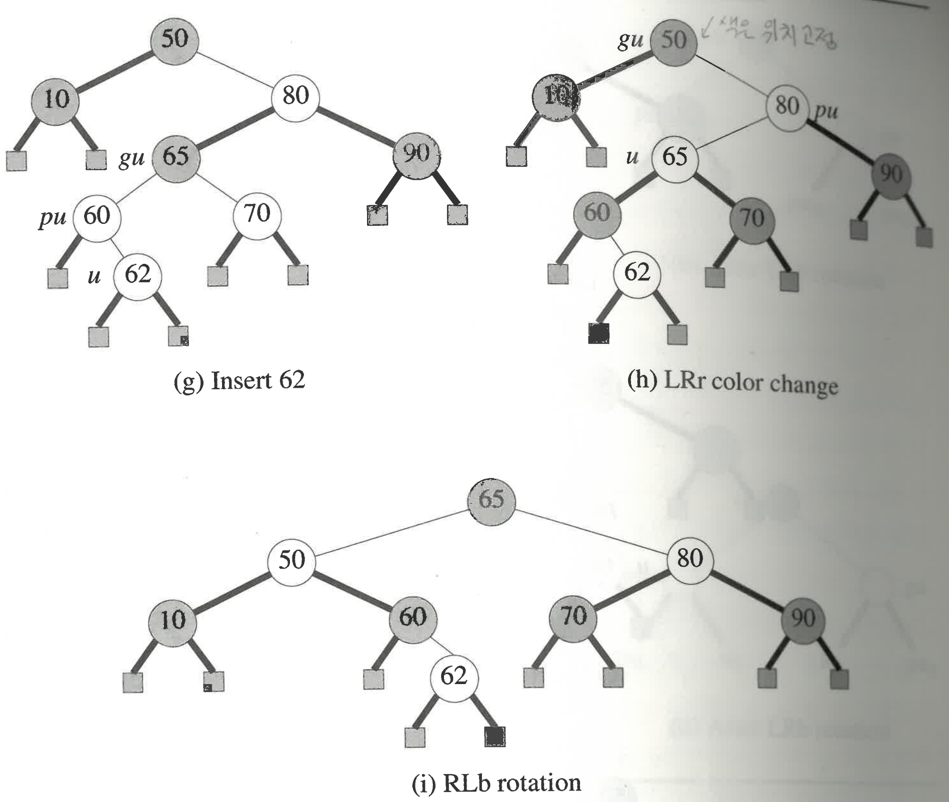

그림 10.18 레드-블랙 트리에 삽입

다. *gu* 노드가 없기 때문에 이 회전에서 RB2 불균형이 성립할 수 없어 완료된다. 루트에서 외부 노드로의 모든 경로는 정확히 2개의 블랙 포인터를 갖는다.

이제 그림 10.18(d)의 트리에 65를 삽입해보자. 결과는 그림 10.18(e)에 나타나 있다. 새로운 노드는 *u* 노드이다. 그 부모와 조부모는 각각 *pu*와 *gu* 노드이다. 그림 10.17의 (c)와 (d)의 회전을 수행해야 하는 LRb 불균형이 발생한다. 회전 결과 그림 10.18(f)의 트리를 얻게 된다.

마지막으로, 그림 10.18(g)의 트리를 얻기 위해 62를 삽입한다. 컬러 변경이 요구되는 LRr 불균형이 발생한다. 그 결과 트리와 새로운 *u*, *pu*, *gu* 노드들은 그림 10.18(h)에 나와 있다. 수행된 컬러 변경은 두 레벨 위까지 RLb 불균형을 초래하므로 이제 RLb 회전을 해야 한다. 그 회전 결과는 그림 10.18(i)의 트리에 나타나 있다. 회전 후에는 더 이상 추가 작업 없이 완료된다. □

10.3.5 레드-블랙 트리에서의 삭제

삭제 변환은 연습문제로 남겨두었다.

10.3.6 레드-블랙 트리의 조인

5.7.5절에서 *threeWayJoin*, *twoWayJoin*, *split*과 같은 이원 탐색 트리의 연산을 정의하였다. 이 연산들은 레드-블랙 트리에서 log 시간 내에 수행될 수 있다. 연산 *threeWayJoin* (*A*, *x*, *B*: *A*는 *small*에 대응하고 *x*는 *mid*, *B*는 *big*에 대응)는 다음과 같이 수행될 수 있다.

경우 1: *A*와 *B*가 같은 랭크를 갖는다면 *x*, *leftChild A*, *rightChild B*의 쌍을 갖는 새로운 루트를 생성함으로써 C가 만들어진다고 하자. 두 링크는 블랙으로 만들어진다. C의 랭크는 A와 B의 랭크보다 하나 높다.

경우 2: 만일 *rank*(*A*) > *rank*(*B*)일 때 *A*에서부터 *rank*(*B*)와 같은 랭크를 갖는 첫 번째 노드 *Y*까지 *rightChild* 포인터를 따라간다면, 성질 RB1에서 RB3까지를 통해 이 노드가 존재함을 보장할 수 있다. *p*(*Y*)가 *Y*의 부모라고 하자. *Y*의 정의로부터 *rank*(*p*(*Y*)) = *rank*(*Y*) + 1이 된다. 그 결과, *p*(*Y*)에서 *Y*로의 포인터는 블랙 포인터이다. *x*, *leftChild Y*, *rightChild B*의 쌍을 갖는 새로운 노드 *Z*를 생성한다(즉, 노드 *Y*와 *Y*의 서브트리는 *Z*의 왼쪽 서브트리가 된다). *Z*는 *p*(*Y*)의 오른쪽 자식으로 만들어지고, *p*(*Y*)에서 *Z*로의 링크는 레드이다. 이 변환은 루트에서 외부 노드까지의 경로상에 있는 블랙 포인터의 수를 변화시키지 않음에 주목하라. 그러나 이 변환은 루트에서 Z까지의 경로 내에 2개의 연속적인 레드 포인터를 포함시키도록 할 수는 있다. 이러한 경우, 상향식 삽입에서 사용되었던 변환을 사용한다. 이런 변환들은 트리의 랭크를 하나씩 증가시킬 수 있다.

경우 3: 만일 *rank*(*A*) < *rank*(*B*)이면, 경우 2와 비슷하다.

***threeWayJoin*의 분석:** 기술된 함수가 정확한지에 대해서는 쉽게 알 수 있다. 경우 1은 O(1) 시간이 걸리고, 나머지 두 가지 경우는 각 레드-블랙 트리의 랭크가 우선한다는 가정 하에서 조인을 수행하는 데 O(|*rank*(*A*)−*rank*(*B*)|) 시간이 걸린다. 따라서 3-원 조인은 O(log *n*)의 시간에 수행될 수 있는데, 이때 *n*은 조인되고 있는 두 트리의 노드 수를 의미한다. 2-원 조인도 비슷한 방법으로 수행될 수 있다. 조인에 필요한 부모들은 루트에서 노드 *Y*까지 움직이는 만큼 스택에 저장될 수 있기 때문에, 조인을 수행하는 데 있어 노드에 부모 데이타 멤버를 추가할 필요는 없다. □

<*twoWayJoin*도 같은 방법으로 수행된다>