# 적은 굴곡점을 가진 이진트리를 그리는 알고리즘

## (An Algorithm for Drawing Binary Trees with Less Bends)

김 성 권 †

(Sung Kwon Kim)

**요 약** 본 논문에서는 $n$개의 노드로 이루어진 이진트리를 평면에 그리는 한 방법인 평면 다각선 상향 순서보존 그리드 방법을 이용하여 높이 $O(n)$, 폭 $O(\log n)$, 면적 $O(n\log n)$은 기존의 결과와 같게 유지하면서, 굴곡점의 수를 $O(n)$에서 $O(n/\log n)$으로 줄이는 알고리즘을 제시한다.

***Abstract*** In this paper we present a planar polyline upward order-preserving grid drawing algorithm for binary trees with $n$ vertices that achieves $O(n)$ height, $O(\log n)$ width and $O(n\log n)$ area, matching the previously known results, and that reduces the number of bends to $O(n/\log n)$ from $O(n)$.

## 1. 서 론

그림 하나가 말 천 마디보다 낫다는 말이 있다. 우리 일상 생활의 많은 분야에서 그림이 정보를 전달하는 방법으로 혹은 의미를 명확히 하는 수단으로 사용된다. 이 때 그림은 명료하고, 보기 편해야만 의미가 있다. 그래프는 추상적인 의미로는 노드와 에지의 집합으로 표현되며, 이를 사람들이 이해하기 쉽도록 평면상에 그림으로 그려주는 경우가 많다. 논문이나 책에 나오는 많은 그래프들이 독자들이 보기에 아무런 거부감이 없게 그려져 있는데 이유는 그 그래프의 특성을 잘 파악하여 나름대로의 미적 기준에 따라 그려져 있기 때문이다.

그래프는 어떤 객체들의 상호 연관 관계를 추상적으로 나타내는 모델링 도구인데, 이 그래프를 평면상에 그리는 그래프 그리기 (graph drawing)는 다양한 분야에 응용을 가지고 있다. 예를 들어, 복잡한 구조를 가지는 전기 기계의 부속품 연결도나, 병렬 컴퓨터와 같은 복잡한 구조를 이해하기 좋게 그리는 것 등 그림 그 자체에 관심이 있는 경우 뿐 아니라, VLSI 설계에서 부품 배치나 연결선 배선을 해결하는 작업과 같은 곳에도 그래프 그리기는 응용될 수 있다. 이런 작업은 그 크기가 엄청나기 때문에 인간의 수작업으로는 불가능하고 컴퓨터의 힘을 빌려야만 가능하다. 따라서 이를 효율적으로 수행할 수 있는 알고리즘의 개발이 필수적이다.

본 논문에서는 그래프의 범위를 좁혀 트리를 평면에 그리는 알고리즘에 대해서 연구하고, 이에 대한 결과를 제시한다. 근본적으로 트리는 추상적인 자료구조로서 그림은 단지 트리의 이해성을 높여 주기 위한 한 도구일 뿐이다. 그러나 응용에 따라서는 트리를 실제로 종이와 같은 평면에 그리고 이 결과를 이용하는 분야가 많이 있다. 이럴 경우 다양한 방법의 트리 그리기 방법이 적용될 수 있다. 가장 보편적인 방법은 각 노드를 정수 좌표를 가진 점으로 그리고, 각 에지는 양끝 노드에 해당하는 두 점을 연결하는 조단 곡선 (Jordan curve)으로 그리는 것이다. 트리를 평면에 그린 결과를 드로잉이라 하는데, 드로잉이 그려질 때 지켜야할 규칙에 따라서 여러 종류의 드로잉 방법을 생각할 수 있다. 참고문헌 [1, 2, 3, 4, 5, 6]에 여러 방법에 따라 트리를 그리는 알고리즘들이 나와 있다. 여러 개의 규칙들이 동시에 한 드로잉에 적용될 수도 있다.

본 논문에서는 **평면 다각선 상향 순서보존 그리드** 방법으로 트리를 그리는 방법에 대해서 연구한다. 이 그리기 방법을 자세히 설명하면 다음과 같다.

(i) **평면** (planar): 평면에 그려진 에지들이 겹치거나

· 본 논문은 1997년 한국학술진흥재단의 공모과제 연구비에 의하여 연구되었음.

† 종신회원 : 중앙대학교 컴퓨터공학과 교수
skkim@cau.ac.kr

논문접수 : 1999년 8월 17일
심사완료 : 1999년 12월 3일

교차하지 않게 그려야 한다. 이는 에지가 교차하거나 겹치게 되면 드로잉을 보는데 이해성이 떨어질 뿐 아니라, 응용에 따라서는 겹치거나 교차하는 것을 금지하는 경우가 있다.

