한국지능시스템학회 논문지 2008, Vol. 18, No. 2, pp. 272-277

Symmerge 알고리즘의 복잡도

Complexity of the Symmerge Algorithm

김복선
Pok-Son Kim

국민대학교 자연과학대학 수학과

요 약

$m \le n$을 만족하는 m과 n을 두 입력수열이라고 했을 때 Symmerge는 비교횟수와 관련해 복잡도 $O(m\log\frac{n}{m})$를 필요로 하는 stable minimum storage 머징 알고리즘이다. 그러므로 비교횟수와 관련된 머징의 점근적 하계 $\Omega(m\log\frac{n}{m})$에 의해 Symmerge 알고리즘은 최적 알고리즘에 해당함을 알 수 있다. Symmerge는 두 입력수열의 분할 (partition)과 로테이션 (rotation)을 통해 얻어지는 수열들에 알고리즘의 재귀적 콜 (recursive call)이 적용되는 divide 와 conquer 기술을 이용한다. 이로 인해 수열들이 반복해서 분할과 로테이션 되는데 특히 재귀의 깊이가 $m-1$ 가 되는 경우에 있어서 두 입력수열의 길이의 관계를 알아보고자 한다.

키워드 : stable 머징, 알고리즘 복잡도

Abstract

Symmerge is a stable minimum storage merging algorithm that needs $O(m\log\frac{n}{m})$ element comparisons, where m and n are the sizes of the input sequences with $m \le n$. Hence, according to the lower bound for merging, the algorithm is asymptotically optimal regarding the number of comparisons. The Symmerge algorithm is based on the standard recursive technique of "divide and conquer". The objective of this paper is to consider the relationship between m and n for the degenerated case where the recursion depth reaches $m-1$.

Key Words : stable merging, algorithm complexity

1. 서 론

Merging [4,5] denotes the operation of rearranging the elements of two adjacent sorted sequences of sizes m and n so that the result forms one sorted sequence of $m+n$ elements. An algorithm merges two adjacent sequences with minimum storage when it needs $O(\log^2(m+n))$bits additional space at most. This form of merging represents a weakened form of in-place merging and allows the usage of a stack that is logarithmically bounded in $m+n$. Minimum storage merging is sometimes also referred to as in situ merging. A merging algorithm is regarded as stable, if it preserves the initial ordering of elements with equal value.Some lower bounds for merging have been proven so far. The lower bound for the number of assignments is $m+n$, because every element may change its position in the sorted result. The lower bound for the number of comparisons is $\Omega(m\log\frac{n}{m})$ for $m \le n$. This can be proven by a combinatorial inspection combined with an argumentation using decision trees. An accurate presentation of these bounds is given by Knuth [1].

A minimum storage merging algorithm was proposed by Dudzinski and Dydek [2] in 1981. They presented a divide and conquer algorithm that is asymptotically optimal regarding the number of comparisons but nonlinear regarding the number of assignments. In 2004 Kim and Kutzner [3] also presented a new stable minimum storage merging algorithm performing $O(m\log\frac{n}{m})$ comparisons and $O((m+n)\log m)$ assignments for two sequences of size m and n $m \le n$. This algorithm is based on a simple strategy of symmetric comparisons, which will be explained in detail by an example. In complexity-analysis we realized that the algorithm may reach the recursion level $m-1$. The objective of this paper is to consider the relationship be-

접수일자 : 2007년 10월 15일
완료일자 : 2008년 1월 30일
This work was supported by the Kookmin University research grant in 2007

tween m and n for the degenerated case where the recursion depth reaches $m-1$.

2. The Symmerge Algorithm

We start with a brief introduction of the Symmerge algorithm [3]. Let us assume that we have to merge the two sequences u=(0,2,5,9) and v=(1,4,7,8).

