한국지능시스템학회 논문지 2010, Vol. 20, No. 3, pp. 428-433

효율적 In-Place Block Rotation 알고리즘과 복잡도 분석

An Efficient In-Place Block Rotation Algorithm and its Complexity Analysis

김복선[1] · 쿠츠너 아네[2]
Pok-Son Kim and Arne Kutzner

[1]국민대학교 수학과
E-mail: pskim@kookmin.ac.kr
[2]한양대학교 정보시스템학과
E-mail: kutzner@hanyang.ac.kr

요 약

u와 v를 두 인접수열 (consecutive sequence)이라고 했을 때 이때 “block rotation”이란 uv를 vu로 바꾸는 연산을 의미한다. 기존에 3개의 block rotation 알고리즘 즉 “BlockRotation", "Juggling" 그리고 "Reversal 알고리즘“이 소개되었는데 최근 우리는 하나의 새로운, QuickRotation 이라고 명명한 block rotation 알고리즘을 소개했다. 우리는 이 논문에서 QuickRotation 알고리즘을 이들 기존의 알고리즘들과 비교해 보이고자 한다. 벤치마킹 뿐만 아니라 복잡도 분석을 통한 비교를 통해 QuickRotation 알고리즘의 우수성을 증명해 보이고자 한다.

키워드 : 알고리즘 분석, 알고리즘의 복잡도, 기초 알고리즘, 블록로테이션.

Abstract

The notion “block rotation” denotes the operation of exchanging two consecutive sequences of elements uv to vu. There are three already well-known block rotation algorithms called BlockRotation, Juggling and Reversal algorithm. Recently we presented a novel block rotation algorithm called QuickRotation. In this paper we compare QuickRotation to these three known block rotation algorithms. This comparison covers a complexity analysis as well as benchmarking and shows that a switch to QuickRotation is almost always advantageous.

Key Words : Algorithm analysis, Algorithm complexity, Elementary algorithm, Block rotation.

1. 서 론

두 수열 $u = x_1 x_2 \cdots x_m$과 $v = y_1 y_2 \cdots y_n (n,m \geq 0)$에 대해 uv로부터 vu를 얻고자 하는 기초적인 문제를 “블록로테이션 (block rotation)"이라고 한다. u와 v 의 길이가 같을 경우 이 문제의 해를 우리는 "block swap"이라고 말한다.

지금까지 3개의 block rotation 알고리즘 (이하: 로테이션 알고리즘) 즉 “BlockRotation (BlockSwapRotation)", "Juggling" 그리고 "Reversal 알고리즘“이 소개되었다. BlockRotation 알고리즘은 재귀적 구조를 띄고 있으며 여러 번에 걸친 block swap의 적용을 통해 문제를 해결한다. 이 알고리즘의 implementation이 C++ Standard Template Library (STL)에 속해 있다. 가장 간단한 구조를 취하고 있는 "Reversal 알고리즘“은 Trabb Pardo [5]에 의해 소개되었으며 3번에 걸쳐 reversal 을 적용해 uv에서 vu를 얻게 되는 $(uv \rightarrow u^r v \rightarrow u^r v^r \rightarrow (u^r v^r)^r = vu)$ 방식을 취한다. "juggling approach“에 기초한 Juggling 알고리즘은 최대공약수 계산에 의존하며 Dudzinski-Dydek에 의해 최초로 소개 되었다 [2]. 기존의 세 알고리즘에 대한 벤치마킹을 포함한 Pseudocode는 [1]에서 소개되고 있다. 최근 우리가 추가로 "floating hole“기술에 기반한 새로운 블록로테이션 알고리즘을 [3]에서 소개했으며 이 알고리즘을 "QuickRotation “이라 명명하고자 한다.

