분할 정복

주어진 문제를 같은 크기의 둘 이상의 부분 문제로 나눈 뒤 각 부분 문제의 답을 이용해 전체를 계산하는 방식
크고 방대한 문제를 조금씩 나눠가며 용이하게 풀 수 있는 문제로 나눠 해결한 후 다시 합치는 방법

분할 (Divid)

문제를 더 이상 분할할 수 없을 때까지 동일한 유형의 여러 하위 문제로 나눈다.

정복 (Conquer), 분할이 필요 없는 작은 문제 (Base Case)

가장 작은 단위의 하위 문제를 해결

조합 (Combine), 병합 (Merge)

하위 문제에 대한 결과를 원래 문제에 대한 결과로 변환

특징

재귀함수로 구성한다.
둘 이상의 같은 크기로 문제를 나눠가며 진행한다.
분할 방식에 따라 시간 복잡도가 크게 달라진다.

장점

병렬적으로 문제를 해결하는데 큰 강점이 있다.
큰 문제를 작은 부분으로 나눠 해결하는 방식을 취한다. (생각하기 편하다.)

단점

과도한 오버헤드가 발생할 수 있다.
경우에 따라 스택 오버플로우가 발생할 수 있다.

Divide and Conquer VS Dynamic Programming

	Divide and Conquer	Dynamic Programming
방식	작은 문제로 분할해 합병하여 해결함	작은 문제를 조합해 큰 문제를 해결함
방향	하향식	상향식
중복	중복이 발생하지 않음	중복이 발생
중복 대처	고려하지 않음	고려하여 제거함 (메모이제이션)

결과적으로 비슷하지만 DP가 Divide and Conquer를 포함한다.
Divide and Conquer ⊂ DP

사용법

분할이 불가능 하면 값을 반환, 분할이 가능하면 분할하여 나중에 값을 결합

```
Funtion F(x)
    if F(x) 분할 불가
        return 계산 결과

    else F(x) 분할 가능
        F(x)를 F(x1)과 F(x2)로 분할
        reust = F(x1)과 F(x2)를 결합 (F(x)를 구한 값)
        return reust
```