(ii) **다각선** (polyline): 각 에지를 선분들이 연속으로 이어져 나오는 단순 (simple) 다각선으로 그리는 것을 말한다. 단순이라는 것은 자신이 자신을 교차해서는 안된다는 것을 의미한다.

(iii) **상향** (upward): 자식노드의 y좌표가 부모노드의 y좌표와 같을 수는 있으나 클 수는 없다. 일반적으로 트리의 에지를 생각할 때 부모노드가 위에 있고 자식노드가 그보다 아래 있는 것으로 생각하는 것이 일반적인 사항이므로, 이를 지키는 것을 말한다.

(iv) **순서보존** (order-preserving): 각 노드에 대해서 자식노드들의 좌우 순서를 지킨다. 자식노드들의 순서를 무시하고 그리는 방법도 있으나, 주어진 트리가 순서가 중요한 (ordered) 트리일 경우에는 드로잉에서도 이를 반드시 지켜야 하는 것을 의미한다.

(v) **그리드** (grid): 문헌에 나와 있는 거의 모든 그래프 드로잉이 이 사항을 만족하므로 일반적으로 생략을 하는 경우가 많은데, 모든 노드는 물론, 모든 굴곡점의 좌표가 정수 좌표를 가져야 한다는 것을 뜻한다. **굴곡점** (bend)라는 것은 다각선에서 한 선분과 그 다음 선분이 만나는 점으로 이곳에서 다각선이 구부러지기 때문에 그렇게 부른다. 각 굴곡점은 일종의 가상적인 노드로 생각할 수 있으므로 이를 요구한다.

드로잉이 있으면 이를 포함하는 (변이 수평이나 수직인) 가장 작은 직사각형을 생각할 수 있는데, 드로잉의 **폭**은 이 직사각형의 폭을 의미하고, 드로잉의 **높이**는 이 직사각형의 높이를 의한다. 또, 드로잉의 **면적**은 이 직사각형의 면적을 의미한다.

트리를 그릴 때는 드로잉 방법에 따라 같은 트리라도 다르게 그려진다. 이 경우에 생각할 수 있는 목적함수가 여러 가지 있다. 먼저, 드로잉의 면적을 최소화하는 것이다. 이는 좁은 면적에 큰 트리를 그리는 것이 여러 면에서 유리하므로 생각할 수 있는 목적함수이다. 다음으로는 드로잉의 폭과 높이의 비, 종횡비 (aspect ratio)를 생각할 수 있다. 종횡비가 1에 가까우면 드로잉이 전체적으로 정사각형과 비슷한 모양이 되고, 그렇지 않으면 폭에 비해 높이가 높은 길쭉한 모양을 할 수도 있고, 반대로 폭이 높이에 비해 큰 옆으로 긴 모양을 할 수도 있다. 트리 드로잉의 응용에 따라서는 종횡비를 마음대로 조절하여 폭과 높이를 정할 수 있으면 좋은 경우가 많다. 에지를 직선이 아닌 다각선으로 그리는 경우는, 다각선의 굴곡점의 수가 되도록 작은 것이 좋다. 에지가 여러 번 구부러지는 복잡한 모습으로 그려지게 되면 보는 사람이 드로잉을 이해하기 힘들기 때문이고, 응용에 따라서는 굴곡점 하나에 대해서 일정 비용을 지불해야 하는 경우도 있기 때문이다.

이런 목적함수는 단독으로 쓰이는 경우는 드물고, 대게 두 가지 이상을 복합적으로 사용하는 경우가 많다. 예를 들어서, 면적을 줄이면서 종횡비를 함께 고려하는 경우도 있으며, 같은 면적이면 굴곡점의 수를 줄이는 것이 좋은 결과이기도 하다.

## 2. 기존의 결과 및 본 논문의 결과

이진 트리 $T$에 관계되는 몇 가지 용어를 정의하고 간다. $T$의 루트의 레벨은 0이다. 부모노드의 레벨이 $i$이면 그 자식노드의 레벨은 $i+1$이다. $T$의 노드 중에 최대 레벨을 $h$라 하면, $T$의 깊이 (depth)는 $h+1$이 된다. $T$의 루트와 단말노드를 연결하는 경로 (path) 중에서 가장 긴 것에 $h+1$개의 노드가 있다는 것과 같은 말이다. $T$의 조각 (fragment)은 $T$의 부분 트리 (partial tree)를 의미한다. 즉, $T$를 그래프라고 생각할 때, 연결된 (connected) 서브그래프에 해당하는 것이 조각이다. $T$의 한 노드에 대해서 그 노드의 모든 후손노드들과 이들을 잇는 에지들로 구성된 트리를 그 노드를 루트로 하는 $T$의 서브트리라 부른다. 서브트리는 조각이 되지만, 역은 성립하지 않는다.

$T$를 $n$개의 노드를 가진 이진트리라 하자. 앞으로 이진트리에 대해서 별다른 언급이 없으면 그 이진트리는 $n$개의 노드를 가지고 있다고 가정한다. 먼저 기존의 결과를 살펴보면서, 본 논문에서는 어느 면을 향상시키고자 하는지를 설명한다.