길이 m의 세 수열이 $4m$에 해당하는 assignments횟수를 소비 하며 순환방식으로 수열의 위치 (position)를 바꿔가는 floating hole기술의 활용을 통해 QuickRotation이 기존의 모든 알고리즘과는 차별화 되는 우수성을 지닐 수 있게 되었다. 이 논문에서 우리는 복잡도 분석을 통해 QuickRotation이 BlockRotation과 Reversal 알고리즘보다 더 적은 assignments 횟수를 요구함을 증명해 보이고자 한다. 또한 벤치마킹을 통해 runtime과 관련해 QuickRotation이 기존 모든 알고리즘 보다 더 적은 (가장 적은) 시간을

접수일자 : 2010년 4월 3일
완료일자 : 2010년 5월 16일
감사의 글 : 본 논문은 본 학회 2010년도 춘계학대회에서 선정된 우수논문입니다.
이 논문은 2008년 정부(교육과학기술부)의 재원으로 한국학술진흥재단 (KRF-2008-531-D00020)과 2009년도 국민대학교 교내연구비 지원을 받아 수행된 연구임.

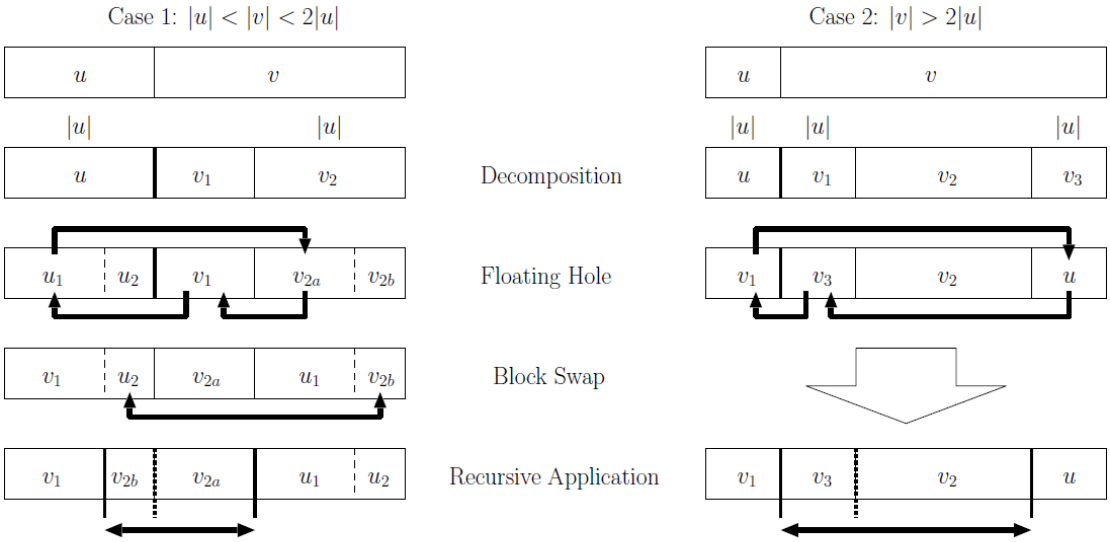

그림 1. QuickRotation 알고리즘의 그래프를 통한 묘사
Fig. 1. Graphical description of the QUICKROTATION algorithm

필요로 함을 보이고자 한다. Juggling 알고리즘은 비록 필요로 하는 assignments 횟수와 관련해 이론적으로 가장 효율적인 알고리즘에 해당하나 실제에 있어서 요구하는 run-time과 관련해 4개의 알고리즘 중 가장 비 효율적인 알고리즘에 해당하는 이례적 사실을 실험결과로 소개한다.

2. QuickRotation 알고리즘

QuickRotation 알고리즘은 Fig 1과 같이 묘사될 수 있으며 또한 다음과 같이 정의된다[3].

QuickRotation Algorithm 의 정의

u와 v를 배열된 두 인접수열 (consecutive sequence)이라고 하자. uv를 vu로 아래와 같이 바꾼다:

(1) If $|u|=0$ or $|v|=0$, then finish.

(2) If $|u|<|v|<2|u|$, then

(a1) Decompose v into v_1v_2, so that $|u|=|v_2|$.

(a2) Decompose u and v_2 into u_1u_2 and $v_{1a}v_{2b}$ respectively, so that $|u_1|=|v_1|=|v_{2a}|$ and $|u_2|=|v_{2b}|$.

(a3) Exchange the elements of u_1, v_1, and v_{2a} circularly using the floating hole technique so that you reorganize $u_1u_2v_1v_{2a}v_{2b}$ into $v_1u_2v_{2a}u_1v_{2b}$.

(a4) Swap u_2 and v_{2b} to change $v_1u_2v_{2a}u_1v_{2b}$ into $v_1v_{2b}v_{2a}u_1u_2$.

