JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 20, No. 2 : 415~420 Feb. 2016

# 선분 그래프의 정점 연결성에 대한 완전 동적 알고리즘

김재훈*

# Fully Dynamic Algorithm for the Vertex Connectivity of Interval Graphs

**Jae-hoon Kim***

Department of Computer Engineering, Busan University of Foreign Studies, Busan 46234, Korea

## 요 약

선분 그래프(interval graph) $G=(V,E)$는 직선 상의 선분들을 나타내는 정점 집합 $V$와 간선 $(i,j)\in E$는 선분 $i$와 $j$가 교차함을 나타내는 간선들의 집합 $E$로 이루어진다. 본 논문에서는 그래프의 여러 특성 중에서 정점 연결성(vertex connectivity)에 주목한다. 특별히 선분들이 겹쳐지는 모습으로 선분 그래프의 정점 연결성을 나타낸다. 또한 선분 그래프에서 정점이나 간선이 추가 되거나 삭제되는 완전 동적 (fully dynamic) 환경에서 정점 연결성을 계산하는 효율적인 알고리즘을 제안할 것이다. 특별한 형태의 선분 트리(interval tree)를 사용하여 새로운 선분이 추가되거나 삭제되는 상황 하에서 정점 연결성을 계산하고 트리를 유지하는데 $O(\log n)$ 시간이 소요됨을 보일 것이다.

## ABSTRACT

A graph $G=(V,E)$ is called an interval graph with a set $V$ of vertices representing intervals on a line such that there is an edge $(i,j)\in E$ if and only if intervals $i$ and $j$ intersect. In this paper, we are concerned in the vertex connectivity, one of various characteristics of the graph. Specifically, the vertex connectivity of an interval graph is represented by the overlapping of intervals. Also we propose an efficient algorithm to compute the vertex connectivity on the fully dynamic environment in which the vertices or the edges are inserted or deleted. Using a special kind of interval tree, we show how to compute the vertex connectivity and to maintain the tree in $O(\log n)$ time when a new interval is added or an existing interval is deleted.

**키워드** : 선분 그래프, 정점 연결성, 완전 동적, 선분 트리, 알고리즘, 선분

Key word : interval graph, vertex connectivity, fully dynamic, interval tree, algorithm, interval

**Received** 23 October 2015, **Revised** 16 November 2015, **Accepted** 30 November 2015

*** Corresponding Author** Jae-hoon Kim(E-mail:jhoon@bufs.ac.kr, Tel:+82-51-509-6226)
Department of Computer Engineering, Busan University of Foreign Studies, Busan 46234, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2016.20.2.415 print ISSN: 2234-4772 online ISSN: 2288-4165

## Ⅰ. 서 론

선분 $\ell = [a, b]$이란, $a \le x \le b$를 만족하는 직선 상의 점 $x$의 집합이다. 두 선분 $\ell_1 = [a_1, b_1]$과 $\ell_2 = [a_2, b_2]$에 대해서, $[a_1, b_1] \cap [a_2, b_2] \neq \varnothing$ 이면, 두 선분 $\ell_1$과 $\ell_2$가 교차한다(intersect)고 한다. 그러면 직선 상에 $n$개의 선분들이 주어질 때, 각 선분을 그래프의 정점에 대응하고 두 선분이 교차하면 대등되는 두 정점 사이에 간선을 연결하여 얻어지는 그래프를 *선분 그래프*(interval graph)라고 한다. 다음 그림 1의 5개의 선분에 대한 선분 그래프가 그림 2에 주어진다.

