう場合は，1を返さずに0を返す．
- アド2：最初のクエリから1ステップずつアド1のシミュレーションをしていく．シミュレーションが問題なく続いている間は常にアド1の0,1を反転させた結果を返す．ただし，アド1のシミュレーションが終了してしまった後は，ひたすら0と答える．

例 3.2 A_1, A_2をそれぞれ

$$A_1\colon G(U,V), G(U,W), G(W,U), G(V,U), G(V,W), G(W,V)$$

$$A_2\colon G(U,V), G(W,U), G(U,W), G(V,U), G(V,W), G(W,V)$$

の順番にクエリするアルゴリズムとする．A_1，A_2 にアド 1 を適用してできるグラフをそれぞれ G_1, G_2 とすると，$\mathrm{cost}(A_1, G_1) = 6 = 3^2 - 3$ となるが，A_2 に アド 1 を適用すると，$\mathrm{cost}(A_2, G_2) = 5 < 3^2 - 3$ となる．つまり，OR 関数の場合 (例 2.8) と異なり，アド 1 だけでコストを最大化できないことがある．同様に，アド 2 だけでもコストを最大化できないことがある．



補題 3.3 アド 1 を適用したとき，ある組辺（p.8 定義 2.2 参照）の 1 本目が 0 ならば 2 本目は 1 である．ただし，この 2 本目をクエリする時点で DAG 判定がまだできていないものとする．

証明　ある組辺の 1 本目が $G(U_i, U_j) = 0$ とする．アド 1 の定義により，U_j から U_i への有向経路が存在する．この組辺の 2 本目が $G(U_j, U_i) = 0$ とすると，U_i から U_j への有向経路が存在し，cycle ができる．2 本目をクエリする時点で DAG 判定がまだできていないという仮定に矛盾しているので，組辺の 1 本目が 0 ならば 2 本目 $G(U_j, U_i) = 1$ である． □

補題 3.4 アド 1 を適用して，判定の結果が DAG であるとき，すべての組辺は，

(1) 2 本ともクエリされていて，0, 1 が 1 本ずつ

(2) 1 本目は 0，2 本目がクエリされていない

のどちらかの状態になっている．

証明　2 本ともクエリされていない組辺があるとすると，そこで長さ 2 の cycle を作ることが考えられるので，判定結果が DAG であることに矛盾する．よって，すべての組辺は少なくとも 1 回はクエリされる．

1 本目が 1 である組辺は，2 本目もクエリしないと判定できないので，このような組辺はすべて 2 本目もクエリされる．アド 1 を適用しているので，2 本目には必ず 0 が返される．よって (1) の状態になる．

1 本目が 0 である組辺は，2 本目がクエリされなければ (2) の状態になる．そうでなければ補題 3.3 により 2 本目は 1 と答える．このとき状態 (1) になる．

以上により，各組辺は (1)，(2) のどちらかの状態に決まる． □

アド 1 を適用したとき，決定性アルゴリズムによってクエリの回数が異なる．n^2-n 手かかるとき，すべての組辺は (1) の状態になるが，n^2-n 手未満のとき (1) の状態の組辺と (2) の状態の組辺が混在する．

定理 3.5 (主定理) アド 1 を適用したとき n^2-n 手未満で DAG 判定できるならば，アド 2 を適用したとき n^2-n 手かかる．

証明 アド 1 を適用して，$k(<n^2-n)$ 手でできたグラフを G とする．G の各組辺は

(1) 2 本ともクエリされていて，0, 1 が 1 本ずつ

(2) 1 本だけクエリされていて，0 が 1 本のみ

のどちらかになっている．同じアルゴリズムでアド 2 を k 手まで適用すると，G の各辺の 0, 1 が反転したグラフ G' ができる．G' の各組辺は，

(3) 2 本ともクエリされていて，0, 1 が 1 本ずつ

(4) 1 本だけクエリされていて，1 が 1 本のみ

のどちらかの状態になっている．G' に cycle が存在しないことを示そう．G' が cycle Γ をもつと仮定する．Γ 上の各々の有向辺 $U_i \to U_j$ に対し，G は U_j から U_i への有向経路を持つことを示す．G' において $U_i \to U_j$ が (3) の場合，G において有向辺 $U_j \to U_i$ が存在する．それ以外，つまり (4) の場合，G において U_iから U_j への有向辺はない．アド 1 の定義により，U_j から U_i への有向経路が存在する．

