c. 위 (a), (b)를 통해 최악의 경우 $\Theta(nk + n\lg(n/k))$ 시간에 실행될 수 있도록 변형된 알고리즘이 주어졌을 때, 그 알고리즘이 표준적인 병합 정렬 알고리즘과 동일한 점근적 수행시간을 갖는 k의 점근적(Θ-표기법 내의) 최댓값을 n의 함수로 표현하라.

d. 실용적으로는 k를 어떻게 정해야 하는가?

2-2 버블 정렬의 타당성

버블 정렬은 비효율적이지만 인기 있는 정렬 알고리즘이다. 이는 정렬되지 않은 이웃 원소를 반복적으로 바꿔 수열을 정렬한다.

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              A[j]와 A[j − 1]을 바꾼다.
```

a. A'을 BUBBLESORT(A)의 결과라고 하자. BUBBLESORT가 타당한지 증명하기 위해 이것이 종료하는지와 $n = A.length$일 때 다음이 성립하는지를 보여야 한다.

$$A'[1] \le A'[2] \le \cdots \le A'[n] \tag{2.3}$$

BUBBLESORT가 정말 정렬할 수 있는지 보이려면 무엇을 더 증명해야 하는가?

다음 두 가지는 부등식 (2.3)을 증명한다.

c. 위 (a), (b)를 통해 최악의 경우 $\Theta(nk + n\lg(n/k))$ 시간에 실행될 수 있도록 변형된 알고리즘이 주어졌을 때, 그 알고리즘이 표준적인 병합 정렬 알고리즘과 동일한 점근적 수행시간을 갖는 k의 점근적(Θ-표기법 내의) 최댓값을 n의 함수로 표현하라.

d. 실용적으로는 k를 어떻게 정해야 하는가?

2-2 버블 정렬의 타당성

버블 정렬은 비효율적이지만 인기 있는 정렬 알고리즘이다. 이는 정렬되지 않은 이웃 원소를 반복적으로 바꿔 수열을 정렬한다.

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              A[j]와 A[j − 1]을 바꾼다.
```

a. A'을 BUBBLESORT(A)의 결과라고 하자. BUBBLESORT가 타당한지 증명하기 위해 이것이 종료하는지와 $n = A.length$일 때 다음이 성립하는지를 보여야 한다.

$$A'[1] \le A'[2] \le \cdots \le A'[n] \tag{2.3}$$

BUBBLESORT가 정말 정렬할 수 있는지 보이려면 무엇을 더 증명해야 하는가?

다음 두 가지는 부등식 (2.3)을 증명한다.

b. 2–4행의 **for** 루프에 대한 루프 불변성을 자세히 서술하고, 그것이 만족되는지를 증명하라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 한다.

c. (b)에서 증명한 루프 불변성의 종료조건을 활용해 부등식 (2.3)을 자연스럽게 증명할 1–4행의 **for** 루프의 불변식을 보여라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 한다.

d. 버블 정렬의 최악의 경우 수행시간은 무엇인가? 그리고 삽입 정렬과 비교하면 어떠한가?

2-3 Horner 공식의 타당성

다항식 계산을 위한 다음과 같은 Horner 공식이 있다.

종합문제

2-1 병합 정렬에서 작은 배열에 대한 삽입 정렬의 적용

병합 정렬은 최악의 경우 $\Theta(n \lg n)$ 시간이 걸리고 삽입 정렬은 $\Theta(n^2)$ 시간이 걸리지만, n이 작으면 상수 인자로 인해 삽입 정렬이 많은 기계에서 더 빠르다. 그러므로 문제 크기가 충분히 작을 때는 삽입 정렬을 적용하는 것이 유용할 수 있다. 즉, 병합 정렬에서 문제 크기가 충분히 작아졌을 때는 재귀 호출의 단위를 "덩어리로 만들어" 삽입 정렬을 적용하는 것이 유용할 수 있다. 이제 변형된 병합 정렬을 고려해 보는데, 어떤 정수 k를 정한 뒤 문제를 분할하다가 크기가 k인 n/k개의 부분 리스트가 되면 삽입 정렬을 이용해 정렬하고, 이를 다시 합치는 것은 일반적인 병합 정렬의 구조를 따르도록 한다.

a. 크기가 각각 k인 n/k개의 부분 리스트를 삽입 정렬을 이용해 최악의 경우 $\Theta(nk)$ 시간에 정렬할 수 있음을 보여라.

b. 이 부분 리스트를 최악의 경우 $\Theta(n \lg(n/k))$ 시간에 병합할 수 있음을 보여라.

종합문제

2-1 병합 정렬에서 작은 배열에 대한 삽입 정렬의 적용

병합 정렬은 최악의 경우 $\Theta(n \lg n)$ 시간이 걸리고 삽입 정렬은 $\Theta(n^2)$ 시간이 걸리지만, n이 작으면 상수 인자로 인해 삽입 정렬이 많은 기계에서 더 빠르다. 그러므로 문제 크기가 충분히 작을 때는 삽입 정렬을 적용하는 것이 유용할 수 있다. 즉, 병합 정렬에서 문제 크기가 충분히 작아졌을 때는 재귀 호출의 단위를 "덩어리로 만들어" 삽입 정렬을 적용하는 것이 유용할 수 있다. 이제 변형된 병합 정렬을 고려해 보는데, 어떤 정수 k를 정한 뒤 문제를 분할하다가 크기가 k인 n/k개의 부분 리스트가 되면 삽입 정렬을 이용해 정렬하고, 이를 다시 합치는 것은 일반적인 병합 정렬의 구조를 따르도록 한다.

a. 크기가 각각 k인 n/k개의 부분 리스트를 삽입 정렬을 이용해 최악의 경우 $\Theta(nk)$ 시간에 정렬할 수 있음을 보여라.

b. 이 부분 리스트를 최악의 경우 $\Theta(n \lg(n/k))$ 시간에 병합할 수 있음을 보여라.

2.3-5

검색 문제를 다시 살펴보자(연습문제 2.1-3 참고). 수열 A가 정렬되어 있으면 수열의 중간값과 찾고자 하는 원소 v를 비교한 수 있어 수열의 절반을 더는 비교할 필요가 없다. ***이진 검색*** 알고리즘은 이런 과정을 반복하는 검색 알고리즘으로, 매번 남아 있는 수열을 이등분하여 검색한다. 반복적인 방법이나 재귀적인 방법을 이용해 이진 검색에 대한 의사코드를 작성하라. 그리고 이진 검색의 최악의 경우 수행시간이 $\Theta(\lg n)$임을 증명하라.

2.3-6

2.1절의 INSERTION-SORT 프로시저에서 5–7행의 **while** 루프가 정렬된 부분 배열 $A[1..j-1]$을 원소 하나씩 역방향으로 훑어보기 위해 선형 검색을 한다는 것을 알 수 있다. 선형 검색 대신 이진 검색(연습문제 2.3-5 참고)을 한다면 삽입 정렬의 최악의 경우 수행시간을 $\Theta(n \lg n)$으로 개선할 수 있을까?

2.3-7 ★

n개 정수의 집합 S와 정수 x가 주어졌을 때, S에 있는 두 원소의 합이 x가 되는 경우가 있는지를 알아내는 $\Theta(n \lg n)$ 시간 알고리즘을 작성하라.

수다. 이제 2^i개의 노드에 대한 재귀 트리의 레벨 수는 $\lg 2^i + 1 = i + 1$이라는 가설이 성립한다고 가정하자(임의의 i에 대해 $\lg 2^i = i$이다). 원래의 입력 크기가 2의 거듭제곱이라 가정했으므로 고려할 다음 입력 크기는 2^{i+1}이다. 리프가 2^{i+1}개인 트리는 리프가 2^i개인 트리보다 한 레벨을 더 가지므로 총 레벨 수는 $(i + 1) + 1 = \lg 2^{i+1} + 1$이다.

점화식 (2.2)의 총 비용을 계산하려면 모든 레벨의 비용을 더하면 된다. $\lg n + 1$ 레벨이 있고 각각은 cn의 비용이 들어 총 비용은 $cn(\lg n + 1) = cn \lg n + cn$이다. 낮은 차수의 항과 상수 c를 무시하면 원하는 결과인 $\Theta(n \lg n)$을 얻는다.

연습문제

2.3-1

그림 2.4를 모델로 하여 배열 $A = \langle 3, 41, 52, 26, 38, 57, 9, 49\rangle$가 주어졌을 때 병합 정렬의 동작을 설명하라.

2.3-2

MERGE 프로시저를, 경계값을 사용하지 않고 배열 L 또는 R이 자신의 원소를 A로 모두 복사하면 비교 작업을 끝내고 나머지 배열에서 복사되지 않고 남은 원소를 A에 모두 복사하도록 재작성하라.

2.3-3

n이 2의 거듭제곱일 때 수학적 귀납법을 이용해 다음 점화식의 해가 $T(n) = n \lg n$임을 보여라.

$$T(n) = \begin{cases} 2 & \text{if } n = 2 \\ 2T(n/2) + n & \text{if } n = 2^k, \text{ for } k > 1 \end{cases}$$

2.3-4

삽입 정렬은 다음과 같이 재귀 호출의 형태로 표현할 수 있다. $A[1..n]$을 정렬하기 위해 $A[1..n-1]$을 재귀적으로 정렬하고 $A[n]$을 정렬된 배열 $A[1..n-1]$에 삽입한다. 이와 같은 재귀 형태의 삽입 정렬에서 최악의 경우 수행시간에 관한 점화식을 써라.

c. 위 (a), (b)를 통해 최악의 경우 $\Theta(nk + n\lg(n/k))$ 시간에 실행될 수 있도록 변형된 알고리즘이 주어졌을 때, 그 알고리즘이 표준적인 병합 정렬 알고리즘과 동일한 점근적 수행시간을 갖는 k의 점근적(Θ-표기법 내의) 최댓값을 n의 함수로 표현하라.

d. 실용적으로는 k를 어떻게 정해야 하는가?

2-2 버블 정렬의 타당성

버블 정렬은 비효율적이지만 인기 있는 정렬 알고리즘이다. 이는 정렬되지 않은 이웃 원소를 반복적으로 바꿔 수열을 정렬한다.

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              A[j]와 A[j − 1]을 바꾼다.
```

a. A'을 BUBBLESORT(A)의 결과라고 하자. BUBBLESORT가 타당한지 증명하기 위해 이것이 종료하는지와 $n = A.length$일 때 다음이 성립하는지를 보여야 한다.

$$A'[1] \le A'[2] \le \cdots \le A'[n] \tag{2.3}$$

BUBBLESORT가 정말 정렬할 수 있는지 보이려면 무엇을 더 증명해야 하는가?

다음 두 가지는 부등식 (2.3)을 증명한다.

b. 2–4행의 **for** 루프에 대한 루프 불변성을 자세히 서술하고, 그것이 만족되는지를 증명하라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 한다.

c. (b)에서 증명한 루프 불변성의 종료조건을 활용해 부등식 (2.3)을 자연스럽게 증명할 1–4행의 **for** 루프의 불변식을 보여라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 한다.

d. 버블 정렬의 최악의 경우 수행시간은 무엇인가? 그리고 삽입 정렬과 비교하면 어떠한가?

2-3 Horner 공식의 타당성

다항식 계산을 위한 다음과 같은 Horner 공식이 있다.

c. 위 (a), (b)를 통해 최악의 경우 $\Theta(nk + n\lg(n/k))$ 시간에 실행될 수 있도록 변형된 알고리즘이 주어졌을 때, 그 알고리즘이 표준적인 병합 정렬 알고리즘과 동일한 점근적 수행시간을 갖는 k의 점근적(Θ-표기법 내의) 최댓값을 n의 함수로 표현하라.

d. 실용적으로는 k를 어떻게 정해야 하는가?

2-2 버블 정렬의 타당성

버블 정렬은 비효율적이지만 인기 있는 정렬 알고리즘이다. 이는 정렬되지 않은 이웃 원소를 반복적으로 바꿔 수열을 정렬한다.

```
BUBBLESORT(A)
1  for i = 1 to A.length − 1
2      for j = A.length downto i + 1
3          if A[j] < A[j − 1]
4              A[j]와 A[j − 1]을 바꾼다.
```

a. A'을 BUBBLESORT(A)의 결과라고 하자. BUBBLESORT가 타당한지 증명하기 위해 이것이 종료하는지와 $n = A.length$일 때 다음이 성립하는지를 보여야 한다.

$$A'[1] \le A'[2] \le \cdots \le A'[n] \tag{2.3}$$

BUBBLESORT가 정말 정렬할 수 있는지 보이려면 무엇을 더 증명해야 하는가?

다음 두 가지는 부등식 (2.3)을 증명한다.

$$P(x)=\sum_{k=0}^{n} a_k x^k$$
$$=a_0 + x(a_1 + x(a_2 + \cdots + x(a_{n-1} + xa_n)\cdots))$$

다음 코드는 계수 $a_0, a_1, \ldots, a_n$과 x값이 주어졌을 때 Horner 공식을 구현한 것이다.

