http://dx.doi.org/10.7236/JIIBC.2013.13.2.103
JIIBC 2013-2-14

Kruskal과 역-삭제 최소신장트리 알고리즘의 효율적 구현 방법

An Efficient Implementation of Kruskal's and Reverse-Delete Minimum Spanning Tree Algorithm

최명복*, 이상운**

Myeong-Bok Choi, Sang-Un Lee

요 약 본 논문은 최소신장트리를 구하는 Kruskal과 역-삭제 알고리즘의 수행 횟수를 줄이는 방법을 제안하였다. 기존의 Kruskal과 역-삭제 알고리즘은 그래프의 모든 간선들을 대상으로 사이클이 발생하는지 여부를 검증한다. 이로 인해 알고리즘 수행 과정에서 이미 최소신장트리를 얻었음에도 불구하고 나머지 간선들에 대해 알고리즘을 추가로 불필요하게 수행하는 문제점을 갖고 있다. 본 논문은 먼저, Kruskal과 역-삭제 알고리즘과 동일하게 모든 간선들을 대상으로 알고리즘은 수행하지만 알고리즘 종료 시점 기준을 적용하여 수행 횟수를 줄이는 "제1방법"을 제안하였다. 다음으로, 최소신장트리에 전혀 영향을 미치지 않는 불필요한 간선을 사전에 제거하고 남은 간선들을 대상으로 최소신장트리를 찾는 "제2방법"을 제안하였다. 제안된 방법들을 실제 그래프들에 적용한 결과 기존의 Kruskal과 역-삭제 알고리즘보다 최소 1.4배에서 최대 3.86배 빨리 알고리즘을 종료시키는 효과를 얻었다. 제안된 2개 방법을 2개 알고리즘에 적용한 4개 알고리즘 중에서 역-삭제 알고리즘 "제2방법"이 가장 빨리 알고리즘을 종료시키는 결과를 얻었다.

Abstract This paper suggests a method to reduce the number of performances of Kruskal and Reverse-delete algorithms. Present Kruskal and Reverse-delete algorithms verify whether the cycle occurs within the edges of the graph. For this reason, they have problems of unnecessarily performing extra algorithms from the edges, even though they've already obtained the minimum spanning tree. This paper, first of all, suggests the 1st method which reduces the no. of performances by introducing stop point criteria of algorithm, but at the same time, performs algorithms from all the edges, just like how Kruskal and Reverse-delete algorithms. Next, it suggests the 2nd method which finds the minimum spanning tree from the remaining edges after getting rid of all the unnecessary edges which are considered not to affect the minimum spanning tree. These suggested methods have an effect of terminating algorithm at least 1.4 times and at most 3.86times than Kruskal and Reverse-delete algorithms, when applied to the real graphs. We have found that the 2nd method of the Reverse-delete algorithm has the fastest speed in terminating an algorithm, among 4 algorithms which are results of the 2 suggested methods being applied to 2 algorithms.

Key Words : Minimum Spanning Tree, Cycle Property, Kruskal Algorithm, Reverse-Delete Algorithm

*종신회원, 강릉원주대학교 멀티미디어공학과
**정회원, 강릉원주대학교, 멀티미디어공학과
접수일자 2013년 1월 7일, 수정완료 2013년 3월 2일
게재확정일자 2013년 4월 12일

Received: 7 January 2013 / Revised: 2 March 2013 /
Accepted: 12 April 2013
*Corresponding Author: cmb5859@gmail.com
Dept. of Multimedia Engineering, Gangnung-Wonju National University, Korea

Ⅰ. 서 론

그래프 $G=(V,E)$는 정점들 (Vertices)과 간선들(Edges)로 구성되어 있으며, 정점들은 연결되어 있고 (Connected), 간선들은 방향성 (Directed) 또는 무방향성 (Undirected)을 갖고 있으며 가중치가 있는 경우 (Weighted)와 없는 경우 (Unweighted)로 구분된다. 그래프가 연결되어 있고 무방향성이며 가중치를 갖고 있는 경우, 모든 정점들을 연결하면서 사이클이 없는 신장트리 (Spanning Tree, ST)를 구할 수 있다. 즉, ST는 사이클이 발생하지 않는 상태에서 그래프의 모든 정점들을 간선으로 연결한 트리이다. 하나의 그래프에서는 다수의 ST가 존재할 수 있다. 최소신장트리 (Minimum Spanning Tree, MST)는 그래프가 갖고 있는 다수의 ST들 중 모든 정점들을 연결하는 간선들의 가중치 합 $(\Sigma w(e))$이 최소가 되면서 사이클이 발생하지 않는 ST를 찾는 것으로 전기, 전화, 가스 또는 수도 등의 분야에 활용될 수 있다.[1] 본 논문은 MST 알고리즘에 초점을 맞춘다.

대표적인 MST 알고리즘으로는 Borůvka[2,3], Prim[4], Kruskal[5]과 역-삭제 (Reverse-Delete) 알고리즘[6]이 있다. Borůvka MST 알고리즘[2,3]은 모든 간선들의 가중치가 서로 상이한 (Distinct) 그래프에서 MST를 찾는 알고리즘으로 1926년에 처음 제안되었으며, 현재는 1950년대에 제안된 Prim과 Kruskal MST 알고리즘이 널리 사용되고 있다. 역-삭제 알고리즘은 Kruskal 알고리즘을 역으로 수행하는 방식으로 현재에는 일반적으로 사용되지 않고 있다.[1]

본 논문은 Kruskal과 역-삭제 알고리즘[5,6]에 초점을 맞춘다. Kruskal 알고리즘[5]은 그래프의 모든 간선들의 가중치를 오름차순으로 정렬시키고 최소 가중치를 갖는 간선부터 시작하여 한번에 하나씩 해당 간선을 신장트리에 추가할 경우 사이클이 발생하지 않으면 신장트리로 추가하는 방법으로 모든 간선들을 대상으로 이 과정을 반복적으로 수행한다. 반면에, 역-삭제 알고리즘은 그래프의 모든 간선들의 가중치를 내림차순으로 정렬시키고, 최대 가중치를 갖고 있는 간선부터 시작하여 한번에 하나씩 해당 간선을 신장트리에서 삭제하였을 경우 임의의 정점을 신장트리에서 분리시키지 않으면 삭제하는 방법으로 모든 간선들을 대상으로 이 과정을 반복적으로 수행한다.

그래프의 모든 정점을 사이클이 발생하지 않게 연결하는 신장트리의 간선의 수 $|e|$는 그래프의 정점의 수 $|v|$ 보다 1개가 적다. 즉, $|e|=|v|-1$이 다. 만약, 그래프의 간선 수가 $2|v|$개로 구성되어 있고, 이상적으로 간선의 가중치를 오름차순으로 정렬시켰을 때 정확히 $|v|-1$개가 최소신장트리를 구성할 수 있다고 가정하여 보자. 이 경우, Kruskal 알고리즘[5]을 적용할 경우 모든 간선을 대상으로 신장트리에 추가할지 여부를 확인해야 하기 때문에 알고리즘을 $2|v|$번 수행 한다. 그러나 $|v|-1$번을 수행하고, 알고리즘을 종료시켜도 동일한 최소신장트리를 찾을 수 있으며, 불필요하게 $|v|+1$번을 수행하게 된다. 반면에 역-삭제 알고리즘[6]은 $|v|+1$개의 최대 가중치를 갖는 간선을 삭제하면 되지만 불필요하게 $|v|-1$번을 추가로 수행하게 된다.