したがって，G は Γ の逆向き，もしくはそれに頂点をいくつか補完した cycle をもつことになり，G が DAG であることに矛盾する．よって，k 手の時点で G' に cycle は存在しない．

G' には (4) の辺があるので，これらの 2 本目をクエリしないと DAG 判定ができない．アド 2 を適用すると，これらの 2 本目にはすべて 0 を返すので，すべての辺をクエリする前に判定できてしまうことはない．ゆえに，アド 2 を適用すると n^2-n 手かかる． □

定理 3.5 より，以下を得る．

系 3.6 (Best et al.(1974) の別証明) non-adaptive algorithm だけを考えるとき，任意の頂点数 n に対し，$D(f) = n^2 - n$

証明　主張 3.5 より，アド 1 で $n^2 - n$ 手かからないアルゴリズムが存在しても，アド 2 を適用すると $n^2 - n$ 手かかることが言えた．すなわち，任意のアルゴリズムに対して $n^2 - n$ 手かかる入力が存在する．よって，題意が示せた．　□

系 3.6 については，村上弘氏によって別証明がされている．証明は後述の付録参照．

4　DAG 判定の randomized complexity

本節では，アルゴリズムとして non-adaptive algorithm を考える．乱数を用いることによって計算量を落とせるかどうかは，決定木計算量にとって基本的な問題である．まず，アルゴリズムとして non-adaptive algorithm を考えたときの $R(f)$ を求める．$n = 2$ のとき $R(f) = D(f)$ となり，$n \geq 3$ のとき $R(f) < D(f)$ となることを示す．すなわち，$n \geq 3$ の場合，乱数の利用によって計算コストの節約ができる．

adaptive algorithm を考えた場合の $R(f)$ も non-adaptive algorithm と同様のことが成り立つことを示す．

なお，本節の結果は，指導教員である鈴木登志雄氏によるものである．

命題 4.1　(1) $n = 2$ のとき，$R(f) = 2$

(2) $n \geq 3$ のとき，$R(f) < n^2 - n$

証明　**(命題 4.1(1))** 完全グラフ G を考える．$G(U_1, U_2)$ を先にクエリする決定性アルゴリズムの G に対するコストは 2 である．$G(U_2, U_1)$ を先にクエリする決定性アルゴリズムの G に対するコストも 2 である．よって，式 (2) の右辺は

$$\sum_{A \in \mathcal{A}_D} \mathrm{prob}[A_R \text{ is } A] \times 2 = 2$$

となる．よって，$R(f) = 2$ が成り立つ．　□

次に，命題 4.1(2) を示すために，以下の補題を示す．

補題 4.2 $n \geq 3$ とする．このとき，頂点数 n の任意の有向グラフ G に対し、ある決定性アルゴリズム A' が存在し，$\mathrm{cost}(A', G) < n^2 - n$ となる．

証明 (i) G が有向 cycle をもつとき
極小有向 cycle のひとつをとり，それを $U_1 \to \cdots \to U_m \to U_1$ とする．このとき，$G(U_1, U_2), \cdots, G(U_{m-1}, U_m), G(U_m, U_1)$ を順次クエリするアルゴリズムを A' とすると，コスト m で「有効 cycle を含む」と判断を下せる．このとき，$m \leq n < n^2 - n$ となる．よって，$\mathrm{cost}(A', G) < n^2 - n$ となる．
(ii) G が有向 cycle をもたないとき
$G(U_i, U_j) = 0$ となるようなすべての $G(U_i, U_j)$ に対し，まず $G(U_i, U_j)$ をクエリするアルゴリズムの一つを A' とする．すると，このような (U_i, U_j) すべてについてクエリし終わった段階で「有向 cycle なし」と判断できる．よって，

$$\mathrm{cost}(A', G) \leq (\text{答えが } 0 \text{ になるクエリの総数}) < n^2 - n$$

ゆえに、題意が示された． □

注意 4.3 我々は連結なグラフのみを考えているから，上記の証明において（答えが 0 になるクエリの総数）$< n^2 - n$ となる．ただし，連結という仮定がなくても補題 4.2 は成り立つ．なぜなら，高々 $n^2 - n - 1$ 個の有向辺についてクエリすれば「有向 cycle なし」と判断できるからである．