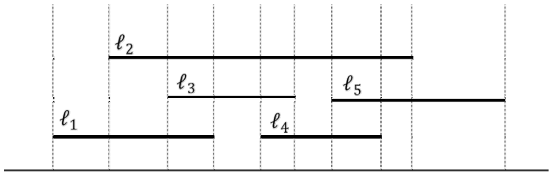


**Fig. 1** Intervals on the line

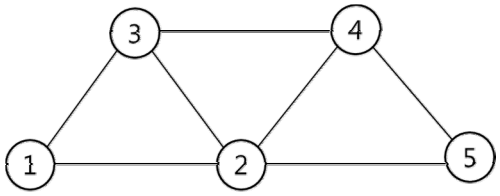


**Fig. 2** Interval graph for the intervals

본 논문에서는 그래프의 연결성(connectivity)에 주목한다. 그래프 $G=(V, E)$에서 임의의 두 정점 $v, w \in V$ 에 대해서, $v$와 $w$사이에 항상 경로가 존재한다면 그리프 $G$는 연결 그래프(connected graph)라고 한다. 그리고 그래프 $G$에서 어떤 두 정점 $v$와 $w$가 존재해서 이 두 정점 사이에 경로가 존재하지 않으면 $G$는 비연결 그래프(disconnected graph)라고 한다. 부분집합 $\overline{V} \subseteq V$에 대해서, $G - \overline{V}$가 비연결 그래프라면 $\overline{V}$를 정점 절단(vertex cut)이라고 한다. 다시 말해서, $\overline{V}$에 속한 정점들을 $G$에서 제거해서 얻어지는 부분 그래프는 비연결된다. 그러면 $k$-정점 절단($k$-vertex cut)은 $k$개의 정점들을 포함하는 정점 절단을 말한다. 그래프 $G$가 적어도 한 쌍의 인접하지 않는 두 정점들을 가지고 있을 때, 그래프 $G$의 정점 연결성(vertex connectivity)이란 $G$가 포함할 수 있는 $k$-정점 절단의 최소 $k$ 값을 말하고 $\kappa(G)$로 나타낸다. 그리고 $\kappa(G) \ge k$이면, 그래프 $G$는 $k$-연결($k$-connected) 되었다고 한다.

본 논문에서는 그래프 $G=(V, E)$에서 정점이나 간선이 추가되고 삭제되는 동적 환경(dynamic environment)에 대해서 다룰 것이다. 주어진 그래프의 변화에 따라서 정점이나 간선의 추가와 삭제가 모두 일어나는 상황을 완전 동적(Fully dynamic)이라고 한다. 반면 정점이나 간선의 추가만 일어나는 상황은 증가(Incremental)라고 하고 삭제만 일어나는 상황은 감소(Decremental)이라고 한다. 본 논문에서는 완전 동적인 상황을 다룰 것이다.

예를 들어, 위의 그림 2의 선분 그래프는 정점 연결성이 2이다. 그런데 그림 3에서와 같이 선분 $\ell_6$이 추가되는 상황을 생각해 보자. 결과적으로 그림 4와 같은 선분 그래프가 얻어지고 정점 연결성은 3이 된다.

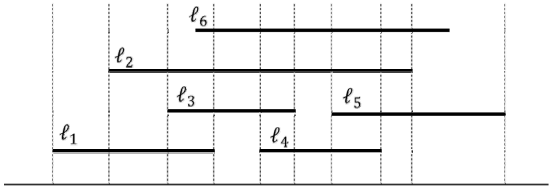


**Fig. 3** The case one interval is inserted

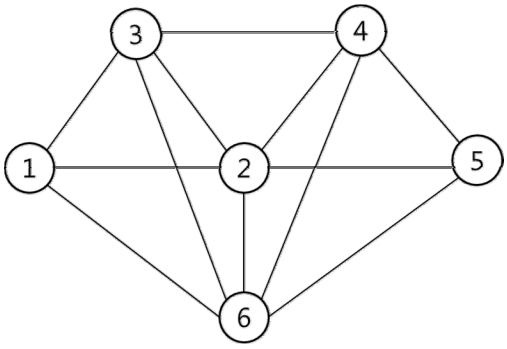


**Fig. 4** Interval graph for Fig. 3

본 논문에서는 주어진 선분 그래프 $G$에 대해서 $G$의 정점 연결성에 대해서 연구할 것이다. 특별히 완전 동적 상황에서, 다시 말해서, 선분 그래프 $G$를 나타내는 선분들에서 하나의 새로운 선분이 추가되거나 존재하

는 선분들 중 하나를 삭제하는 상황에서 선분 그래프 $G$의 정점 연결성을 효율적으로 계산하는 알고리즘에 대해서 연구할 것이다.

## Ⅱ. 관련 연구

선분 그래프는 직선 상의 선분들의 상호 교차 여부를 나타내는 그래프로서 유전학(genetic), 분자 생물학(molecular biology), 정보 검색(information retrieval) 등에서 응용될 수 있다[1]. 일반 그래프에서는 *NP-complete*인 여러 문제들이 선분 그래프에서는 다항시간 알고리즘이 존재함이 알려져 있다[1-5].