(a5) Apply the algorithm recursively on v_{2b} and v_{2a}.

(3) If $|v|\geq 2|u|$, then

(b1) Decompose v into $v_1v_2v_3$, so that $|u|=|v_1|=|v_3|$.

(b2) Apply floating hole to u and v_1, and v_3 to-change $uv_1v_2v_3$ into $v_1v_3v_2u$.

(b3) Recur on v_3 and v_2.

(4) If $|v|<|u|<2|v|$, then apply (2) to u and v symmetrically in the same way.

(5) If $|u|\geq 2|v|$, then apply (3) to u and v symmetrically in the same way.

QuickRotation의 Pseudocode Implementation

앞의 알고리즘의 Pseudocode Implementation은 Algorithm 1과 같이 묘사되며 이러한 표기 방식은 [6]을 따랐다.

3. 복잡도

(u,v)를 초기 recursion level 0에서의 입력수열이라고 하자. 그리고 다른 언급이 없는 한 $m=|u|\geq 0$, $n=|v|\geq 0, m\leq n$라고 하자. $m=0$ 또는 $n=0$일 경우 QuickRotation 알고리즘은 멈춘다. $m\neq 0, n\neq 0$ 그리고 (u',v')를 recursion level 1의 입력수열이라고 하자. 이제 QuickRotation 알고리즘이 요구하는 assignments 횟수를 계산하는 재귀식을 정의하고자 한다. 재귀식을 정의할 수 있기 위해 다음과 같이 3가지의 경우를 구별 한다:

경우 1: $n=m$
QuickRotation은 $3m$ assignments를 요구하며 알고리즘은 멈춘다.

경우 2: $m<n<2m$ 이 경우 아래의 세 가지 사항이 성립 한다:

1a) $|u'|=m-(n-m)=2m-n, |v'|=n-m$.

1b) 등식 $|u'|+|v'|=2m-n+n-m=m$으로부터 $(n+m)-m=n$개의 원소가 자신들의 마지막 자리 (로테이션이 된 후의 자리)로 이동됨을 알 수 있다.

1c) 요구되는 assignments 횟수는 $4(n-m)+3(m-(n-m))=n+2m$이다.

경우 3: $n\geq 2m$
이 경우 아래의 세 가지 사항이 성립 한다:

2a) $|u'|=m, |v'|=n-2m$.

2b) 등식 $|u'|+|v'|=m+n-2m=n-m$으로 부터 $2m$개의 원소가 자신들의 마지막 자리로 이동됨을 알 수 있다.

2c) 요구되는 assignments 횟수는 $4m$이다.

Algorithm 1 QuickRotation Algorithm

```
QuickRotation(A, s, l, r)
1  ▷ u is in A[s : s+l-1], v is in A[s+l : s+l+r-1]
2  while l > 0 and r > 0
3        do if l < r
4              then δ ← r − l
5                   if l > δ
6                     then FloatingHole(A, s, s+l, s+r, δ)
7                          s ← s+δ, l ← l−δ
8                          SwapRanges(A, s, s+r, l)
9                          r ← δ
10                    else FloatingHole(A, s, s+l, s+r, l)
11                         s ← s+l, r ← δ−l
12             else δ ← l − r
13                  if r > δ
14                    then s ← s+r
15                         FloatingHole(A, s−δ, s+r, s, δ)
16                         SwapRanges(A, s−r, s+δ, r−δ)
17                         r ← r−δ, l ← δ
18                    else FloatingHole(A, s, s+l, s+δ, r)
19                         s ← s+r, l ← δ−r
```

명백히 recursion depth가 깊어짐에 따라 입력수열 (u',v')의 길이는 짧아진다. 그러므로 QuickRotation이 요구하는 assignments 횟수를 계산하는 재귀적 함수 $\rho(m,n)$를 다음과 같이 정의할 수 있다.