命題 4.1(2) **の証明** 決定性アルゴリズム全体の集合上の一様分布を A_R とする．今，G を頂点数 n の任意の有向グラフとする．

補題 4.2 により，$\mathrm{cost}(A', G) < n^2 - n$ となる決定性アルゴリズム A' がある．A_R において $\mathrm{prob}[A_R \text{ is } A'] > 0$ であるから，

$$R(f) = ((2) \text{ 右辺}) < n^2 - n$$

となる． □

系 3.6 と命題 4.1 を合わせると，以下を得る．

命題 4.4 non-adaptive アルゴリズムのみを考えるとき，

- $n = 2$ の場合　$R(f) = D(f) = 2^2 - 2 = 2$
- $n \geq 3$ の場合　$R(f) < D(f) = n^2 - n$

命題 4.5 命題4.1は，adaptive algorithmを含めて考える場合にも同じ形で成り立つ．

証明 non-adaptive algorithmを考えた場合，adaptive algorithmも含めて考えた場合のrandomized complexityをそれぞれ$R^{\mathrm{nonad}}(f)$，$R^{\mathrm{ad}}(f)$と表記する．

$n=2$のときすべてのアルゴリズムがnon-adaptiveであるから，以下が成り立つ．

$$R^{\mathrm{ad}}(f) = R^{\mathrm{nonad}}(f) = 2$$

$n \geq 3$のときadaptive algorithmはnon-adaptive algorithmを含めたアルゴリズムの一種なので，以下が成り立つ．

$$R^{\mathrm{ad}}(f) \leq R^{\mathrm{nonad}}(f) < n^2 - n$$

□

5 adaptive algorithmの場合のdeterministic complexity

本節では，アルゴリズムとしてadaptive algorithmを含めた場合を考える．頂点数が3の場合$D(f) = 3^2 - 3 = 6$になり，non-adaptive algorithmだけ考えたときと同様のことが成り立つことを示す．

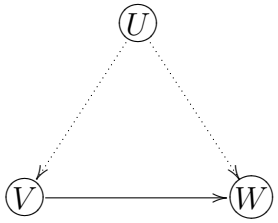
グラフを構成する途中の状態，すなわちいくつかの有向辺について有無が決定されている状態を考える．これを，各成分が1（辺有り），0（辺無し），?（未定義）であるような行列で表そう．

上記のような行列Gが与えられた時，Gの転置行列${}^t\!G$を考える．このとき，任意の$(0 \leq k \leq n^2 - n)$に対し以下が成り立つ．

補題 5.1 以下二つの主張は同値である．

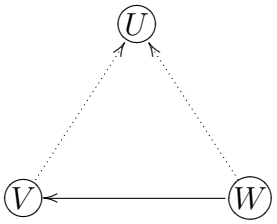
- あるadversaryをGに適用すると，DAGであるか判定するまでに全体でk手かかる．
- あるadversaryを${}^t\!G$に適用するとDAGであるか判定するまでに全体でk手かかる．

例 5.2 $G = \begin{pmatrix} 0 & 0 & 0 \\ ? & 0 & 1 \\ ? & ? & 0 \end{pmatrix}$



のとき，

${}^{t}G = \begin{pmatrix} 0 & ? & ? \\ 0 & 0 & ? \\ 0 & 1 & 0 \end{pmatrix}$



となる．

証明 (補題 5.1) U, V を G の異なる 2 頂点とする．$\Rightarrow$ の証明をいえば十分である（$\Leftarrow$ は同様）．

ある adversaryα を G に適用すると，DAG か判定するまでに全体で k 手かかるとする．${}^{t}G$ に対する adversary${}^{t}\alpha$ を以下のように定める．G に対する α の動きをシミュレートしていく．${}^{t}G$ について「U から V への辺ががあるか？」と尋ねられたら G について「V から U への辺があるか？」を α に尋ねる．そしてその応答（1 または 0）を，${}^{t}\alpha$ の応答とする．G における cycle は，${}^{t}G$ における逆向きの cycle に対応するから，α を G に適用したときと同じだけ，${}^{t}\alpha$ を ${}^{t}G$ に適用したときに手数がかかる．□