그래프와 관련된 여러 문제들 중 정점 연결성 문제는 가장 대표적인 문제들 중 하나이다. 일반 그래프에서의 정점 연결성 문제는 $O(\kappa mn\sqrt{n})$ 시간 알고리즘이 존재 한다[6]. 여기서, $\kappa$는 그래프의 정점 연결성이다. 특별한 그래프들에 대한 정점 연결성 문제들이 연구되었다. 특히 [7]에서 사다리꼴 그래프(trapezoidal graph)의 정점 연결성을 계산하는 $O(n^2)$ 시간 알고리즘이 제안되었고, [8]에서 수행 시간이 $O(n\log n)$으로 향상되었다. [9]에서는 선분 그래프의 $k$-정점 연결성을 결정하는 병렬 알고리즘이 제안되었다.

정점 또는 간선들이 추가되거나 삭제되는 동적 환경하에서의 그래프 문제들에 대한 많은 연구들이 있었다[10]. [11]에서는 그래프가 연결되었는지 여부를 결정하는 연결성(connectivity) 문제에서 완전 동적 상황을 가정하였다. 저자들은 간선이 추가되거나 삭제되는 경우에 $O(\log^2 n)$-*amortized* 시간에 문제를 해결하는 알고리즘을 제안하였다. 또한 완전 동적 상황 하에서 최소 신장 트리(minimum spanning tree)를 유지하는 $O(\log^4 n)$-*amortized* 시간 알고리즘도 제안되었다. [12]에서는 완전 동적 상황 하에서 선분 그래프를 인식하는 문제를 다루었고, 정점의 추가와 삭제 상황에서 $O(n)$ 시간에 답하는 알고리즘을 제안하였다.

## Ⅲ. 알고리즘 및 분석

선분 그래프 $G=(V,E)$가 주어지고, $G$를 나타내는 $n$개의 선분들을 $\ell_1, \ell_2, ..., \ell_n$이라고 하자. 그리고 편의상 선분들의 끝점들은 모두 다르다고 가정한다. 그러면 선분 그래프 $G$의 정점 연결성에 대해서 생각한다.

우선 선분들에 대해서, 선분들과 수직인 직선 $L$을 생각해 보자. $L$의 왼쪽에 적어도 하나의 선분이 존재하고, 다시 말해서, 적어도 한 선분의 오른쪽 끝점이 $L$보다 작고, $L$의 오른쪽에 적어도 하나의 선분이 존재하면, 다시 말해서, 적어도 한 선분의 왼쪽 끝점이 $L$보다 크면, $L$을 선분들의 절단선(cutting line)이라고 한다. 또한 임의의 절단선 $L$에 대해서, $L$과 교차하는 선분들의 수를 생각할 때, 모든 절단선 $L$에 대한 이 수들의 최소값을 주어진 선분들의 절단수(cutting number)라고 한다. 다음 정리에서 선분 그래프의 정점 연결성이 선분들의 절단수와 같음을 증명할 것이다.

**정리 3. 1** 선분 그래프 $G=(V,E)$가 주어지고 $G$를 나타내는 선분들이 $\ell_1, \ell_2, ..., \ell_n$이라고 할 때, $G$의 정점 연결성이 $k$일 필요 충분 조건은 선분들의 절단수가 $k$이다.

**증명.** 선분 그래프 $G$의 정점 연결성이 $\kappa(G)$라 하고 선분들의 절단수를 $k$라고 하자. 우선 $k \geq \kappa(G)$임을 보인다.

임의의 절단선 $L$을 생각하자. $L$과 교차하는 선분들에 대응되는 그래프 $G$의 정점들의 집합을 $\overline{V}$라고 하자. 절단선 $L$의 정의에 의해서 $L$의 왼쪽에 적어도 하나의 선분 $\ell_L$과 오른쪽에 적어도 하나의 선분 $\ell_R$이 존재한다. 그러면 이 두 선분에 대응되는 정점 $v_L$과 $v_R$은 그래프 $G$에서 $\overline{V}$에 속한 정점들을 제거해서 얻어지는 그래프 $G-\overline{V}$에 속한다. 하지만 $v_L$과 $v_R$를 연결하는 경로가 존재하지 않는다. 왜냐하면, 연결하는 경로가 존재하기 위해서는 경로 상에 $L$과 교차하는 적어도 하나의 선분을 나타내는 정점이 존재하여야만 한다. 따라서 $G-\overline{V}$는 비연결이다. 결과적으로 $\overline{V}$는 정점 절단이다. 이것이 $k \geq \kappa(G)$임을 증명한다.