본 논문에서는 이와 같이 불필요하게 수행되는 알고리즘 수행 횟수를 줄이는 방법을 제안한다. 2장에서는 Kruskal과 역-삭제 알고리즘[5,6]을 고찰해보고, 실제 그래프에 적용하여 MST를 찾고 불필요하게 수행되는 문제점을 고찰해 본다. 3장에서는 Kruskal과 역-삭제 알고리즘[5,6]의 불필요한 수행 횟수를 줄일 수 있는 2가지 방법을 제안한다. 4장에서는 실제 그래프들을 대상으로 제안된 알고리즘의 효율성을 검증해 본다.

Ⅱ. 관련 연구와 연구 배경

1. Kruskal과 역-삭제 MST 알고리즘

그래프 $G=(V,E)$에서 간선은 무방향성이므로 $\{x,y\}$로 표기한다. 왜냐하면 (x,y) 표기는 순서쌍으로 방향성을 가지고 있음을 의미하기 때문이다.[7] $G=(V,E)$에서 정점의 개수를 $|v|$, 간선의 개수를 $|e|$라 하자. 그래프에서 간선의 최대 수는 완전 그래프 (Complete Graph)의 간선 수이다. 완전 그래프는 하나의 정점이 다른 모든 정점들과 인접한 (간선들로 연결된) 그래프로, 간선의 수는 $\frac{|v|(|v|-1)}{2}$ 개 이다.[8] 신장트리는 $|v|$개의 정점을 사이클이 발생하지 않게 간선들로 모두 연결하는 경우이므로 간선의 수는 $|v|-1$ 개로 구성되어 있다. Kruskal MST 알고리즘은 연결된 그래프 $G=(V,E)$와 가중치 함수 $w:E\rightarrow\mathbb{N}$이라 가정하여 다음과 같이 수행된다.[5,9]

(1) E에 있는 모든 가중치를 갖는 간선들을 오름차순으

로 정렬시킨다.

(2) 최소 가중치를 갖는 임의의 간선을 선택하여 부분 신장트리 (Partial Spanning Tree)를 구성한다.

(3) E에서 다음 최소 가중치를 갖는 간선을 한번에 하나씩 선택하여 해당 간선이 T에 추가하였을 때 사이클이 발생하지 않는다면 T에 추가하고, 그렇지 않으면 다음 최소 가중치를 갖는 간선을 선택한다.

(4) E에서 더 이상 선택할 간선이 없으면 알고리즘을 종료한다.

반면에, 역-삭제 MST 알고리즘[6]은 Kruskal MST 알고리즘을 역으로 수행하는 방식으로 다음과 같이 수행된다.

(1) T는 모든 정점들과 간선들로 구성되어 있다.

(2) E에 있는 모든 가중치를 갖는 간선들을 내림차순으로 정렬시킨다.

(3) 최대 가중치를 갖는 임의의 간선을 한 번에 하나씩 선택하여 해당 간선을 T에서 삭제하였을 때 T의 임의의 정점을 분리 (Disconnect)시키지 않으면 삭제한다. 그렇지 않으면 삭제하지 않는다.

(4) E에서 더 이상 선택할 간선이 없을 때까지 (3)을 반복적으로 수행하고 알고리즘을 종료한다.

2. 알고리즘 적용 문제점과 연구 배경

Kruskal과 역-삭제 MST 알고리즘을 그림 1의 G_1 그래프에 적용하여 MST를 구하여 보자. G_1 그래프는 Peiper[10]로부터 인용되었으며, 모든 간선들의 가중치가 상이한 경우이다. G_1 그래프에 대해 Kruskal 알고리즘을 적용하여 MST를 구하는 과정은 그림 2에 제시하였다 .Kruskal MST 알고리즘을 적용한 결과 $|e_{mst}| = 5$ 와 $\Sigma w(e) = 21$ 를 얻었으며, 사이클이 발생하지 않아 MST를 얻음을 알 수 있다.

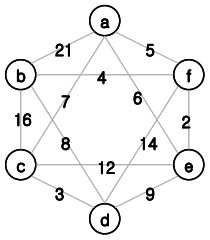


그림 1. G_1 그래프
Fig. 1. Graph G_1

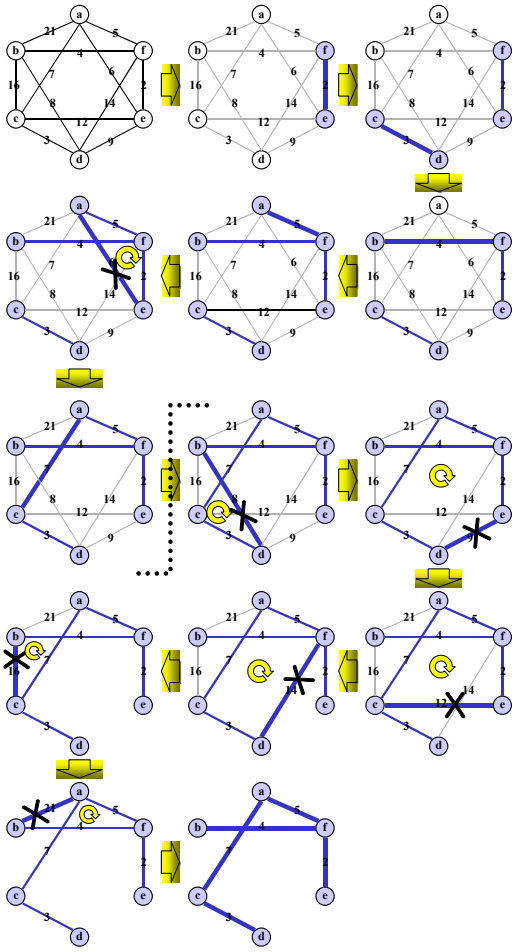


그림 2. G_1 그래프의 Kruskal 알고리즘 적용 MST
Fig. 2. MST applied to kruskal algorithm with graph G_1

Kruskal MST 알고리즘은 E의 간선들의 가중치를 대상으로 오름차순으로 정렬시키고 한 번에 하나씩 선택하여 사이클이 발생하지 않으면 신장트리를 구성한다. 또한, E의 모든 간선들을 대상으로 한다. 여기서 발생하는 문제점은 MST의 간선의 수가 $|v|-1$개가 되는 간선 $w(\{a,c\})=7$을 연결한 시점에서 이미 MST를 얻었음에도 불구하고 $w(\{b,d\})=8, w(\{d,e\})=9, w(\{c,e\})=12, w(\{d,f\})=14, w(\{b,c\})=16,$ $\})=16, w(\{a,b\})=21$를 추가로 확인하는 과정을 불필요하게 수행하였다.

G_1 그래프에 대해 역-삭제 알고리즘을 적용하여 MST를 구하는 과정은 그림 3에 제시하였다. 역-삭제 MST 알고리즘은 E의 간선들의 가중치를 대상으로 내림차순으

로 정렬시키고 가중치가 가장 큰 간선부터 한 번에 하나씩 선택하여 임의의 한 정점을 분리시키지 않는 한 제거하는 방식이다. 또한, E의 모든 간선들을 대상으로 한다. 여기서 발생하는 문제점은 간선 $w(\{a,e\})=6$을 제거한 시점에서 이미 MST를 얻었으며, 이 때 MST의 간선의 수는 $|v|-1$이다. 결국, $w(\{a,f\})=5, w(\{b,f\})=4, w(\{c,d\})=3,\ \ w(\{e,f\})=2$은 불필요하게 추가로 알고리즘이 수행된 경우이다.