```
1  y = 0
2  for i = n downto 0
3      y = a_i + x · y
```

a. Horner 공식을 구현한 이 코드의 점근적 수행시간을 Θ-표기법으로 나타내라.

b. 다항식의 각 항을 처음부터 일일이 계산하는 단순한 다항식 계산 알고리즘의 의사코드를 작성하라. 이 알고리즘의 수행시간은 무엇인가. Horner 공식과 비교하면 어떠한가?

c. 다음 루프 불변성을 생각하라.

2-3행에서 while 루프가 각 반복을 시작할 때 다음과 같다.

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$

항을 갖지 않는 경우의 합은 0으로 해석하라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 하며, 이 루프 불변성을 이용해 루프가 종료될 때 $y = \sum_{k=0}^{n} a_k x^k$라는 것도 보여라.

d. 끝으로, 주어진 코드가 계수가 $a_0, a_1, \ldots, a_n$인 다항식을 올바르게 계산함을 보여라.

a. Horner 공식을 구현한 이 코드의 점근적 수행시간을 Θ-표기법으로 나타내라.

b. 다항식의 각 항을 처음부터 일일이 계산하는 단순한 다항식 계산 알고리즘의 의사코드를 작성하라. 이 알고리즘의 수행시간은 무엇인가. Horner 공식과 비교하면 어떠한가?

c. 다음 루프 불변성을 생각하라.

2–3행에서 while 루프가 각 반복을 시작할 때 다음과 같다.

$$y = \sum_{k=0}^{n-(i+1)} a_{k+i+1} x^k$$

항을 갖지 않는 경우의 합은 0으로 해석하라. 증명 과정은 이 장에서 보인 루프 불변성 증명 구조를 잘 따라야 하며, 이 루프 불변성을 이용해 루프가 종료될 때 $y = \sum_{k=0}^{n} a_k x^k$라는 것도 보이라.

d. 끝으로, 주어진 코드가 계수가 $a_0, a_1, \ldots, a_n$인 다항식을 올바르게 계산함을 보여라.

2-4 역위

$A[1..n]$을 n개의 서로 다른 원소의 배열이라고 하자. $i < j$이고 $A[i] > A[j]$라면 (i, j) 쌍은 A의 **역위** 한 예가 된다.

a. 배열 $\langle 2, 3, 8, 6, 1 \rangle$의 역위 다섯 개를 나열하라.

b. 집합 $\{1, 2, \ldots, n\}$으로 이루어진 배열이 가장 많은 역위를 갖는 경우는 언제인가? 그리고 이때 역위는 몇 개인가?

c. 삽입 정렬의 수행시간과 입력 배열의 역위 개수 사이에는 어떤 관계가 있는가? 이 답을 증명하라.

d. 원소가 n개인 임의의 순열에서 역위의 개수를 최악의 경우 $\Theta(n \lg n)$ 시간에 알아내는 알고리즘을 제시하라(힌트: 병합 정렬을 변형해보라).

$$f(n)=O(g(n)) \iff g(n)=\Omega(f(n))$$
$$f(n)=o(g(n)) \iff g(n)=\omega(f(n))$$

점근적 표기에서 이런 특성이 성립하므로 두 실수 a와 b의 비교와 두 함수 f와 g의 점근적 비교의 유사성을 다음과 같이 비유할 수 있다.

$f(n) = O(g(n)) \approx a \leq b$

$f(n) = \Omega(g(n)) \approx a \geq b$

$f(n) = \Theta(g(n)) \approx a = b$

$f(n) = o(g(n)) \approx a < b$

$f(n) = \omega(g(n)) \approx a > b$

$f(n) = o(g(n))$이면 $f(n)$이 $g(n)$보다 **점근적으로 작다**고 하고, $f(n) = \omega(g(n))$이면 $f(n)$이 $g(n)$보다 **점근적으로 크다**고 한다.

그러나 실수의 특성 중 다음 하나는 점근적 표기에서 적용되지 않는다.

삼분법: 임의의 두 실수 a와 b에 대해 $a < b, a = b, a > b$ 중 정확히 한 가지만 성립한다.

임의의 두 실수는 항상 비교할 수 있지만 함수를 언제나 점근적으로 비교할 수 있는 것은 아니다. 즉, 두 함수 $f(n)$과 $g(n)$에 대해서 $f(n) = O(g(n))$과 $f(n) = \Omega(g(n))$, 어느 것도 성립하지 않는 경우가 있다. 예를 들면, 함수 n과 $n^{1+\sin n}$은 $n^{1+\sin n}$의 지수 값이 0과 2 사이의 모든 값을 가지면서 진동하므로 점근적 표기를 사용해 비교할 수 없다.

연습문제

3.1-1

$f(n)$과 $g(n)$이 점근적으로 음이 아닌 함수라 하자. $\max(f(n), g(n)) = \Theta(f(n)+g(n))$임을 증명하라 (단, Θ-표기의 기본 정의를 사용해야 한다).

3.1-2

$b > 0$인 임의의 실수 상수 a와 b에 대해 다음을 보여라.

$$(n + a)^b = \Theta(n^b) \tag{3.2}$$

3.1-3

"알고리즘 A의 수행시간은 적어도 $O(n^2)$이다"라는 문장이 의미가 없는 이유를 설명하라.

3.1-4

$2^{n+1} = O(2^n)$인가, $2^{2n} = O(2^n)$인가?

3.1-5

정리 3.1을 증명하라.

3.1-6

알고리즘의 수행시간이 $\Theta(g(n))$일 필요충분조건은 그 알고리즘의 최악의 경우 수행시간이 $O(g(n))$이고 최선 수행시간이 $\Omega(g(n))$임을 증명하라.

3.1-7

$o(g(n)) \cap \omega(g(n))$은 공집합임을 증명하라.

3.1-8

앞에서 배운 독립적으로 다른 비율로 무한으로 가는 두 개의 매개변수 n과 m을 가진 경우를 확장할 수 있다. 주어진 함수 $g(n, m)$에 대해 $O(g(n, m))$을 다음과 같은 함수들의 집합으로 나타낸다.

$$O(g(n,m)) = \{f(n,m) : \text{모든 } n \geq n_0 \text{과 } m \geq m_0 \text{에 대해 } 0 \leq f(n,m) \leq cg(n,m) \text{인 양의 상수 } c, n_0, m_0 \text{이 존재한다.}\}$$

이에 맞추어 $\Omega(g(n, m))$과 $\Theta(g(n, m))$에 해당하는 정의를 써라.

이것은 i번째 피보나치 수 F_i가 $\phi^i/\sqrt{5}$를 가장 가까운 정수로 반올림한 값과 같음을 의미한다. 따라서 피보나치 수는 지수적으로 증가한다.

연습문제

3.2-1

$f(n)$과 $g(n)$이 단조증가 함수면 $f(n) + g(n)$, $f(g(n))$도 단조증가 함수임을 보여라. 그리고 $f(n)$과 $g(n)$이 음이 아니라는 조건이 더 주어지면 $f(n) \cdot g(n)$가 단조증가함도 보여라.

3.2-2

식 (3.16)을 증명하라.

3.2-3

식 (3.19)를 증명하라. 또한 $n! = \omega(2^n)$ 그리고 $n! = O(n^n)$을 증명하라.

3.2-4 ★

함수 $\lceil \lg n \rceil !$은 다항식적으로 한정되는가, 함수 $\lceil \lg \lg n \rceil !$은 다항식적으로 한정되는가?

3.2-5 ★

$\lg(\lg^* n)$와 $\lg^*(\lg n)$ 중 어느 것이 점근적으로 더 큰가?

3.2-6

황금율 ϕ와 그 컬레수 $\hat{\phi}$가 식 $x^2 = x + 1$을 만족시킴을 증명하라.

3.2-7

i번째 피보나치 수가 다음 식을 만족한다는 것을 귀납법으로 증명하라. 여기서 ϕ는 황금률이고 $\hat{\phi}$은 그 컬레수다.

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

종합문제

3-1 다항식의 점근적 특성

$a_d > 0$일 때 다음 식이 n에 대한 d차 다항식이고 k가 상수라고 하자.

$$p(n) = \sum_{i=0}^{d} a_i n^i$$

점근적 표기의 정의를 사용해 다음 특성을 증명하라.

a. $k \geq d$이면 $p(n) = O(n^k)$이다.

b. $k \leq d$이면 $p(n) = \Omega(n^k)$이다.

c. $k = d$이면 $p(n) = \Theta(n^k)$이다.

d. $k > d$이면 $p(n) = o(n^k)$이다.

e. $k < d$이면 $p(n) = \omega(n^k)$이다.

3-2 상대적인 점근적 증가

다음 표에 있는 식의 쌍 (A, B) 각각에 대해 A가 B의 O, o, Ω, ω인지 밝히라. $k \geq 1$, $\epsilon > 0$, $c > 1$은 상수라고 가정한다. 표의 각 칸에 "yes"나 "no"로 표시하라.

	A	B	O	o	Ω	ω	Θ
a.	$\lg^k n$	n^ϵ					
b.	n^k	c^n					
c.	$\sqrt{n}$	$n^{\sin n}$					
d.	2^n	$2^{n/2}$					
e.	$n^{\lg c}$	$c^{\lg n}$					

3-3 점근적 증가율에 따라 순서 매기기

a. 다음 함수를 증가 순서에 따라 번호를 매겨라. 즉, $g_1 = \Omega(g_2)$, $g_2 = \Omega(g_3)$, ..., $g_{29} = \Omega(g_{30})$을 만족하는 함수의 배열 $g_1, g_2, \ldots, g_{30}$을 찾아라. 그리고 작성한 목록을 $f(n)$과 $g(n)$이 같은 클래스에 속할 필요충분조건이 $f(n) = \Theta(g(n))$인 동류항(equivalence class)으로 나누어라.

$\lg(\lg^* n)$ $2^{\lg^* n}$ $(\sqrt{2})^{\lg n}$ n^2 $n!$ $(\lg n)!$

$(\frac{3}{2})^n$ n^3 $\lg^2 n$ $\lg(n!)$ 2^{2^n} $n^{1/\lg n}$

$\ln\ln n$ $\lg^* n$ $n \cdot 2^n$ $n^{\lg\lg n}$ $\ln n$ 1

$2^{\lg n}$ $(\lg n)^{\lg n}$ e^n $4^{\lg n}$ $(n+1)!$ $\sqrt{\lg n}$

$\lg^*(\lg n)$ $2^{\sqrt{2\lg n}}$ n 2^n $n \lg n$ $2^{2^{n+1}}$

b. (a) 부분의 모든 함수 $g_i(n)$에 대해 $O(g_i(n))$도 아니고 $\Omega(g_i(n))$도 아닌, 음이 아닌 함수 $f(n)$의 예를 하나만 들라.

3-4 점근적 표기의 특성

$f(n)$과 $g(n)$을 점근적으로 양인 함수라고 하자. 다음 가정이 옳은지 또는 옳지 않은지 증명하라.

a. $f(n) = O(g(n))$이면 $g(n) = O(f(n))$이다.

b. $f(n) + g(n) = \Theta(\min(f(n), g(n)))$이다.

c. 충분히 큰 모든 n에 대해 $\lg(g(n)) \geq 1$이고 $f(n) \geq 1$이라고 할 때, $f(n) = O(g(n))$이면 $\lg(f(n)) = O(\lg(g(n)))$이다.

d. $f(n) = O(g(n))$이면 $2^{f(n)} = O\left(2^{g(n)}\right)$이다.

e. $f(n) = O\left((f(n))^2\right)$이다.

f. $f(n) = O(g(n))$이면 $g(n) = \Omega(f(n))$이다.

g. $f(n) = \Theta(f(n/2))$이다.

h. $f(n) + o(f(n)) = \Theta(f(n))$이다.

이와 같이 분할정복 기법이 주먹구구식 방법보다 점근적으로 더 빠른 알고리즘을 만들어 낼 수 있음을 알게 되었다. 병합 정렬과 최대 부분 배열 문제를 통해 분할정복 기법이 얼마나 강력한 힘을 발휘할 수 있는지에 대한 느낌을 조금 갖게 된 것이다. 분할정복 기법은 종종 점근적으로 가장 빠르게 푸는 알고리즘을 만들어내지만, 그보다 더 빠른 알고리즘을 만들 수도 있다. 연습문제 4.1-5와 같이 최대 부분 배열 문제를 푸는 선형 시간 알고리즘이 존재하며 이 알고리즘은 분할정복 기법을 사용하지 않는다.

연습문제

4.1-1

A의 모든 원소가 음수일 때 FIND-MAXIMUM-SUBARRAY가 반환하는 값은 무엇인가?

4.1-2

최대 부분 배열 문제를 푸는 주먹구구식 방법에 대한 의사코드를 작성하라. 작성한 프로시저는 $\Theta(n^2)$ 시간에 수행되어야 한다.

4.1-3

당신의 컴퓨터를 사용해 최대 부분 배열 문제를 푸는 주먹구구식 알고리즘과 재귀 알고리즘을 둘 다 구현하라. 재귀 알고리즘이 주먹구구식 알고리즘을 넘어설 때의 교차 지점이 되는 문제의 크기 n_0은 무엇인가? 그리고 문제의 크기가 n_0보다 작을 때 주먹구구식 알고리즘을 사용하도록 재귀 알고리즘의 베이스 케이스를 수정하라. 이로 인해 교차 지점이 달라지는가?

4.1-4

최대 부분 배열 문제를 수정해 빈 부분 수열을 결과로 허용한다고 가정하자. 빈 부분 배열의 합은 0이다. 빈 부분 수열을 허용하지 않는 알고리즘이 빈 부분 배열을 결과로 허용하도록 하려면 어떻게 수정해야 할까?

4.1-5

다음 아이디어를 이용해 최대 부분 배열 문제를 푸는 재귀적이지 않은 선형 시간 알
고리즘을 개발하라. 배열의 왼쪽 끝에서 시작해 오른쪽으로 진행하면서 지금까지의
최대 부분 배열을 기록한다. $A[1..j]$의 최대 부분 배열을 알고 있으면 다음 관찰을
이용해 인덱스 $j+1$에서 끝나는 최대 부분 배열을 찾을 수 있도록 확장할 수 있다.
$A[1..j+1]$의 최대 부분 배열은 $1 \le i \le j+1$에 대해 $A[1..j]$의 최대 부분 배열이
거나 부분 배열 $A[i..j+1]$이다. 인덱스 j로 끝나는 최대 부분 배열을 알고 있다는
사실에 기반해서 $A[i..j+1]$ 형태의 최대 부분 배열을 상수 시간에 구하라.

4.1-5

다음 아이디어를 이용해 최대 부분 배열 문제를 푸는 재귀적이지 않은 선형 시간 알고리즘을 개발하라. 배열의 왼쪽 끝에서 시작해 오른쪽으로 진행하면서 지금까지의 최대 부분 배열을 기록한다. $A[1..j]$의 최대 부분 배열을 알고 있으면 다음 관찰을 이용해 인덱스 $j+1$에서 끝나는 최대 부분 배열을 찾을 수 있도록 확장할 수 있다. $A[1..j+1]$의 최대 부분 배열은 $1 \le i \le j+1$에 대해 $A[1..j]$의 최대 부분 배열이거나 부분 배열 $A[i..j+1]$이다. 인덱스 j로 끝나는 최대 부분 배열을 알고 있다는 사실에 기반해서 $A[i..j+1]$ 형태의 최대 부분 배열을 상수 시간에 구하라.

4.2 행렬 곱셈을 위한 스트라센 알고리즘

행렬에 대해서 알고 있다면 행렬을 어떻게 곱하는지도 알고 있을 것이다(그렇지 않다면 부록 D의 D.1절을 읽어보기 바란다). $A = (a_{ij})$와 $B = (b_{ij})$가 $n \times n$ 정사각 행렬들일 때 두 행렬의 곱 $C = A \cdot B$의 원소 c_{ij}를 다음과 같이 정의한다. 여기서 $i, j = 1, 2, \ldots, n$이다.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \tag{4.8}$$

n^2개의 행렬 원소를 계산해야 하고, 각 원소는 n개 값의 합이다. 다음 프로시저는 $n \times n$ 행렬 A와 B를 받아서 곱셈을 한 후 $n \times n$ 행렬곱 C를 반환한다. 각 행렬은 행의 수를 나타내는 인자 *rows*를 가지고 있다고 가정한다.

