2003－MPS－47　(8)
2003／12／12

衝突問題に対する量子アルゴリズムにおけるソーティング方法の選択について

[1]大久保 誠也　[2]西野 哲朗

あらまし: 衝突問題とは, 集合 X と関数 $f : X \to Z$ が与えられたときに, $f(x) = f(y)$ を満たすような対 $(x, y) \in X \times X$ を発見する問題である. このような対 (x, y) のことを衝突という. 本論では, Brassard らによって提案された, 衝突問題に対する量子アルゴリズムの時間計算量を解析する. その量子アルゴリズムは, ソーティング・アルゴリズムを部分ステップとして含んでいるので, そのアルゴリズムの計算量は, 用いられるソーティングアルゴリズムの計算量に依存する. そこで, 我々は, 種々の状況において, その量子アルゴリズムに適したソーティング・アルゴリズムについて議論する. 本論の解析においては, マージソート, 基数ソート, バケットソートを取り上げる. 最後に, 上記量子アルゴリズムの, 時間計算量と領域計算量の関係についても考察する.

How to Find a Suitable Sorting Algorithm for a Quantum Algorithm for the Collision-Finding Problem

[1]Seiya Okubo　[2]Tetsuro Nishino

Abstract: The collision-finding problem is a problem, given a set X and a function $f : X \to Z$, to find an element $(x, y) \in X \times X$ such that $f(x) = f(y)$. Such a pair (x, y) is called a collision. In this paper, we analyse the time complexity of the quantum algorithm for the collision-finding problem proposed by Brassard et al. Since the quantum algorithm contains a sorting algorithm as a substep, the complexity of the algorithm depends on the complexity of the incorporated sorting algorithm. Thus, we discusses suitability of some sorting algorithms for the quantum algorithm in several settings. In our analysis, we adopt merge sort, radix sort, and bucket sort algorithms. Finally, we study the relationships between the time complexity and the space complexity of the above quantum algorithm.

1　はじめに

1985 年に，D.Deutsch は量子コンピュータのモデル化を行い, 量子力学に基づいた新しい計算モデルとして，量子 Turing 機械を提案した．さらに, 1994 年に P.W.Shor は，整数の因数分解を多項式時間内に高い成功確率で行う量子アルゴリズムを示した．また，1996 年には L.K.Grover が，解の探索問題に対する効率的量子アルゴリズムを設計した．

量子コンピュータ上で効率良く解ける, もうひとつの問題として，**衝突問題 (collision problem)** が知られている [1]．現在, 一般に広く使われているディジタル署名は，RSA 暗号により暗号化して送信されてきた署名を復号化したものと，送られてきた文章のハッシュ値を比較することによって，文章が正式なものであるか否かを判断している．このディジタル署名の安全性は，$f(x) = f(y)$ を満たすような，x とは異なる y を発見することが難しいという仮定に基づいている．ここで，このような x と y を**衝突 (collision)** という．つまり，衝突を発見することができると，上のような文章を改竄することが可能となる．このような衝突を発見する問題を衝突問題という．

本論文では，論文 [1] で, Brassard らにより示された衝突問題のアルゴリズムでサブルーチンとして用いられている, ソーティング・アルゴリズムを変更することにより，量子アルゴリズム全体の計算量にどのような変化が生じるかについて考察する．その結果, Brassard らの量子アルゴリズムを量子コンピュータ上で動作させるときに, 実際のメモリサイズと実行ステップ数にどのような関係があるかについても明らかとなり, 本量子アルゴリズムの種々の応用において, どのソーティング・アルゴリズムを用いればよいかが明確になった.

2　Grover のアルゴリズム

Grover のアルゴリズムが対象とする**検索問題**とは, システムが $S_1, S_2, S_3, \cdots, S_N$(ただし $N = 2^n$) とラベル付けされた状態を取るとしたときに, これらの状態の中から条件 $C(S) = 1$ を満たす唯一の状態を捜し出す問題である（どのような状態 S に対しても $C(S)$ は単位時間で条件を満たすか否か判定できるとする）．また, 各状態は n ビットの記号列で表され, 条件を満たさない $C(S)$ に対しては $C(S) = 0$ が成り立つする. 検索問題を解くには, 古典的アルゴリズムでは平均 $0.5N$ 回オラクルにアクセスする必要があるが, Grover のアルゴリズムでは, $O(\sqrt{N})$ 回のオラクルへのアクセスで十分である [2]. また, 検索問題において, 条件 $C(S) = 1$ を満たす状態が t 個存在する場合, Grover のアルゴリズムは期待時間 $O\left(\sqrt{N/t}\right)$ でこの問題を解くこ