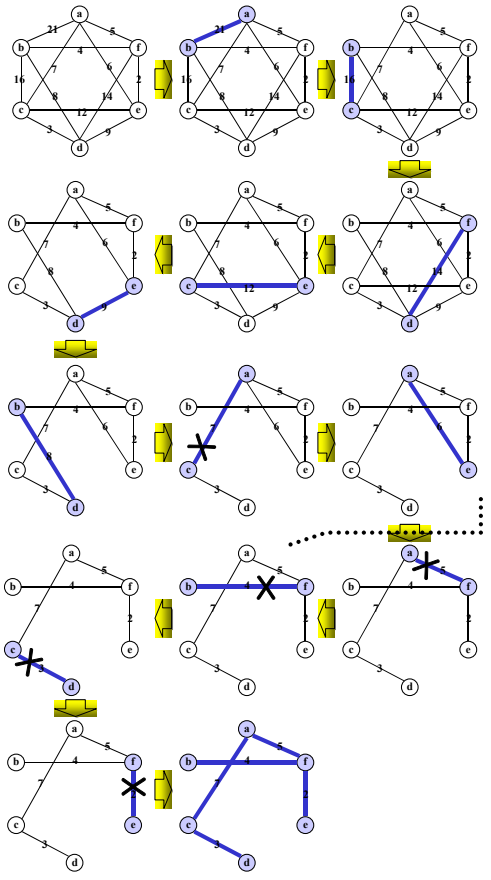


그림 3. G_1 그래프의 역-삭제 알고리즘 적용 MST
Fig. 3. MST applied to reverse-delete algorithm with graph G_1

즉, 2개 알고리즘 모두 이미 MST를 얻은 상태에서 불필요하게 추가적으로 알고리즘이 수행된다. 3장에서는 이렇게 추가적으로 알고리즘이 수행되는 것을 방지하여 알고리즘을 보다 빠르게 종료를 시킬 수 있는 방법을 제안한다.

Ⅲ. Kruskal과 역-삭제 MST 알고리즘의 효율적 구현 방법

1. 제안 알고리즘

본 논문은 Kruskal과 역-삭제 MST 알고리즘의 수행 횟수를 줄일 수 있는 2가지 방법을 제안한다. 이를 "[제1방법]"과 "[제2방법]"이라 하자.

Kruskal 알고리즘 [제1방법]은 Kruskal 알고리즘과 동일하게 모든 간선들을 대상으로 하며, 그래프의 모든 정점 집합 V와 부분 신장트리 집합 T를 갖는다. 부분 신장트리 집합 T는 부분집합 T_i $(i=1,2,\cdots)$를 갖고 있다. 초기치로 V는 모든 정점들의 집합으로, $T_i=\{\varnothing\}$으로 설정한다.

(1) 그래프의 가중치에 대해 $|v|\times|v|$ 정방행렬 (Square Matrix)을 작성하여 상삼각행렬 (Upper Triangle Matrix)에 대해서만 모든 간선들의 가중치를 오름차순으로 정렬시킨다.

(2) 가중치 간선들을 대상으로 최소 가중치를 갖는 간선부터 한 번에 하나씩 선택하여 사이클이 발생하지 않으면 부분 신장트리 부분집합 T_i에 저장한다. 이 과정을 가중치의 오름차순으로 반복적으로 수행한다.

if $x\in T_i, y\in T_j$ then $e_{mst}\leftarrow\{x,y\}$, $T_i=T_i+T_j$
else if $x\in T_i, y\in T_i$ then skip /* 사이클 발생
else if $x\in T_i, y\notin T_i$ then $e_{mst}\leftarrow\{x,y\}$, $T_i=T_i+y$.

사이클 발생 여부는 $\{x,y\}\in T_i$ 즉, 2개의 정점 x와 y가 임의의 어느 한 부분신장트리 부분집합 T_i에 모두 포함되는 경우이다. 이 경우, 해당 간선을 부분신장트리에 추가하지 않는다. 이 방법은 최소신장트리의 사이클 속성 (Cycle Property)에 기반을 두고 있다. 사이클 속성은 임의의 간선을 부분신장트리에 추가시 사이클이 발생한다면, 새로 추가되는 간선의 가중치가 가장 큰 값을 갖고 있게 되며, 최소신장트리는 가장 큰 가중치를 갖는 간선을 포함하지 않는다는 속성이다.

(3) 알고리즘은 $|e_{mst}|=|v|-1$이 되는 시점에서 종료시킨다. 이 기준은 $V=\{\varnothing\}$이면서 T가 하나의 부분집합만으로 구성되는 경우이다.

역-삭제 알고리즘 [제1방법]은 역-삭제 알고리즘과 동일하게 모든 간선들을 대상으로 하며,

(1) 그래프의 가중치 정방행렬을 작성하여 상삼각행렬에 대해서만 모든 간선들의 가중치를 내림차순으로 정렬시킨다.

(2) 최대 가중치 간선부터 해당 간선이 삭제되었을 때 T에 있는 임의의 정점을 분리시키지 않는 한 제거한다. 간선 제거는 다음 기준을 적용한다.

각 정점의 차수 $d_G(v)$, $|v|$와 $|e|$를 구한다.

for $i \leftarrow |e|$ downto $|e|=|v|-1$

do 최대 가중치 간선 $\{u,v\}$의 $N_G(u)$와 $N_G(v)$를 구한다.

단, $N_G(u) \leftarrow N_G(u) \setminus v$, $N_G(v) \leftarrow N_G(v) \setminus u$.

if $N_G(u) \cap N_G(v) \neq \varnothing$ or $\{N_G(u), w\}$, $\{N_G(v), w\}$인 w 존재 then $|e|=|e|-1$, $E \leftarrow E \setminus \{u,v\}$.

(3) 알고리즘은 $|e_{mst}|=|v|-1$이 되는 시점에서 종료시킨다. 이 기준은 알고리즘 시작 초기에 $|e_{mst}|=|e_t|$로 설정하고, 해당 간선이 제거될 때마다 $|e_{mst}|-1$을 수행하면서, $|e_{mst}|=|v|-1$이 되는 시점에서 알고리즘이 종료된다.

[제2방법]은 E의 모든 정점을 대상으로 하지 않고, 사전에 불필요한 간선을 가능한 많이 제거한 상태에서 MST를 구성하는 방법이다. 그래프의 간선의 수를 그림 4와 같이 표현하여 보자. 여기서, 우리가 찾고자 하는 간선의 수는 $|e_{mst}|=|v|-1$이며, $|e_\alpha|$는 MST를 찾는 과정에서 사이클이 발생하여 제거되는 간선의 수, $|e_\beta|$는 사전에 제거할 수 없는 사이클이 발생하는 불필요한 간선의 수, $|e_\gamma|$는 사전에 어떠한 기준을 적용하여 제거할 수 있는 사이클이 발생하는 불필요한 간선의 수로 정의한다.

