보조정리 10.1: 루트로부터 외부 노드로의 경로의 길이를 경로상의 포인터의 수로 정하자. 레드-블랙 트리에서 루트로부터 외부 노드로의 2개의 경로 P, Q가 있을 때, $length(P) \leq 2length(Q)$이다.

증명: 임의의 레드-블랙 트리에서 루트의 랭크를 r이라고 하자. $RB1'$로부터 루트에서 외부 노드로의 경로상에 있는 마지막 포인터는 블랙이다. $RB2'$로부터 2개의 연속적인 레드 포인터를 갖는 경로는 존재하지 않는다. 그러므로 각 레드 포인터의 뒤에는 블랙 포인터가 오게 된다. 결과적으로 루트에서 외부 노드로의 각 경로는 r과 $2r$ 사이의 포인터를 갖게 되므로 $length(P) \leq 2length(Q)$이다. 상한선이 존재할 수 있는지 알아보기 위해 그림 10.15의 레드-블랙 트리를 보자. 루트로부터 5의 왼쪽 자식까지의 경로는 길이가 4인 반면, 80의 오른쪽 자식까지의 경로의 길이는 2이다. □

보조정리 10.2: 레드-블랙 트리(외부 노드를 포함하지 않는)의 높이를 h라고 하자. 트리의 내부 노드의 수를 n이라 하고 루트의 랭크를 r이라 하면, 다음이 성립한다.

(a) $h \leq 2r$

(b) $n \geq 2^r - 1$

(c) $h \leq 2\log_2(n+1)$

증명: 보조정리 10.1의 증명으로부터 루트에서 외부 노드로의 경로가 $length > 2r$을 만족하는 것은 존재하지 않음을 알 수 있다. 그러므로 $h \leq 2r$이다.(제거된 외부 노드들과 함께 그림 10.15의 레드-블랙 트리의 높이는 $2r = 4$이다.)

루트의 랭크는 r이므로 레벨 1에서 r까지에는 외부 노드가 없고, 이러한 레벨들에는 $2^r - 1$개의 내부 노드들이 있다. 따라서 이것은 내부 노드 총 수의 최소한의 수가 된다.(그림 10.15의 레드-블랙 트리에서 레벨 1과 2는 $3 = 2^2 - 1$의 내부 노드를 가진다. 레벨 3과 4에는 내부 노드들이 더 있다.)

(b)로부터 $r \leq \log_2(n+1)$이다. 이 부등식과 (a)로부터 (c)가 산출된다. □

레드-블랙 트리의 높이는 최대 $2\log_2(n+1)$이므로 $O(h)$ 시간 내에 수행되는 탐색·삽입·삭제 알고리즘은 $O(\log n)$의 복잡도를 갖는다.

레드-블랙 트리의 최악의 경우 높이는 동일한 (내부) 노드 수를 갖는 AVL 트리의 최악의 경우 높이[대략 $1.44\log_2(n+2)$] 이상이라는 점에 주목하라.

10.3.2 레드-블랙 트리의 표현

레드-블랙 트리를 정의할 때는 외부 노드들을 포함시키는 것이 더 편리하지만, 구현에 있어서는 외부 노드를 표현하기 위해 물리적인 노드보다는 널 포인터를 이용한다. 또한 포인터와 노드의 컬러는 밀접한 연관이 있으므로 각 노드와 함께 그 노드의 컬러만을 저장하거나, 그 자식을 가리키는 두 포인터의 컬러를 저장할 필요가 있다. 노드의 컬러를 저장하기 위해서는 노드당 추가적인 1비트가 필요하며, 포인터의 컬러를 저장하기 위해서는 노드당 2비트가 필요하다. 두 전략 모두 거의 비슷한 양의 공간이 요구되므로, 레드-블랙 트리 알고리즘을 실행시킨 결과로 나온 실제 실행 시간을 기반으로 둘 중 하나를 선택하면 된다.

여기서는 삽입과 삭제 연산에 관한 노드에 관해서만 컬러가 어떻게 변하는지를 명백히 언급할 것이다. 해당되는 포인터의 컬러가 어떻게 변하는지는 추론할 수 있다.

10.3.3 레드-블랙 트리에서의 탐색

레드-블랙 트리는 일반적인 이원 탐색 트리의 탐색에서 사용하는 알고리즘(프로그램 5.15)을 이용하여 탐색할 수 있다. 이 알고리즘은 $O(h)$의 복잡도를 가지므로 레드-블랙 트리에서는 $O(\log n)$의 복잡도를 가진다. 일반적인 이원 탐색 트리와 AVL 트리, 그리고 레드-블랙 트리에 동일한 탐색 알고리즘을 사용하고 AVL 트리의 최악의 경우 높이가 최소이므로, 탐색이 주가 되는 응용 프로그램에서는 AVL 트리가 최악의 경우에 최고의 성능을 보일 것이라고 기대한다.

10.3.4 레드-블랙 트리에서의 삽입

원소들은 일반적인 이진 트리에서 사용된 방법(프로그램 5.17)을 이용하여 삽입될 수 있다. 레드-블랙 트리에 새로운 노드가 붙었을 때, 그 노드에 컬러를 지정해줄 필요가 있다. 삽입 전에 트리가 비어 있었다면, 새로운 노드는 루트가 되고 그 컬러는 블랙이 될 것이다(성질 RB1 참조). 삽입 전에 트리가 비어 있지 않았다고 하자. 새로운 노드의 컬러가 블랙으로 주어진다면, 루트에서 새로운 노드의 자식인 외부 노드까지의 경로상에 별도의 블랙 노드를 갖게 될 것이다. 반면에 새로운 노드의 컬러가 레드로 지정된다면 2개의 연속적인 레드 노드를 가지게 될지도 모른다. 새로운 노드를 블랙으로 만드는 것은 성질 RB3을 위배되지만, 새로운 노드를 레드로 만드는 것은 성질 RB2에 위배될 수도 있고 위배되지 않을 수도 있다. 여기서는 새로운 노드를 레드로 만들 것이다.

새로운 노드를 레드로 만드는 것이 성질 RB2에 대해 위배될 경우 그 트리는 불균형하게 된 것이다. 새로운 노드 u, 그 노드의 부모 pu, 그리고 u의 조부모인 gu를 검사하여