정의 1. QuickRotation 알고리즘이 요구하는 assignments 횟수를 계산하는 함수 $\rho(m,n)$, $m \geq 0$, $n \geq 0$ 는 다음과 같이 재귀적으로 정의 된다:

$$
\begin{aligned}
\rho(0,n) &= 0 \\
\rho(m,0) &= 0 \\
\rho(m,m) &= 3m \\
\rho(m,n) &= \rho(n,m) \quad \text{if } m > n \\
\rho(m,n) &= \begin{cases} n+2m+\rho(2m-n,n-m) & \text{if } m < n < 2m \\ 4m+\rho(m,n-2m) & \text{if } n \geq 2m \end{cases}
\end{aligned}
$$

앞으로 $m_k(n_k)$는 재귀의 높이 (recursion height) k에서 짧은 (긴) 입력수열의 길이를 나타낸다 (Fig. 2 참조). h를 uv를 vu로 로테이션 시키는데 있어서 요구되는 재귀적 call의 전체 횟수라면 재귀의 높이 h에서 $m_h = m$과 $n_h = n$이 된다. QuickRotation 알고리즘이 $O(m+n)$ assignments와 $O(1)$ extra space를 요구하는 알고리즘에 해당함을 증명하면 다음과 같다.

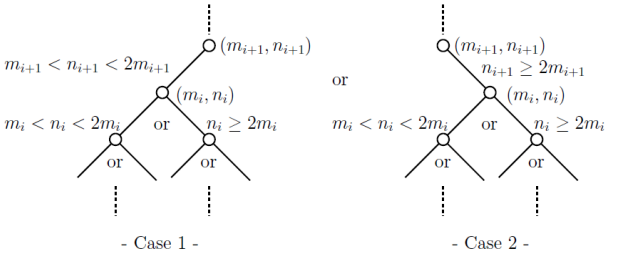

그림 2. 복잡도 계산 $\rho(m,n) < 2n+3m$의 설명

Fig. 2. Illustration of the Complexity Calculation $\rho(m,n) < 2n+3m$

정리 2. QuickRotation 알고리즘은 $2n+3m$ 보다 더 적은 횟수의 assignments를 요구 한다.

증명. recursion height에 관해 귀납적으로 증명해 보이고자 한다.

Induction base: $\rho(0,n)=0$, $\rho(m,0)=0$, $\rho(m,m)=3m$이므로 성립한다.

Induction step: QuickRotation 알고리즘이 (m_i,n_i)에 대해 $2n_i+3m_i$보다 더 적은 횟수의 assignments를 필요로 한다고 가정하자 즉 $\rho(m_i,n_i) < 2n_i+3m_i$ 라고 가정하자. 이 때 우리는 다음과 같은 두 경우를 나누어 각각의 경우에 대해 성립함을 보여야 한다 (Fig. 2 참조).

경우 1: $n_{i+1} < 2m_{i+1}$. 이 경우 아래의 등식이 성립한다.

$\rho(m_{i+1},n_{i+1}) = 2m_{i+1}+n_{i+1}+\rho(2m_{i+1}-n_{i+1},n_{i+1}-m_{i+1})$

그러면 $m_i = \min(2m_{i+1}-n_{i+1}, n_{i+1}-m_{i+1})$,

$n_i = \max(2m_{i+1}-n_{i+1}, n_{i+1}-m_{i+1})$,

$m_i+n_i = m_{i+1}, m_i \leq m_{i+1}$ 와 $n_i \leq m_{i+1}$ 가 성립한다.

가정 $\rho(m_i,n_i) < 2n_i+3m_i$에 의해 다음을 얻는다.

$$
\begin{aligned}
\rho(m_{i+1},n_{i+1}) &= 2m_{i+1}+n_{i+1}+\rho(m_i,n_i) \\
&< 2m_{i+1}+n_{i+1}+3m_i+2n_i \\
&= 4m_{i+1}+n_{i+1}+m_i \\
&= 4m_{i+1}+2n_{i+1}-m_{i+1} \\
&= 3m_{i+1}+2n_{i+1}
\end{aligned}
$$

위의 계산에서 $m_i = \min(2m_{i+1}-n_{i+1}, n_{i+1}-m_{i+1})$이므로 $m_i \leq n_{i+1}-m_{i+1}$이 성립한다.