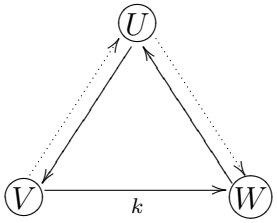
補題 5.3 頂点数が 3 のとき，任意のアルゴリズムに対して，次のような入力がある：1 手目が 0 かつ $3^2 - 3 = 6$ 手かかる．

証明 あるアルゴリズム A が存在して以下が成り立つとする：任意の入力に対し，1 手目が 0 ならば 6 手かからない

このような A を一つ固定する．また，A に対して 1 手目に 0 と答える入力のうち最もコストの高い G を一つ固定する．そのコストを $k\,(k < 6)$ とする．k 手目に $0, 1$ どちらを返しても終了する．

(i) k 手目が組辺の 1 本目の場合

k 手目を 1 としたとき，長さ 3 の cycle ができて終了するので，考えられるグラフの形は U, V, W の置換を除いて下図のみで，このとき $k = 5$ である．



- 4 手目が $G(W,U)$ のとき，$G(W,U)$ の値を 0 に変更することにより，6 手かかる adversary の存在を示せる.（以下、「$G(W,U)=0$ に変更すると 6 手かかる」と表す.）この場合，5 手目に 1 と答えればよい.

- 4 手目が $G(U,V)$ の場合も同様.

- 4 手目が $G(U,W)$ のとき
  3 手目が $G(W,U)$ のとき．1 手目への応答が 0 であるという仮定により，2 手目は $G(U,V)$ である．$G(U,V)=0$ に変更すると，6 手かかる．この場合，{U,W} と {V,W} それぞれが cycle を作るか調べねばならないからである.
  3 手目が $G(U,V)$ のとき，$G(U,V)=0$ に変更すると、6 手かかる．この場合，$G(V,W)$ と $G(W,V)$ は先にきかれた方を 1，後のを 0，$G(U,W)$ は 0 と答えればよい.

- 4 手目が $G(V,U)$ のとき，4 手目が $G(U,W)$ の場合と同様にして以下を得る.
  3 手目が $G(U,V)$ のとき．1 手目への応答が 0 であるという仮定により，2 手目は $G(W,U)$ である．$G(W,U)=0$ に変更すると 6 手かかる.
  3 手目が $G(W,U)$ のとき，$G(W,U)=0$ に変更すると，6 手かかる.

以上より，$k<6$ が最大であることに矛盾する.

(ii) k 手目が組辺の 2 本目で，もう一方の辺が 0 の場合

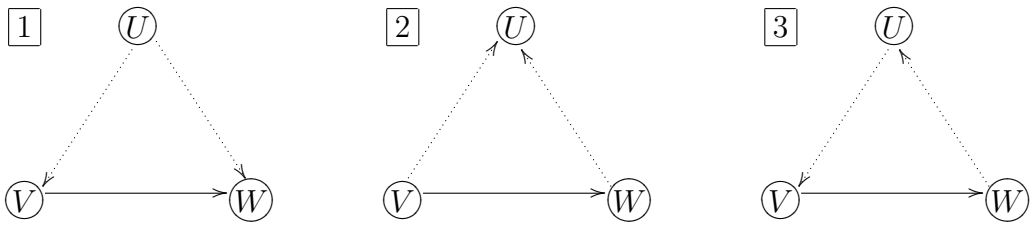
もう一方の辺が 0 なので，終了条件から k 手目が 1 のとき長さ 3 の cycle を作って終了する．他の 2 つの組辺には、k 手目の辺と cycle を作るように 1 の辺が 1 本ずつある．$k<6$ より，これらの組辺のうち少なくともどちらか一つは，もう一方の辺がクエリされていない状態である．しかしこの場合，k 手目で 0 を返したときに cycle が含まれないと断定することはできず，矛盾する．ゆえに，k 手目のもう一方の辺が 0 であることはない.

(iii) k 手目が組辺の 2 本目で，もう一方の辺が 1 の場合

$G(W,V)$ を k 手目にクエリするとして，一般性を失わない.