```
SQUARE-MATRIX-MULTIPLY(A, B)
1  n = A.rows
2  C는 새로운 n × n 행렬이라 하자.
```

4.1-5

다음 아이디어를 이용해 최대 부분 배열 문제를 푸는 재귀적이지 않은 선형 시간 알고리즘을 개발하라. 배열의 왼쪽 끝에서 시작해 오른쪽으로 진행하면서 지금까지의 최대 부분 배열을 기록한다. $A[1..j]$의 최대 부분 배열을 알고 있으면 다음 관찰을 이용해 인덱스 $j+1$에서 끝나는 최대 부분 배열을 찾을 수 있도록 확장할 수 있다. $A[1..j+1]$의 최대 부분 배열은 $1 \le i \le j+1$에 대해 $A[1..j]$의 최대 부분 배열이거나 부분 배열 $A[i..j+1]$이다. 인덱스 j로 끝나는 최대 부분 배열을 알고 있다는 사실에 기반해서 $A[i..j+1]$ 형태의 최대 부분 배열을 상수 시간에 구하라.

4.2 행렬 곱셈을 위한 스트라센 알고리즘

행렬에 대해서 알고 있다면 행렬을 어떻게 곱하는지도 알고 있을 것이다 (그렇지 않다면 부록 D의 D.1절을 읽어보기 바란다). $A = (a_{ij})$와 $B = (b_{ij})$가 $n \times n$ 정사각 행렬들일 때 두 행렬의 곱 $C = A \cdot B$의 원소 c_{ij}를 다음과 같이 정의한다. 여기서 $i, j = 1, 2, \ldots, n$이다.

$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj} \tag{4.8}$$

n^2개의 행렬 원소를 계산해야 하고, 각 원소는 n개 값의 합이다. 다음 프로시저는 $n \times n$ 행렬 A와 B를 받아서 곱셈을 한 후 $n \times n$ 행렬곱 C를 반환한다. 각 행렬은 행의 수를 나타내는 인자 *rows*를 가지고 있다고 가정한다.

SQUARE-MATRIX-MULTIPLY(A, B)

1 $n = A.rows$

2 C는 새로운 $n \times n$ 행렬이라 하자.

주의: 연습문제 4.2-3, 4.2-4, 4.2-5는 스트라센 알고리즘의 변형에 관한 것이지만, 이 문제를 풀기 전에 4.5절을 먼저 읽어보아야 한다.

4.2-1

스트라센 알고리즘을 이용해 다음 행렬 곱을 계산하는 과정을 보여라.

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}\begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$$

4.2-2

스트라센 알고리즘의 의사코드를 작성하라.

4.2-3

$n \times n$ 행렬을 곱할 때 n이 정확히 2의 거듭제곱이 아니라면 스트라센 알고리즘을 어떻게 수정해야 하는가? 결과로 나온 알고리즘이 $\Theta(n^{\lg 7})$ 시간에 수행됨을 보여라.

4.2-4

k개의 곱셈을 이용해 3×3 행렬을 곱할 수 있다고 하면(곱셈의 교환 법칙은 가정하지 않고), $n \times n$ 행렬을 $o(n^{\lg 7})$ 시간에 곱할 수 있는 가장 큰 k는 무엇인가? 이 알고리즘의 수행시간은 어떻게 되는가?

4.2-5

V. Pan은 132,464번의 곱셈으로 68×68 행렬을 곱하는 법과, 143,640번의 곱셈으로 70×70 행렬을 곱하는 법, 155,424번의 곱셈으로 72×72 행렬을 곱하는 법을 찾아냈다. 분할정복 행렬 곱셈 알고리즘으로 이용했을 때 어느 방법이 가장 좋은 점근적 수행시간을 보이는가? 스트라센 알고리즘과 비교하면 어떤가?

4.2-6

스트라센 알고리즘을 서브루틴으로 사용했을 때 $kn \times n$ 행렬과 $n \times kn$ 행렬의 곱셈을 얼마나 빨리 계산할 수 있는가? 입력 행렬의 순서를 바꾸었을 때 같은 질문에 답하라.

이것은 점화식 (4.19)와 매우 비슷하다. 실제로 이 새로운 점화식은 같은 해, $S(m) = O(m \lg m)$을 가진다. $S(m)$을 $T(n)$으로 다시 바꾸면 다음을 얻는다.

$$T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$$

연습문제

4.3-1

$T(n) = T(n-1) + n$의 해가 $O(n^2)$임을 보여라.

4.3-2

$T(n) = T(\lceil n/2 \rceil) + 1$의 해가 $O(\lg n)$임을 보여라.

4.3-3

$T(n) = 2T(\lfloor n/2 \rfloor) + n$의 해가 $O(n \lg n)$임을 보았다. 이 점화식의 해가 $\Omega(n \lg n)$도 됨을 보여라. 그리고 해가 $\Theta(n \lg n)$이 된다고 결론지어라.

4.3-4

점화식 (4.19)에 대해 다른 귀납적 가정을 함으로써 $(T) = 1$이라는 한계 조건을 그대로 두고서도 어려움을 해결할 수 있음을 보여라.

4.3-5

$\Theta(n \lg n)$이 병합 정렬의 점화식 (4.3)의 "정확한"해임을 보여라.

4.3-6

$T(n) = 2T(\lfloor n/2 \rfloor + 17) + n$의 해가 $O(n \lg n)$임을 보여라.

4.3-7

4.5절의 마스터 방법을 사용하면, 점화식 $T(n) = 4T(n/3) + n$의 해가 $T(n) = \Theta(n^{\log_3 4})$임을 보일 수 있다. $T(n) \leq cn^{\log_3 4}$라는 가정으로 치환법 증명을 할 수 없음을 보여라. 그리고 치환법 증명을 하기 위해 저차항을 어떻게 빼야 하는지 설명하라.

4.3-8

4.5절의 마스터 방법을 사용하면 점화식 $T(n) = 4T(n/2) + n$의 해가 $T(n) = \Theta(n^2)$임을 보일 수 있다. $T(n) \leq cn^2$이라는 가정으로 치환법 증명을 할 수 없음을 보여라. 그리고 치환법 증명을 하기 위해 저차항을 어떻게 빼야 하는지 설명하라.

4.3-9

변수를 바꿈으로써 점화식 $T(n) = 3T(\sqrt{n}) + \log n$을 풀라. 구한 해가 점근적으로 정확해야 한다. 값이 정수인지 아닌지는 생각하지 않아도 좋다.

[illegible] 내부 노드가 비게 된다. 결과적으로 재귀 트리의 아래쪽 레벨은 총 비용에 cn보다 적게 기여한다. 모든 비용의 정확한 계산을 할 수 있지만 여기서는 단지 치환법에 쓰기 위한 추측식을 제안하기 위해 노력하고 있음을 잊지 말자. 조잡함을 참고 $O(n \lg n)$이 상한이라는 추측이 옳다는 것을 보이자.

실제로 $O(n \lg n)$이 점화식 해의 상한이라는 것을 검증하기 위해 치환법을 사용할 수 있다. d가 적당한 양의 상수일 때 $T(n) \le dn \lg n$이라는 것을 보인다. $d \ge c/(\lg 3 - (2/3))$일 때 다음과 같이 할 수 있다.

$$
\begin{aligned}
T(n) &\le T(n/3) + T(2n/3) + cn \\
&\le d(n/3)\lg(n/3) + d(2n/3)\lg(2n/3) + cn \\
&= (d(n/3)\lg n - d(n/3)\lg 3) \\
&\qquad + (d(2n/3)\lg n - d(2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg(3/2)) + cn \\
&= dn\lg n - d((n/3)\lg 3 + (2n/3)\lg 3 - (2n/3)\lg 2) + cn \\
&= dn\lg n - dn(\lg 3 - 2/3) + cn \\
&\le dn\lg n
\end{aligned}
$$

그러므로 재귀 트리에서 굳이 더 정확한 비용을 계산할 필요가 없다.

연습문제

4.4-1

점화식 $T(n) = 3T(\lfloor n/2 \rfloor) + n$에 대해 재귀 트리를 사용해 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-2

점화식 $T(n) = T(n/2) + n^2$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-3

점화식 $T(n) = 4T(n/2 + 2) + n$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-4

점화식 $T(n) = 2T(n-1) + 1$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-5

점화식 $T(n) = T(n-1) + T(n/2) + n$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-6

c가 상수일 때 재귀 트리를 써서 점화식 $T(n) = T(n/3) + T(2n/3) + cn$의 해가 $\Omega(n \lg n)$임을 보여라.

4.4-7

c가 상수일 때 $T(n) = 4T(\lfloor n/2 \rfloor) + cn$에 대한 재귀 트리를 그리고 그 해의 정확한 점근적 한계를 제시하라. 아울러 구한 한계를 치환법으로 검증하라.

4.4-8

$a \geq 1$과 $c > 0$이 상수일 때 재귀 트리를 사용해 점화식 $T(n) = T(n-a) + T(a) + cn$에 대해 점근적으로 정확한 해를 구하라.

4.4-9

α는 범위가 $0 < \alpha < 1$인 상수고 $c > 0$ 또한 상수일 때, 재귀 트리를 사용해 점화식 $T(n) = T(\alpha n) + T((1-\alpha)n) + cn$에 대해 점근적으로 정확한 해를 구하라.

4.4-2

점화식 $T(n) = T(n/2) + n^2$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-3

점화식 $T(n) = 4T(n/2 + 2) + n$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-4

점화식 $T(n) = 2T(n - 1) + 1$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-5

점화식 $T(n) = T(n - 1) + T(n/2) + n$에 대해 재귀 트리를 사용해서 좋은 점근적 상한을 구하라. 답을 검증하기 위해 치환법을 사용하라.

4.4-6

c가 상수일 때 재귀 트리를 써서 점화식 $T(n) = T(n/3) + T(2n/3) + cn$의 해가 $\Omega(n \lg n)$임을 보여라.

4.4-7

c가 상수일 때 $T(n) = 4T(\lfloor n/2 \rfloor) + cn$에 대한 재귀 트리를 그리고 그 해의 정확한 점근적 한계를 제시하라. 아울러 구한 한계를 치환법으로 검증하라.

4.4-8

$a \geq 1$과 $c > 0$이 상수일 때 재귀 트리를 사용해 점화식 $T(n) = T(n - a) + T(a) + cn$에 대해 점근적으로 정확한 해를 구하라.

여기서는 $a = 7, b = 2, f(n) = \Theta(n^2)$이므로 $n^{\log_b a} = n^{\log_2 7}$이다. $\log_2 7$을 $\lg 7$로 다시 쓰고 $2.80 < \lg 7 < 2.81$임을 상기하면, $\epsilon = 0.8$에 대해 $f(n) = O(n^{\lg 7 - \epsilon})$임을 알 수 있다. 또 다시 경우 1을 적용하면 해 $T(n) = \Theta(n^{\lg 7})$을 얻을 수 있다.

연습문제

4.5-1

마스터 방법을 사용해 다음 점화식들의 정확한 점근적 한계를 구하라.

a. $T(n) = 2T(n/4) + 1$

b. $T(n) = 2T(n/4) + \sqrt{n}$

c. $T(n) = 2T(n/4) + n$

d. $T(n) = 2T(n/4) + n^2$

4.5-2

Caesar 교수는 스트라센 알고리즘보다 점근적으로 더 빠른 행렬 곱셈 알고리즘을 개발하기 원한다. 그의 알고리즘은 각 행렬을 $n/4 \times n/4$ 크기로 나누고, 분할과 결합에 $\Theta(n^2)$ 시간이 걸리는 분할정복 알고리즘을 사용한다. 그가 스트라센 알고리즘을 이기려면 얼마나 많은 부분 문제를 만들어야 하는지 알아야 한다. 그의 알고리즘이 a개의 부분 문제를 만든다면 수행시간 $T(n)$에 대한 점화식은 $T(n) = aT(n/4) + \Theta(n^2)$이 된다. Caesar 교수의 알고리즘이 스트라센 알고리즘보다 빠르기 위한 가장 큰 정수 a는 무엇인가?

4.5-3

마스터 방법을 사용해 이진 검색의 점화식 $T(n) = T(n/2) + \Theta(1)$의 해가 $T(n) = \Theta(\lg n)$임을 보이라(이진 검색에 대한 설명은 연습문제 2.3-5를 참고).

4.5-4

마스터 방법이 점화식 $T(n) = 4T(n/2) + n^2 \lg n$에 적용될 수 있는가? 그 이유는 무엇인가? 이 점화식의 점근적 상한을 제시하라.

$$T(n) = 7T(n/2) + \Theta(n^2)$$

여기서는 $a = 7$, $b = 2$, $f(n) = \Theta(n^2)$이므로 $n^{\log_b a} = n^{\log_2 7}$이다. $\log_2 7$을 $\lg 7$로 다시 쓰고 $2.80 < \lg 7 < 2.81$임을 상기하면, $\epsilon = 0.8$에 대해 $f(n) = O(n^{\lg 7 - \epsilon})$임을 알 수 있다. 또 다시 경우 1을 적용하면 해 $T(n) = \Theta(n^{\lg 7})$을 얻을 수 있다.