**[소정리 2.1]** $T$와 상수 $\alpha, 0<\alpha<1$가 주어질 때, 면적 $O(n)$, 폭 $O(n^{\alpha})$, 높이 $O(n^{1-\alpha})$인 평면 다각선 상향 드로잉을 $O(n)$ 시간에 구할 수 있다.

소정리 2.1의 증명은 논문 [4]에 나와 있다. 소정리 2.1의 드로잉은 순서보존이 아니고, 경우에 따라서는 굴곡점의 수가 최대 $O(n)$이 될 수 있다. 만약 순서보존을 하고자하는 경우에는 면적이 더 많은 면적이 필요하며, 그만큼의 면적으로 항상 가능하다는 것이 논문 [4]의 결과로 아래 소정리에 나와 있다.

**[소정리 2.2]** 소정리 2.1과 같은 방법이면서 동시에 순서보존인 경우에는, (1) 면적이 최저 $\Omega(n\log n)$이 필요한 이진트리가 있으며, (2) 어떤 이진트리에 대해서도 높이 $O(n)$, 폭 $O(\log n)$, 면적 $O(n\log n)$인 평면 다각선

상향 드로잉을 $O(n)$ 시간에 구할 수 있다.

논문 [4]에 나와 있는 소정리 2.2의 (2)에 대한 알고리즘을 간단하게 재귀적으로 설명할 수 있다. 그림 1처럼 $T$의 루트를 $v$라하고, 왼쪽 서브트리를 $A$, 오른쪽 서브트리를 $B$라하고 이들을 루트 (즉, $v$의 자식노드)를 각각 $a, b$라 한다. $A$와 $B$를 재귀적으로 그려서 드로잉을 얻는다. $T$ 전체를 그리기 위해서 루트 $v$를 위에 놓고 그 아래에 $A$의 드로잉과 $B$의 드로잉을 수직으로 쌓는다. 즉, $v, a, b$가 같은 수직선에 오도록 쌓는다. 두 드로잉 중에 어느 것을 위에 놓느냐는 $A$의 노드 수와 $B$의 노드 수를 비교해서 작은 것을 위에 놓는다. 이제, 두 에지 $(v, a)$와 $(v, b)$를 연결하는데, 먼저 $A$의 드로잉이 $B$의 드로잉 보다 위에 있는 경우에는 그림 2(a) 처럼 $(v, a)$는 바로 수직선으로 연결할 수 있다. $b$가 $v$의 오른쪽 자식이므로 $(v, b)$는 $A$의 드로잉의 오른쪽을 돌아서 연결하면 된다. $B$의 드로잉이 위에 있는 경우에는 $(v, b)$는 바로 수직선으로 연결할 수 있다. 그림 2(b)를 보면 쉽게 이해할 수 있다.

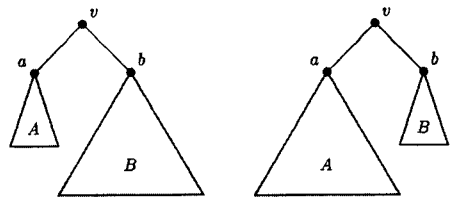



그림 1

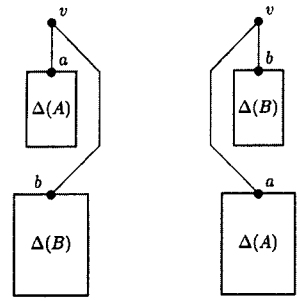



그림 2

이렇게 얻어진 드로잉에서는 모든 노드들이 같은 수직선 위에 놓인다. 앞에서 $v, a, b$가 같은 수직선에 오도록 놓았기 때문에, 이를 재귀적으로 $A, B$에 적용하면 쉽게 알 수 있다. 따라서 드로잉의 높이는 $O(n)$이 된다. 두 에지 $(v, a)$와 $(v, b)$ 중 한 에지는 수직으로 연결되고, 다른 에지는 $A$나 $B$의 드로잉 중에서 위에 놓인 것을 왼쪽이나 오른쪽으로 돌게 되어 있으므로 위에 놓인 드로잉의 폭을 하나 증가시킨다. 아래 놓인 것의 폭은 불변이다. $A$와 $B$ 중에서 노드의 수가 작은 것을 위에 놓는 이유는 노드의 수가 작은 것의 드로잉이 일반적으로 노드의 수가 큰 것의 드로잉보다 폭이 작기 때문이다. $w(T)$를 $T$의 드로잉의 폭이라고 하면, 그림 2(a)의 경우는 $w(T) \le \max\{w(A)+1, w(B)\}$이고, 그림 2(b)의 경우는 $w(T) \le \max\{w(A), w(B)+1\}$이다. 이를 확장시키기 위해서 $width(n)$을 $n$개의 노드로 구성된 임의의 이진트리의 드로잉의 최대 폭이라 하면, $width(n) \le \max\{width(n_1), width(n_2)+1\}$이 된다. 여기서 $n = n_1 + n_2 + 1$로서 $n_1$과 $n_2$는 루트의 두 서브트리의 노드 수를 각각 나타내며, $n_1 \ge n_2$이다. 이 식을 $width(1) = 0$이라는 사실을 이용하여 풀면 $width(n) = O(\log n)$이 되어서, 드로잉의 폭은 최대 $O(\log n)$ 된다는 것을 확인할 수 있다.