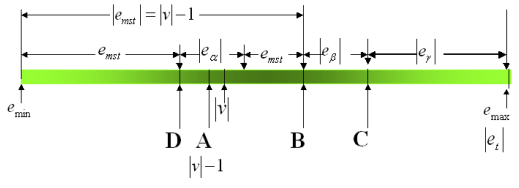


그림 4. 그래프의 간선의 수 표현
Fig. 4. Edge number representation of graph

그래프의 모든 간선들의 가중치를 오름차순으로 정렬하였을 경우, 최소 가중치로 구성된 MST의 간선들은 이상적으로는 $|e_{mst}|=|v|-1$인 A 지점이어야 한다. 그러나 현실적으로는 적은 가중치를 갖고 있음에도 불구하고 사이클을 발생시키는 경우가 존재하여 $|e_\beta|$를 포함한 B 지점까지 검증해야만 한다. 따라서 Kruskal MST 알고리즘[5]을 적용할 경우, B 지점을 찾는 것이 최적의 해법 (Optimal Solution)이 된다. 그러나 현실적으로 사이클이 발생하는 간선을 사전에 검증할 방법이 없어 Kruskal 알고리즘[5]은 불필요하게 $|e_\beta|+|e_\gamma|$를 포함하여 최대 가중치를 갖고 있는 $e_{\max}$까지 모든 간선들을 대상으로 알고리즘이 수행된다. 여기서 알고리즘 수행에 적용되는 간선의 총 수를 e_c라 하자. 만약, MST에 전혀 기여를 하지 않아 불필요한 간선들 $|e_\beta|+|e_\gamma|$ 중에서 어느 정도의 간선들 (여기서는 $|e_\gamma|$)을 기준에 따라 사전에 제거할 수 있다면 알고리즘을 수행하는 시간을 단축시킬 수 있을 것이다. [제2방법]은 이러한 이론에 기반을 두고 있다. MST에 기여를 하지 못하는 불필요한 간선을 사전에 가능한 많이 제거하는 $|e_\gamma|$의 개수는 다음 기존을 적용한다. 이를 사전처리 (Pre-Processing)라 하자.

(1) 그래프의 가중치 정방행렬을 작성한다.

(2) 모든 행 (또는 열) 정점들에 대해 최대값을 갖는 가중치를 삭제한다. 단, 첫 번째 행 정점은 차수가 3 이상일 때, 두 번째부터는 차수가 2 이상일 때만 적용한다. 또한, 최대 가중치가 동일한 값이 다수 존재하면 첫 번째 위치한 값을 제거한다.

(3) (2)에서 삭제한 모든 간선 $\{x,y\}$에 대해 $\{y,x\}$ 간선 가중치를 삭제한다.

(4) 만약, 어느 한 정점에 대해 가중치를 갖는 간선들이 모두 삭제되는 경우, 최소 가중치를 갖는 간선은 삭제하지 않는다.

이 방법은 그래프의 간선들이 너무 많이 존재하여 어느 한 정점을 기준으로 하였을 경우, 해당 정점에 부속된 간선들 중에서 최대 가중치를 갖는 간선을 삭제하여도 MST를 얻는 데는 지장을 주지 않을 수 있다는 가정에 근거한다. 그러나 이 이론은 아직 증명된 바는 없으며, 본 논문에서는 이 이론이 가능하다고 가정한다. 따라서 사전처리를 거친 결과 얻어진 간선들은 $|e_c|=|e_{mst}|+|e_\alpha|+|e_\beta|$가 된다.

Kruskal 알고리즘 [제2방법]은 사전처리 기준을 적용하여 선택된 간선들을 대상으로 최소 가중치를 갖는 e_{min}에서 시작하여 C 지점까지의 간선들을 대상으로 하여 정확히 B 지점에서 알고리즘이 종료된다.

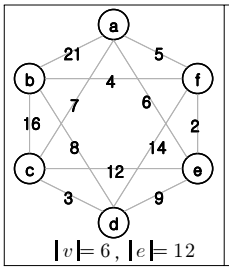
역-삭제 알고리즘 [제2방법]은 사전처리 기준을 적용하여 선택된 간선들을 대상으로 최대 가중치를 갖는 C 지점에서 시작하여 해당 간선이 신장트리 T에서 삭제되면 임의의 정점을 분리시키지 않는 한 삭제하면서 정확히 D 지점에서 알고리즘이 종료된다.

본 논문에서 제안하는 4가지 방법 중 역-삭제 알고리즘 [제2방법]이 가장 적은 알고리즘 수행 횟수를 나타낼 수 있다.

2. 제1방법 적용

먼저, 주어진 G_1 그래프의 가중치 간선들에 대해 $|v|\times|v|$ 정방행렬을 그림 5와 같이 작성하여 상삼각행렬에 대한 간선들의 가중치를 모두 선택한다. 이를 Candidate Edges라 하자. 여기서는 $w(\{x,y\})$를 줄여서 $\{x,y\}$로 표기하였다.

먼저, Kruskal 알고리즘 [제1방법]은 표 1과 같이 수행된다. Kruskal 알고리즘[5]은 간선의 수 12개 모두에 대해 사이클을 확인하므로 12회 반복 수행으로 5개의 간선을 얻는다. 반면에 Kruskal 알고리즘 [제1방법]은 12개의 간선에 대해 6회 수행에서 알고리즘이 종료되고 5개의 간선을 얻는다.



	a	b	c	d	e	f
a		**21**	**7**		**6**	**5**
b	21		**16**	**8**		**4**
c	7	16		**3**	**12**	
d		8	3		**9**	**14**
e	6		12	9		**2**
f	5	4		14	2	

그림 5. G_1 **그래프의 가중치 정방행렬**
Fig. 5. Weighted square matrix with graph G_1

다음으로 역-삭제 알고리즘 [제1방법]은 표 2와 같이 수행된다. 역-삭제 알고리즘은 간선의 수 12개 모두에 대해 사이클을 확인하므로 12회 반복 수행으로 5개의 간선을 얻는다. 반면에 역-삭제 알고리즘 [제1방법]은 12개의 간선에 대해 8회 수행에서 알고리즘이 종료되고 5개의 간선을 얻는다.

표 1. G_1 **그래프의 Kruskal 알고리즘 [제1방법]**
Table 1. Kruskal algorithm of graph G_1 [first method]

Candidate Edges			T	V	T Edges
E	오름차순 정렬	사이클 확인			
$\{a,b\}=21$	$\{e,f\}=2$	×	$\{\varnothing\}$	$\{a,b,c,d,e,f\}$	$\{\varnothing\}$
$\{a,c\}=7$	$\{c,d\}=3$	×	$\{e,f\}$	$[a,b,c,d]$	$\{e,f\}=2$
$\{a,e\}=6$	$\{b,f\}=4$	×	$\{e,f\}\{c,d\}$	$[a,b]$	$\{c,d\}=3$
$\{a,f\}=5$	$\{a,f\}=5$	×	$\{b,e,f\}\{c,d\}$	$[a]$	$\{b,f\}=4$
$\{b,c\}=16$	$\{a,e\}=6$	○	$\{a,b,e,f\}\{c,d\}$	$\{\varnothing\}$	$\{a,f\}=5$
$\{b,d\}=8$	$\{a,c\}=7$	×	$\{a,b,e,f\}\{c,d\}$	$\{\varnothing\}$	–
$\{b,f\}=4$	$\{b,d\}=8$	–	$\{a,b,c,d,e,f\}$	$\{\varnothing\}$	$\{a,c\}=7$
$\{c,d\}=3$	$\{d,e\}=9$	–	**(알고리즘 종료)**		
$\{c,e\}=12$	$\{c,e\}=12$	–			
$\{d,e\}=9$	$\{d,f\}=14$	–			
$\{d,f\}=14$	$\{b,c\}=16$	–			
$\{e,f\}=2$	$\{a,b\}=21$	–			
$\|e_t\|=12$				수행횟수 : 6회	$\|e_{mst}\|=5$