연습문제

4.5-1

마스터 방법을 사용해 다음 점화식들의 정확한 점근적 한계를 구하라.

a. $T(n) = 2T(n/4) + 1$

b. $T(n) = 2T(n/4) + \sqrt{n}$

c. $T(n) = 2T(n/4) + n$

d. $T(n) = 2T(n/4) + n^2$

4.5-2

Caesar 교수는 스트라센 알고리즘보다 점근적으로 더 빠른 행렬 곱셈 알고리즘을 개발하기 원한다. 그의 알고리즘은 각 행렬을 $n/4 \times n/4$ 크기로 나누고, 분할과 결합에 $\Theta(n^2)$ 시간이 걸리는 분할정복 알고리즘을 사용한다. 그가 스트라센 알고리즘을 이기려면 얼마나 많은 부분 문제를 만들어야 하는지 알아야 한다. 그의 알고리즘이 a개의 부분 문제를 만든다면 수행시간 $T(n)$에 대한 점화식은 $T(n) = aT(n/4) + \Theta(n^2)$이 된다. Caesar 교수의 알고리즘이 스트라센 알고리즘보다 빠르기 위한 가장 큰 정수 a는 무엇인가?

4.5-3

마스터 방법을 사용해 이진 검색의 점화식 $T(n) = T(n/2) + \Theta(1)$의 해가 $T(n) = \Theta(\lg n)$임을 보여라(이진 검색에 대한 설명은 연습문제 2.3-5를 참고).

4.5-4

마스터 방법이 점화식 $T(n) = 4T(n/2) + n^2 \lg n$에 적용될 수 있는가? 그 이유는 무엇인가? 이 점화식의 점근적 상한을 제시하라.

종합문제

4-1 점화식의 예들

다음 점화식 각각에 대해 $T(n)$의 점근적 상한과 하한을 제시하라. $n \leq 2$에 대해 $T(n$
이 상수라고 가정한다. 상한과 하한을 가능한 정확하게 하고 답이 옳음을 보여라.

a. $T(n) = 2T(n/2) + n^4$

b. $T(n) = T(7n/10) + n$

c. $T(n) = 16T(n/4) + n^2$

d. $T(n) = 7T(n/3) + n^2$

e. $T(n) = 7T(n/2) + n^2$

f. $T(n) = 2T(n/4) + \sqrt{n}$

g. $T(n) = T(n-2) + n^2$

종합문제

4-1 점화식의 예들

다음 점화식 각각에 대해 $T(n)$의 점근적 상한과 하한을 제시하라. $n \leq 2$에 대해 $T(n)$이 상수라고 가정한다. 상한과 하한을 가능한 정확하게 하고 답이 옳음을 보여라.

a. $T(n) = 2T(n/2) + n^4$

b. $T(n) = T(7n/10) + n$

c. $T(n) = 16T(n/4) + n^2$

d. $T(n) = 7T(n/3) + n^2$

e. $T(n) = 7T(n/2) + n^2$

f. $T(n) = 2T(n/4) + \sqrt{n}$

g. $T(n) = T(n-2) + n^2$

연습문제

4.6-1 •

추가 입력의 실수 α가 어느 범위의 양의 상수인 경우에 식 (4.27)의 n_0에 대한 간단하고 정확한 식을 제시하라.

4.6-2 •

$k \geq 0$일 때, $f(n) = \Theta(n^{\log_b a} \lg^k n)$이라면 마스터 점화식이 $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$의 해를 가짐을 보여라. 간단하게 하기 위해 정식을 b의 거듭제곱에 대해서만으로 제한하라.

4.6-3 •

어떤 상수 $c < 1$에 대한 정규 조건 $af(n/b) \leq cf(n)$이 만족되면 $f(n) = \Omega(n^{\log_b a + \varepsilon})$인 상수 $\varepsilon > 0$이 존재하므로, 마스터 정리의 경우 3에 불필요한 말이 들어 있음을 보여라.

종합문제

4-1 점화식의 예들

다음 점화식 각각에 대해 $T(n)$의 점근적 상한과 하한을 제시하라. $n \leq 2$에 대해 $T(n)$이 상수라고 가정한다. 상한과 하한을 가능한 정확하게 하고 답이 옳음을 보여라.

a. $T(n) = 2T(n/2) + n^4$

b. $T(n) = T(7n/10) + n$

c. $T(n) = 16T(n/4) + n^2$

d. $T(n) = 7T(n/3) + n^2$

e. $T(n) = 7T(n/2) + n^2$

f. $T(n) = 2T(n/4) + \sqrt{n}$

g. $T(n) = T(n-2) + n^2$

4-2 인자를 넘겨주는 비용

이 책 전체에 걸쳐 프로시저 호출 동안 원소의 개수가 N인 배열을 넘겨 준다 해도 인자를 넘겨주는 것은 상수 시간이 걸린다고 가정한다. 보통 배열 자체가 아니라 배열의 포인터가 넘겨지므로 이 가정은 대부분의 시스템에서 유효하다. 이 문제에서는 매개변수를 넘겨주는 세 가지 전략들의 관계를 조사한다.

1. 배열이 포인터로 넘겨진다. 걸리는 시간은 $\Theta(1)$이다.
2. 배열이 복사되어 넘겨진다. N이 배열의 크기일 때 걸리는 시간은 $\Theta(N)$이다.
3. 호출된 프로시저에 필요한 부분 범위에 대해서만 배열이 복사되어 넘겨진다. 부분 배열 $A[p..q]$가 넘겨진다고 할 때 걸리는 시간은 $\Theta(q - p + 1)$이다.

a. 정렬된 배열에서 숫자를 찾아내는 재귀적인 이진 검색 알고리즘을 고려하자(연습문제 2.3-5 참고). 배열이 위 세 가지 방법 각각을 사용해 넘겨질 때 이진 검색의 최악의 경우 수행시간에 대한 점화식을 제시하고, 그 점화식의 해에 대한 적절한 상한을 제시하라. N을 원래 문제의 크기라 하고 n을 부분 문제의 크기라 하자.

b. 2.3.1절의 MERGE-SORT 알고리즘에 대해 (a) 부분을 다시 풀라.

4-3 좀 더 많은 점화식의 예

다음에 주어진 점화식에서 $T(n)$의 점근적 상한과 하한을 각각 제시하라. $T(n)$이 충분히 작은 n에 대해 상수라고 가정한다. 상한과 하한을 가능한 정확하게 하고 답이 옳음을 보이라.

a. $T(n) = 4T(n/3) + n \lg n$

b. $T(n) = 3T(n/3) + n/\lg n$

c. $T(n) = 4T(n/2) + n^2\sqrt{n}$

d. $T(n) = 3T(n/3 - 2) + n/2$

e. $T(n) = 2T(n/2) + n/\lg n$

f. $T(n) = T(n/2) + T(n/4) + T(n/8) + n$

g. $T(n) = T(n - 1) + 1/n$

h. $T(n) = T(n - 1) + \lg n$

i. $T(n) = T(n-2) + 1/\lg n$

j. $T(n) = \sqrt{n}T(\sqrt{n}) + n$

4-4 피보나치 수

이 문제에서는 피보나치 수의 특성을 발전시켜 생각하는데, 이는 점화식 (3.22)에 의해 정의된다. 피보나치 점화식을 풀기 위해 생성 함수 기법을 사용할 수 있다. *생성 함수*(또는 *형식적 멱급수*) $\mathcal{F}$는 다음과 같이 정의한다.

$$\begin{aligned}\mathcal{F}(z) &= \sum_{i=0}^{\infty} F_i z^i \\ &= 0 + z + z^2 + 2z^3 + 3z^4 + 5z^5 + 8z^6 + 13z^7 + 21z^8 + \cdots\end{aligned}$$

여기서 F_i는 i번째 피보나치 수다.

a. $\mathcal{F}(z) = z + z\mathcal{F}(z) + z^2\mathcal{F}(z)$임을 보여라.

b. $\phi = \dfrac{1+\sqrt{5}}{2} = 1.61803\ldots$

와

$$\hat{\phi} = \frac{1-\sqrt{5}}{2} = -0.61803\ldots$$

에 대해 다음을 보여라.

$$\begin{aligned}\mathcal{F}(z) &= \frac{z}{1-z-z^2} \\ &= \frac{z}{(1-\phi z)(1-\hat{\phi}z)} \\ &= \frac{1}{\sqrt{5}}\left(\frac{1}{1-\phi z} - \frac{1}{1-\hat{\phi}z}\right)\end{aligned}$$

c. 다음을 보여라.

$$\mathcal{F}(z) = \sum_{i=0}^{\infty} \frac{1}{\sqrt{5}}(\phi^i - \hat{\phi}^i)z^i$$

d. (c) 부분을 이용해, $i > 0$에 대해 가장 가까운 정수로 반올림해서 $F_i = \phi^i/\sqrt{5}$임을 증명하라(*힌트*: $|\hat{\phi}| < 1$이라는 것을 이용하라).

예를 들어, RANDOM(0, 1)은 각각 1/2의 확률로 0과 1을 리턴한다. RANDOM(3, 7)을 호출하면 각각 1/5의 확률로 3, 4, 5, 6, 7을 리턴한다. RANDOM이 리턴하는 정수는 이전의 호출로 리턴된 값과는 독립이다. RANDOM을 $(b - a + 1)$개의 면을 가진 주사위를 굴리는 것과 같다고 생각할 수 있다(실제로 대부분의 프로그래밍 환경에서는 ***의사 난수 생성기***를 제공하는데, 이는 통계적으로 난수처럼 "보이는" 수를 리턴하는 결정론적(deterministic) 알고리즘이다).

랜덤화된 알고리즘의 수행시간을 분석할 때 난수 생성기가 리턴하는 값의 분포에 대한 수행시간의 평균값을 구한다. 이 수행시간을 입력이 랜덤한 알고리즘의 평균 수행시간과 구분하기 위해 ***기대 수행시간***이라 부른다. 대체로 입력의 확률적 분포를 알 때는 평균 수행시간을 분석하고 알고리즘 그 자체가 랜덤화된 경우에는 기대 수행시간을 분석한다.

연습문제

5.1-1

HIRE-ASSISTANT 프로시저의 4행에서 어떤 지원자가 가장 적합한지를 결정할 수 있다면 지원자들의 전체 순위를 알 수 있게 됨을 보이라.

5.1-2 ★

RANDOM(0, 1)만 호출해 동작하도록 RANDOM(a, b)의 구현을 작성하라. 그리고 그 알고리즘에서 예상되는 수행시간을 a와 b의 함수로 표현하라.

5.1-3 ★

0과 1을 각각 1/2의 확률로 출력하고자 한다. 함수 BIASED-RANDOM은 $0 < p < 1$인 p의 확률로 1을 출력하고 $1 - p$의 확률로 0을 출력하지만, p값은 모른다. BIASED-RANDOM을 서브 루틴으로 사용해 0과 1을 똑같이 1/2의 확률로 리턴하는 알고리즘을 제시하라. 그리고 이 알고리즘의 기대 수행시간을 p의 함수로 표현하라.

5.2 지표 확률 변수

지표 확률 변수를 사용하면 고용 문제를 포함해 많은 알고리즘을 분석할 수 있다. 지표 확률 변수는 확률과 기댓값 사이의 변환을 용이하게 한다. 표본 공간 S와 사건 A가 주어졌을 때 사건 A에 관한 ***지표 확률 변수*** $\mathrm{I}\{A\}$는 다음과 같이 정의된다.

보조정리 5.2

지원자가 무작위 순서로 온다고 할 때, HIRE-ASSISTANT 알고리즘은 총 $O(c_h \ln n)$의 평균 고용 비용이 든다.

증명 고용 비용에 대한 정의와 식 (5.5)에 의해 바로 만족된다. 이는 기대 고용 횟수가 대략 $\ln n$임을 증명한다. ■

고용 비용의 기댓값은 최악의 고용 비용 $O(c_h n)$보다 상당히 향상된다.

연습문제

5.2-1

HIRE-ASSISTANT에서 지원자가 무작위 순서로 온다고 할 때 한 번만 고용할 확률은 얼마인가? 그리고 정확히 n번 고용할 확률은 얼마인가?

5.2-2

HIRE-ASSISTANT에서 지원자가 무작위 순서로 온다고 할 때 정확하게 두 번만 고용할 확률은 얼마인가?

5.2-3

지표 확률 변수를 사용해 주사위를 n번 던진 것의 합에 대한 기댓값을 구하라.

5.2-4

지표 확률 변수를 사용해 ***모자 검사 문제***로 알려진 다음 문제를 풀라. n명의 모든 고객이 음식점에 들어가면서 모자 관리자에게 모자를 넘겨준다. 모자 관리자가 고객에게 무작위로 돌려준다. 그러면 자신의 모자를 제대로 받은 고객 수에 대한 기댓값은 어떻게 되는가?

5.2-5

$A[1..n]$을 n개의 서로 다른 숫자들의 배열이라 하자. 만약 $i < j$이고 $A[i] > A[j]$이면 쌍 (i, j)를 A에 대한 ***역위***라고 한다(역위에 관한 더 많은 내용은 종합문제 2-4를 참고). A의 원소는 항상 $< 1, 2, \ldots, n >$의 균등 임의 순열이다. 지표 확률 변수를 사용해 역위들의 개수에 대한 기댓값을 구하라.

연습문제

5.3-1

Marceau 교수는 보조정리 5.5의 증명에 사용된 루프 불변성에 반대한다. 그는 첫 반복 이전에 루프 불변성 조건을 만족하는지 의문을 품고 있다. 그의 논리는 빈 부분 배열이 어떠한 0 순열들도 포함하지 않는다고 쉽사리 선언할 수 있다는 것이다. 그러므로 빈 부분 배열이 0 순열을 포함할 확률은 0이어야 한다. 이는 첫 반복 이전에 루프 불변성을 깬다. 함수 RANDOMIZE-IN-PLACE를 첫 반복 이전에 관련된 루프 불변성이 비지 않은 부분 배열에 적용할 수 있도록 다시 작성하고 작성한 함수에 대한 것으로 보조정리 5.5의 증명을 바꿔라.

5.3-2

Kelp 교수는 동일한 순열을 제외하고는 모든 순열을 임의로 만들 수 있도록 하는 함수를 만들기로 결심했다. 그는 다음 프로시저를 제안했다. 이 코드는 Kelp 교수가 의도한대로 동작하는가?