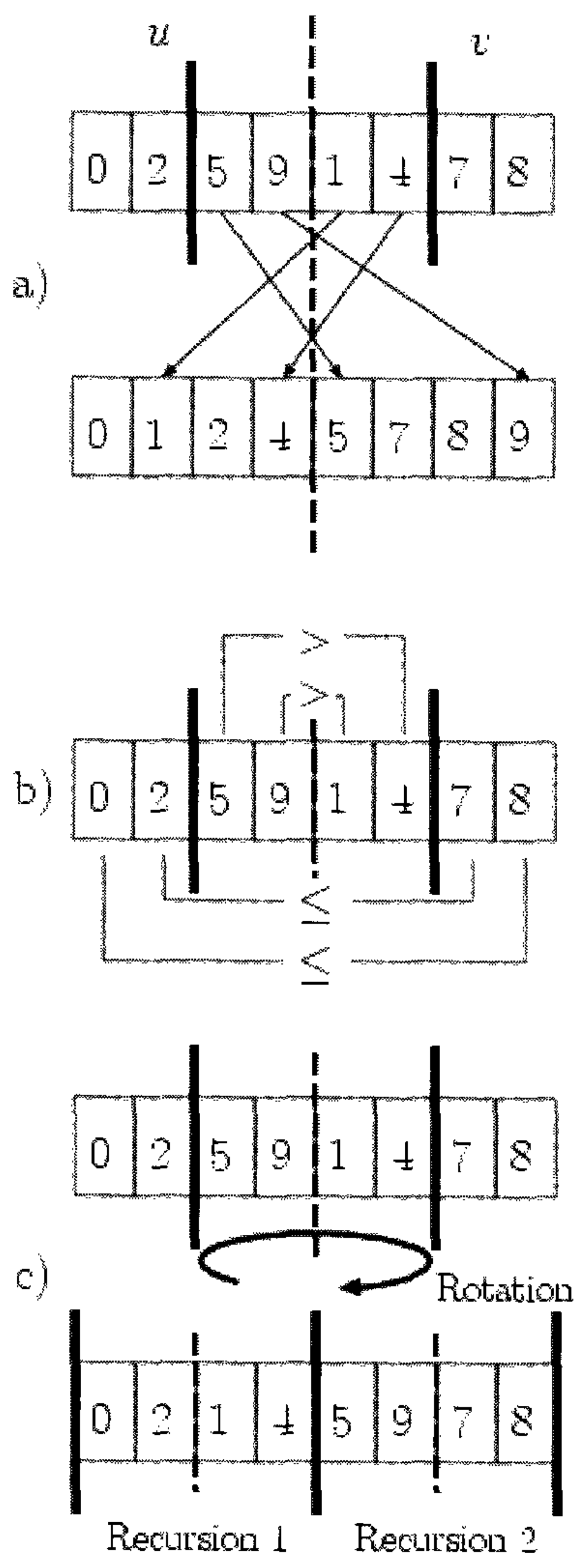


Fig. 1. Symmerge example

When we compare the input with the sorted result, we can see that in the result the last two elements of u occur on positions belonging to v, and the first two elements of v appear on positions belonging to u (see Fig. 1). So, 2 elements should be exchanged between u and v. The kernel of the algorithm is to compute this number of side-changing elements efficiently and then to exchange such a number of elements. This number can be determined by a process of symmetrical comparisons of elements that happens according to the following principle:We start at the leftmost element in u and at the rightmost element in v and compare the elements at these positions. We continue doing so by *symmetrically comparing* element-pairs from the outsides to the middle. Fig. 1 shows the resulting pattern of mutual comparisons for the example. There can occur at most one position, where the relation between the compared elements alters from 'not greater' to 'greater'. In Figure 1 two thick lines mark this position. These thick lines determine the number of side-changing elements as well as the bounds for the rotation. Then by recursive application of this technique to the arising subsequences we get a sorted result. Due to this technique of *symmetric comparisons* the algorithm has been called Symmerge.So far we introduced the computation of the number of side-changing elements as linear process of symmetric comparisons. But this computation may also happen in the style of a binary search. Then only $\lfloor \log(\min(|u|,|v|)) \rfloor + 1$ comparisons are necessary to compute the number of side-changing elements.

The input lengths $|u|$ and $|v|$ of the above example are equal. However, if the lengths of u and v are different, for example $u < v$, we decompose the longer sequence v into three parts v_1wv_2 so that the middle part w has the same size as the shorter sequence u. Applying the technique of symmetric comparisons described above to u and w, the bounds for the rotation are determined.