$k = 4$ のとき

4 手で終了するグラフについて，3 手目までの形は（U, V, W の置換を除いて）次の 3 つに限る．



ここで，6 手かかる adversary が存在することを示す．

1 最初の 3 手のクエリと応答が，集合として $\{G(U,V) = 0, G(V,W) = 1, G(U,W) = 0\}$ であるとき，クエリされる順番で場合分けする．

(1.1) 順番が $G(U,V), G(V,W), G(U,W)$ のとき，2 手目を $G(V,W) = 0$ に変更すると 6 手かかる．なぜなら，以下 (1.1.1)〜(1.1.4) のようにして adversary を作れるからである．

(1.1.1) 3 手目が $G(W,U)$ のときは 0 と答え，4, 5 手目は 1 と答えればよい．

(1.1.2) 3 手目が $G(U,W)$ のときは 1 と答え，$G(V,U)$ と $G(W,V)$ は先にきかれた方を 1，後のを 0，$G(W,U)$ は 0 と答えればよい．

(1.1.3) 3 手目が $G(V,U)$ のときは 1 と答え，$G(W,V)$ と $G(W,U)$ は先にきかれた方を 0，後のを 1，$G(U,W)$ は 1 と答えればよい．

(1.1.4) 3 手目が $G(W,V)$ のときは 1 と答え，$G(V,U)$ と $G(W,U)$ は先にきかれた方を 0，後のを 1，$G(U,W)$ は 1 と答えればよい．

(1.2) 順番が $G(U,V), G(U,W), G(V,W)$ のとき，2 手目 $G(U,W) = 1$ に変更すると 6 手かかる．

(1.2.1) 3 手目が $G(W,U)$ のときは 0 と答える．(1.1.3) と同様にすればよい．

(1.2.2) 3 手目が $G(V,U)$ のときは 0 と答え，$G(V,W)$ と $G(W,V)$ は先にきかれた方を 1，後のを 0，$G(W,U)$ は 0 と答えればよい．

(1.2.3) 3 手目が $G(V,W)$ のときは 0 と答える．(1.1.2) と同様にすればよい．

(1.2.4) 3 手目が $G(W,V)$ のときは 1 と答え，4, 5 手目は 0 と答えればよい．

(1.3) $G(U,W), G(U,V), G(V,W)$ のとき，2 手目 $G(U,V)=1$ に変更すると 6 手かかる．(1.2) と同様にすればよい．

(1.4) $G(U,W), G(V,W), G(U,V)$ のとき，3 手目 $G(U,V)=1$ に変更すると 6 手かかる．(1.2.4) と同様にすればよい．

[2] 最初の 3 手のクエリと応答が，集合として $\{G(V,U)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．補題 5.1 により，[1] に帰着する．

[3] 最初の 3 手のクエリと応答が，集合として $\{G(U,V)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．

(3.1) $G(U,V), G(W,U)G(V,W)$ のとき，$G(V,W)=0$ に変更すると 6 手かかる．(1.1.1) と同様にすればよい．

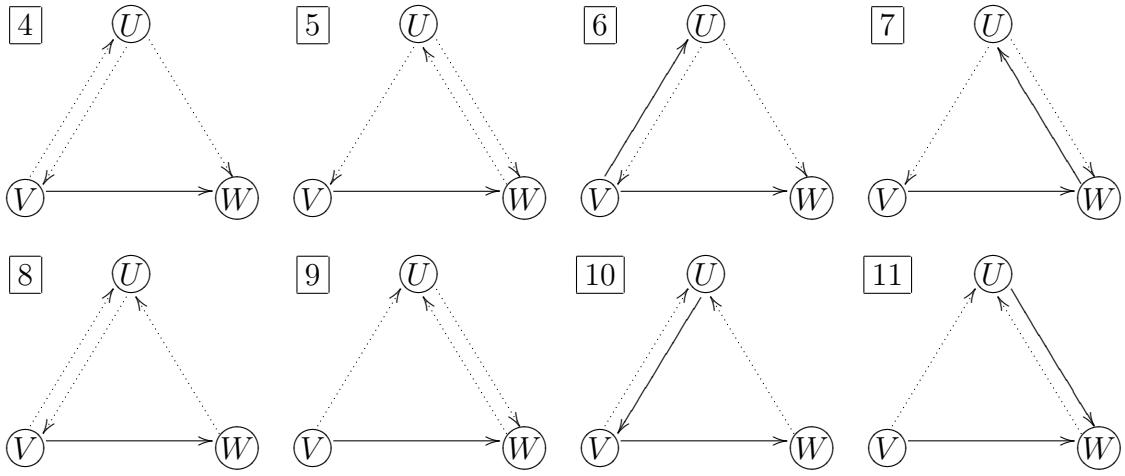
(3.2) $G(U,V), G(V,W), G(W,U)$ のとき，$G(V,W)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．