DP

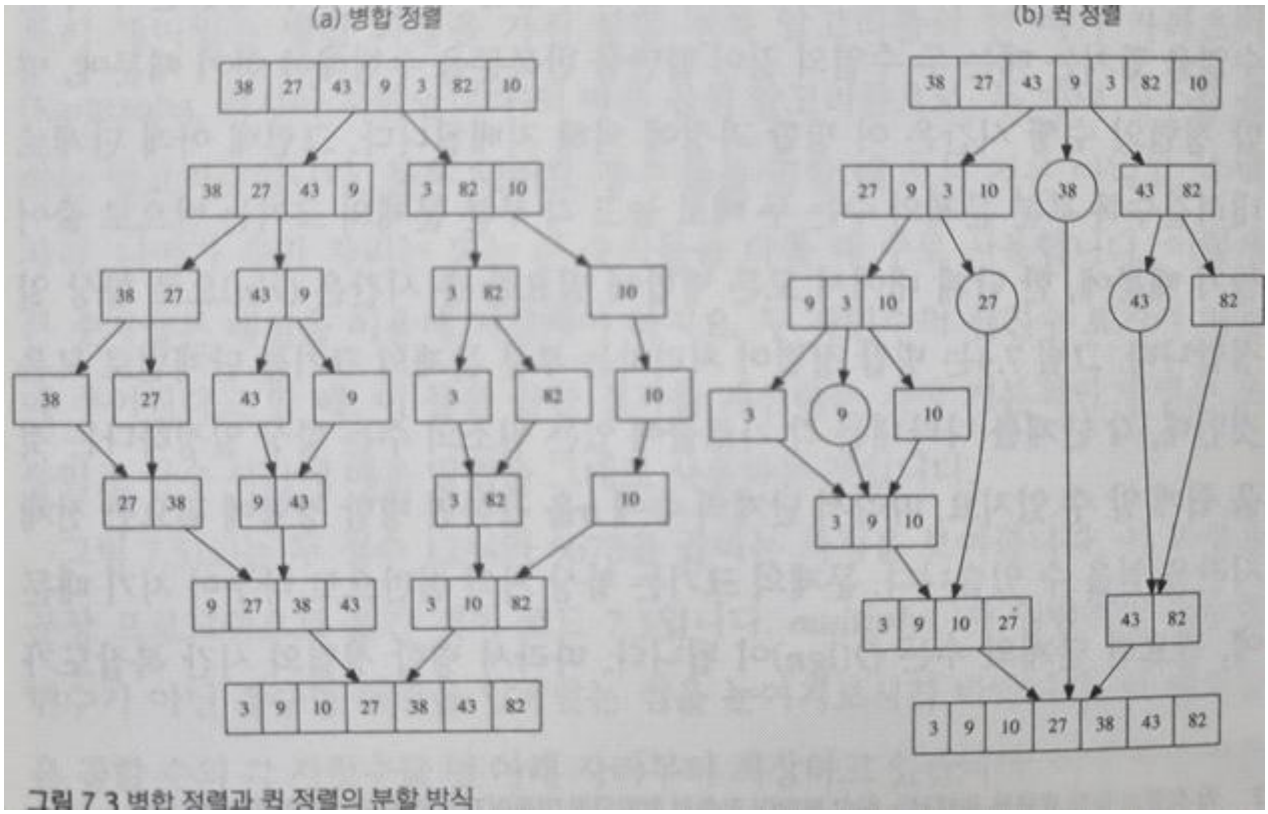

그림 7.3 병합 정렬과 퀵 정렬의 분할 방식

Divide and Conquer

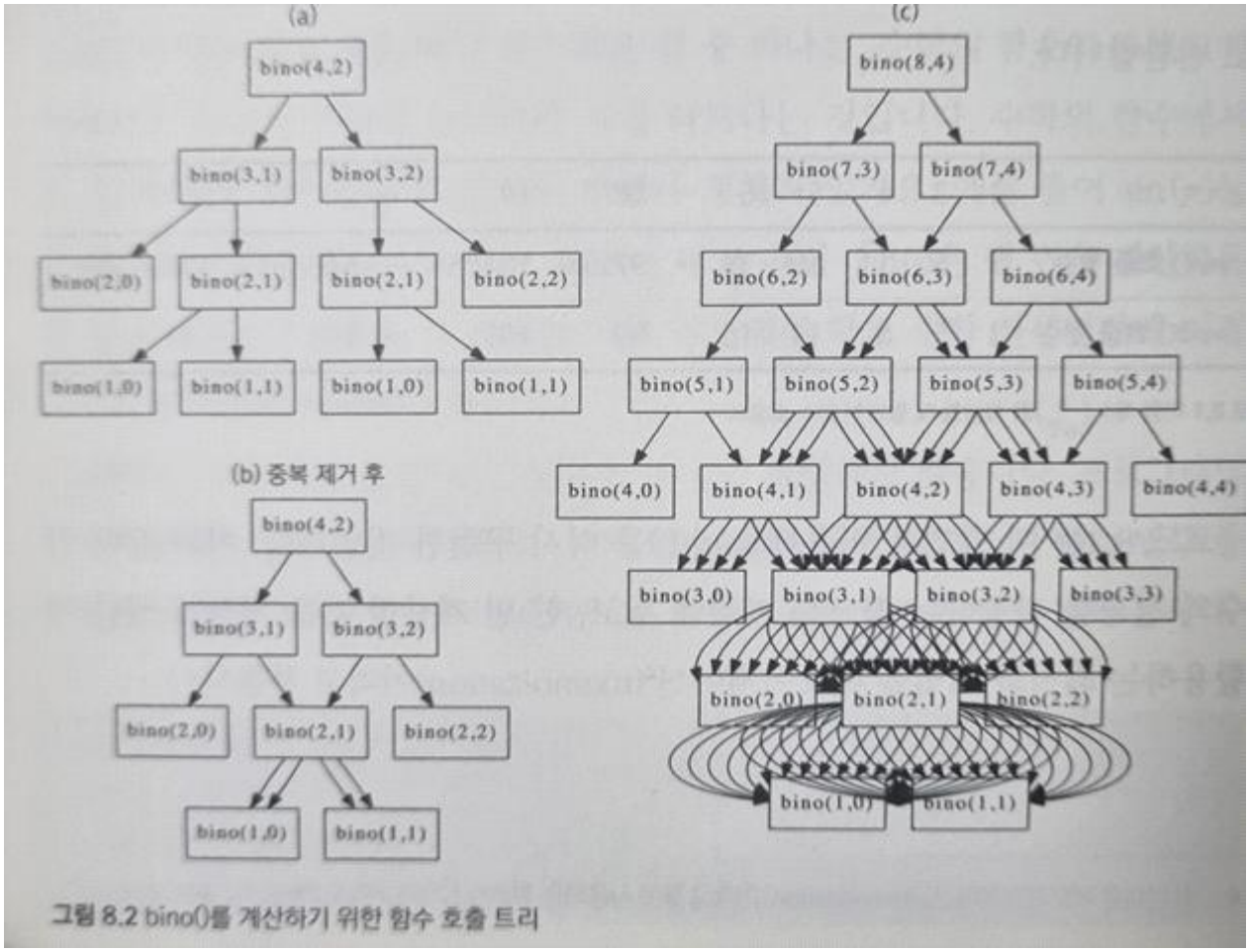

그림 8.2 bino()를 계산하기 위한 함수 호출 트리