이 드로잉의 굴곡점의 수를 생각해 보면, 두 에지 $(v, a)$와 $(v, b)$ 중 한 에지는 수직으로 연결되어 굴곡점이 안 생기지만, 다른 에지는 $A$나 $B$의 드로잉 하나를 돌게 되므로 굴곡점이 적어도 하나 생긴다. 만약 이진트리 $T$의 모든 내부 노드 (internal node)가 자식노드를 두 개씩 가지고 있으면, 이 자식노드로 가는 두 에지중 하나에는 반드시 굴곡점이 생기므로 전체적으로 $n/2$개의 에지에 적어도 하나의 굴곡점이 생긴다는 것을 알 수 있다. 논문 [4]는 주로 드로잉의 면적을 줄이는 관점에서 저술된 것이 여서, 그 논문과 소정리 2.2에는 굴곡점의 수에 대해서는 분석과 언급이 없다.

본 논문에서는 같은 드로잉 방법으로 이진트리를 그려서 굴곡점의 수를 줄여 최대 $O(n/\log n)$가 되도록 그릴 수 있음을 보인다.

## 3. 평면 다각선 상향 순서보존 드로잉에서 굴곡점의 수 줄이기

본 장에서는 다음과 같은 이미 잘 알려진 전략을 이용하여 원하는 목표를 달성한다. $n$개의 노드로 구성된 이진트리 $T$에서 $i$개의 에지를 제거하면 $T$는 $i+1$개의 조각으로 나누어진다. 조각들 역시 이진트리이다. Shin, Kim, Chwa [6]는 아래의 소정리를 증명하고, 이를 이용하여 여러 가지 트리 드로잉에 응용하였다.

**[소정리 3.1]** $T$에서 $\theta(n/\log n)$개의 에지를 제거하면

$\theta(n/\log n)$개의 조각이 생기는데, 각 조각의 크기 (즉, 조각에 포함되는 노드의 수)가 최대 $O(\log n)$이 되도록 에지들을 선택할 수 있다. 이는 $O(n)$ 시간에 가능하다.

소정리 3.1의 중요 점은 조각들의 크기가 전체적으로 균등하다는 것이다. 소정리 3.1은 $T$를 그리는데 이용할 수 있다. 즉, $T$를 그리기 위해서 먼저 소정리 3.1을 이용하여 각각의 크기가 $O(\log n)$인 $\theta(n/\log n)$개의 조각으로 나눈다. 각 조각에 대해서 3.1 절에서 설명할 방법을 이용하여 드로잉을 얻는다. 이제 이 드로잉들을 모두 모아서 3.2 절에서 설명할 방법에 의해서 놓일 곳을 결정한 다음, 소정리 3.1에 의해서 제거된 에지들을 다시 복원하여 조각들 사이에 잘 그려 넣으면 전체 $T$의 드로잉이 완성된다. 따라서, 위 방법으로 $T$를 그리기 위해서는 먼저, 각 조각을 어떤 방법에 의해서 그리는지를 설명해야 하고, 둘째로는 조각 드로잉들을 어떤 방식으로 배열하는가를 생각해야 하며, 마지막으로 제거된 에지들을 복원할 때 어떻게 해야 하는지를 보이면 된다.

소정리 3.1에 의해서 얻어진 $T$의 조각들 중에서 단 하나의 노드로만 구성된 것을 **단순** (trivial) 조각이라 부르고, 둘 이상의 노드로 구성된 것을 **비단순** (nontrivial) 조각이라 부른다.

소정리 3.1에 의해서 얻어진 $T$의 조각들, 각각에 대해서 그 안에 들어 있는 에지들과 노드들을 모두 뭉쳐서 하나의 수퍼노드를 만든다. 그리고 소정리 3.1에서 제거된 에지들로 이 수퍼노드들을 연결한다. 즉, 에지 $e=(u,v)$가 제거된 에지라면 이 에지를 $u$가 속한 수퍼노드와 $v$가 속한 수퍼노드를 연결하는데 사용한다. 그러면 $\theta(n/\log n)$개의 수퍼노드로 구성된 트리가 생긴다. 이를 **조각트리** (fragment tree)라 부르고, 이를 $FT$로 표시한다.