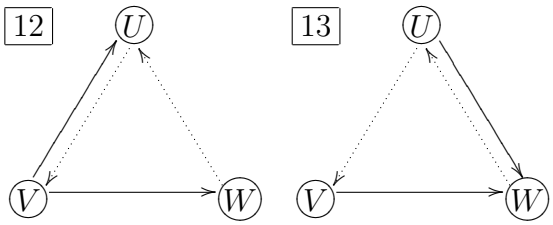
(3.3) $G(W,U), G(U,V), G(V,W)$ のとき，$G(V,W)=0$ に変更すると 6 手かかる．(1.1.1) と同様にすればよい．

(3.4) $G(W,U), G(V,W), G(U,V)$ のとき，$G(V,W)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．

$k=5$ のとき

5 手で終了するグラフについて，4 手目までの形は（U,V,W の置換を除いて）次の 10 個に限る．





ここで，6 手かかる adversary が存在することを示す．

4 最初の 4 手のクエリと応答が，集合として $\{G(U,V)=0, G(U,W)=0, G(V,W)=1, G(V,U)=0\}$ であるとき．

(4.1) 4 手目 が $G(V,U)$ のとき
3 手目の時点では，1 の 3 手目までと同じである．1 では 3 手以前に戻ると 6 手かかる adversary が存在するので，この場合も 6 手かかる adversary が存在する．

(4.2) 4 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．5 手目に 0 と答えればよい．

(4.3) 4 手目が $G(U,W)$ のとき，$G(U,W)=1$ に変更すると 6 手かかる．5 手目に 0 と答えればよい．

(4.4) 4 手目が $G(V,W)$ のとき

(4.4.1) 3 手目が $G(U,W)$ のとき，$G(U,W)=1$ に変更すると 6 手かかる．(1.2.2) と同様にすればよい．

(4.4.2) 3 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．(1.1.3) と同様にすればよい．

(4.4.3) 3 手目が $G(V,U)$，2 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．(1.2) と同様にすればよい．同様に 3 手目が $G(V,U)$，2 手目が $G(U,W)$ のときも，$G(U,W)=1$ に変更すると 6 手かかる．

5 最初の 4 手のクエリと応答が，集合として $\{G(U,V)=0, G(U,W)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．

(5.1) 4 手目が $G(W,U)$ のとき，3 手目までは 1 と同じなので，1 により 6 手かかる．

(5.2) 4 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．5 手目に 0 と答えればよい．

(5.3) 4 手目が $G(U,W)$ のとき，3 手目までは $\boxed{3}$ と同じなので，$\boxed{3}$ により 6 手かかる．

(5.4) 4 手目が $G(V,W)$ のとき，3 手目までは (4.4) と同じ形をしている．(4.4) により 6 手かかる．

$\boxed{6}$ 最初の 4 手のクエリと応答が，集合として $\{G(U,V)=0, G(U,W)=0, G(V,U)=1, G(V,W)=1\}$ であるとき．

(6.1) 4 手目が $G(V,U)$ のとき，3 手目までは $\boxed{1}$ と同じなので，6 手かかる．

(6.2) 4 手目が $G(U,W)$ のとき $G(U,W)=1$ に変更すると 6 手かかる．5 手目に 0 と答えればよい．

(6.3) 4 手目が $G(V,W)$ のとき

(6.3.1) 3 手目が $G(V,U)$，2 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．(1.2) と同様にすればよい．同様に，3 手目が $G(V,U)$，2 手目が $G(U,W)$ のとき，$G(U,W)=1$ に変更すると 6 手かかる．

(6.3.2) 3 手目が $G(U,V)$ のとき，1 手目への応答が 0 であるという仮定により，2 手目は $G(V,U)$ である．$G(V,U)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．

(6.3.3) 3 手目が $G(U,W)$ のとき，1 手目への応答が 0 であるという仮定により，2 手目は $G(V,U)$ である．$G(V,U)=0$ に変更すると 6 手かかる．この場合，$\{U,W\}$ と $\{V,W\}$ それぞれが cycle を作るか調べねばならないからである．