경우 2: $n_{i+1} \geq 2m_{i+1}$. 이 경우

$\rho(m_{i+1},n_{i+1}) = 4m_{i+1}+\rho(m_{i+1},n_{i+1}-2m_{i+1})$이 성립한다. 그러면

$m_i = \min(m_{i+1}, n_{i+1}-2m_{i+1})$,

$n_i = \max(m_{i+1}, n_{i+1}-2m_{i+1})$,

$m_i+n_i = n_{i+1}-m_{i+1}, m_i \leq m_{i+1}$ 와 $m_{i+1} \leq n_i$

를 얻는다. 가정 $\rho(m_i,n_i) < 2n_i+3m_i$에 의해 다음이 성립한다.

$$
\begin{aligned}
\rho(m_{i+1},n_{i+1}) &< 4m_{i+1}+2n_i+3m_i \\
&= 4m_{i+1}+2n_{i+1}-2m_{i+1}+m_i \\
&\leq 2m_{i+1}+2n_{i+1}+m_i \\
&\leq 3m_{i+1}+2n_{i+1}
\end{aligned}
$$

앞의 증명의 경우 1로부터 다음의 결론을 얻을 수 있다.

따름정리 3. $m \leq n < 2m$라고 가정하자. 그러면 QuickRotation 알고리즘은 $5m+n$ 보다 더 적은 횟수의 assignments를 필요로 한다.

4. 다른 알고리즘과의 비교

기존에 이미 알려진 3개의 로테이션 알고리즘인 Block-Rotation, Juggling 그리고 Reversal 알고리즘의 복잡도와 QuickRotation 알고리즘의 복잡도를 이 단원에서 비교하고자 한다.

표 1. BlockSwap과 QuickRotation의 비교
Table 1. Comparison of BlockSwap and QuickRotation

알고리즘	요구되는 Assignments 횟수	제자리를 찾아가게 되는 원소의 개수
BlockSwap	$3m_i$ if $m_i = n_i$	$m_i + n_i$
	$3m_i$ otherwise	m_i
SpeedRotation	$3m_i$ if $m_i = n_i$	$m_i + n_i$
	$2m_i + n_i$ if $m_i < n_i < 2m_i$	n_i
	$4m_i$ if $n_i \geq 2m_i$	$2m_i$

※ 본 표에 주어진 각각의 값은 임의의 recursion level i 와 관련된 값이다. 즉 recursion level i에서 요구되는 assignments 횟수와 제자리를 찾아가게 되는 원소의 개수를 의미한다.

4.1 BlockRotation

BlockRotation 알고리즘은 다음과 같이 정의되어 질 수 있다:

(1) If $|u| = 0$ or $|v| = 0$, then finish.
(2) If $|u| < |v| < 2|u|$, then
  (a1) Decompose v into $v_1 v_2$, so that $|u| = |v_2|$.
  (a2) Swap the elements of u and v_2 to change uv_1v_2 into v_2v_1u.
  (a3) Apply the algorithm recursively on v_2 and v_1.
(3) If $|u| \geq |v|$, then apply (2) to u and v symmetrically in the same way.

QuickRotation이 기존의 알고리즘 중 최적알고리즘에 해당하는 BlockRotation보다도 평균적으로 20-30%정도 더 적은 runtime을 필요로 한다는 실험결과를 얻을 수 있었다 (Fig. 3 참조). 이 실험결과로부터 두 알고리즘 모두 재귀적 구조에 기반을 두고 있는 관계로 QuickRotation이 Block-Rotation 보다 더 적은 element assignments 횟수를 요구하리라는 추측을 할 수 있다. 그래서 임의의 recursion level i에서 두 알고리즘이 각각 요구하는 element assignments 횟수와 제자리를 이미 찾게 되는 원소의 수를 비교해 보았으며 그 결과 표 1 과 같은 사실을 알아낼 수 있었다. 표 1에서 $m = |u|, n = |v|, m \leq n$라고 했을 때 m_i와 n_i는 각각 임의의 recursion level i (initially $m_0 = m$ and $n_0 = n$)에서 짧은 그리고 긴 입력수열의 길이를 나타낸다. 표 1로부터 우리는 임의의 recursion level i에서 명백히 QuickRotation이 BlockRotation보다 assignments 횟수와 관련해 더 효율적인 (제자리를 찾게 되는 원소의 수에 대해 QuickRotation이 요구하는 assignments 횟수가 BlockRotation이 요구하는 assignments 횟수보다 같거나 더 적음) 알고리즘에 해당한다는 사실을 알 수 있다.