```
PERMUTE-WITHOUT-IDENTITY(A)
1  n = A.length
2  for i = 1 to n − 1
3      swap A[i] with A[RANDOM(i + 1, n)]
```

5.3-3

부분 배열 $A[i..n]$에서 임의의 원소와 원소 $A[i]$를 바꾸는 대신에 배열의 다른 임의의 곳으로부터 임의의 원소와 바꾼다고 가정하자.

```
PERMUTE-WITH-ALL(A)
1  n = A.length
2  for i = 1 to n
3      swap A[i] with A[RANDOM(1, n)]
```

이 코드는 균등 임의 순열을 만드는가? 대답에 대한 이유를 설명하라.

5.3-4

Armstrong 교수는 균등 임의 순열을 만들기 위한 다음 함수를 제안했다.

```
PERMUTE-BY-CYCLIC(A)
1  n = A.length
2  B[1..n]를 새 배열이라 하자.
3  offset = RANDOM(1, n)
4  for i = 1 to n
5      dest = i + offset
6      if dest > n
7          dest = dest − n
8      B[dest] = A[i]
9  return B
```

각 원소 $A[i]$가 B에서 임의의 특별한 위치에서 끝날 확률이 $1/n$임을 보여라. 그리고 결과로 얻는 순열이 균등 임의 순열이 아님을 보임으로써 Armstrong 교수가 틀렸음을 보여라.

5.3-5 ★

프로시저 PERMUTE-BY-SORTING에 있는 배열 P에서 모든 원소가 유일하게 되는 확률은 적어도 $1 - 1/n$임을 증명하라.

5.3-6

두 개 이상의 우선순위가 같은 경우를 다루기 위한 알고리즘 PERMUTE-BY-SORTING를 구현하는 방법을 설명하라. 즉, 우선순위가 같은 것이 두 개 이상 있을 때도 이 알고리즘은 균등 임의 순열을 만들어야 한다.

적분을 이용해 상한과 하한을 구한다. 부등식 (A.12)에 의해 다음 식을 얻는다.

$$\int_1^n \frac{1}{x}\,dx \le \sum_{i=1}^{n-1} \frac{1}{i} \le \int_{1-1}^{n-1} \frac{1}{x}\,dx$$

정적분을 계산하면 다음 부등식이 성립한다.

$$\frac{k}{n}(\ln n - \ln k) \le \Pr\{S\} \le \frac{k}{n}(\ln(n-1) - \ln(k-1))$$

이것은 $\Pr\{S\}$에 대해 상당히 정확한 한계를 구해 준다. 이 문제에서 성공 확률을 최대화하는 것이 목적이므로 $\Pr\{S\}$의 하한을 최대화할 수 있는 k를 구하도록 하자(그리고 하한에 대한 식이 상한에 대한 식에 비해 최대화하기에도 더 쉽다). 식 $(k/n)(\ln n - \ln k)$를 k에 관해 미분을 하면 다음 식을 구하게 된다.

$$\frac{1}{n}(\ln n - \ln k - 1)$$

미분값 0이 되는 지점을 찾으면, $\ln k = \ln n - 1 = \ln(n/e)$ 즉, $k = n/e$일 때 확률의 하한이 최대가 됨을 알 수 있다. 그래서 $k = n/e$로 택하는 전략은 $1/e$ 이상의 확률로 가장 뛰어난 지원자를 고용하는 데 성공할 것이다.

연습문제

5.4-1

여러분과 생일이 같은 사람이 있을 확률이 적어도 1/2이 되려면 방에 얼마나 많은 사람이 있어야 하는가, 적어도 두 사람이 11월 17일에 생일일 확률이 1/2보다 크려면 얼마나 많은 사람이 있어야 하는가?

5.4-2

b개의 주머니에 공을 던진다. 던지는 작업은 독립이고 임의의 공이 각 주머니에 들어갈 확률은 모두 같다. 적어도 하나의 주머니에 두 개의 공이 들어가도록 하기 위한 공 던지기 횟수의 기댓값은 얼마인가?

5.4-3 ★

생일에 관한 역설을 분석하는데, 생일이 서로 독립인 것이 중요한가? 아니면 쌍마다 독립인 것으로 충분한가? 대답에 합당한 이유를 제시하라.

(연습문제 6.1-2 참고). 힙에 대한 기본 연산의 수행시간은 오래 살려도 트리의 높이에 비례하는 정도기 때문에 $O(\lg n)$이다. 이 장에서는 몇 가지 기본 프로시저를 제시하고 정렬 알고리즘과 우선순위 큐 자료구조에서 이것이 사용되는 방법을 보인다.

- MAX-HEAPIFY 프로시저는 최대 힙 특성을 유지하는 데 핵심 역할을 한다. 이 프로시저는 $O(\lg n)$ 시간에 실행된다.
- BUILD-MAX-HEAP 프로시저는 정렬되지 않은 입력 배열로부터 최대 힙을 만든다. 이 프로시저는 선형 시간에 실행된다.
- HEAPSORT 프로시저는 배열을 내부 정렬한다. 이 프로시저는 $O(n \lg n)$ 시간에 실행된다.
- MAX-HEAP-INSERT, HEAP-EXTRACT-MAX, HEAP-INCREASE-KEY, HEAP-MAXIMUM 프로시저는 힙 자료구조가 우선순위 큐로 쓰일 수 있게 해준다. 이 프로시저들은 $O(\lg n)$ 시간에 실행된다.

연습문제

6.1-1

높이가 h인 힙의 최대, 최소 원소 수는 각각 얼마인가?

6.1-2

원소 수가 n인 힙의 높이는 $\lfloor \lg n \rfloor$임을 보여라.

6.1-3

최대 힙의 모든 서브 트리에서 서브 트리의 루트가 그 서브 트리에서 최댓값을 가짐을 보여라.

마스터 정리(정리 4.1)의 두 번째 경우를 적용하면 이 관계식의 해는 $T(n) = O(\lg n)$이다. 그리고 높이가 h인 노드에서 MAX-HEAPIFY를 실행할 때 $O(h)$이라고 서술하기도 한다.

연습문제

6.2-1

그림 6.2를 참고해 배열 $A = \langle 27, 17, 3, 16, 13, 10, 1, 5, 7, 12, 4, 8, 9, 0\rangle$에 대한 MAX-HEAPIFY$(A, 3)$의 동작을 보여라.

6.2-2

MAX-HEAPIFY 프로시저를 참고해 최소 힙을 위한 프로시저 MIN-HEAPIFY(A, i)의 의사코드를 작성하라. MAX-HEAPIFY와 비교할 때 MIN-HEAPIFY의 수행시간은 얼마나 되는가?

6.2-3

원소 $A[i]$가 자식들보다 클 때 MAX-HEAPIFY(A, i)를 호출한 결과는 얼마인가?

6.2-4

$i > A.heap\text{-}size/2$인 때 MAX-HEAPIFY(A, i)를 호출한 결과는 얼마인가?

6.2-5

MAX-HEAPIFY의 코드는 상수 인자를 생각해보면 매우 효율적이다. 그러나 컴파일러에 따라 10행의 재귀 호출을 비효율적인 코드로 만들 수 있다. 재귀 호출 대신 반복 제어 구조(루프)를 사용해 효율적인 MAX-HEAPIFY를 작성하라.

6.2-6

힙의 크기가 n일 때 MAX-HEAPIFY의 최악의 경우 수행시간은 $\Omega(\lg n)$임을 보여라 (*힌트:* 힙의 노드 수가 n개일 때, 루트에서 리프로 내려가는 경로의 모든 노드에서 MAX-HEAPIFY가 재귀 호출되도록 각 노드에 값을 주어라).

6.1-4

최대 힙에서 모든 원소가 서로 다를 때 가장 작은 원소는 어디에 존재하겠는가?

6.1-5

정렬된 배열은 최소 힙인가?

6.1-6

수열 (23, 17, 14, 6, 13, 10, 1, 5, 7, 12)는 최대 힙인가?

6.1-7

원소 수가 n개인 힙을 배열로 나타낼 때 $\lfloor n/2 \rfloor + 1, \lfloor n/2 \rfloor + 2, \ldots, n$번째 노드가 리프임을 보여라.

마지막의 합계는 식 (A.8)에 $x = 1/2$을 대입하여 구할 수 있다.

$$\sum_{h=0}^{\infty} \frac{h}{2^h} = \frac{1/2}{(1-1/2)^2}$$
$$= 2$$

따라서 BUILD-MAX-HEAP의 수행시간 상한은 다음과 같다.

$$O\left(n \sum_{h=0}^{\lfloor \lg n \rfloor} \frac{h}{2^h}\right) = O\left(n \sum_{h=0}^{\infty} \frac{h}{2^h}\right)$$
$$= O(n)$$

그러므로 정렬되지 않은 배열에서 최대 힙을 만드는 것은 선형 시간에 가능하다.

BUILD-MAX-HEAP에서 3행의 MAX-HEAPIFY를 MIN-HEAPIFY(연습문제 6.2-2 참고)로 바꾼 BUILD-MIN-HEAP 프로시저를 사용해 최소 힙을 만들 수 있다. BUILD-MIN-HEAP 프로시저는 정렬되지 않은 배열로부터 선형 시간에 최소 힙을 만들어 낸다.

연습문제

6.3-1

그림 6.3을 참고해 배열 $A = \langle 5, 3, 17, 10, 84, 19, 6, 22, 9 \rangle$에 대한 BUILD-MAX-HEAP의 동작을 보여라.

6.3-2

BUILD-MAX-HEAP의 2행을 보자. 인덱스 i를 1부터 $\lfloor A.length/2 \rfloor$까지 증가시키지 않고 $\lfloor A.length/2 \rfloor$부터 1까지 감소시키는 이유는 무엇인가?

6.3-3

원소 수가 n개인 어떤 힙에서도 높이 h인 노드는 최대 $\lceil n/2^{h+1} \rceil$개임을 보여라.