(6.4) 4 手目が $G(U,V)$ のとき

(6.4.1) 3 手目が $G(U,W)$ のとき，$G(U,W)=1$ に変更すると 6 手かかる．4, 5 手目に 0 と答えればよい．

(6.4.2) 3 手目が $G(V,U)$ のとき，$G(V,U)=0$ に変更すると 6 手かかる．(1.1.2) と同様にすればよい．

(6.4.3) 3 手目が $G(V,W)$ のとき，1 手目への応答が 0 であるという仮定により，2 手目は $G(V,U)$ である．$G(V,U)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．

7 最初の 4 手のクエリと応答が，集合として $\{G(U,V)=0, G(U,W)=0, G(V,W)=1, G(W,U)=1\}$ であるとき．

(7.1) 4 手目が $G(W,U)$ のとき，3 手目までは 1 と同じなので，6 手かかる．

(7.2) 4 手目が $G(U,W)$ のとき，1 手目は $G(U,V)$ に決まる．2 手目が $G(V,W)$ のとき，$G(V,W)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．同様に，2 手目が $G(W,U)$ のとき，$G(W,U)=0$ に変更すると 6 手かかる．

(7.3) 4 手目が $G(U,V)$ のとき，1 手目は $G(U,W)$ に決まる．

(7.3.1) 3 手目が $G(V,W)$，2 手目が $G(W,U)$ のとき，$G(W,U)=0$ に変更すると 6 手かかる．(6.3.3) と同様にすればよい．

(7.3.2) 3 手目が $G(W,U)$，2 手目が $G(V,W)$ のとき，$G(W,U)=0$ に変更すると 6 手かかる．$G(U,V)$ と $G(V,U)$ は先にきかれた方を 1，後のを 0，$G(W,V)$ は 0 と答えればよい．

(7.4) 4 手目が $G(V,W)$ のとき

(7.4.1) 3 手目が $G(W,U)$，2 手目が $G(U,V)$ のとき，$G(U,V)=1$ に変更すると 6 手かかる．(1.2) と同様にすればよい．同様に 3 手目が $G(W,U)$，2 手目が $G(U,W)$ のとき，$G(U,W)=1$ に変更すると 6 手かかる．

(7.4.2) 3 手目が $G(U,W)$ のとき，1 手目は $G(U,V)$，2 手目は $G(W,U)$ に決まる．$G(W,U)=0$ に変更すると 6 手かかる．(1.1) と同様にすればよい．

(7.4.3) 3 手目が $G(U,V)$ のとき，1 手目は $G(U,W)$，2 手目は $G(W,U)$ に決まる．$G(W,U)=0$ に変更すると 6 手かかる．(6.3.3) と同様にすればよい．

8 最初の 4 手のクエリと応答が，集合として $\{G(U,V)=0, G(V,U)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．命題 5.1 により，5 に帰着する．

9 最初の 4 手のクエリと応答が，集合として $\{G(U,W)=0, G(V,U)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．命題 5.1 により，4 に帰着する．

10 最初の4手のクエリと応答が，集合として $\{G(U,V)=1, G(V,U)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．命題5.1により，7に帰着する．

11 最初の4手のクエリと応答が，集合として $\{G(U,W)=1, G(V,U)=0, G(V,W)=1, G(W,U)=0\}$ であるとき．命題5.1により，6に帰着する．

12 最初の4手のクエリと応答が，集合として $\{G(U,V)=0, G(V,U)=1, G(V,W)=1, G(W,U)=0\}$ であるとき．

(12.1) 4手目が $G(V,U)$ のとき，3手目までは3と同じなので，6手かかる．

(12.2) 4手目が $G(V,W)$ のとき，$G(V,W)=0$ に変更すると6手かかる．5手目に1と答えればよい．

(12.3) 4手目が $G(U,V)$ のとき1手目は $G(W,U)$ に決まる．

(12.3.1) 3手目が $G(V,U)$，2手目が $G(V,W)$ のとき，$G(V,W)=0$ に変更すると6手かかる．(1.1) と同様にすればよい．

(12.3.2) 3手目が $G(V,W)$，2手目が $G(V,U)$ のとき，$G(V,W)=0$ に変更すると6手かかる．(1.1.2) と同様にすればよい．