[6]에 나와 있는 또 다른 중요 결과 중의 하나는 소정리 3.1에서 에지를 제거하여 조각들을 만들 때 다음 소정리를 만족하도록 할 수 있다는 것이다.

**[소정리 3.2]** (1) $FT$는 이진트리이다. (2) $FT$의 한 수퍼 노드가 두 개의 자식노드를 가지면 그 수퍼 노드는 단순 조각에서 생긴 것이다.

소정리 3.2의 (2)가 의미하는 것은 비단순 조각에서 만들어진 수퍼노드는 $FT$내에서 많아야 하나의 자식노드만을 가진다는 것이다. 이 사실은 조각들을 각각 그린 후, 제거된 에지들을 다시 복원하여 이 에지들을 그릴 때 면적이나 굴곡점의 수를 줄이는데 결정적인 역할을 한다. 하나의 수퍼노드에 여러 개의 에지들이 있게 되면 이들을 복원하여 그리는데 그 만큼 더 복잡해진다. 따라서, 비단순조각에 해당하는 수퍼노드를 $FT$에서 보면 최대 두 개의 에지가 연결되어 있는데, 하나는 이 조각과 그 부모를 연결하는 것이고, 다른 하나는 (만약 있다면) 이 조각과 그 자식노드을 연결하는 것이다. 이들을 $T$에서 더 자세히 보면 첫 번째 에지는 이 비단순조각의 루트가 다른 조각의 노드와 연결되는 것이고 두 번째 에지는 이 비단순조각의 한 노드 (이를 이 조각의 연결노드라 부른다)가 다른 조각의 노드에 연결되는 것이다. 즉, 루트와 연결노드를 통해서 이 비단순조각은 외부의 다른 조각들과 연결된다.

### 3.1 조각 그리기

이 절에서는 조각을 그리는 방법을 설명하는데, 단순 조각의 경우 노드 하나로만 구성되어 있어서 그리기가 아주 쉽다. 그러므로 비단순조각을 어떻게 그리는지를 설명한다. $F$를 $m$개의 노드로 구성된 비단순조각이라 하자. 조각 역시 이진트리이므로 이진트리에서 정의된 용어를 그대로 사용할 수 있다. $F$를 평면 직선 상향 순서보존 방법으로 그려서, 폭과 높이가 각각 $O(m)$이고 면적이 $O(m^2)$인 드로잉을 얻는 알고리즘을 먼저 소개한다. 즉, 에지를 단 하나의 직선 선분으로 그리는 것이다. $F$의 깊이를 $h$라 하자. $F$의 각 레벨 $i$ ($0\le i\le h-1$)에 대해서 그 레벨에 있는 모든 노드들을 왼쪽에서 오른쪽으로 순서대로 순서번호를 1,2,...으로 붙인다. $F$의 노드 $v$가 레벨 $i$에 있고 그 레벨에서의 순서 번호가 $j$이면, $v$를 좌표 $(j,h-i)$에 그린다. 각 에지는 양끝 점에 해당하는 두 노드를 직선으로 연결하게 그린다. 그러면 $F$의 루트는 좌표 $(1,h)$에 위치하고, 그 자식노드들은 만약 두 개 다 있다면 좌표 $(1,h-1)$과 $(2,h-1)$에 위치한다. 이런 식으로 모든 노드들이 위치한다. 그림 3의 예에 대한 드로잉이 그림 4 (a)에 있다. 이렇게 그린 드로잉이 평면 직선 상향 순서보존 드로잉이 된다는 사실을 확인하는 것은, 드로잉이 레벨별로 이루어지고 모든 에지들이 이웃한 레벨들 사이에만 연결된다는 점을 생각하면 쉽게 가능하다. 드로잉의 높이는 $F$의 깊이 $h$와 같고, 드로잉의 폭은 가장 많은 노드를 가진 레벨의 노드 수와 같다. 따라서, 폭과 높이 모두 $O(m)$이고, 면적은 $O(m^2)$가 된다.

여기서 눈여겨볼 점은 이 드로잉에서 $F$의 루트는 맨위 가장 왼쪽에 자리 잡고 있다는 것이다. 이 점이 중요한 이유는 $F$의 드로잉이 다른 조각의 드로잉과 연결되어야 하는데 특히 $F$의 루트는 위쪽에서 다른 조각으로부터 내려오는 에지에 연결되어야 하므로 이를 위한 공간을 미리 생각해야 하는데 루트의 위치가 이를 위한

좋은 위치라는 것이다. 또하나 생각해야 할 것이 $F$의 연결노드이다. 앞에서 언급했듯이 연결노드는 비단순조각의 경우 최대 하나이므로, 만약 $F$가 하나를 가지고 있다면, 이 연결노드에서 나가는 에지가 다른 조각의 루트에 연결되는 통로를 미리 생각해서 마련해 두어야 한다. 이 통로를 어떻게 준비하는가를 설명한다. 굴곡점의 수를 줄이는 것이 목적이므로 되도록 굴곡이 작은 통로를 생각하는 것이 바람직하다.