[1]電気通信大学大学院 電気通信学研究科 電子情報学専攻 The Graduate School of Electro-Communications
e-mail: s-okubo@ice.uec.ac.jp
[2]電気通信大学 電気通信学部 情報通信工学科 情報メディア工学講座 The University of Electro-Communications
e-mail: nishino@ice.uec.ac.jp

とができる．この Grover のアルゴリズムの一般化を，以後 Grover(F, y_0) と書く．

3 衝突問題

本論で扱う衝突問題は，以下のように定義される．

定義 1 (衝突問題，collision problem) 集合 X，関数 $f : X \to Z$ が与えられたときに，$f(x) = f(y)$ を満たす異なる要素の組 $(x, y) \in X \times X$ を発見せよ．また，そのような組 (x, y) を **衝突** という．

定義 2 (r-to-one 関数) 関数 $f : X \to Y$ は，任意の $y \in Y$ に対し，$|\{x|x \in X, f(x) = y\}| = r$ を満たすとき，r-*to-one* であると言う．

論文 [1] で，関数 F が r-to-one であるときの衝突問題に対する，以下の量子アルゴリズムが示された．

Collision(F, k)

1. $|\mathbf{X}| = N, |\mathbf{K}| = k, \mathbf{K} \subseteq \mathbf{X}$ をランダムに抽出し，$(x,\ F(x))$,$x \in \mathbf{K}$ からなるテーブル L を作成する．
2. 第 2 成分 $F(x)$ に従って L をソートする．
3. L 内に衝突が含まれていないかをチェックする．具体的には，$F(x_0) = F(x_1)$ となる 2 つの要素 $(x_0, F(x_0)), (x_1, F(x_1)) \in L$ が存在しているか否かを点検する．もし存在したなら，ステップ 6 に進む．
4. $x_1 =$ **Grover**$(H, 1)$ を計算する．
   ただし，$H : X \to \{0.\ 1\}$ は以下のように定義する．$H(x) = 1 \Leftrightarrow (x_0, F(x)) \in L$ だが，$x \neq x_0$ である $x_0 \in \mathbf{K}$ が存在する．
5. $(x_0, F(x_1)) \in L$ を二分探索を用いて発見する．
6. 衝突の組 (x_0, x_1) を出力する．

関数 f が r-to-1 であったとき，次のような事実が示されている [1]:

1. ステップ 2 では，ソーティングを実行するために $l \log l = \sqrt[3]{N/r} \log N$ 回の比較が必要である．
2. ステップ 4 では，Grover のアルゴリズムを実行するのに $\sqrt[3]{N/r} \log N$ 回の比較が必要である．

全体として，Collision(F, k) の実行のためには，$O(\sqrt[3]{N/r} \log N)$ 回の比較が必要となる．

以下，本論文では，関数 F を評価するためには，$O(T)$ ステップが必要であるとの仮定して，Collision(F, k) の実行に必要な時間量を評価する．Collision(F, k) の実行においては，その第 2 ステップでソーティングを行っているので，アルゴリズム全体の時間量は，この第 2 ステップで用いるソーティング・アルゴリズムの時間量に依存する．そこで，種々の場合において，Collision(F, k) の実行には，どのようなソーティング・アルゴリズムが適しているのかについて考察を行っていく．

4 衝突問題の量子時間量

以下では，F の関数値の長さが $O(n)$ になる場合について考える．ここで，$2^n = N$ である．さらに，関数 F の計算に必要な時間量を $O(T)$ ステップとし，ステップ 2 で用いるソーティング・アルゴリズムの種類にしたがって，量子アルゴリズム全体の時間量を評価する．

4.1 マージソートを用いた場合

マージソートを用いた場合は，ステップ 2 の実行に $O(kT + k \log k)$ ステップ必要であり，ステップ 4 で $O(\sqrt{N/rk}(T + \log k))$ ステップ必要である．したがって，全体では $O((T + \log k)(k + \sqrt{N/rk}))$ ステップが必要である．

この場合，最も時間量が小さくなるのは $k = \sqrt[3]{N/r}$ としたときで $O((T + \log N)\sqrt[3]{N/r})$ ステップとなる．

4.2 基数ソートを用いた場合

基数ソートの場合に必要な時間量 t は，以下のようにして評価できる．k 個の要素のテーブルを作成するのに $O(kT)$ ステップ，テーブルを $f(x)$ の値に従ってソートするのに $O(k \log_u N)$ ステップ，Grover のアルゴリズムで所望の要素を検索するのに $O((T + \log k)\sqrt{N/rk})$ ステップが，それぞれ必要である．したがって，全部で $O(kT + k \log_u N + (T + \log k)\sqrt{N/rk})$ ステップ必要となる．