이제 더 나아가 QuickRotation의 우수성을 모든 input $(m,n), m > 0,\ n > 0, m < n$으로 일반화 시켜 보이고자, BlockRotation 알고리즘이 요구하는 assignments 횟수를 계산하는 재귀함수 $\eta(m,n)$을 아래와 같이 정의 한다:

정의 4. 함수 $\eta(m,n),\ m \geq 0,\ n \geq 0$ 은 다음과 같이 재귀적으로 정의 된다:

$$\begin{aligned}
\eta(0,n) &= 0 \\
\eta(m,0) &= 0 \\
\eta(m,m) &= 3m \\
\eta(m,n) &= \eta(n,m) \quad \text{if } m > n \\
\eta(m,n) &= 3m + \eta(m, n-m) \quad \text{if } m < n
\end{aligned}$$

이제 모든 $(m,n), m > 0, n > 0, m \neq n$ 에 대해 Quick-Rotation이 BlockRotation보다 더 적은 횟수의 assignments를 요구함을 증명하고자 한다.

정리 5. 모든 $(m,n), m > 0, n > 0, m < n$ 에 대해 $\rho(m,n) < \eta(m,n)$이 성립한다.
증명. $n = m + r, 0 < r < m$ 과 $n = qm + r, q > 1, 0 \leq r < m$ 인 두 경우로 나누어 정리가 성립함을 보이자.
경우 1: $n = m + r, 0 < r < m$.
이 경우 다음의 등식이 성립한다.

$$\begin{aligned}
\eta(m,n) &= 3m + \eta(m,r) \\
&= 3m + \eta(r,m) \\
&= 3m + 3r + \eta(r, m-r) \\
&= n + 2m + 2r + \eta(r, m-r)
\end{aligned}$$

$$\begin{aligned}
\rho(m,n) &= n + 2m + \rho(2m - n, n - m) \\
&= n + 2m + \rho(m - r, r) \\
&= n + 2m + \rho(r, m - r)
\end{aligned}$$

η 와 ρ가 재귀함수이고 더욱이 $n + 2m < n + 2m + 2r$, $r < m$, $m - r < n$이 성립하므로 $\rho(m,n) < \eta(m,n)$이다.
경우 2: $n = qm + r, q > 1, 0 \leq r < m$.
q가 짝수인 경우와 홀수인 경우로 나누어 성립함을 보이고자 한다.
경우 2.1: $n = 2km + r, k \geq 1, 0 \leq r < m$.
이 경우 아래의 등식이 성립한다.

$$\begin{aligned}
\eta(m,n) &= 3 \cdot 2km + \eta(m,r) \\
&= 3(n - r) + \eta(r,m)
\end{aligned}$$

$$\begin{aligned}\rho(m,n) &= 4m\cdot\lfloor n/2m\rfloor+\rho(m,n-\lfloor n/2m\rfloor\cdot 2m)\\ &= 4mk+\rho(m,n-2km)\\ &= 2(n-r)+\rho(m,r)\\ &= 2(n-r)+\rho(r,m)\end{aligned}$$

η 와 ρ가 재귀함수이고 더욱이
$2(n-r)<3(n-r), r<m, m<n$이 성립하므로
$\rho(m,n)<\eta(m,n)$이다.
경우 2.2: $n=(2k+1)m+r, k\geq 1, 0\leq r<m$.
이 경우 아래의 등식이 성립한다.

$$\begin{aligned}\eta(m,n) &= 3(2k+1)m+\eta(m,r)\\ &= n+4km+2m-r+\eta(r,m)\\ &= n+4km+2m+2r+\eta(r,m-r)\end{aligned}$$

$$\begin{aligned}\rho(m,n) &= 4m\cdot\lfloor n/2m\rfloor+\rho(m,n-\lfloor n/2m\rfloor\cdot 2m)\\ &= 4mk+\rho(m,m+r)\\ &= 4mk+m+r+2m+\rho(m-r,r)\\ &= n+2mk+2m+\rho(r,m-r)\end{aligned}$$

그러므로 $\rho(m,n)<\eta(m,n)$이다.