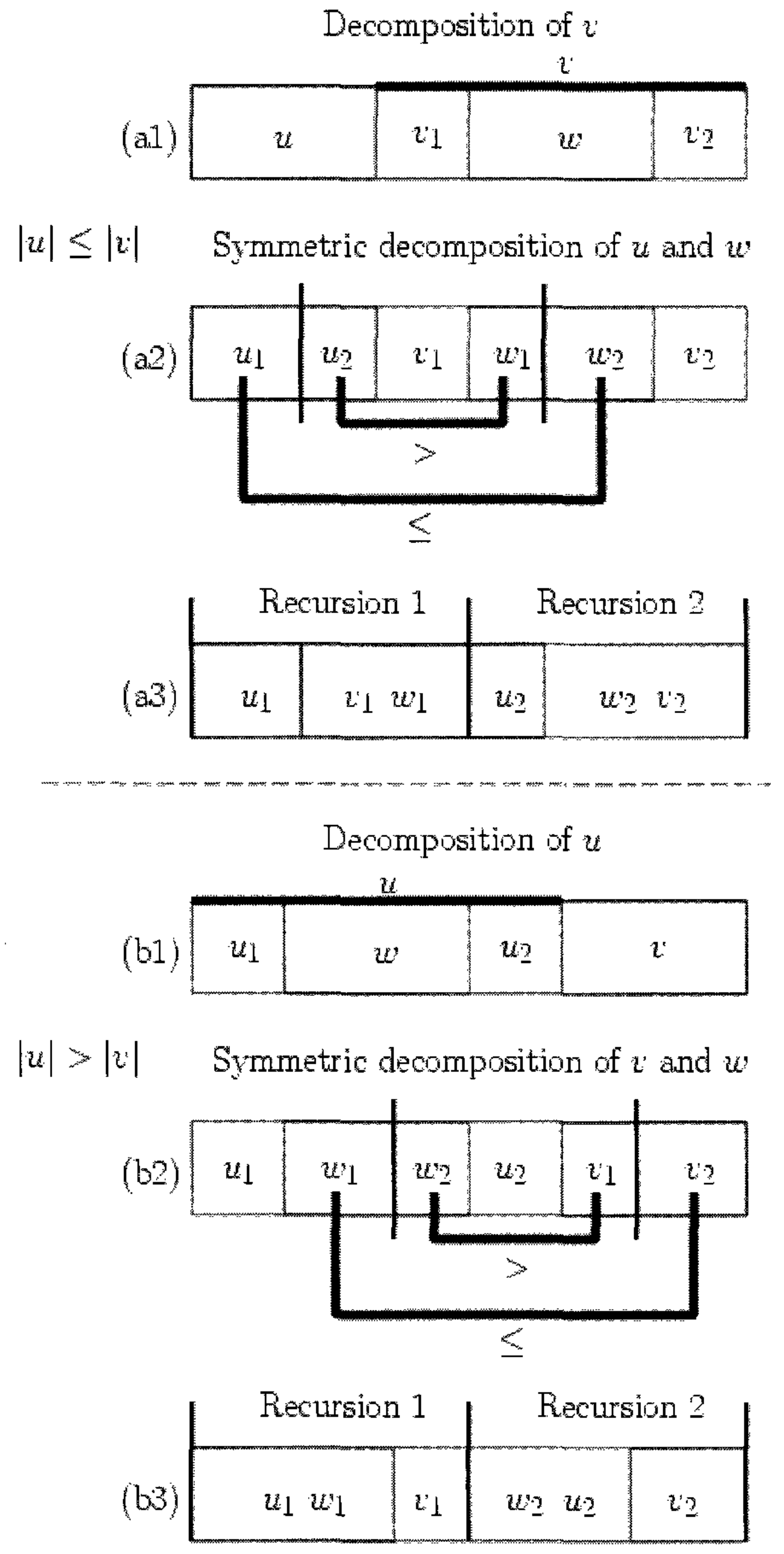
2.1 Formal Definition

Let u and v be two adjacent ascending sorted sequences. We define $u \le v$ ($u < v$) iff. $x \le y$ ($x < y$) for all elements $x \in u$ and for all elements $y \in v$. We merge u and v as follows:If $|u| \le |v|$, then(a1) we decompose v into v_1wv_2 such that $|w| = |u|$ and either $|v_2| = |v_1|$ or $|v_2| = |v_1| + 1$.(a2) we decompose u into u_1u_2 ($|u_1| \ge 0, |u_2| \ge 0$) and w into w_1w_2 ($|u_1| \ge 0, |u_2| \ge 0$) such that $|u_1| = |w_2|$, $|u_2| = |w_1|$ and $u_1 \le w_2$, $u_2 > w_1$.(a3) we recursively merge u_1 with v_1w_1 as well as u_2 with w_2v_2. Let u' and v' be the resulting sequences, respectively.