最適な場合を求めるために，以下の 4 つの場合について考える．

(1) $T > \log k$ かつ $T > \log_u N$ のとき

時間量は $O(T(k + \sqrt{N/rk}))$ ステップである．時間量が最小となるのは，$k = \sqrt[3]{N/r}$ のときで $O(T\sqrt[3]{N/r})$ ステップとなる．

(2) $T > \log k$ かつ $T < \log_u N$ のとき

時間量は $O(k \log_u N + T\sqrt{N/rk})$ ステップである．時間量が最小となるのは，$k \log_u N = T\sqrt{N/rk}$ のとき，すなわち

$k = \sqrt[3]{T^2 N/(r \log_u^2 N)}$ のとき で，$O(\sqrt[3]{T^2 N \log_u N}/r)$ ステップとなる．

(3) $T < \log k$ かつ $T > \log_u N$ のとき

時間量は $O(kT + \log k\sqrt{N/rk})$ ステップである．時間量が最小となるのは $k = \frac{\sqrt[3]{N/r}\sqrt[3]{\log N^2}}{\sqrt[3]{T^2}}$ のときで，$\Omega(\sqrt[3]{T}\sqrt[3]{N/r}\sqrt[3]{\log N^2})$ ステップとなる．

(4) $T < \log k$ かつ $T < \log_u N$ のとき

時間量は $O(k \log_u N + \log k \sqrt{N/rk})$ ステップである. 時間量が最小となるのは,

$$k' \log_u N = \sqrt{N/rk'} \log k' \quad (1)$$

を満たす k' のときである.

この k' を直接求めることは困難なので,

$$k'' \log_u N = \sqrt{N/rk''} \log N$$

を満たす k'', すなわち $k'' = \sqrt[3]{(N \log^2 N)/(r \log_u^2 N)}$ について考える. この k'' を式 1 の左辺の k' に代入すると以下を得る.

$$O(k' \log_u N) = O(\sqrt[3]{\frac{N \log^2 N \log_u N}{r}}) \quad (2)$$

k'' を式 1 の右辺の k' に代入すると以下を得る.

$$O(\sqrt{N/(rk')} \log k') = O(\sqrt[3]{\frac{N \log^2 N \log_u N}{r}}) \quad (3)$$

式 (2)(3) より, $k'' = \sqrt[3]{(N \log^2 N)/(r \log_u^2 N)}$ のときに時間量は最小となり, $O(\sqrt[3]{\frac{N \log^3 N}{r \log u}})$ となる.

	merge sort	radix sort $T > \log_u N$	radix sort $T < \log_u N$	backet sort
$T \geq \log k$				
$k < \sqrt[3]{N/r}$	$O(T\sqrt{N/rk})$	$O(T\sqrt{N/rk})$	$O(T\sqrt{N/rk})$	$O(T\sqrt{N/rk})$
$k \geq \sqrt[3]{N/r}$	$O(kT)$	$O(kT)$		$O(kT)$
$k > \sqrt[3]{\frac{T^2 N}{r \log_u^2 N}}$			$O(k \log_u N)$	
$T < \log k$				
$k < \sqrt[3]{N/r}$	$O(\sqrt{N/rk} \log k)$	$O(\sqrt{N/rk} \log k)$	$O(\sqrt{N/rk} \log k)$	$O(T\sqrt{N/rk})$
$k \geq \sqrt[3]{N/r}$	$O(k \log k)$			$O(kT)$
$k > \sqrt[3]{\frac{N \log^2 N}{r \log_u^2 N}}$			$O(k \log_u N)$	
$k > \sqrt[3]{\frac{N \log^2 N}{r T^2}}$		$O(kT)$		

表 1: 時間量の比較

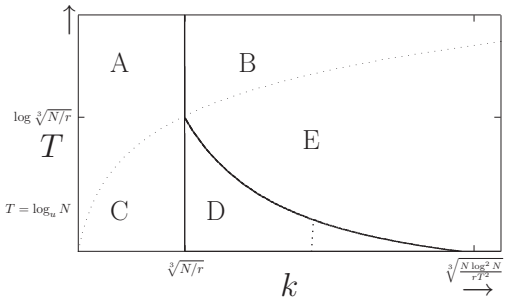

図 1: 最適な k の選択方法

4.3 バケットソートを用いた場合

バケットソートを用いた場合は, ステップ 2 で $O(kT + k) = O(kT)$ ステップ必要であり , ステップ 4 で $O(T\sqrt{N/rk})$ ステップ必要である. したがって, 全体で $O(T(\sqrt{N/rk} + k))$ ステップが必要となる.