표 2. G_1 **그래프의 역-삭제 알고리즘 [제1방법]**
Table 2. Reverse-delete algorithm of graph G_1 [first method]

Candidate Edges			사이클	간선 삭제	$\|e_{mst}\|$
E	내림차순 정렬	사이클 확인			
$\{a,b\}=21$	$\{a,b\}=21$	○	$a-f-b-a$	$\{a,b\}=21$	12
$\{a,c\}=7$	$\{b,c\}=16$	○	$b-d-c-b$	$\{b,c\}=16$	11
$\{a,e\}=6$	$\{d,f\}=14$	○	$d-e-f-d$	$\{d,f\}=14$	10
$\{a,f\}=5$	$\{c,e\}=12$	○	$c-d-e-c$	$\{c,e\}=12$	9
$\{b,c\}=16$	$\{d,e\}=9$	○	$d-b-f-e-d$	$\{d,e\}=9$	8
$\{b,d\}=8$	$\{b,d\}=8$	○	$b-f-a-c-d-b$	$\{b,d\}=8$	7
$\{b,f\}=4$	$\{a,c\}=7$	×	×	–	6
$\{c,d\}=3$	$\{a,e\}=6$	○	$a-f-e-a$	$\{a,e\}=6$	6
$\{c,e\}=12$	$\{a,f\}=5$	–	**(알고리즘 종료)**		5
$\{d,e\}=9$	$\{b,f\}=4$	–			
$\{d,f\}=14$	$\{c,d\}=3$	–			
$\{e,f\}=2$	$\{e,f\}=2$	–			
$\|e_t\|=8$				수행횟수 : 8회	$\|e_{mst}\|=5$

3. 제2방법 적용

먼저, 사전처리 단계에서 주어진 G_1 그래프의 가중치 간선들에 대한 그림 5의 $|v|\times|v|$ 정방행렬에 대해 각 행의 정점들에 대한 최대 가중치를 갖는 간선을 삭제하여 그림 6과 같이 얻는다. 상삼각행렬에 대한 간선들의 가중치를 모두 선택한다.

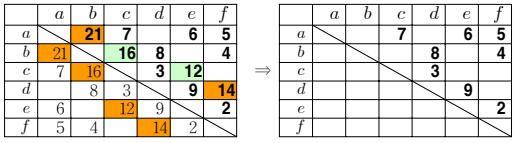


그림 6. G_1 그래프의 불필요한 가중치 제거 상삼각행렬
Fig. 6. Unnecessary weight deletion upper triangular matrix with graph G_1

Kruskal 알고리즘 [제2방법]은 표 3과 같이 수행된다. 제안된 제2방법은 12개의 간선에 대해 8개를 대상으로 하여 6회 수행 만에 알고리즘이 종료되고 5개의 간선을 얻는다. 역-삭제 알고리즘 [제2방법]은 표 4와 같이 수행된다. 12개의 간선에 대해 8개를 대상으로 하여 4회 수행만에 알고리즘이 종료되고 5개의 간선을 얻는다. 따라서 알고리즘 수행 속도가 가장 빠른 방법이다.

표 3. G_1 그래프의 Kruskal 알고리즘 [제2방법]
Table 3. Kruskal algorithm of graph G_1 [second method]

Candidate Edges			T	V	T Edges
E	오름차순 정렬	사이클 확인			
$\{a,c\}=7$	$\{e,f\}=2$	×	$\{\varnothing\}$	$\{a,b,c,d,e,f\}$	$\{\varnothing\}$
$\{a,e\}=6$	$\{c,d\}=3$	×	$\{e,f\}$	$[a,b,c,d\}$	$\{e,f\}=2$
$\{a,f\}=5$	$\{b,f\}=4$	×	$\{e,f\}\{c,d\}$	$[a,b\}$	$\{c,d\}=3$
$\{b,d\}=8$	$\{a,f\}=5$	×	$\{b,e,f\}\{c,d\}$	$[a\}$	$\{b,f\}=4$
$\{b,f\}=4$	$\{a,e\}=6$	○	$\{a,b,e,f\}\{c,d\}$	$\{\varnothing\}$	$\{a,f\}=5$
$\{c,d\}=3$	$\{a,c\}=7$	×	$\{a,b,e,f\}\{c,d\}$	$\{\varnothing\}$	–
$\{d,e\}=9$	$\{b,d\}=8$	–	$\{a,b,c,d,e,f\}$	$\{\varnothing\}$	$\{a,c\}=7$
$\{e,f\}=2$	$\{d,e\}=9$	–	**(알고리즘 종료)**		
$\lvert e_t\rvert=8$				수행횟수 : 6회	$\lvert e_{mst}\rvert=5$

표 4. G_1 그래프의 역-삭제 알고리즘 [제2방법]
Table 2. Reverse-delete algorithm of graph G_1 [second method]

Candidate Edges			사이클	간선 삭제	$\lvert e_{mst}\rvert$
E	내림차순 정렬	사이클 확인			
$\{a,c\}=7$	$\{d,e\}=9$	○	$d-b-f-e$	$\{d,e\}=9$	8
$\{a,e\}=6$	$\{b,d\}=8$	○	$b-f-a-c-d$	$\{b,d\}=8$	7
$\{a,f\}=5$	$\{a,c\}=7$	×	$a-f-e-a$	–	6
$\{b,d\}=8$	$\{a,e\}=6$	○	$a-f-e$	$\{a,e\}=6$	6
$\{b,f\}=4$	$\{a,f\}=5$	–	**(알고리즘 종료)**		5
$\{c,d\}=3$	$\{b,f\}=4$	–			
$\{d,e\}=9$	$\{c,d\}=3$	–			
$\{e,f\}=2$	$\{e,f\}=2$	–			
$\lvert e_t\rvert=8$				수행횟수 : 4회	$\lvert e_{mst}\rvert=5$

제안한 2개 방법을 적용하여 최소신장트리를 얻은 결과는 그림 7과 같다. 결국, 4개 모델은 알고리즘 수행 횟수는 다르지만 모두 동일한 최소신장트리를 얻을 수 있었다.

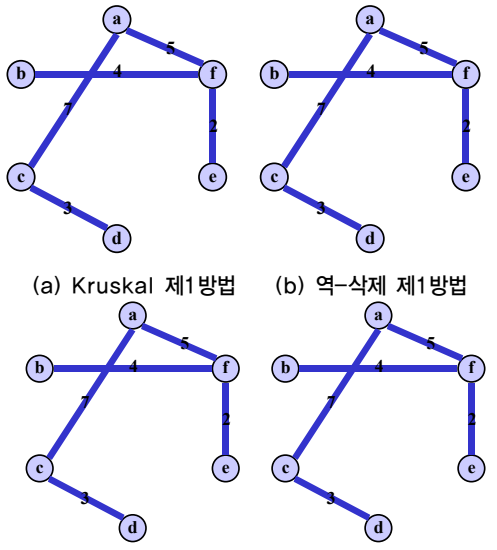
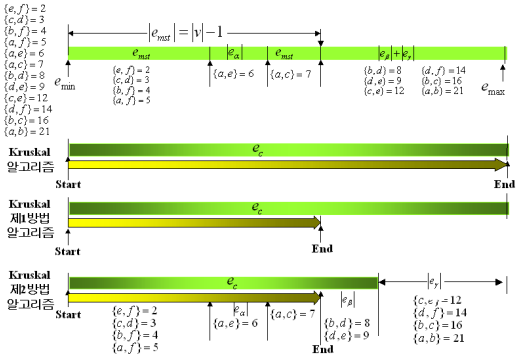
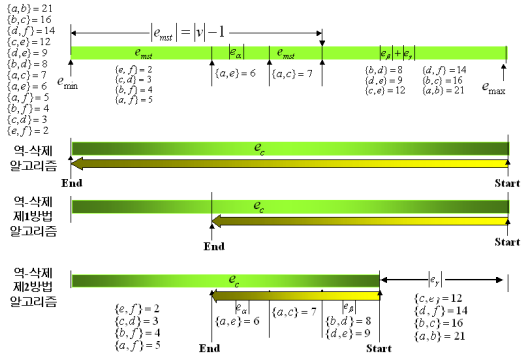