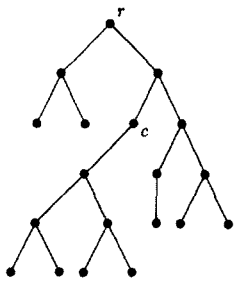
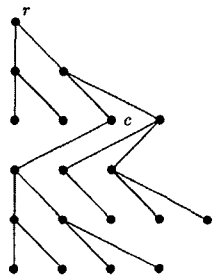
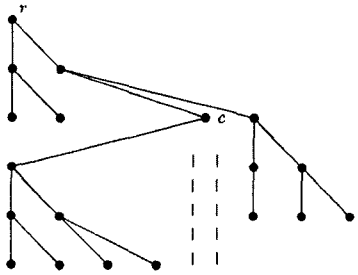



그림 3




(a)




(b)

그림 4

아이디어는 전체적으로 드로잉은 앞의 방법과 동일하게 하면서 연결노드 $c$의 수직으로 아래쪽에는 어떤 노드나 에지도 위치하지 않도록 노드들을 움직이는 것이다. 그렇게 함으로써 $c$에서 나가는 에지가 수직으로 아래로 갈 수 있는 직선 통로를 확보한다.

먼저 $F$의 루트를 $r$이라 하고, $F$에서 $r$과 $c$를 연결하는 경로를 $P$라 한다. $F-P$에 있는 노드들을 다음과 같이 두 집합 $L,R$으로 나눈다. 노드 $v\in F-P$에 대해서 $v$에서 $r$을 향해서 올라갈 때, 처음으로 만나는 $P$의 노드를 $w$라 하고, $w$의 자식노드 중 $P$에 있지 않은 노드를 $u$라 한다. 다시 말해, $v$에서 $r$을 향해서 올라가면 $u$를 만나고 바로 다음에 $w$를 만난다. 이 경우 $u$가 $w$의 왼쪽 자식노드이면, $v$를 $L$에 넣고, 아니면 $R$에 넣는다.

$F$에서 $c$의 레벨을 $k$라하고 그 레벨에서 순서번호를 $l$이라 하자. 이제 $F$의 모든 노드에 대해서 그 노드가 위치할 좌표를 새로 결정한다. 먼저, 레벨 $1,2,\ldots,k-1$에 속하는 모든 노드들과 레벨 $k,k+1,\ldots,h$에 있으면서 집합 $L$에 속한 모든 노드들은 앞서와 동일하게 좌표 $(j,h-i)$에 위치시킨다. $i$는 노드의 레벨, $j$는 그 레벨에서의 순서번호이다. 노드 $c$는 좌표 $(q+1,h-k)$에 위치시킨다. 여기서, 레벨 $k,k+1,\ldots,h$에서 $L$에 속하는 노드만을 셀 때 가장 많은 노드를 가진 레벨의 노드 수를 $q$라 한다. 레벨 $k$에 있으면서 $R$에 속하는 순서번호 $j$인 노드는 좌표 $(q-l+j+1,h-k)$에 위치시킨다. 레벨 $k+1,k+2,\ldots,h$에 있으면서 집합 $R$에 속하는 노드는 좌표 $(q+j+1,h-i)$에 위치시킨다. $i$는 노드의 레벨, $j$는 그 레벨에서 $R$에 속하는 노드들만을 왼쪽에서 오른쪽으로 번호를 $1,2,\ldots$로 붙일 때의 순서번호이다. 그림 4 (b)를 보면 $q=4$이므로 연결노드 $c$의 $x$좌표가 $q+1=5$가 되었다.

이 드로잉에서 보면, 노드 $c$는 좌표 $(q+1,h-k)$에 있고 이 점에서 수직으로 내려가는 반직선에는 어떤 노드도 있지 않고 어떤 에지도 이를 교차하지 않는다. 이 반직선이 나중에 $c$에서 나가는 에지를 위한 통로가 된다. 그림 4 (b)에 이 통로가 점선으로 표시되어 있다.

이렇게 만들어진 드로잉은 폭이 두배 정도 늘지만 높이는 변함없다. 따라서 면적 역시 두배로 늘지만 아직 $O(m^2)$으로 변함없다.

**[소정리 3.3]** $m$개의 노드로 이루어진 비단순 조각에 대해서 폭 $O(m)$, 높이 $O(m)$ 면적 $O(m^2)$인 평면 직선 상향 드로잉을 선형시간에 구할 수 있다. 만약 이 비단순 조각이 연결노드를 가지고 있으면, 이 연결노드에서 수직으로 내려가는 반직선이 드로잉을 교차하지 않게 만들 수 있다.