else

(b1) we decompose u into u_1wu_2 such that $|w| = |u|$ and either $|u_2| = |u_1|$ or $|u_2| = |u_1| + 1$.(b2) we decompose v into v_1v_2 ($|v_1| \ge 0, |v_2| \ge 0$) and w into w_1w_2 ($|w_1| \ge 0, |w_2| \ge 0$) such that $|v_1| = |w_2|$, $|v_2| = |w_1|$ and $w_1 \le v_2$, $w_2 > v_1$.(b3) we recursively merge u_1w_1 with v_1 as well as w_2u_2 with v_2. Let u' and v' be the resulting sequences, respectively.

$u'v'$ then contains all elements of u and v in sorted order.



Decomposition of v
(a1) u v_1 w v_2
$|u| \le |v|$ Symmetric decomposition of u and w
(a2) u_1 u_2 v_1 w_1 w_2 v_2
$>$
$\le$
Recursion 1 Recursion 2
(a3) u_1 $v_1\ w_1$ u_2 $w_2\ v_2$
Decomposition of u
(b1) u_1 w u_2 v
$|u| > |v|$ Symmetric decomposition of v and w
(b2) u_1 w_1 w_2 u_2 v_1 v_2
$>$
$\le$
Recursion 1 Recursion 2
(b3) $u_1\ w_1$ v_1 $w_2\ u_2$ v_2

Fig. 2. Illustration of Symmerge

Fig. 2. contains an accompanying graphical description of the process described above. The steps (a1) and (b1) manage the situation of input sequences of different length by cutting a subsection w in the middle of the longer sequence as ''active area''. This active area has the same size as the shorter of either input sequences. The decomposition formulated by the steps (a2) and (b2) can be achieved efficiently by applying the principle of the symmetric comparisons between the shorter sequence u (or v) and the active area w. After the decomposition step (a2) (or (b2)), the subsequence $u_2v_1w_1$(or $w_2u_2v_1$) is rotated so that we get the sub-sequences $u_1v_1w_1$ and $u_2w_2v_2$($u_1w_1v_1$ and $w_2u_2v_2$). The treatment of pairs of equal elements as part of the ''outer blocks'' (u_1,w_2 in (a2) and w_1,v_2 in (b2)) avoids the exchange of equal elements and so any reordering of these.

It is obvious that Symmerge is stable because of the decomposition condition $u_1 \le w_2$, $u_2 > w_1$ (or $w_1 \le v_2$, $w_2 > v_1$).

3. Complexity

3.1 Worst Case Complexity

Unless stated otherwise, let us denote $m = |u|$, $n = |v|$, $m \le n$ $k = \lfloor \log m \rfloor$. Further let m_j^i and n_j^i denote the minimum and maximum of lengths of sequences merging on the ith recursion group for $i = 0,1,2,\cdots,k$ and $j = 1,2,\cdots,2^i$ (initially $m_1^0 = m$ and $n_1^0 = n$). A recursion group consists of one or several recursion levels and comprises $2^i(i = 0,1,\cdots,k)$ subsequence mergings at most (see figure 3). In the special case where each subsequence merging always triggers two nonempty recursive calls - in this case the recursion depth becomes exactly $k = \lfloor \log m \rfloor$ -, recursive groups and recursive levels are identical, but in general for the recursion depth dp it holds $k = \lfloor \log m \rfloor \le dp \le m$. Further, for each recursion group $i = 0,1,2,\cdots,k$, it holds $\sum_{j=1}^{2^i}(m_j^i + n_j^i) \le m + n$. The worst case complexity of Symmerge regarding the number of comparisons and assignments are given in [3] as follows:

Lemma 1. If $k = \sum_{j=1}^{2^i} k_j$ for any $k_j > 0$ and integer $i \ge 0$, then $\sum_{j=1}^{2^i} \log k_j \le 2^i \log(k/2^i)$.