그림 7. G_1 그래프의 최소신장트리
Fig. 7. Minimum spanning tree of graph G_1

G_1 그래프에 대해 제안된 알고리즘을 기존의 Kruskal과 역-삭제 알고리즘과 비교하여 설명하면 그림 8과 같다.



(a) Kruskal 알고리즘



(b) 역-삭제 알고리즘

그림 8. 알고리즘 수행 횟수 비교
Fig. 8. Running time comparison of algorithms

Ⅳ. 알고리즘 적용성 평가

본 장에서는 그림 9와 같이 8개 그래프에 대해 알고리즘 적용성을 평가해 본다. G_2는 Wikipedia[14], G_3, G_4, G_6, G_9은 Peiper[10-13]에서, G_5, G_7, G_8는 Chen[9]에서 인용되었다.

1. 알고리즘 적용

그림 9의 8개 그래프에 대해 제안된 4개 알고리즘을 적용한 결과는 그림 10 ~ 그림 17에 제시하였다. 편의상 알고리즘 수행 과정을 제시하지 않았으며, 상단 좌측에는 가중치 정방행렬을, 우측에는 $|e_\gamma|$를 삭제하고 남아 있는 가중치들을 표현하였다. 하단에는 4개의 알고리즘으로 얻은 최소신장트리를 표현하였다. 8개 그래프 모두에서 제안된 4개 알고리즘 모두 최소신장트리를 얻었음을 알 수 있다.