最も時間量が小さくなるのは, $k = \sqrt[3]{N/r}$ としたときで $O(T\sqrt[3]{N/r})$ ステップとなる.

4.4 最適なアルゴリズムの選択

以上の事をまとめると表 1 のようになる.

時間量は, ソーティングの時間量と T の関係, ソーティングの時間量と Grover アルゴリズムの時間量の関係で場合分けされる.

底 u を $O(1)$ と取ったときはマージソートを使用したときと同じ時間量になる. 底 u を $O(N)$ と取ったときはソートに必要な時間はバケットソートと同じ時間量になるが, その後のバイナリサーチに必要な時間量が異なるため, 全体の時間量は一致しない. 底 u の値の取り方によって, 基数ソートの時間量は変化する.

使用できる領域量が事前にわかっているならば, マージソートの代わりに基数ソートを用いることにより, 時間量を少なくすることができる.

また, $T < \log_u N$ かつ $T > \log k$ という状況は起こり得ない. $T < \log k$ のときのみ, 基数ソートの時間量は u の値によって変化する. もし $T < \log_u N$ ならば, u を大きく取れば時間量を減らすことができる. $\log k > T > \log_u N$ である場合は, 時間量は u に影響されない.

図 1 において, マージソート, バケットソートを用いた場合における最適な k の選択を表しているのは $k = \sqrt[3]{N/r}$ (図中, 黒細線) である. そして, 基数ソートを用いた場合における最適な k の選択を表しているのは $T = \frac{\sqrt[3]{N/r} \sqrt[3]{\log N^2}}{\sqrt[3]{T^2}}$ である (図中, 太黒線). この線により近い k を取ることで, より高速にアルゴリズムは実行される.

たとえば $T = \log \sqrt[4]{N}$ のとき, 領域量が無限に取れるのであれば, 図中の各線上の k を選択するのが良い選択である. 一方, 領域が $10\sqrt[3]{N/r}$ しか利用することができないのであれば, マージソートと基数ソートを用いる場合は曲線上の k (つまり, $k = \sqrt[3]{N/r}$) を選択するのがもっとも良く, 基数ソートを用いた場合は曲線に最も近い k (つまり, $k = 10\sqrt[3]{N/r}$) と選択するのが良い.

また, $k = \sqrt[3]{N/r}$ におけるマージソートを用いた場合と基数ソートを用いた場合の時間計算量は一致する. つまり, 領域量が $\sqrt[3]{N/r}$ よりも大きいのであれば, マージソートを用いるよりも基数ソートを用いた方が時間計算量を減らすことができると考えられる.

	merge sort	radix sort		bucket sort
		$T > \log_u N$	$T < \log_u N$	
$T \geq \log k$				
$k < \sqrt[3]{N/r}$	$O(T^2N)$	$O(T^2uN/(rk))$	$O(T^2uN/(rk))$	$O(T^2N)$
$k \geq \sqrt[3]{N/r}$	$O(k^3T^2)$	$O(k^2T^2u)$		$O(k^3T^2)$
$\sqrt[3]{\frac{T^2N}{r\log_u^2 N}}$			$O(uk^2\log_u^2 N)$	
$T < \log k$				
$k < \sqrt[3]{N/r}$	$O(N\log^2 k)$	$O(N\log^2 ku/(rk))$	$O(N\log^2 ku/(rk))$	$O(T^2N)$
$k \geq \sqrt[3]{N/r}$	$O(k^3\log^2 k)$			$O(k^3T^2)$
$k > \frac{\sqrt[3]{N/r}\sqrt[3]{\log N^2}}{\sqrt[3]{T^2}}$		$O(k^2T^2u)$		
$k > \sqrt[3]{\frac{N\log^2 N}{r\log_u^2 N}}$			$O(uk^2\log_u^2 N)$	