4.2 Reversal 알고리즘

Reversal 알고리즘이 요구하는 assignments 횟수를 함수 $\xi(m,n)$로 묘사하고자 하며 [4]에서 이 함수식이 모든 $m>0, n>0$에 대해 $\xi(m,n)=3(\lfloor(m+n)/2\rfloor+\lfloor n/2\rfloor+\lfloor m/2\rfloor)$이 됨이 소개 되었다. 이제 QuickRotation이 Reversal 알고리즘보다 더 적은 횟수의 assignments를 요구함을 증명하고자 한다.

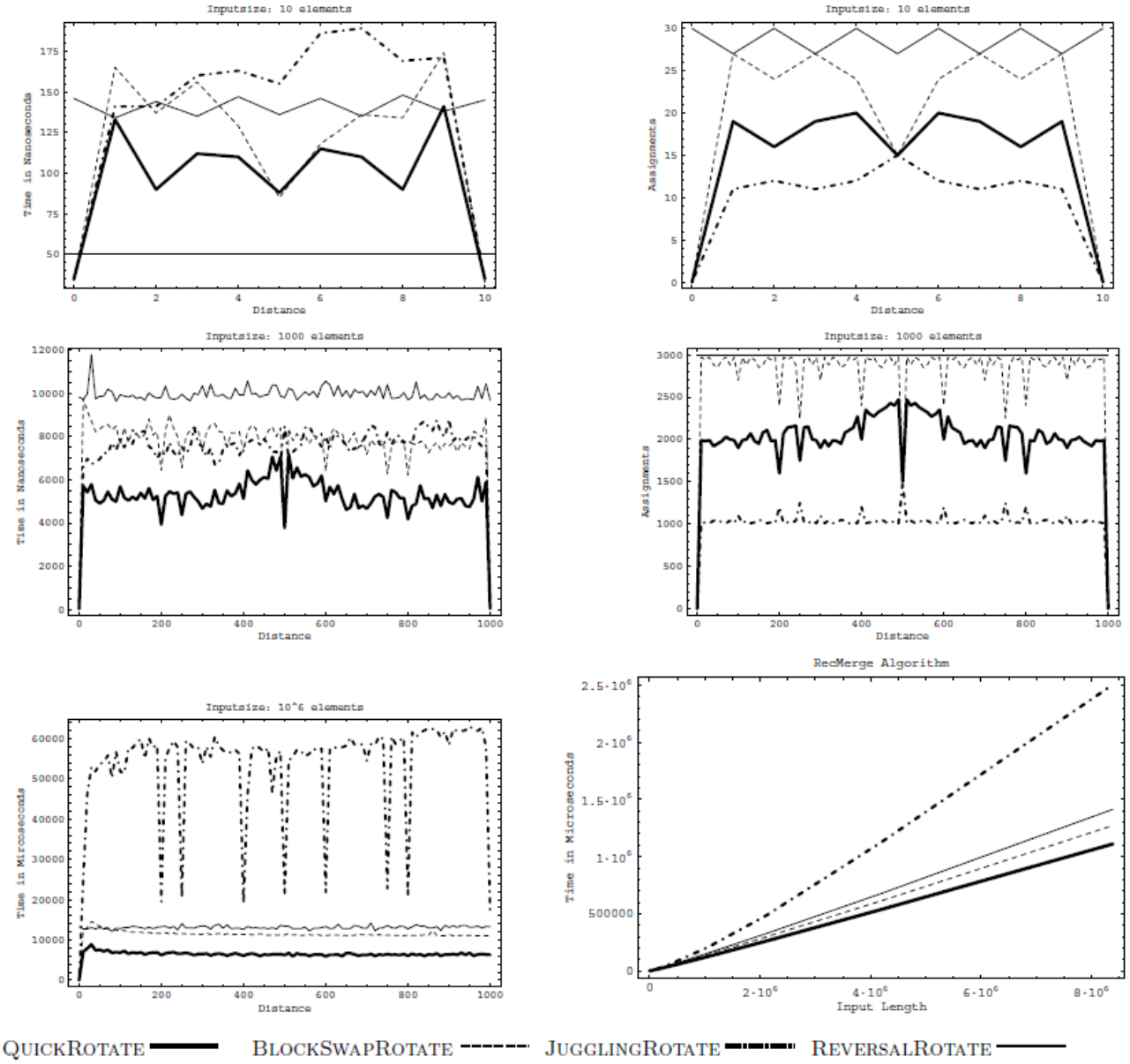

그림 3. 여러 block rotation 알고리즘들의 비교
Fig. 3. Comparison of several block rotation algorithms