### 3.2 조각 드로잉 배치와 제거된 에지 복원하기

이 절에서는 앞절에서 만들어진 조각들의 드로잉을 어떻게 배치하고, 제거된 에지들을 어떤 방법으로 이 드로잉들 사이로 연결하여 전체 드로잉을 완성하는지를 설명한다.

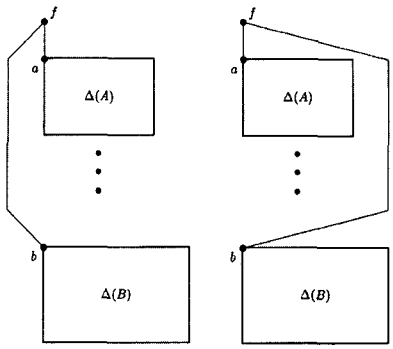
우선 3.1절의 방법으로 그려진 조각 $F$의 드로잉을 $\Delta(F)$라 표시하자. $F$가 단순조각이면 $\Delta(F)$는 단지 노드 하나로만 이루지지만, $F$가 비단순조각일 경우는 소정리 3.1에 따라 $m=O(\log n)$이 되므로 소정리 3.3에 의해서 얻어진 $\Delta(F)$의 폭과 높이가 모두 $O(\log n)$이 된다.

소정리 3.2에 의하면 조각 트리 $FT$는 $\theta(n/\log n)$개의 노드로 구성된 이진트리이다. 물론, 자세히 보면 각 노드가 하나의 조각에 해당하는 수퍼노드지만, 우선은 일단 그냥 일반 노드라고 생각하고, 소정리 2.2의 (2)에 해당하는 알고리즘을 $FT$에 적용한다. 그러면 높이 $O(n/\log n)$, 폭 $O(\log n)$, 면적 $O(n)$인 $FT$의 드로잉 $\Delta(FT)$가 얻어진다. 우리가 원하는 것은 $T$의 드로잉이므로, $\Delta(FT)$의 노드마다 그에 해당하는 조각의 드로잉(위에서 말한 $\Delta(F)$)을 그 노드 대신에 끼워 넣으면 $T$의 드로잉 $\Delta(T)$를 얻을 수 있다. 이 끼워 넣기를 어떻게 하는가는 다음에 설명하기로 하고, 일단 끼워 넣기가 끝난 후의 드로잉를 생각해 보자. $\Delta(FT)$의 각 노드가 폭과 높이가 각각 $O(\log n)$인 드로잉을 바꾸므로 전체 드로잉의 폭과 높이도 비례해서 커진다. 먼저, $\Delta(FT)$에서 $O(n/\log n)$개의 노드가 수직으로 쌓여있던 것이 각 노드가 높이 $O(\log n)$인 드로잉으로 바뀌므로 전체의 높이는 $O(n/\log n)*O(\log n)$으로 $O(n)$이 되고, 폭은 원래의 폭 $O(\log n)$에 새로 폭이 $O(\log n)$만큼 더 해지므로 전체의 폭은 변함없이 $O(\log n)$이다. 그래서 면적은 $O(n\log n)$이 된다.

이제 문제는 $\Delta(FT)$의 각 노드 $F$를 그것의 드로잉 $\Delta(F)$으로 끼워 넣는 것이다. 앞으로는 $FT$ 혹은 $\Delta(FT)$의 노드라는 말과 그 노드가 나타내는 조각이라는 말을 혼용해서 사용한다. 예를 들어, $F$를 $FT$ 혹은 $\Delta(FT)$ 입장에서 보면 노드에 해당하지만, $F$ 자체를 놓고 보면 조각에 해당한다는 것이다.

기본적으로 지켜야 할 것은 모든 $\Delta(F)$를 끼워 넣은 후에 이들의 왼쪽 변들을 모두 한 수직선에 맞춘다는 것이다. $\Delta(F)$에서 보면 루트들이 맨위 가장 왼쪽에 있으므로 이 루트들이 모두 같은 수직선 위에 오도록 맞춘다는 것이다.

먼저, $F$가 단순 조각인 경우에 $\Delta(F)$가 노드 $f$로만 이루어져 있으므로 쉽다. $FT$에서 $F$의 두 자식노드를 $A$, $B$라하고, $A$의 루트를 $a$, $B$의 루트를 $b$라 한다. $\Delta(FT)$에 있는 두 에지 $(F,A)$와 $(F,B)$를 $\Delta(T)$에서는 $(f,a)$와 $(f,b)$로 대신 연결하면 된다. 그림 5에 있는 것처럼, 둘 중 하나는 직선으로 연결되고, 다른 하나는 두 개의 굴곡점을 가진다.