Proof. We show the lemma through induction on i.Induction base: If $i = 1$, the followings hold:

$$\sum_{j=1}^{2} \log k_j = \log k_1 + \log k_2$$

$$\log \frac{k}{2} = \log \frac{k_1 + k_2}{2}$$

Since log-function is concave,
$\frac{1}{2}\sum_{j=1}^{2} \log k_j \le \log(\frac{k_1 + k_2}{2})$ Thus $\sum_{j=1}^{2} \log k_j \le 2\log(\frac{k}{2})$

Induction step: Assuming that the result holds for i, we can confirm its validity for $i+1$ as follows:

$$k = \sum_{j=1}^{2^{i+1}} k_j = \sum_{j=1}^{2^i} k_j + \sum_{j=2^i+1}^{2^{i+1}} k_j$$

Let T_1 be $\sum_{j=1}^{2^i} k_j$ and let T_2 be $\sum_{j=2^i+1}^{2^{i+1}} k_j$.

Then

$$\sum_{j=1}^{2^{i+1}} \log k_j = \sum_{j=1}^{2^i} \log k_j + \sum_{j=2^i+1}^{2^{i+1}} \log k_j$$

$\le 2^i\log\frac{T_1}{2^i}+2^i\log\frac{T_2}{2^i}$, by our inductive assumption. By applying our inductive assumption once more, we obtain the result.

$$\sum_{j=1}^{2^{i+1}}\log k_j \le 2^i\,(\log\frac{T_1}{2^i}+\log\frac{T_2}{2^i})$$

$$\le 2^i\,(2\log\frac{\frac{T_1+T_2}{2^i}}{2}) = 2^{i+1}\log\frac{k}{2^{i+1}} \qquad \square$$