```
HEAPSORT(A)
1  BUILD-MAX-HEAP(A)
2  for i = A.length downto 2
3      exchange A[1] with A[i]
4      A.heap-size = A.heap-size - 1
5      MAX-HEAPIFY(A, 1)
```

그림 6.4는 최대 힙이 초기에 만들어진 다음 힙 정렬 동작의 예를 보여준다. 2–5행에 있는 **for** 루프의 반복은 항상 최대 힙으로 시작한다.

BUILD-MAX-HEAP이 $O(n)$ 시간 걸리고 MAX-HEAPIFY를 $n-1$번 실행하는 것이 각각 $O(\lg n)$ 시간이 걸리므로 HEAPSORT의 수행시간은 $O(n \lg n)$이다.

연습문제

6.4-1

그림 6.4를 참고해 배열 $A = \langle 5, 13, 2, 25, 7, 17, 20, 8, 4 \rangle$에 대한 HEAPSORT의 동작을 그려라.

6.4-2

다음 루프 불변성을 사용해 HEAPSORT가 정확히 동작함을 증명하라.

2–5행에 있는 **for** 루프에서 반복을 시작할 때마다 부분 배열 $A[1..i]$는 $A[1..n]$에서 작은 쪽 원소 i개로 구성된 최대 힙이고, 부분 배열 $A[i+1..n]$은 $A[1..n]$에서 큰 쪽 원소 $n-i$개가 정렬되어 있는 배열이다.

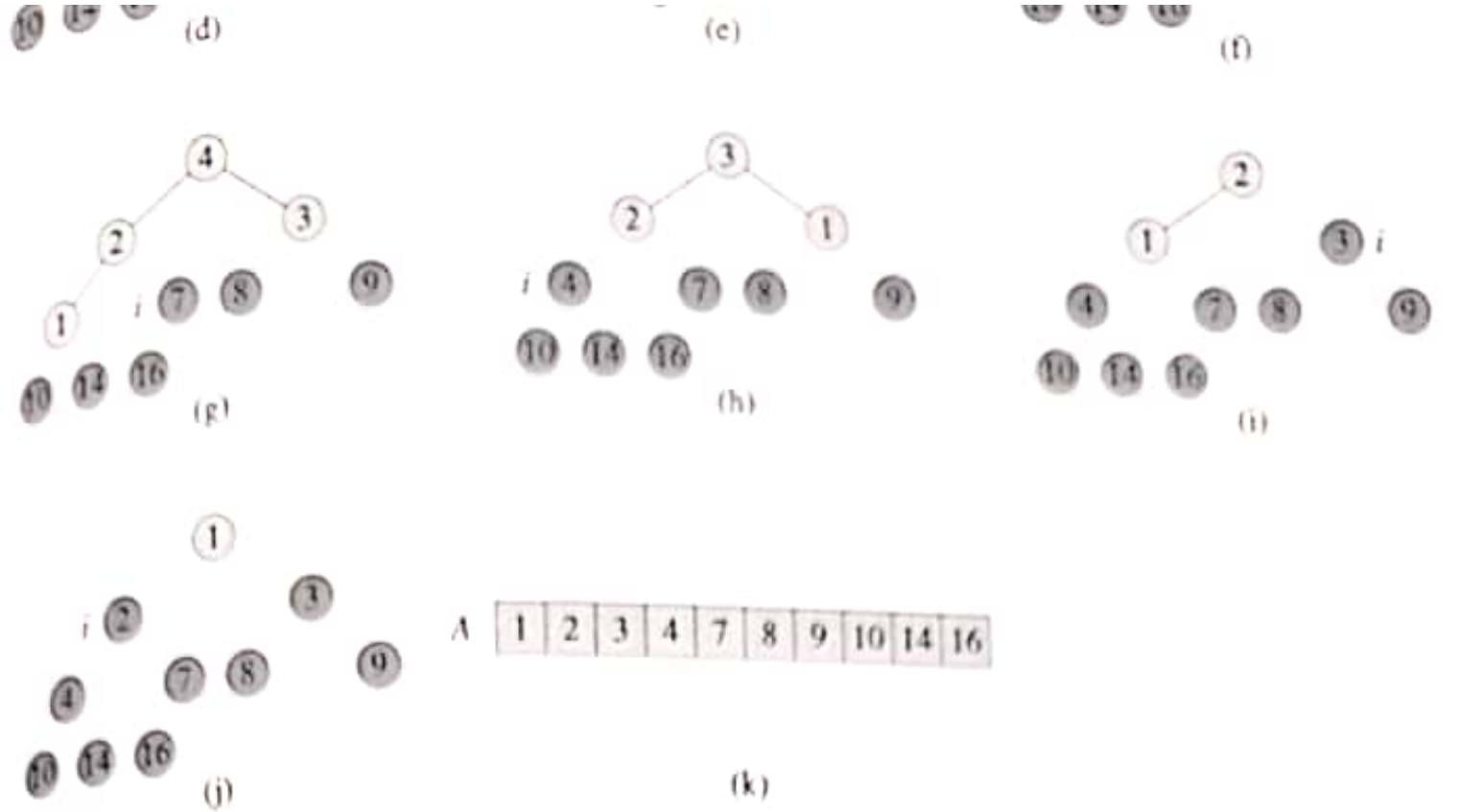

그림 6.4 HEAPSORT의 동작. (a) BUILD-MAX-HEAP을 이용해 만들어진 직후의 최대 힙 자료구조. (b)–(j) 5행을 반복할 때마다 MAX-HEAPIFY를 호출한 직후의 상태. 각 시점에서 i의 값을 보여준다. 연한 회색 노드만 힙에 남는다. (k) 결과로 나타나는 정렬된 배열 A.

6.4-3

오름차순으로 정렬된 n개짜리 배열 A를 힙 정렬하는 데 걸리는 시간은 얼마나 되는가? 내림차순인 경우에는 어떠한가?

6.4-4

힙 정렬의 최악의 경우 수행시간은 $\Omega(n \lg n)$임을 보여라.

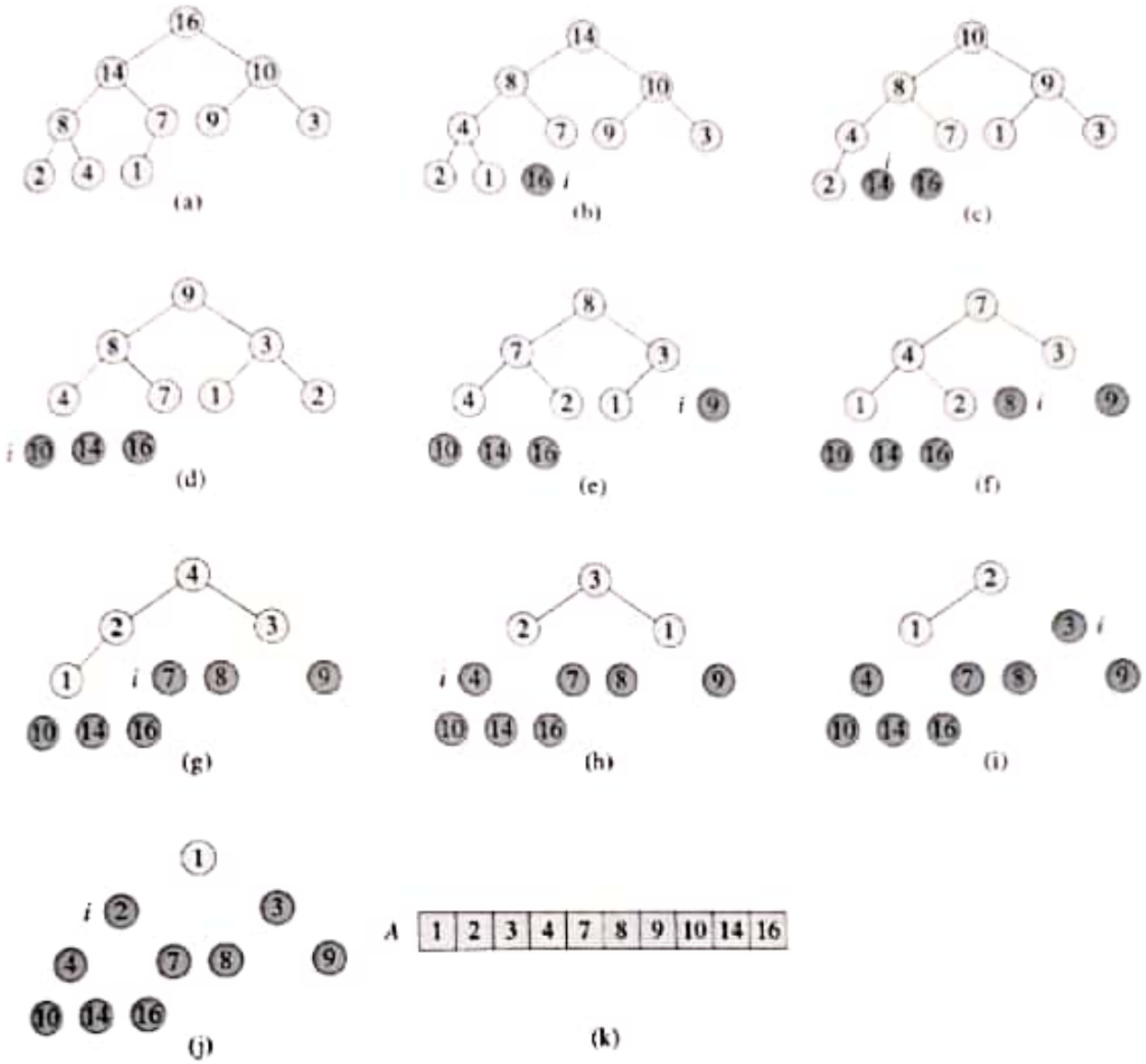

그림 6.4 HEAPSORT의 동작. (a) BUILD-MAX-HEAP을 이용해 만들어진 직후의 최대 힙 자료구조. (b)–(j) 5행을 반복할 때마다 MAX-HEAPIFY를 호출한 직후의 상태. 각 시점에서 i의 값을 보여준다. 연한 회색 노드만 힙에 남는다. (k) 결과로 나타나는 정렬된 배열 A.

6.4-3

오름차순으로 정렬된 n개짜리 배열 A를 힙 정렬하는 데 걸리는 시간은 얼마나 되는가? 내림차순인 경우에는 어떠한가?

6.4-4

힙 정렬의 최악의 경우 수행시간은 $\Omega(n \lg n)$임을 보이라.

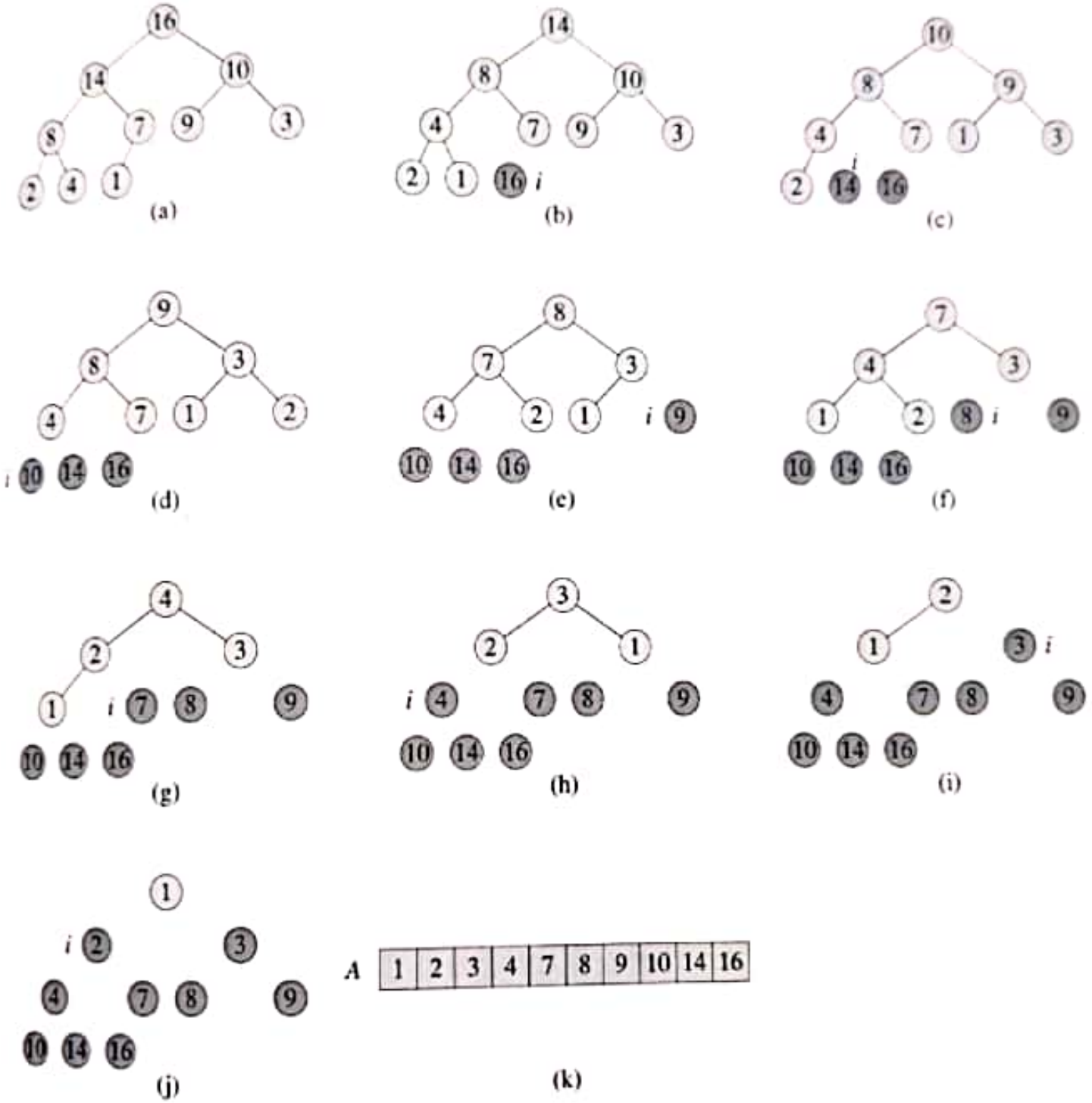

그림 6.4 HEAPSORT의 동작. (a) BUILD-MAX-HEAP을 이용해 만들어진 직후의 최대 힙 자료구조. (b)–(j) 5행을 반복할 때마다 MAX-HEAPIFY를 호출한 직후의 상태. 각 시점에서 i의 값을 보여준다. 연한 회색 노드만 힙에 남는다. (k) 결과로 나타나는 정렬된 배열 A.