(12.4) 4手目 $G(W,U)$ のとき，1手目は $G(U,V)$ に決まる．

(12.4.1) 3手目が $G(V,W)$ のとき，$G(V,W)=0$ に変更すると6手かかる．(1.1.3) と同様にすればよい．

(12.4.2) 3手目が $G(V,U)$ のとき，$G(V,U)=0$ に変更すると6手かかる．(1.2.2) と同様にすればよい．

13 最初の4手のクエリと応答が，集合として $\{G(U,V)=0, G(U,W)=1, G(V,W)=1, G(W,U)=0\}$ であるとき．命題5.1により，12に帰着する．

以上より、$k<6$ が最大であることに矛盾する．

(i)〜(iii) より，背理法の仮定に矛盾することが言えたので，題意を示せた． □

補題5.3より，以下が言える．

命題 5.4 頂点数が3のとき，

$$D(f)=3^2-3=6$$

6　今後の課題について

$D(f)$ について，本研究では non-adaptive algorithm に限定しているため，Best et al.[2] と完全に同じ結果を出すには至っていない．本研究の手法の応用範囲を広げていくのが今後の課題である．特に，Best et al.[2] のようなアルゴリズムに制約のない結果に対して，本研究の手法による別証明を与えることが今後の課題として興味深い．

7　付録

系 3.6 の別証明　$D(f) < n^2 - n$ と仮定する．すなわち，あるアルゴリズム A_0 が存在し，A_0 はどんな入力に対しても $n^2 - n$ 手未満で DAG 判定ができると仮定する．$\cdots(*)$　A_0 のもとでは $n^2 - n$ 手未満で終了するので，$n^2 - n$ 手目はクエリされない．その辺を U_i から U_j への辺とする．今，$U_k \neq U_i, U_j$ をとる．

$G(U_k, U_l) = 1(U_l \neq U_i, U_j, U_k), G(U_k, U_i) = 1, G(U_j, U_k) = 1, G(U_i, U_j)$ を除くその他の辺を 0 とするグラフを G とする．このとき，G は $n^2 - n - 1$ 手までクエリされている．$(*)$ より DAG 判定ができており，G は DAG である．

G において，$G(U_i, U_j) = 1$ としたグラフを G' とすると，G' は有向 cycle をもつので DAG ではない．すなわち，$n^2 - n$ 手目までクエリしなければ DAG 判定ができない入力が存在し，背理法の仮定に矛盾する．ゆえに，題意が示せた．　□

謝辞

学部生のときからご指導くださいました，鈴木登志雄准教授に厚く御礼申し上げます．お忙しい中，親身にご指導してくださったおかげで，本論文を書き上げることができました．また，研究集会で発表をする貴重な経験をさせていただいたことにも，感謝いたします．

同研究室の水澤勇気氏にも，本研究に関してアドバイスを頂きました．この場を借りて御礼申し上げます．

最後に，修士課程に進学する機会を与えてくれ，今日まで支えてくれた家族に感謝いたします．

参考文献

[1] S.Arora，B.Barak，*Computational complexity*：*A modern approach*，Cambridge university press，Cambridge，2009.

[2] M.R.Best，P.van Emde Boas，H.W.Lenstra Jr.，A sharpened version of the Aanderaa-Rosenberg conjecture，*Report* ZW 30/74，Mathematisch Centrum Amsterdam，1974.

[3] H.Buhrman，E.Spaan，L.Torenvliet，The relative power of logspace and polynomial time reductions，*Computational complexity*，3(1993)，pp. 231-244.

[4] T.H.Cormen，C.E.Leiserson，R.L.Rivest，C.Stein，*Introduction to algorithms*，*Third edition*，MIT Press，2009．邦訳：アルゴリズムイントロダクション第 3 版　第 2 巻＝高度な設計と解析手法・高度なデータ構造・グラフアルゴリズム，浅野哲夫，岩野和生，梅尾博司，山下雅史，和田幸一 共訳，近代科学社，2012.

[5] R.Holt，E.Reingold，On the time required to detect cycles and connectivity in graphs，*Mathematical systems theory*，6(1972)，pp. 103-106.

[6] J.Kahn，M.Saks，D.Sturtevant，A topological approach to evasiveness，*Combinatorica*，4(1984)，pp. 297-306.

[7] M.Saks，A.Wigderson，Probabilistic boolean decision trees and the complexity of evaluating game trees，in；*Proc. 27th Annual IEEE Symposium on Foundations of Computer Science(FOCS)*，pp. 29-38，1986.

[8] 鈴木譲，ベイジアンネットワーク入門 確率的知識情報処理の基礎，培風館，2009.