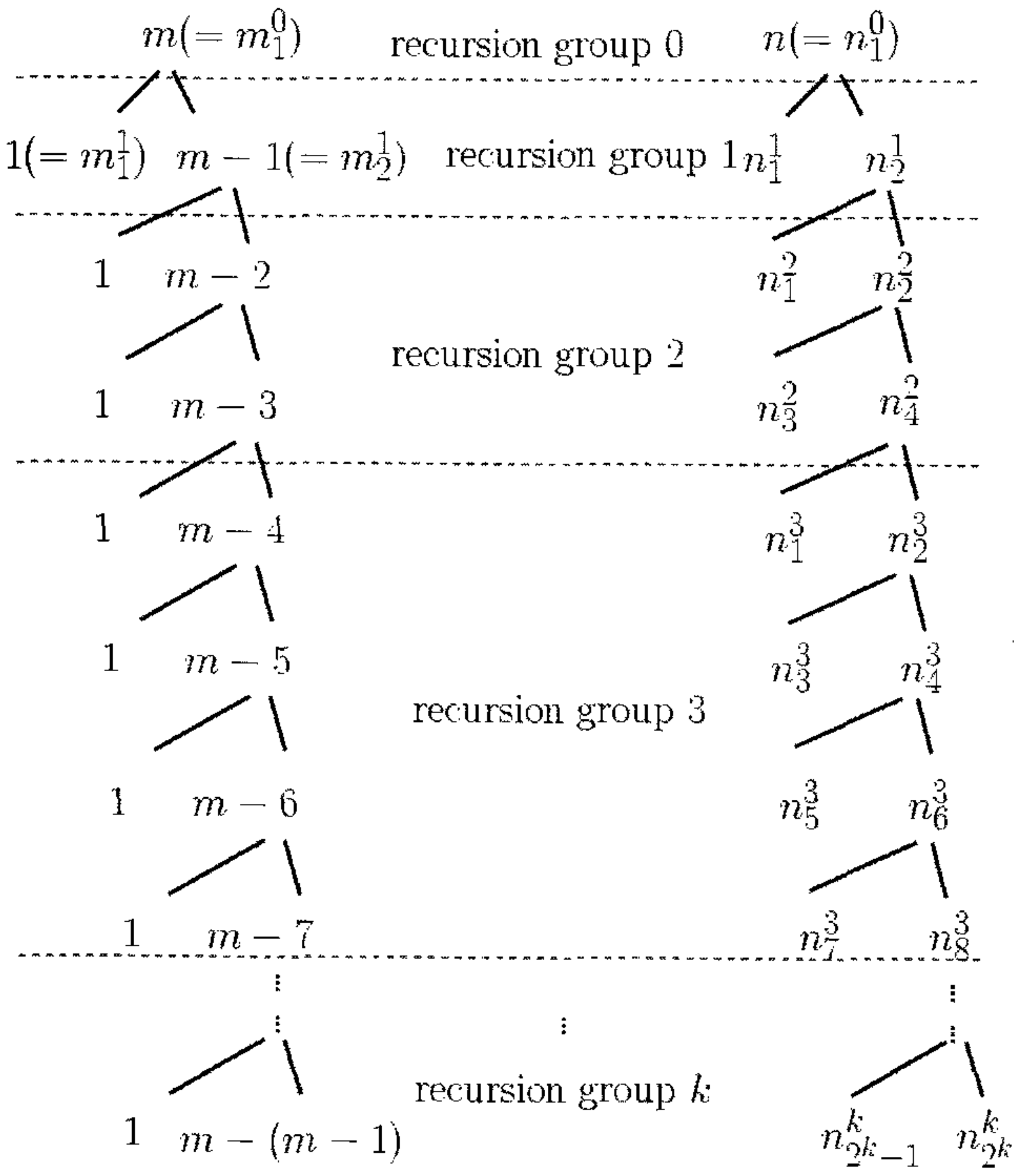


Fig. 3. Construction of recursion groups

Theorem 1. ([3] Theorem 1) The Symmerge algorithm needs $O(m\log\frac{n}{m})$ comparisons.

proof. The number of comparisons for the binary search for recursion group 0 is equal to

$$\lfloor\log m\rfloor+1 \le \lfloor\log m+n\rfloor+1.$$

For the recursion group 1 we need at most

$$\log(m_1^1+n_1^1)+1+\log(m_2^1+n_2^1)+1$$

comparisons, and so on. For the recursion group i we need at most $\sum_{j=1}^{2^i}\log(m_j^i+n_j^i)+2^i$ comparisons. Since

$$\sum_{j=1}^{2^i}(m_j^i+n_j^i)=m+n,$$

it holds

$$\sum_{j=1}^{2^i}\log(m_j^i+n_j^i)+2^i \le 2^i\log((m+n)/2^i)+2^i$$

by Lemma 1. So the overall number of comparisons for all $k+1$ recursion groups is not greater than

$$\sum_{i=0}^{k}(2^i+2^i\log((m+n)/2^i)) = 2^{k+1}-1$$
$$+(2^{k+1}-1)\log(m+n)-\sum_{i=0}^{k}i2^i.$$

Since $\sum_{i=0}^{k}i2^i=(k-1)2^{k+1}+2$, algorithm Symmerge needs at most

$$2^{k+1}-1+(2^{k+1}-1)\log(m+n)-(k-1)2^{k+1}-2$$
$$=2^{k+1}\log(m+n)-k2^{k+1}+2^{k+2}-\log(m+n)-3$$
$$=2m(\log(m+n)-\log m+2)-\log(m+n)-3$$
$$=2m(\log\frac{m+n}{m}+2)-\log(m+n)-3$$
$$=O(m\log(\frac{n}{m}+1))$$

comparisons.

$\square$

Theorem 2. ([3] Theorem 3) If we take the rotation algorithm given in [2], then Symmerge requires $O((m+n)\log m)$ element assignments.

proof. Inside each recursion group $i=0,1,\cdots,k$ disjoint parts of u are merged with disjoint parts of v. Hence each recursion group i comprises at most

$$\sum_{j=1}^{2^i}((m_j^i+n_j^i)+\gcd(m_j^i,n_j^i))$$
$$\le m+n+\sum_{j=1}^{2^i}m_j^i$$
$$=2m+n$$

assignments resulted from rotations. So the overall number of assignments for all k recursion groups is less than

$$(2m+n)(k+1)=(2m+n)\log m+2m+n$$
$$=O((m+n)\log m).$$

$\square$

3.2 Relation of the Input-Sizes m and n

So far we grouped recursion levels so that each recursion group consists of one or several recursion levels and comprises $2^i\,(i=0,1,\cdots,\lfloor\log m\rfloor)$ subsequence mergings at most. We resolve them again. So, let m_j^i and n_j^i denote sizes of sequences merged on the ith recursion level $(i=0,1,2,\cdots,m-1)$. Initially $m_1^0=m$ and $n_1^0=n$. Now we consider the relationship between m and n for the degenerated case where the recursion depth reaches $m-1$, where each (m_2^i,n_2^i) is partitioned to $(1,n_1^{i+1})$ and $(m_2^i-1\ (=m_2^{i+1}),n_2^{i+1})$ as shown in Fig 4.