다음으로 $\kappa(G) \geq k$임을 증명할 것이다. 역으로 $k > \kappa(G)$가 참이라고 가정하고 최소 개수의 정점 절단을 $C=\{v_1, v_2, ..., v_{\kappa(G)}\}$라고 하자. 이 정점 절단에 속한 정점들을 제거하고 나면, 그래프 $G$는 적어도 두 개의 연결 성분(connected component) $G_1$과 $G_2$가 남는다.

일반성의 손실 없이(Without loss of generality), $G_1$에 속하는 정점들을 나타내는 선분들의 오른쪽 끝점들은 $G_2$에 속하는 정점들을 나타내는 선분들의 왼쪽 끝점들보다 작다. 왜냐하면, $G_1$과 $G_2$에 속하는 정점을 나타내는 각각의 선분을 $\ell_1 = [a, b]$과 $\ell_2 = [c, d]$라고 하자. 그러면 일반성 손실 없이, $b < c$라고 가정할 수 있다. 그런데 $G_2$에 속하는 정점의 선분 $\ell_3 = [e, f]$가 $e \leq b$라고 하면, $\ell_3$가 $\ell_1$과 교차하지 않으므로 $f < a$. 하지만, 선분 $\ell_3$와 $\ell_2$는 연결되어 있으므로 이 두 선분의 정점을 연결하는 경로 상의 적어도 한 정점의 선분은 $\ell_1$과 교차해야만 하고 이것은 모순이다.

따라서 $G_1$에 속하는 정점들을 나타내는 선분들의 오른쪽 끝점들 중 최대값을 $r_1$이라 하고, $G_2$에 속하는 정점들을 나타내는 선분들의 왼쪽 끝점들 중 최소값을 $l_2$라고 하면, 정점 절단 $C$에 속한 정점들에 대응되는 선분들은 구간 $(r_1, l_2)$과 교차한다. 따라서 이 구간을 통과하는 임의의 수직선 $L$을 생각한다. 수직선 $L$과 교차하는 선분들의 개수를 $h$라고 하면, $\kappa(G) \geq h$. 처음의 가정에 의해서 $k > \kappa(G) \geq h$이 되는데, 이것은 절단수 $k$의 정의에 의해서 모순이다. 따라서 $\kappa(G) \geq k$임을 증명한다.

$n$개의 선분이 주어진 정적인 상황에서 선분들의 절단수를 구하는 알고리즘은 선분들의 끝점의 좌표들을 정렬해서 왼쪽부터 오른쪽으로 고려하면서 현재까지 고려한 선분들의 절단수를 업데이트하면 $O(n)$만에 계산할 수 있다. 다시 말해서, 선분 그래프의 정점 연결성을 $O(n)$ 시간 만에 계산할 수 있다.

우리는 선분들이 추가되거나 삭제되는 동적 상황을 다룰 것이다. 이런 동적 상황에서 선분 그래프 $G$의 정점 연결성을 효율적으로 계산하는 알고리즘을 생각할 것이다.

동적 환경을 다루기 위해서 우리는 선분 트리(interval tree)라는 특별한 자료구조를 사용할 것이다. 선분 트리 $T$는 레드-블랙 트리(red-black tree), AVL 트리 등과 같은 균형 이진 탐색 트리(balanced binary search tree, BST) 형태로 만든다. 따라서 선분 트리의 정점의 개수가 $n$개일 때, 트리의 높이는 항상 $O(\log n)$으로 유지되고, 따라서 삽입, 삭제 연산을 $O(\log n)$ 시간에 수행할 수 있다.