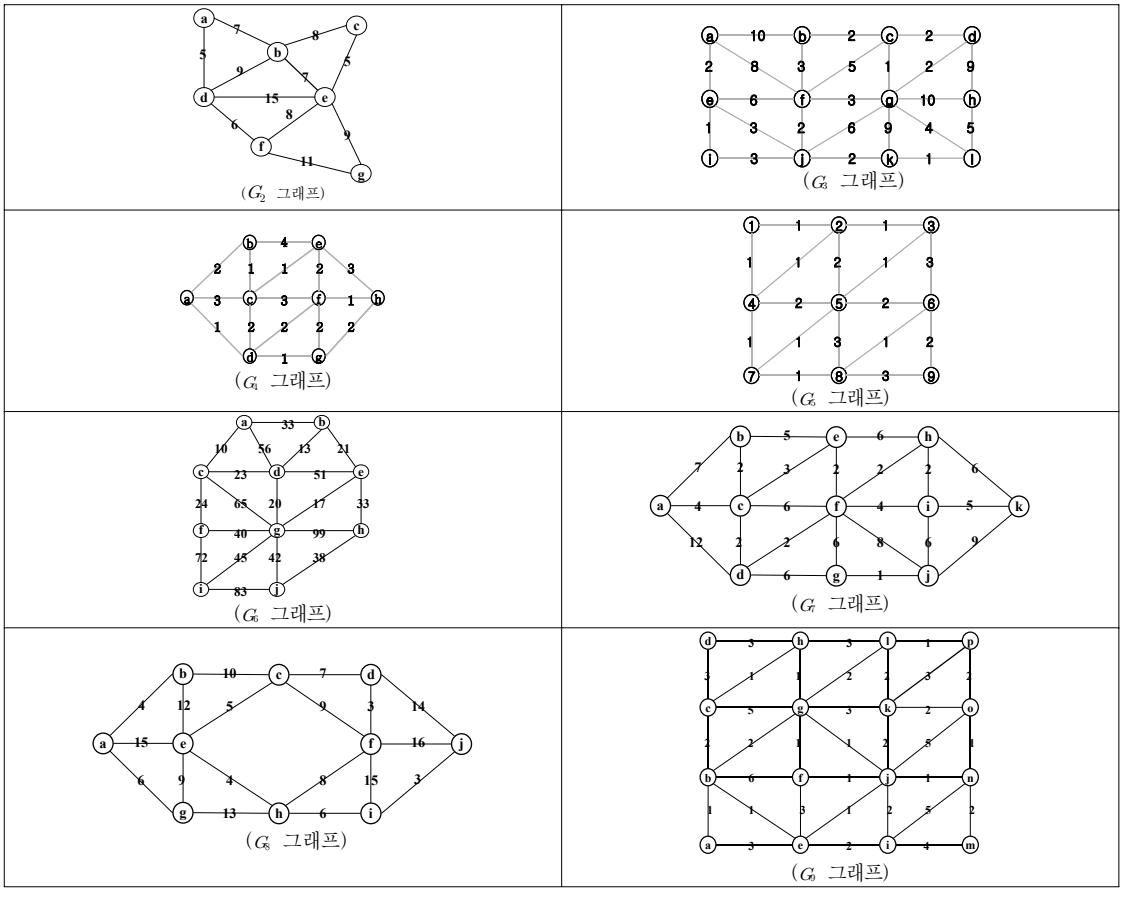


그림 9. 알고리즘 평가에 사용된 그래프
Fig. 9. Graphs used to test the algorithms

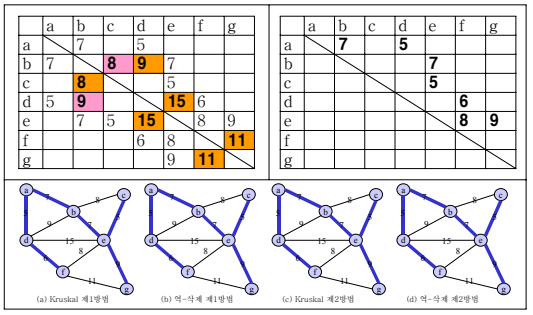


그림 10. G_2 그래프의 MST
Fig. 10. MST of graph G_2

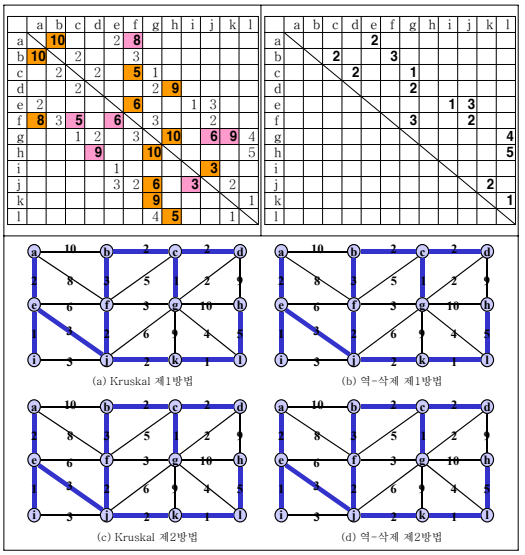


그림 11. G_3 그래프의 MST
Fig. 11. MST of graph G_3

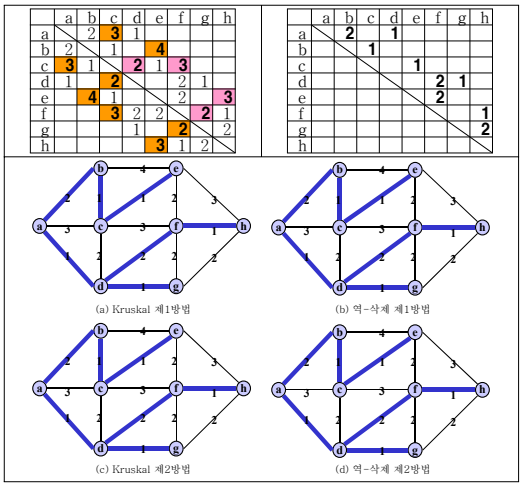


그림 12. G_4 그래프의 MST
Fig. 12. MST of graph G_4

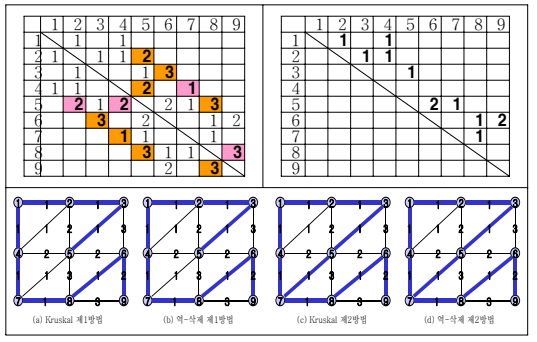


그림 13. G_5 그래프의 MST
Fig. 13. MST of graph G_5

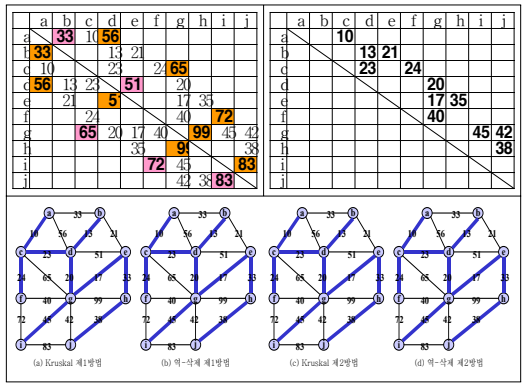


그림 14. G_6 그래프의 MST
Fig. 14. MST of graph G_6

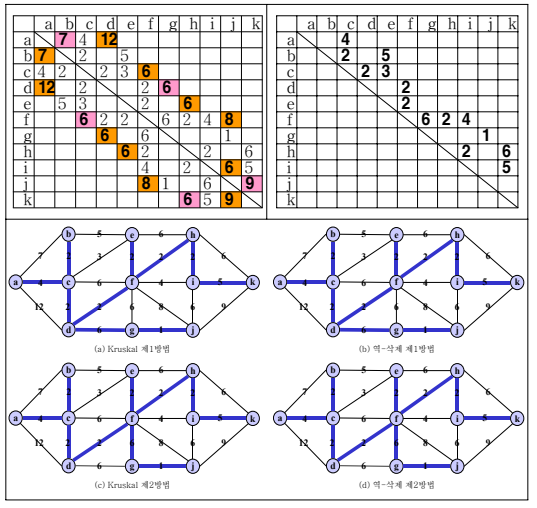


그림 15. G_7 그래프의 MST
Fig. 15. MST of graph G_7

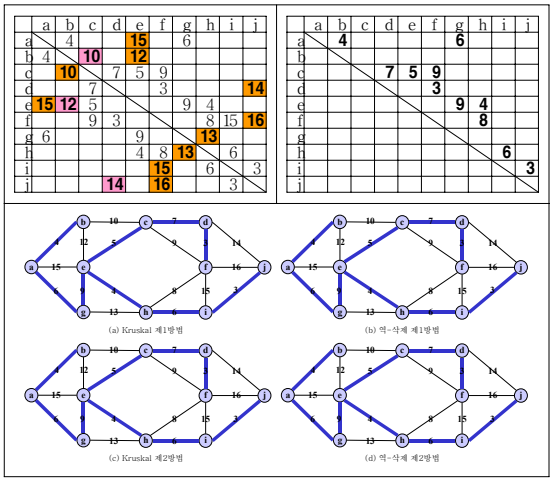


그림 16. G_8 그래프의 MST
Fig. 16. MST of graph G_8

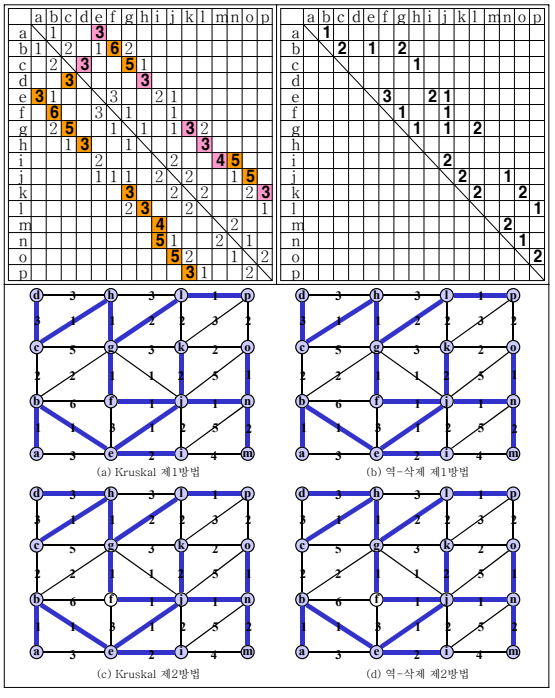


그림 17. G_9 그래프의 MST
Fig. 17. MST of graph G_9

2. 적용 결과 분석

본 논문에 적용된 9개 그래프에 대한 알고리즘 수행 횟수를 종합하여 표 5에 제시하였다. 제안된 4개 알고리즘이 기존의 Kruskal과 역-삭제 알고리즘에 비해 알고리즘 수행횟수 단축율이 얼마나 되는지를 비교한 결과는 표 6에 제시하였다. 이는 표 5의 결과로부터 유도되었다.

표 5. MST 알고리즘의 수행 횟수 비교
Table 5. Running time comparison of MST algorithm

그래프	$\|v\|$	$\|e\|$	구분	Kruskal과 역-삭제 알고리즘	[제1방법] Kruskal 알고리즘	[제1방법] 역-삭제 알고리즘	[제2방법] Kruskal 알고리즘	[제2방법] 역-삭제 알고리즘
G_1	6	12	대상 간선 수	12	12	12	8	8
			수행 횟수	12	6	8	6	4
G_2	7	11	대상 간선 수	11	11	11	7	7
			수행 횟수	11	9	6	7	2
G_3	12	23	대상 간선 수	23	23	23	14	14
			수행 횟수	23	16	17	14	8
G_4	8	15	대상 간선 수	15	15	15	9	9
			수행 횟수	15	8	9	7	2
G_5	9	16	대상 간선 수	16	16	16	10	10
			수행 횟수	16	13	13	10	7
G_6	10	19	대상 간선 수	19	19	19	12	12
			수행 횟수	19	13	15	12	8
G_7	11	22	대상 간선 수	22	22	22	14	14
			수행 횟수	22	14	15	13	7
G_8	16	33	대상 간선 수	33	33	33	23	23
			수행 횟수	33	23	26	22	16
G_9	10	18	대상 간선 수	18	18	18	11	11
			수행 횟수	18	11	10	11	3