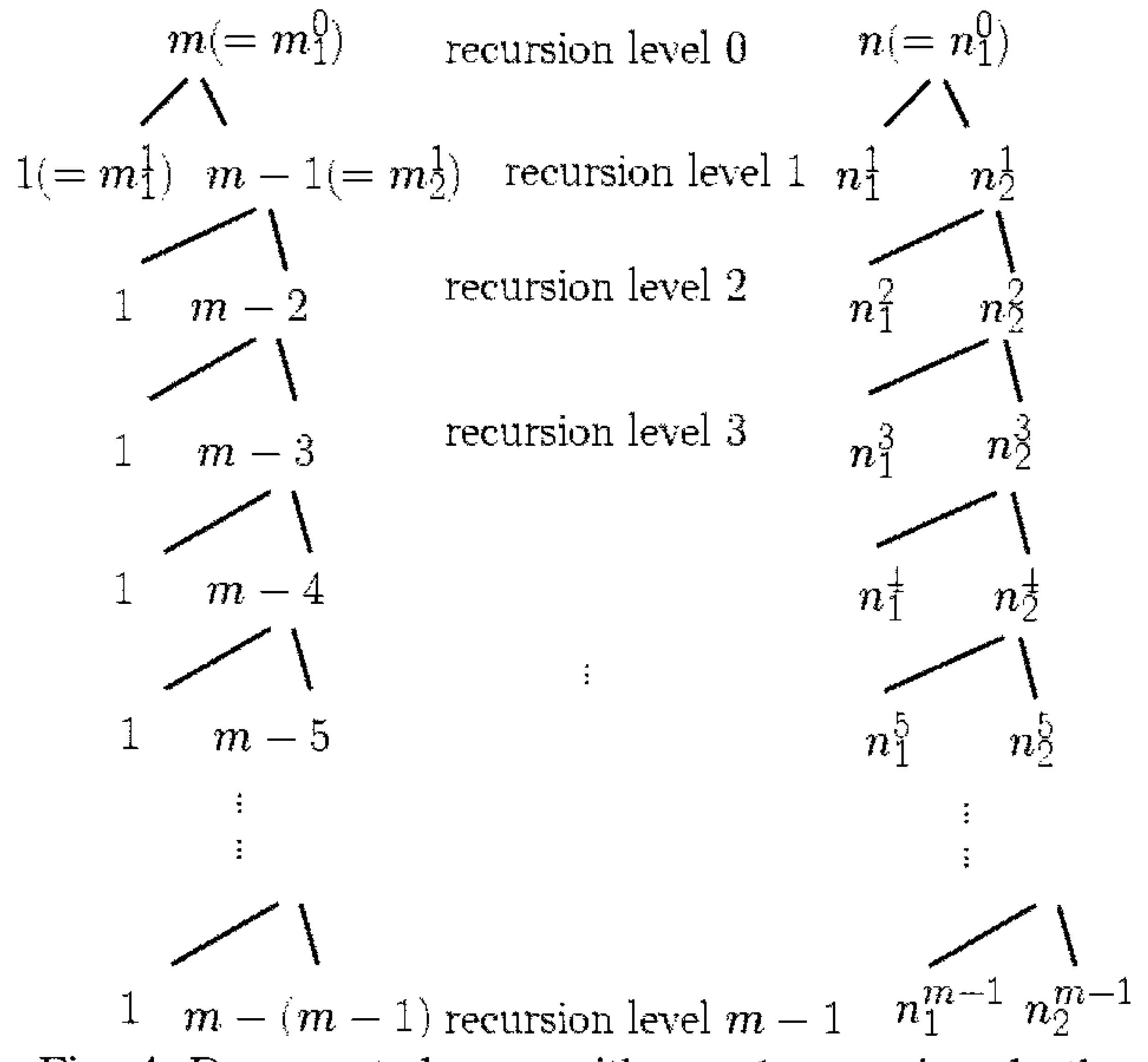


Fig. 4. Degenerated case with $m-1$ recursion depth

Theorem 3. If Symmerge algorithm reaches the recursion depth $m-1$ for two input sequences of sizes m and n $(m \le n)$, then $n \ge 2^{m-1}(m+2)-m$.