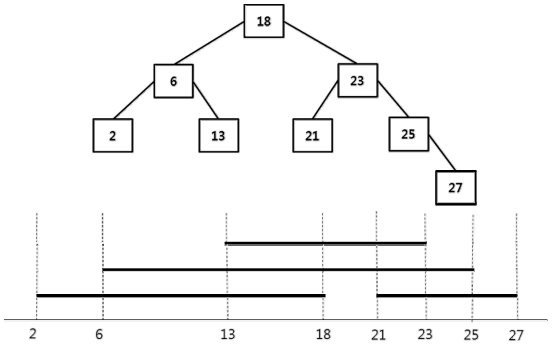


**Fig. 5** Interval tree for intervals

직선 상에 $n$개의 선분들이 주어질 때, 이 선분들의 끝점들을 $p_1$, ..., $p_{2n}$이라고 하자. 선분 트리 $T$는 위의 그림 5와 같이 이 끝점들을 키 값으로 갖는 균형 이진 탐색 트리로 만든다. 그림 5에서는 4개의 선분들이 주어질 때, 8개의 끝점들의 좌표를 키 값으로 하는 탐색 트리가 만들어졌다.

선분 트리 $T$의 각 노드에는 끝점의 좌표뿐만 아니라 추가적인 정보들을 저장할 것이다. 우선 끝점들에 대해서 선분의 왼쪽 끝점 $p_i$에는 $v(p_i) = 1$을 부여하고 오른쪽 끝점 $p_j$에는 $v(p_j) = -1$을 부여한다. 여기서 $v(p_i) = 1$은 새로운 선분이 시작됨으로 인해서 선분들이 겹치는 개수가 1개 늘어남을 의미하고, $v(p_j) = -1$은 선분이 끝나서 겹치는 선분들이 1개 줄었음을 의미한다. 또한 점들을 왼쪽에서부터 스캔하고 다시 오른쪽에서부터 스캔하면서 점 $p_i$의 왼쪽 부분에 존재하는 오른쪽 끝점들의 개수 $l(p_i)$(여기서, $p_i$가 오른쪽 끝점인 경우에 왼쪽 부분에 포함된다)와 $p_i$의 오른쪽 부분에 존재하는 왼쪽 끝점들의 개수를 $r(p_i)$(여기서, $p_i$가 왼쪽 끝점인 경우에 오른쪽 부분에 포함되지 않는다)를 계산한다. 그러면 $l(p_i)$와 $r(p_i)$가 모두 0이 아니면, 점 $p_i$를 주요점(primary point) 이라고 부른다. 그리고 $s(i, j) = v(p_i) + \cdots + v(p_j)$라고 하면, $s(1, j)$는 점 $p_1$, ..., $p_j$을 포함하는 선분만을 고려해서 $p_j$를 넘어서는 순간 겹치는 선분들의 수를 나타낸다.

선분 트리 $T$에서 점 $p_i$를 키 값으로 가지는 노드 $x$는 자신의 부트리에 속하는 점들이 $p_j$, ..., $p_k$일 때, $w[x] = s(j, k)$를 저장한다. 이 값은 트리 $T$의 아래쪽

노드에서 위쪽 노드 방향으로 계산될 수 있다. $x$의 왼쪽 자식 노드를 $l[x]$라고 하고 오른쪽 자식 노드를 $r[x]$라고 할 때, $w[x]=w[l[x]]+v(p_i)+w[r[x]]$로 구해진다.

또한 노드 $x$에서 다음과 같이 정의된 $o[x]$를 저장한다. $m$은 $p_i$가 주요점일 때 $w[l[x]]+v(p_i)$로 정의하고, 주요점이 아니면 $\infty$로 정의한다. 그러면

$$o[x]=\min\{o[l[x]],m, \quad w[l[x]]+v(p_i)+o[r[x]]\} \tag{1}$$

로 정의한다. 식 1의 값은 노드 $x$를 루트로 하는 부트리에 속하는 노드들에 대응하는 선분들만을 고려해서 이 선분들의 절단수를 결정하는 수직선 $L$이 좌부트리의 구간에 존재하거나, 노드 $x$에 대응하는 점 $p_i$ 직후에 존재하거나, 우부트리의 구간에 존재하는 경우를 고려해서 결정한다. 따라서 루트 노드 $\overline{x}$에 저장되는 $o[\overline{x}]$가 주어진 모든 선분들의 절단수와 같다.