정리 6. 모든 $(m,n), m \le n, n \ge 4$ 에 대해 $\rho(m,n) < \xi(m,n)$이 성립한다.
증명. m과 n이 짝수인 경우
$\xi(m,n) = 3(\lfloor (m+n)/2 \rfloor + \lfloor n/2 \rfloor + \lfloor m/2 \rfloor)$
$= 3m+3n$이 성립한다. 정리 2에 의해
$\rho(m,n) < 2n+3m$이므로 $\rho(m,n) < \xi(m,n)$를 만족한다.
모든 나머지 경우에 대해
$\xi(m,n) = 3(\lfloor (m+n)/2 \rfloor + \lfloor n/2 \rfloor + \lfloor m/2 \rfloor)$
$= 3m+3n-3$이 되고 그러므로 $n \ge 4$에 대해
$\rho(m,n) < 2n+3m < \xi(m,n)$이 성립한다.

정리 7. 모든 $(m,n), m \le n, 1 \le m \le 3, 2 \le n \le 3$ 에 대해 $\rho(m,n) < \xi(m,n)$이 성립한다.
증명. 각각의 경우에 대해 계산해 보므로 정리가 성립함을 쉽게 알 수 있다.

위의 두 정리로부터 아래의 따름정리가 성립한다.

따름정리 8.
모든 $(m,n), m \le n, m > 0, n > 0, (m,n) \ne (1,1)$ 에 대해 $\rho(m,n) < \xi(m,n)$이 성립한다.

4.3 Juggling 알고리즘

Juggling 알고리즘이 가장 적은 횟수의 assignments를 요구하는 사실이 [2]에서 소개 되었다. 이러한 이론적으로 옵티멀한 특성에도 불구하고 Juggling 알고리즘이 실제에 있어서 가장 오랜 runtime을 필요로 하는 알고리즘에 해당함을 다음 단원에 소개 하고자 한다. 이로부터 더 적은 횟수의 assignments 요구가 항상 더 적은 runtime을 요구하지는 않는다는 사실에 대한 하나의 반례로 Juggling 알고리즘을 들 수 있음을 알 수 있다.

5. 벤치마킹

QuickRotation 알고리즘에 대해 기존의 3개의 알고리즘과의 비교를 통해 벤치마킹 결과를 얻을 수 있었다. Figure 3에서 볼 수 있듯 QuickRotation 알고리즘이 4개의 알고리즘 중 runtime과 관련해 가장 효율적인 알고리즘에 해당함을 알 수 있다.

참 고 문 헌

[1] J. Bentley. *Programming Pearls*. Addison-Wesley, Inc, 2nd edition, 2000.
[2] K. Dudzinski and A. Dydek. "On a stable storage merging algorithm." *Information Processing Letters*, 12(1):58, February 1981.
[3] P. S. Kim and A. Kutzner, "An Improved Algorithmic Solution for the Problem of Block -Rotation." *In 10th International Symposium on Advanced Intelligent Systems*, pp. 161-164, Busan, Korea, August 17-19, 2009.
[4] van Leeuwen, J. "The Complexity of Data Organisation." *Mathematical Centre Tracts 81* (Mathematical Centre, Amsterdam), 1976.
[5] L. T. Pardo. "Stable sorting and merging with optimal space and time bounds," *SIAM Journal on Computing*, 6(2), pp. 351-372, 1977.
[6] T. Cormen, C. Leiserson, R. Rivest and C. Stein, *Introduction to Algorithms*, MIT Press, 2001.

저 자 소 개

김복선 (Pok-Son Kim)
한국지능시스템학회 협력이사
현재 국민대학교 수학과 부교수
제 18권 2호 참조

쿠츠너 아네 (Arne Kutzner)
1999년 : 독일 University of Frankfurt 컴퓨터과학부 공학박사
2001년 : 독일 Dresdner 은행
2002년 : DLogistics Korea 대표이사
2003년~2008년 : 서경대학교 전자상거래과 교수
2009년~현재 : 한양대학교 제II공대 정보 시스템학과 부교수

관심분야 : Algorithm, Algorithm Engineering, Programming Languages
E-mail : kutzner@hanyang.ac.kr