表 2: $k < u$ の場合の St^2 のオーダー

	merge sort	radix sort		bucket sort
		$T > \log_u N$	$T < \log_u N$	
$T \geq \log k$				
$k < \sqrt[3]{N/r}$	$O(T^2N/r)$	$O(T^2N/r)$	$O(T^2N/r)$	$O(T^2N/r)$
$k \geq \sqrt[3]{N/r}$	$O(k^3T^2)$	$O(k^3T^2)$		$O(k^3T^2)$
$\sqrt[3]{\frac{T^2N}{r\log_u^2 N}}$			$O(k^3\log_u^2 N)$	
$T < \log k$				
$k < \sqrt[3]{N/r}$	$O(N\log^2 k/r)$	$O(N\log^2 k/r)$	$O(N\log^2 k/r)$	$O(T^2N/r)$
$k \geq \sqrt[3]{N/r}$	$O(k^3\log^2 k)$			$O(k^3T^2)$
$k > \frac{\sqrt[3]{N/r}\sqrt[3]{\log N^2}}{\sqrt[3]{T^2}}$		$O(k^3T^2)$		
$k > \sqrt[3]{\frac{N\log^2 N}{r\log_u^2 N}}$			$O(k^3\log_u^2 N)$	

表 3: $k > u$ の場合の St^2 のオーダー.

5 時間量と領域量のトレードオフ

Brassard らの量子アルゴリズムの, 時間量と領域量の関係を調べるため, St^2 の値を評価する. ここで, t はアルゴリズムの時間量, S は領域量である.

結果をまとめると, 表 2, 3 のようになる. 基数ソートを用いた場合は, K の要素数と 底 u のどちらが領域量に効いてくるかによって場合分けされる.

基数ソートを用いたときもマージソートを用いたときも, k の増加に従って St^2 の値は増加していく. 最小値は $k = 1$ のときに得られる. マージソートと基数ソートの間で, 最適な点における St^2 のオーダーを比べた場合, 基数ソートを用いた方が大きくなる. つまり, 基数ソートを用いることで時間計算量を減少させることはできるが, その分 St^2 の値は大きくなる.

6 衝突問題と電子署名

電子署名では, 以下のような手続きによって文章の正当性が確かめられる.

1. アリスからボブに文章を送るものとする. アリスは文章を関数によってハッシュし, 得られたハッシュ値を秘密鍵で暗号化することにより, 電子署名を作成する.
2. アリスはボブにテキストと電子署名を送る.
3. ボブは電子署名を公開鍵で復号化し, テキストをハッシュして得られた値と比較し, 文章の正当性を確かめる.

MD5 暗号などの電子書名の場合, 入力メッセージがどのような長さだろうと, 128bit の長さの署名が作成される. そこで, 任意の長さの入力と, 関数はどのような入力も定数長の長さの出力に変換する縮小関数がブラックボックスとして与えられたときに, 衝突を発見することを考える.

この問題に対して, 第 4 節の結果を適用して考えてみる. 実際の暗号で使用されるハッシュ関数には, 入力のそれぞれの bit が, 互いに干渉しあうことが要求される. したがって, 長さ n のメッセージを暗号化するには, $O(n^2)$ から $O(n^3)$ ステップ程度の計算量が必要である. 重複度は, 128bit に縮むことから推測できる. 例えば, 129 bit 長のメッセージの署名は, 平均的に見て 2-to-one であることが期待される. この場合は, どのソーティング法を用いたとしても, 最適な点での時間量に差は生じない. また, マージソートと基数ソートを用いた場合で St^2 のオーダーに差は生じない.

将来, $T < \log N$ を満たすハッシュ関数が発見された場合には, 本論で行った議論が, 暗号の分野で役に立つものと考えられる.

7 考察

本論では, 論文 [1] で示された, 衝突問題に対する量子アルゴリズムにおいて, ソーティング方法を変更することにより, 計算量全体にどのような変化が起きるかを考察した. そして, ある特定の条件下では, 基数ソートを用いることにより, 時間量のオーダーを減少させることが可能であるという結論を得た. 一方, 基数ソートを用いたとしても, 領域量を多く取らなければ時間量が減ることはなく, また基数ソートを用いると, 領域量と時間量のトレードオフの観点からは, あまり好ましくないこともわかった. 将来, Brassard らのアルゴリズムを実装することを考えると, より詳細な計算量評価が, 今後必要となるものと考えられる.

参考文献

[1] G.Brassard, P.Høyer and A.Tapp: "Quantum Algorithm for the Collision Problem", *quant-ph/*9705002 (1997).

[2] L.K.Grover: "Quantum Mechanics Helps in Searching for a Needle in a Haystack", *Physical Review Letters*, Vol.79, No.2, pp.325-328 (1997).

[3] Gilles Brassard, Peter Høyer, Michele Mosca, Alain Tapp: "Quantum Amplitude Amplification and Estimation" *quant-ph/*0005055 (2000).

[4] 西野哲朗 :「量子コンピュータの理論」, 培風館, 2002.