선분 트리 $T$에서 각 노드 $x$에 키 값 $p_i$, $w[x]$, $o[x]$를 저장하고 이 값들은 위에 설명한 것처럼 트리의 아래로부터 위쪽으로 계산될 수 있다. 또한 루트 노드 $\overline{x}$에 저장된 $o[\overline{x}]$는 선분들의 절단수, 다시 말해서, 선분 그래프의 정점 연결성과 같다.

선분 트리 $T$를 가지고 있으면 새로운 선분이 추가되거나 존재하는 선분을 제거하는 경우에 루트 노드로 가는 경로 상의 노드들에 저장된 $w[]$와 $o[]$를 업데이트할 수 있다. 이것은 $O(\log n)$ 시간에 수행될 수 있다.

**정리 3. 2** 선분 그래프 $G=(V,E)$에서 한 정점이나 간선이 추가되고 삭제되는 경우에 $O(\log n)$ 시간에 정점 연결성을 구할 수 있다.

## Ⅳ. 결 론

본 논문에서는 선분 그래프의 정점 연결성 문제를 다루었다. 특별히 정점들 또는 간선들이 추가되거나 삭제되는 완전 동적 상황 하에서 정점 연결성을 효율적으로 계산하는 알고리즘에 대해서 연구하였다. 정점 연결성을 선분들의 절단수로 표현하고 선분 트리를 사용해서 절단수를 효율적으로 계산하는 방법을 소개하였다. 선분 트리를 유지하고 절단수를 계산하는데 $O(\log n)$ 시간이면 충분함을 보였다.

향후에 선분 그래프의 다른 특성들을 완전 동적 상황 하에서 효율적으로 계산하는 문제를 연구 할 수 있을 것이다.


## ACKNOWLEDGMENTS

This work was supported by the research grant of the Busan University of Foreign Studies in 2015


## REFERENCES


[ 1 ] M. C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs,* New York, NY: Academic Press, 1980.

[ 2 ] J. M. Keil, “Finding hamiltonian circuits in interval graphs,” *Information Processing Letters,* vol. 20, pp. 201-206, May 1985.

[ 3 ] G. Ramalingam and C. Pandu Rangan, “A unified approach to domination problems on interval graphs,” *Information Processing Letters,* vol. 27, pp. 271-274, April 1988.

[ 4 ] A. Srinivasa Rao and C. Pandu Rangan, “Linear algorithm for domatic number problem on interval graphs,” *Information Processing Letters,* vol. 33, pp. 29-33, Oct. 1989.

[ 5 ] H. Broersma, Jiri Fiala, P. A. Golovach, T. Kaiser, D. Paulusma, and A. Proskurowski, “Linear-time algorithms for scattering number and hamilton-connectivity of interval graphs,” *Journal of Graph Theory,* vol. 79, pp. 282-299, Aug. 2015.

[ 6 ] S. Even and R. E. Tarjan, “Network flow and testing graph connectivity,” *SIAM Journal on Computing,* vol. 4, pp. 507-518, Oct. 1975.

[ 7 ] P. K. Ghosh and M. Pal, “An Efficient algorithm to solve connectivity problem on trapezoid graphs,” *Journal of Applied Mathematics and Computing,* vol. 24, pp. 141-154, May 2007.

[ 8 ] A. Ilic, “Efficient algorithm for the vertex connectivity of trapezoid graph," *Information Processing Letters,* vol. 113,

pp. 398-404, May 2013.

[9] T. W. Kao and S. J. Horng, “Computing k-vertex connectivity on an interval graph," in *Proceeding of the 13th International Conference on Parallel Processing*, pp. 218-221, 1994.

[10] D. Eppstein, Z. Galil, G, and F. Italiano, *Agorithms and Theoretical Computing Handbook,* CRC Press, 1999.

[11] J. Holm, K. de Lichtenberg, and M. Thorup, “Poly-logarithmic deterministic fully dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity,” *Journal of ACM,* vol. 48, pp. 723-760, July 2001.

[12] C. Crespelle, “Fully dynamic representations of interval graphs," in *Proceeding of the 35th International Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 77-87, 2009.

**김재훈(Jae-Hoon Kim)**

1994 서강대학교 수학과 이학사
1996 KAIST 수학과 이학석사
2003 KAIST 전산과 공학박사
2003~ 부산외국어대학교 컴퓨터공학과 교수
※관심분야 : 알고리즘, 최적화, 스케줄링