(a) (b)

그림 5

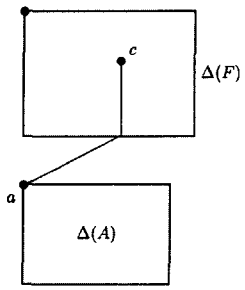



그림 6

이제, $F$가 비단순조각인 경우를 설명한다. 이 경우는 소정리 3.2 (2)에 의해서, 자식노드가 있으면 꼭 하나 있다. $F$가 자식노드를 가지고 있지 않으면 더 이상 할 일이 없으므로, 자식노드를 하나 가지고 있는 경우만 생각한다. $FT$에서 $F$의 자식노드에 해당하는 조각을 $A$, $A$의 루트를 $a$라하고, $F$의 연결노드를 $c$라 한다. 다시 말해, 에지 $(c,a)$를 연결해야 한다. $\Delta(FT)$의 에지 $(F,A)$가 $\Delta(T)$에서는 $(c,a)$로 연결하면 된다. 그런데, $\Delta(A)$는 $\Delta(F)$ 바로 밑에 나온다. $\Delta(F)$를 둘러싸는 가장 작은 (변이 수직 수평인) 직사각형 $R$을 고려하자. $\Delta(F)$에서 연결노드 $c$에서 수직으로 내려오면 $R$의 아래 변과 교차하는데, 그 교차점에서 바로 $a$로 (기울어진) 직선으로 연결한다. 이렇게 연결한 $(c,a)$는 하나의 굴곡점

을 가진다. $c$에서 수직으로 내려오는 통로가 있다는 것은 소정리 3.3에서 설명했다. 그림 6을 보면 이 과정을 이해할 수 있다.

**[정리 3.1]** $n$개의 노드로 이루어진 이진트리에 대해서 높이 $O(n)$, 폭 $O(\log n)$, 면적 $O(n\log n)$이면서 굴곡점의 수가 $O(n/\log n)$인 평면 다각선 상향 드로잉을 $O(n)$ 시간에 구할 수 있다.

**증명:** 높이, 폭, 면적에 대해서는 앞에서 설명했고, 수행시간이 $O(n)$이 된다는 것은 쉽게 알 수 있으므로, 굴곡점의 수에 대해서만 분석하면 된다. 위에서 얻어진 드로잉 $\Delta(T)$를 보면, 먼저 각 조각 드로잉 $\Delta(F)$는 모두 직선 상향이므로 굴곡점이 없다. 굴곡점을 가지고 있는 에지들은 이 조각들을 연결하는 에지들 뿐인데, 이 에지들은 그림 5, 6에 있듯이, 에지당 최대 두개의 굴곡점만을 가진다. 따라서 많아야 $O(n/\log n)$개의 굴곡점이 $\Delta(T)$에 있다. ■

## 4. 결 론

본 논문에서는 $n$개의 노드로 이루어진 이진트리를 평면에 그리는 한 방법인 평면 다각선 상향 순서보존 그리드 방법을 이용하여 높이 $O(n)$, 폭 $O(\log n)$, 면적 $O(n\log n)$은 기존의 결과와 같게 유지하면서, 굴곡점의 수를 $O(n)$에서 $O(n/\log n)$으로 줄이는 방법을 제시하였다.

## 참 고 문 헌


[1] T. Chan, M.T. Goodrich, S.R. Kosaraju, and R. Tamassia, Optimizing area and aspect ratio in staright-line orthogonal tree drawings, Proc. Graph Drawing '96, Lecture Notes in Computer Science, vol. 1190, pp. 63-75, 1997

[2] P. Crescenzi, G. Di Battista, and A. Piperno, A note on optimal area algorithms for upward drawings of binary trees, Computational Geometry: Theory and Applications, vol. 2, pp. 187--200, 1992.

[3] P. Crescenzi, P. Penna and A. Piperno, Linear area upward drawings of AVL trees, Computational Geometry: Theory and Applications, Volume 9, pp. 25-42, 1998.

[4] A. Garg, M.T. Goodrich, and R. Tamassia, Area-efficient upward tree drawings, Proc. 9th ACM Symposium on Computational Geometry, pp. 359--368, 1993.

[5] S.K. Kim, Logarithmic width, linear area upward drawing of AVL trees, Information Processing Letters, vol. 63, pp. 303--307, 1997.

[6] C.S. Shin, S.K. Kim, and K.Y. Chwa, Area-efficient algorithms for upward straight-line tree drawings, Proc. 2nd International Conference on Computing and Combinatorics, COCOON '96, Lecture Notes in Computer Science, vol. 1090, pp. 106--116, 1996.



김 성 권

1981년 2월 서울대학교 계산통계학과(학사). 1983년 2월 한국과학기술원(석사). 1983년 3월 ~ 1985년 9월 목포대학교 재직. 1990년 8월 University of Washington 전산학과(박사). 1991년 3월 ~ 1996년 2월 경성대학교 재직. 1996년 3월 ~ 현재 중앙대학교 컴퓨터공학과 재직중. 관심분야는 알고리즘, 컴퓨터이론.