```
1  A.heap-size = A.heap-size + 1
2  A[A.heap-size] = −∞
3  HEAP-INCREASE-KEY(A, A.heap-size, key)
```

MAX-HEAP-INSERT의 수행시간은 힙의 크기가 n일 때 $O(\lg n)$이다.

요약하면, 힙은 크기가 n인 집합에 대한 어떤 우선순위 큐 연산이라도 $O(\lg n)$ 시간에 해낼 수 있다.

연습문제

6.5-1

힙 $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1\rangle$에 대해 HEAP-EXTRACT-MAX의 동작을 보여라.

6.5-2

힙 $A = \langle 15, 13, 9, 5, 12, 8, 7, 4, 0, 6, 2, 1\rangle$에 대해 MAX-HEAP-INSERT(A,10)의 동작을 보여라.

6.5-3

최소 힙에서 최소 우선순위 큐를 구현하기 위한 프로시저 HEAP-MINIMUM, HEAP-EXTRACT-MIN, HEAP-DECREASE-KEY, MIN-HEAP-INSERT의 의사코드를 작성하라.

6.5-4

키값을 원하는 값으로 증가시키고자 할 때 MAX-HEAP-INSERT의 2행에서 들어오는 노드의 키값을 굳이 $-\infty$로 해 놓는 이유는 무엇인가?

6.5-5

다음 루프 불변을 사용해 HEAP-INCREASE-KEY가 정확히 동작함을 증명하라.

4-6행에 있는 **while** 루프를 반복하기 시작할 때마다 배열 $A[1..A.heap\text{-}size]$는 최대 힙 특성을 만족한다. 다만 만족하지 않는 예외가 하나 있을 수 있는데, 바로 $A[i]$가 $A[\text{PARENT}(i)]$보다 큰 경우다.

HEAP-INCREASE-KEY를 호출 할 때 부분 배열 $A[1..A.heap\text{-}size]$가 최대 힙 특성을 만족한다고 가정해도 된다.

6.5-6

보통 HEAP-INCREASE-KEY의 5행의 교환 과정을 수행하기 위해 3개의 할당문이 필요하다. INSERTION-SORT의 안쪽 루프의 아이디어를 활용해 1개의 할당문으로 줄일 수 있음을 보여라.

6.5-7

우선순위 큐로 어떻게 FIFO 큐를 구현할 수 있는지 보여라. 그리고 우선순위 큐로 어떻게 스택을 구현할 수 있는지 보여라(큐와 스택은 10.1에서 정의한다).

6.5-8

HEAP-DELETE(A, i) 연산은 힙 A에서 노드 i를 삭제한다. 최대 힙에서 원소 수가 n개일 때 $O(\lg n)$ 시간에 동작하는 HEAP-DELETE를 구현하라.

6.5-9

정렬된 배열 k개를 정렬된 배열 한 개로 합치는 $O(n \lg k)$ 알고리즘을 제시하라. 여기서 n은 모든 입력 배열에 있는 원소 개수의 총합이다(*힌트*: k-원 병합을 할 때 최소 힙을 이용하라).

4–6행에 있는 **while** 루프를 반복하기 시작할 때마다 배열 $A[1..A.heap\text{-}size]$는 최대 힙 특성을 만족한다. 다만 만족하지 않는 예외가 하나 있을 수 있는데, 바로 $A[i]$가 $A[\text{PARENT}(i)]$보다 큰 경우다.

HEAP-INCREASE-KEY를 호출 할 때 부분 배열 $A[1..A.heap\text{-}size]$가 최대 힙 특성을 만족한다고 가정해도 된다.

6.5-6

보통 HEAP-INCREASE-KEY의 5행의 교환 과정을 수행하기 위해 3개의 할당문이 필요하다. INSERTION-SORT의 안쪽 루프의 아이디어를 활용해 1개의 할당문으로 줄일 수 있음을 보여라.

6.5-7

우선순위 큐로 어떻게 FIFO 큐를 구현할 수 있는지 보여라. 그리고 우선순위 큐로 어떻게 스택을 구현할 수 있는지 보여라(큐와 스택은 10.1에서 정의한다).

6.5-8

HEAP-DELETE(A, i) 연산은 힙 A에서 노드 i를 삭제한다. 최대 힙에서 원소 수가 n개일 때 $O(\lg n)$ 시간에 동작하는 HEAP-DELETE를 구현하라.

6.5-9

정렬된 배열 k개를 정렬된 배열 한 개로 합치는 $O(n \lg k)$ 알고리즘을 제시하라. 여기서 n은 모든 입력 배열에 있는 원소 개수의 총합이다(힌트: k-원 병합을 할 때 최소 힙을 이용하라).

[illegible] 간단히 하기 위해 다음과 같이 가정해보자. 트리의 각 레벨에는 좋은 분할과 나쁜 분할이 번갈아가며 나타나고, 좋은 분할은 최적의 분할이고 나쁜 분할은 최악의 분할이라 하자. 그림 7.5(a)는 재귀 트리에 있는 연속된 두 레벨에서 일어나는 분할을 보여준다. 트리의 루트에서 분할 비용은 n이고, 생성되는 부분 배열의 크기는 각각 $n-1$과 0이다. 즉, 최악의 경우다. 다음 레벨에서는 크기가 $n-1$인 부분 배열이 각각 $(n-1)/2-1$과 $(n-1)/2$인 두 부분 배열로 분할된다. 즉 최적인 경우다. 부분 배열의 크기가 0일 때 한계 조건의 비용은 1이라고 하자.

나쁜 분할 이후에 좋은 분할이 이어지는 조합은 크기가 각각 0, $(n-1)/2-1$, $(n-1)/2$인 세 부분 배열을 만들고, 비용은 $\Theta(n)+\Theta(n-1)=\Theta(n)$만큼 든다. 이 경우는 점근적으로는 그림 7.5(b)와 같이 크기가 $(n-1)/2$인 부분 배열 두 개를 만들면서 비용이 $\Theta(n)$만큼 드는 아주 균형잡힌 분할에 비해 나쁘지 않다. 직관적으로 나쁜 분할의 비용 $\Theta(n-1)$은 좋은 분할의 비용 $\Theta(n)$에 흡수될 수 있으므로 결과적으로 좋은 분할이 된다. 따라서 좋은 분할과 나쁜 분할이 번갈아가면서 발생할 때 퀵 정렬의 수행시간은 여전히 $O(n \lg n)$으로 좋은 분할만 나타나는 경우의 수행시간과 비슷하다. 다만 O-표기법에 숨겨진 상수는 조금 더 커진다. 7.4.2절에서 랜덤화된 퀵 정렬 수행시간의 평균값에 대해 엄밀한 분석을 제시한다.

연습문제

7.2-1

이 절의 시작 부분에서 주장했듯이 재귀 관계식 $T(n) = T(n-1) + \Theta(n)$의 해가 $T(n) = \Theta(n^2)$임을 치환법을 이용해 보여라.

7.2-2

배열 A에서 모든 원소의 값이 동일할 때 QUICKSORT의 수행시간은 어떻게 되는가?

7.2-3

배열 A가 내림차순으로 정렬되어 있고 원소가 서로 다를 때 QUICKSORT의 수행시간이 $\Theta(n^2)$임을 보여라.

7.2-4

은행은 주로 계좌의 거래 내역을 거래 시간 순으로 기록하지만 많은 사람이 수표가 번호순으로 되어 있는 고객 내역서를 받고 싶어한다. 사람들은 보통 수표를 수표 번호순으로 사용하고 상인들은 비교적 신속하게 이를 현금화한다.[1] 따라서 거래 시간 순서를 수표 번호 순서로 바꾸는 문제는 거의 정렬된 입력을 정렬하는 문제다. INSERTION-SORT가 QUICKSORT보다 이 문제를 잘 풀지를 논하라.

7.2-5

퀵 정렬이 모든 단계에서 $1-\alpha$대 α 비율로 분할한다고 하자. 여기서 $0 < \alpha \leq 1/2$이고 α는 상수다. 재귀 트리에서 리프의 최소 깊이는 근사값으로 $-\lg n / \lg \alpha$이고, 최대 깊이는 근사값으로 $-\lg n / \lg(1-\alpha)$임을 보여라(반올림은 고려하지 않아도 된다).

7.2-6 ★

$0 < \alpha \leq 1/2$인 모든 상수 α와 임의의 입력 배열에 대해 PARTITION이 $1-\alpha$ 대 α보다 균등한 분할을 할 확률은 근사값으로 $1-2\alpha$임을 증명하라.

7.2-2

배열 A에서 모든 원소의 값이 동일할 때 QUICKSORT의 수행시간은 어떻게 되는가?

7.2-3

배열 A가 내림차순으로 정렬되어 있고 원소가 서로 다를 때 QUICKSORT의 수행시간이 $\Theta(n^2)$임을 보여라.

7.2-4

은행은 주로 계좌의 거래 내역을 거래 시간 순으로 기록하지만 많은 사람이 수표가 번호순으로 되어 있는 고객 내역서를 받고 싶어한다. 사람들은 보통 수표를 수표 번호순으로 사용하고 상인들은 비교적 신속하게 이를 현금화한다.[1] 따라서 거래 시간 순서를 수표 번호 순서로 바꾸는 문제는 거의 정렬된 입력을 정렬하는 문제다. INSERTION-SORT가 QUICKSORT보다 이 문제를 잘 풀지를 논하라.

7.2-5

퀵 정렬이 모든 단계에서 $1-\alpha$대 α 비율로 분할한다고 하자. 여기서 $0 < \alpha \leq 1/2$이고 α는 상수다. 재귀 트리에서 리프의 최소 깊이는 근사값으로 $-\lg n / \lg \alpha$이고, 최대 깊이는 근사값으로 $-\lg n / \lg(1-\alpha)$임을 보여라(반올림은 고려하지 않아도 된다).

7.2-6 ★

$0 < \alpha \leq 1/2$인 모든 상수 α와 임의의 입력 배열에 대해 PARTITION이 $1-\alpha$ 대 α보다 균등한 분할을 할 확률은 근사값으로 $1-2\alpha$임을 증명하라.

3 return PARTITION(A, p, r)

새로운 퀵 정렬은 PARTITION을 호출하는 대신 RANDOMIZED-PARTITION을 호출한다.

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      RANDOMIZED-QUICKSORT(A, p, q − 1)
4      RANDOMIZED-QUICKSORT(A, q + 1, r)
```

이 알고리즘은 다음 절에서 분석한다.

연습문제

7.3-1

왜 랜덤화된 알고리즘의 평균인 경우만 분석하고 최악의 경우는 분석하지 않는가?

7.3-2

프로시저 RANDOMIZED-QUICKSORT를 실행하는 동안 최악의 경우 난수 생성기 RANDOM을 몇 번이나 호출하는가? 최적인 경우에는 어떠한가? Θ-표기법으로 답하라.

$$= O(n \lg n) \tag{7.4}$$

따라서 RANDOMIZED-PARTITION을 이용하면 퀵 정렬의 평균 수행시간은 개체들의 값이 서로 다를 때 $O(n \lg n)$이다.

연습문제

7.4-1

다음 재귀 관계식에서 $T(n) = \Omega(n^2)$임을 보여라.

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n - q - 1)) + \Theta(n)$$

7.4-2

퀵 정렬의 수행시간이 최적인 경우 $\Omega(n \lg n)$임을 보여라.