표 6. 제안 알고리즘의 수행 횟수 성능 비교
Table 6. Running time comparison of the proposed algorithm

그래프	$\|v\|$	$\|e\|$	[제1방법] 알고리즘 수행횟수 단축(배) Kruskal 알고리즘	[제1방법] 알고리즘 수행횟수 단축(배) 역-삭제 알고리즘	[제2방법] 간선 제거 수	[제2방법] 알고리즘 수행횟수 단축(배) Kruskal 알고리즘	[제2방법] 알고리즘 수행횟수 단축(배) 역-삭제 알고리즘
G_1	6	12	2.00	1.50	4	2.00	3.00
G_2	7	11	1.22	1.83	4	1.57	5.50
G_3	12	23	1.44	1.35	9	0.61	2.88
G_4	8	15	1.88	1.67	6	2.14	7.50
G_5	9	16	1.23	1.23	6	1.60	2.29
G_6	10	19	0.68	1.27	7	1.58	2.38
G_7	11	22	1.57	0.68	8	1.69	3.14
G_8	16	33	1.43	1.27	10	1.50	2.06
G_9	10	18	1.64	1.80	7	1.64	6.00
최소 값			0.68	0.68	4	0.61	2.06
최대 값			2.00	1.83	10	2.14	7.50
평 균			1.45	1.40	6.78	1.59	3.86

비교 결과 Kruskal과 역-삭제 알고리즘 [제1방법]은 평균적으로 기존의 Kruskal과 역-삭제 알고리즘에 비해 평균적으로 1.4배 정도 빨리 알고리즘을 종료시키는 효과를 얻었다. 반면에, Kruskal 알고리즘 [제2방법]은 약 1.6배 정도 빨리 종료시킬 수 있는 효과를 얻어 [제1방법]과 별다른 차이를 보이지 않는데 비해, 역-삭제 알고리즘 [제2방법]은 3.86배 빨리 알고리즘을 종료시키는 효과를 얻었다. 결국, Kruskal 알고리즘[제1방법]이 역-

삭제 알고리즘 [제1방법]방법 보다 약간 빠른 결과를, Kruskal 알고리즘 [제2방법]이 Kruskal 알고리즘 [제1방법] 보다 약간 빠른 결과를 나타내었다. 다른 3가지 알고리즘보다 역-삭제 알고리즘 [제2방법]이 가장 빨리 알고리즘을 종료시키는 효과를 얻었다.

V. 결론

본 논문은 최소신장트리를 구하는 Kruskal과 역-삭제 알고리즘을 실제 그래프에 적용하여 문제점을 고찰하였다. 기존의 Kruskal과 역-삭제 알고리즘은 그래프의 모든 간선들을 대상으로 사이클이 발생하는지 여부를 검증한다. 이로 인해 알고리즘 수행 과정에서 이미 최소신장트리를 얻었음에도 불구하고 나머지 간선들에 대해 알고리즘을 추가로 불필요하게 수행하는 문제점을 갖고 있다.

본 논문은 먼저, Kruskal과 역-삭제 알고리즘과 동일하게 모든 간선들을 대상으로 알고리즘은 수행하지만 알고리즘 종료 시점 기준을 적용하여 수행 횟수를 줄이는 [제1방법]을 제안하였다. 다음으로, 최소신장트리에 전혀 영향을 미치지 않는 불필요한 간선을 사전에 제거하고 남은 간선들을 대상으로 최소신장트리를 찾는 [제2방법]을 제안하였다.

제안된 방법들을 실제 그래프들에 적용한 결과 기존의 Kruskal과 역-삭제 알고리즘보다 최소 1.4배에서 최대 3.86배 빨리 알고리즘을 종료시키는 효과를 얻었다. 제안된 2개 방법을 2개 알고리즘에 적용한 4개 알고리즘 중에서 역-삭제 알고리즘 [제2방법]이 가장 빨리 알고리즘을 종료시키는 결과를 얻었다.

참 고 문 헌

[1] Wikipedia, "Minimum Spanning Tree," http://en.wikipedia.org/ wiki/Minimum_spanning_tree, Wikimedia Foundation, Inc., 2007.

[2] O. Borůvka, "O Jistem Problemu Minimalnim," Prace Mor. Prrodved. Spol. V Brne (Acta Societ. Natur. Moravicae), Vol. III, No. 3, pp. 37-58, 1926.

[3] J. Nešetřil, E. Milková, and H. Nešetřilová, "Otakar Borůvka on Minimum Spanning Tree Problem (Translation of the both 1926 Papers, Comments, History)," DMATH: Discrete Mathematics, Vol. 233, 2001.

[4] R. C. Prim, "Shortest Connection Networks and Some Generalisations," Bell System Technical Journal, Vol. 36, pp. 1389-1401, 1957.

[5] J. B. Kruskal, "On the Shortest Spanning Subtree and The Traveling Salesman Problem," Proceedings of the American Mathematical Society, Vol. 7, pp. 48-50, 1956.

[6] Wikipedia,"Reverse-Delete Algorithm," http://en.wikipedia.org/ wiki/Reverse_delete_algorithm, Wikimedia Foundation, Inc., 2007.

[7] Wikipedia, "Graph (mathematics)," http://en.wikipedia.org/wiki/ Graph_(mathematics), Wikimedia Foundation, Inc., 2010.

[8] Wikipedia, "Glossary of Graph Theory," http://en.wikipedia.org/ wiki/Glossary_of_graph_theory, Wikimedia Foundation, Inc., 2010.

[9] WWL. Chen, "Discrete Mathematics," Department of Mathematics, Division of ICS, Macquarie University, Australia, http://www.maths.mq.edu.au/~wchen/lndmfolder/lndm.html, 2003.

[10] C. Peiper, CS 400 - Data Structures for Non CS-Majors," http:// www.cs.uiuc.edu/class/fa05/cs400/_labs/Lab12/suuri/, 2005.

[11] H. K. Park, C. H. Lee, "Embedded System Implementation of Tree Routing Structure for Ubiquitous Sensor Network," Journal of the Korea Academia-Industrial cooperation Society, v.11 no.10, pp.4531-4535, October 2011.

[12] W. J. Wang, C. S. Han, "Bond Graph Modeling, Analysis and Control of Dual Stage System," Journal of the Korea Academia-Industrial cooperation Society, v.13, no.4, pp.1453-1459, April 2012.

[13] M. B. Choi, S. U. Lee, "A Prim Minimum Spanning Tree Algorithm for Directed Graph," IWIT, v.12 no.3, 2012.

[14] Wikipedia, "Prim's Algorithm," http://en.wikipedia.org/wiki/ Prims_algorithm, Wikimedia Foundation, Inc., 2010.

저자 소개

최 명 복(종신회원)

- 1997년 ~ 현재 : 강릉원주대학교 멀티미디어공학과 교수
- 2004년 1월 ~ 현재 : 한국인터넷방송통신학회 이사

<주관심분야 : 알고리즘 등>

- E-mail : cmb5859@gmail.com

이 상 운(정회원)

- 2007년 3월 ~ 현재 : 강릉원주대학교 과학기술대학 멀티미디어공학과 부교수

<주관심분야 :소프트웨어공학 등>

- E-mail : sulee@gwnu.ac.kr