Proof. At the beginning the longer sequence v of size n is decomposed into three parts so that the middle part has the same size as the shorter sequence u of size m. Afterwards the technique of symmetric comparisons is applied. As result m and n are partitioned to (m_1^1, n_1^1) and (m_2^1, n_2^1) where it holds $m_1^1 = 1$,

$$m_2^1 = m-1,$$

$$n_1^1 = \frac{n-m}{2} + m - 1 \text{ and}$$

$$n_2^1 = \frac{n-m}{2} + 1.$$

To reach the next recursion level, it must be satisfied that

$$m-1 \le n_2^1 = \frac{n-m}{2} + 1.$$

Suppose that, just as on the first recursion level,

$$(m-1(=m_2^1), \frac{n-m}{2}+1(=n_2^1))$$

is again partitioned to $(1, n_1^2)$ and $(m-2, n_2^2)$ on the second recursion level. Then

$$m-2 \le n_2^2 = \frac{\frac{n-m}{2}+1-(m-1)}{2}+1$$

$$= \frac{n-m+2-2(m-1)+2^2}{2^2}$$

$$= (n-\sum_{i=0}^{1} 2^i(m-i) + \sum_{i=0}^{1} 2^{i+1})/2^2.$$

On the kth recursion level, suppose

$$(m-(k-1)(=m_2^{k-1}), n_2^{k-1})$$

is partitioned to $(1, n_1^k)$ and $(m-k, n_2^k)$, where

$$n_2^k = (n-m+2-2(m-1)+2^2-2^2(m-2)$$
$$+2^3+\cdots-2^{k-1}(m-(k-1))+2^k)/2^k$$
$$= (n-\sum_{i=0}^{k-1} 2^i(m-i) + \sum_{i=0}^{k-1} 2^{i+1})/2^k.$$

In order to reach the next recursion level $k+1$, it must be satisfied that $m-k \le n_2^k$, and so on. Hence, to reach the recursion depth $m-1$, we need the assumption $m-(m-1) \le n_2^{m-1}$, where

$$n_2^{m-1} = (n-m+2-2(m-1)+2^2$$
$$-2^2(m-2)+2^3-\cdots-2^{i-1}(m-(i-1))$$
$$+2^i-\cdots-2^{m-2}(m-(m-2))+2^{m-1})/2^{m-1}$$
$$= (n-\sum_{i=0}^{m-2} 2^i(m-i) + \sum_{i=0}^{m-2} 2^{i+1})/2^{m-1}$$
$$= (n-(m-2)(\sum_{i=0}^{m-2} 2^i) + \sum_{i=1}^{m-2} i2^i)/2^{m-1}.$$ Therefore

$$2^{m-1} \le n-(m-2)(2^{m-1}-1)+2^{m-1}(m-3)+2$$
$$= n-2^{m-1}(m+1)+m)$$

Hence $2^{m-1}(m+2)-m \le n$

□

4. Conclusion

The Symmerge algorithm is a simply structured stable merging algorithm. It merges two sorted sequences on the foundation of a divide and conquer strategy. In [3] it was already proved that Symmerge is asymptotically optimal regarding the number of comparisons, where the corresponding proof relies on the construction of recursion groups as central technique. In the context of this proof the inequality given in Theorem 3 is of significant importance. Here we proved the correctness of this inequality in full detail as completion of the work presented in [3].

참 고 문 헌

[1] D. E. Knuth, "The Art of Computer Programming," Addison-Wesley, Vol. 3: *Sorting and Searching*, 1973.

[2] K. Dudzinski and A. Dydek, "On a Stable Storage Merging Algorithm," *Information Processing Letters*, Vol. 12, No. 1, pp. 5-8, 1981.

[3] P. S. Kim and A. Kutzner, "Stable Minimum Storage Merging by Symmetric Comparisons," In Albers, S., Radzik, T. (eds.), Algorithms-ESA 2004, Springer, *Lecture Notes in Computer Science* 3221, pp. 714-723, 2004.

[4] L. T. Pardo. "Stable sorting and merging with optimal space and time bounds," *SIAM Journal on Computing*, 6(2):351-372, June 1977.
[5] J. Salowe and W. Steiger. "Simplified stable merging tasks," *Journal of Algorithms*, 8:557-571, 1987.

저 자 소 개

Pok-Son Kim (김복선)
She has been assistant professor of Department of Mathematics, Kookmin University, Seoul, Korea.
Her research interests are complexity theory, merging and sorting algorithms as well as scheduling problems.

Phone : +82-2-910-4747
Fax : +82-2-910-4739
E-mail : pskim@kookmin.ac.kr