7.4-3

$q^2 + (n - q - 1)^2$은 $q = 0, 1, \ldots, n - 1$에서 $q = 0$ 또는 $q = n - 1$일 때 최대가 됨을 보여라.

7.4-4

RANDOMIZED-QUICKSORT의 평균 수행시간이 $\Omega(n \lg n)$임을 보여라.

7.4-5

실제 문제에서 입력이 "거의" 정렬되어 있는 경우 삽입 정렬의 빠른 수행시간을 이용해 퀵 정렬의 수행시간을 향상시킬 수 있다. 원소가 k개보다 작은 부분 배열에서 퀵 정렬이 호출되면 부분 배열을 정렬하지 않고 간단히 리턴되게 한다. 최상위 레벨에서 퀵 정렬 재귀 호출이 끝난 다음 정렬을 마치기 위해 전체 배열에 대해서 삽입 정렬을

실행한다. 이 알고리즘이 평균 $O(nk + n\lg(n/k))$ 시간에 동작함을 보여라. k를 어떻게 정하는 것이 좋겠는가? 이론적, 실제적 측면을 모두 고려해 답하라.

7.4-6 ★

배열 A에서 원소 세 개를 임의로 선택하여 이 중 중앙값(세 원소 중 가운데 것)으로 분할하도록 PARTITION 프로시저를 수정하라. 최악의 경우 α대 $(1-\alpha)$ 분할을 할 확률을 $0 < \alpha < 1$ 범위에서 α의 함수로 근사 계산하라.

종합문제

7-1 Hoare 분할의 정확성

이 장의 PARTITION은 원래의 분할 알고리즘이 아니다. 다음은 C. A. R. Hoare가 고안한 원래의 분할 알고리즘이다.

HOARE-PARTITION(A, p, r)

```
1  x = A[p]
2  i = p − 1
3  j = r + 1
4  while TRUE
5      repeat
6          j = j − 1
7      until A[j] ≤ x
8      repeat
```

정렬을 사용하고, 이 경우 수행시간은 $\Theta(n)$이다.

계수 정렬은 비교 정렬이 아니므로 8.1절에서 증명한 하한 $\Omega(n \lg n)$의 제약을 받지 않는다. 실제로 코드 어디에서도 입력 원소끼리 비교하는 부분이 없다. 대신 계수 정렬은 원소의 실제 값을 배열의 인덱스로 사용한다. $\Omega(n \lg n)$이라는 정렬의 하한은 비교 정렬이 아니면 적용되지 않는다.

계수 정렬의 중요한 특징은 ***안정성***(stable)을 가진다는 점이다. 이는 출력 배열에서 값이 같은 숫자가 입력 배열에 있던 것과 같은 순서로 나타나는 것을 뜻한다. 즉, 두 숫자가 같을 때는 입력 배열에서 먼저 나타나는 것이 출력 배열에서도 먼저 나타난다. 보통 안정성이라는 특성이 중요할 때는 정렬되는 원소에 부속 데이터가 붙어 다닐 때뿐이다. 계수 정렬의 안정성은 다른 이유로도 중요한데, 계수 정렬이 종종 기수 정렬의 서브 루틴으로 쓰이기 때문이다. 다음 절에서 기수 정렬이 정확하게 동작하기 위해 계수 정렬의 안정성이 필수임을 보게 될 것이다.

연습문제

8.2-1

그림 8.2를 참고해 배열 $A = \langle 6, 0, 2, 0, 1, 3, 4, 6, 1, 3, 2 \rangle$에 대한 COUNTING-SORT의 동작을 설명하라.

8.2-2

COUNTING-SORT가 안정된 정렬임을 증명하라.

8.2-3

COUNTING-SORT 프로시저의 10행 루프 서두를 다음과 같이 수정했다고 하자. 이 알고리즘이 잘 동작함을 보여라. 수정한 알고리즘은 안정된 정렬인가?

```
10  for j = 1 to A.length
```

8.2-4

0과 k 사이의 정수 n개가 주어질 때, 입력을 전처리한 후 $[a..b]$ 범위에 들어가는 수가 몇 개나 있는지 질문하면 $O(1)$ 시간에 응답하는 알고리즘을 작성하라. 전처리 시간은 $\Theta(n+k)$만 사용해야 한다.

8.3 기수 정렬

기수 정렬(radix sort)은 이제는 컴퓨터 박물관에서나 볼 수 있는 카드 정렬 기계에서 사용하는 알고리즘이다. 카드는 80열로 구성되고, 각 열에는 12개의 자리가 있어 그 중 한 군데에 구멍을 뚫을 수 있다. 정렬기는 기계적으로 "프로그래밍되어" 있어서 각 카드에서 열을 살펴 보고 구멍이 뚫린 자리에 따라 12개의 통 중 하나로 카드를 분배한다. 그리고 나서 작업자는 첫째 구멍이 뚫린 카드가 둘째 구멍이 뚫린 카드 위에 오고 하는 식으로 카드를 통별로 모을 수 있다.

10진수라면 각 열은 10개 장소만을 사용한다(다른 두 장소는 숫자가 아닌 문자를 인코딩하기 위해서 사용한다). d자리 숫자는 d개의 열을 점유한다. 카드 정렬기는 한 번에 한 열만 볼 수 있으므로 d자리 숫자 카드 n개를 정렬하는 문제에는 정렬 알고리즘이 필요하다.

직관적으로 *가장 높은 자리*에 따라 숫자를 정렬하고, 결과가 담긴 통들 각각을 재귀적으로 정렬한 다음, 순서대로 모으려고 생각할 수 있다. 그러나 각 통을 정렬할 때마다 통 10개 중 9개는 어딘가에 저장되어 있어야 하므로 이 방법은 중간 과정에서 유지해야 하는 임시 카드 묶음이 많이 생긴다(연습문제 8.3-5참고).

이로써 식 (8.2)가 증명된다.

이 기댓값을 식 (8.1)에 대입하면 버킷 정렬의 평균 수행시간이 $\Theta(n)+n\cdot O(2-1/n)=\Theta(n)$라고 결론지을 수 있다.

입력이 균일한 분포가 아니더라도 버킷 정렬은 여전히 선형 시간에 실행될 수 있다. 버킷에 있는 원소 수에 대한 제곱의 합계가 총 원소 수에 대해 선형인 특성을 입력이 가지고 있는 한 식 (8.1)에 따라서 버킷 정렬은 선형 시간에 실행될 것이다.

연습문제

8.4-1

그림 8.4를 참고해 배열 $A=(.79, .13, .16, .64, .39, .20, .89, .53, .71, .42)$에 대한 BUCKET-SORT의 동작을 그려라.

8.4-2

버킷 정렬의 최악의 경우 수행시간이 $\Theta(n^2)$인 이유를 설명하라. 이 알고리즘을 수정해 평균 수행시간은 선형으로 유지시키고 최악의 경우 수행시간을 $O(n\lg n)$이 되게 하는 간단한 방법은 무엇인가?

8.4-3

확률 변수 X를 제대로 된 동전을 두 번 던질 때 앞면이 나오는 횟수라 하자. $\mathrm{E}[X^2]$은 얼마인가, 그리고 $\mathrm{E}^2[X]$는 얼마인가?

8.4-4 ★

단위 원에서 $0<x_i^2+y_i^2\le 1$인 점 $p_i=(x_i,y_i)$ n개가 주어진다($i=1,2,\ldots,n$). 그리고 이 점이 균일하게 분포되어 있다고 하자. 다시 말해 원의 어느 부분에서든 점을 찾을 확률은 그 부분의 면적에 비례한다. 점 n개를 원점으로부터 거리 $d_i=\sqrt{x_i^2+y_i^2}$를 기준으로 정렬하는 평균 수행시간 $\Theta(n)$ 알고리즘을 설계하라(힌트: BUCKET-SORT에서 버킷 크기가 단위 원에서 점의 균등 분포를 반영하도록 설계하라).

8.4-5 ★

확률 변수 X에 대한 **확률 분포 함수** $P(x)$를 $P(x)=\Pr\{X\le x\}$로 정의한다. $O(1)$ 시간에 계산할 수 있는 연속 확률 분포 함수 P에서 확률 변수 $X_1,\ldots,X_n$의 목록을 뽑아내고 이 숫자를 평균적으로 선형 시간에 정렬하는 알고리즘을 제시하라.

연습문제

8.3-1

그림 8.3을 참고해 다음 영어 단어 목록을 RADIX-SORT로 정렬하는 동작을 설명하라. <COW, DOG, SEA,RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX >

8.3-2

삽입 정렬, 병합 정렬, 힙 정렬, 퀵 정렬 중에서 안정된 정렬인 것들은 무엇인가? 어떠한 정렬 알고리즘도 안정된 정렬로 만드는 간단한 방법을 제시하라. 그리고 이 방법을 사용할 경우 시간과 공간이 추가로 얼마나 드는가?

8.3-3

귀납법을 이용해 기수 정렬이 잘 동작함을 증명하라. 증명의 어느 부분에서 중간 정렬이 안정된 정렬이라는 가정이 필요한가?

8.3-4

0에서 $n^3 - 1$ 범위에 있는 정수 n개를 $O(n)$ 시간에 정렬하는 방법을 보여라.

8.3-5 ★

이 절 처음의 카드 정렬 알고리즘에서 d자리 10진수를 정렬하는 데 최악의 경우 정렬 과정이 정확하게 몇 번 필요한가? 그리고 최악의 경우 작업자는 카드를 몇 묶음이나 유지하고 있어야 하는가?

8.4 버킷 정렬

버킷 정렬(bucket sort)은 입력이 균등 분포(uniform distribution)를 하고 있음을 가정하며, 평균 $O(n)$의 수행시간을 가진다. 계수 정렬처럼 버킷 정렬은 입력에 관해 몇

연습문제

15.1-1

식 (15.4)가 식 (15.3)과 초기 조건 $T(0) = 1$로부터 얻어짐을 보여라.

15.1-2

다음의 "그리디(greedy)" 전략이 막대를 자르는 데 항상 최적의 방법을 결정하지 못함을 반례(counterexample)를 통해 보여라. 길이가 i인 막대의 밀도를 p_i/i로, 즉 인치당 값으로 정의하라. 길이가 n인 막대에 대한 그리디 전략은 $1 \le i \le n$에 대해 최대 밀도를 갖는 길이가 i인 첫 번째 조각을 잘라낸다. 그런 다음 길이가 $n - i$인 나머지 조각에 그리디 전략을 계속 적용한다.

15.1-3

막대 자르기 문제를 변형하여 각 막대에 대한 가격 p_i 외의 각 자르기가 특정한 비용 c를 발생한다고 생각하자. 하나의 해에 대한 수익은 조각들의 가격 총 합에서 자르는 비용을 빼 주어야 한다. 이렇게 변형된 문제를 풀기 위한 동적 프로그래밍 알고리즘을 보여라.

15.1-4

값뿐만 아니라 실제 해도 함께 리턴하도록 MEMOIZED-CUT-ROD를 수정하라.

15.1-5

피보나치 수는 재귀식 (3.22)에 의해 정의된다. n번째 피보나치 수를 계산하기 위한 $O(n)$ 시간의 동적 프로그래밍 알고리즘을 만들라. 그리고 부분 문제 그래프를 그려라. 그래프에서 정점과 간선의 수는 몇 개인가?

```
3  else ( 을 출력한다.
4       PRINT-OPTIMAL-PARENS(s, i, s[i, j])
5       PRINT-OPTIMAL-PARENS(s, s[i, j] + 1, j)
6       ")"를 출력한다.
```

그림 15.5의 예에서 PRINT-OPTIMAL-PARENS$(s, 1, 6)$은 $((A_1(A_2A_3))((A_4A_5)A_6))$을 출력한다.

연습문제

15.2-1

행렬 차원에 대한 수열 p가 $(5, 10, 3, 12, 5, 50, 6)$인 행렬 곱셈의 최적 괄호 묶는 방법을 찾아라.

15.2-2

행렬 $\langle A_1, A_2, \ldots, A_n \rangle$과 MATRIX-CHAIN-ORDER로 구한 테이블 s와 인덱스 i 및 j가 주어졌을 때, 실제로 최적의 행렬 곱셈을 수행하는 MATRIX-CHAIN-MULTIPLY$(A, s, 1, n)$ 재귀 알고리즘을 제시하라(최초의 호출은 MATRIX-CHAIN-MULTIPLY$(A, s, 1, n)$이 될 것이다).

15.2-3

대입법을 사용해 점화식 (15.6)에 대한 해가 $\Omega(2^n)$임을 보여라.

15.2-4

체인 길이가 n인 행렬 체인 곱에 대한 부분 문제 그래프를 설명하라. 정점을 몇 개 가지는가, 간선을 몇 개 가지며 어떤 간선을 가지는가?

15.2-5

테이블 엔트리 $m[i, j]$가 MATRIX-CHAIN-ORDER의 호출에서 다른 테이블 엔트리를 계산하는 동안 참조되는 횟수를 $R(i, j)$라 하자. 전체 테이블에 대한 참조의 총 개수가 다음과 같음을 보여라(*힌트:* 식 (A.3)이 유용할 것이다).

연습문제

15.3-1

행렬 곱 문제에서 스칼라 곱의 최소한의 개수를 결정할 때, 가능한 괄호 묶는 경우를 모두 나열해서 최소한의 곱셈의 횟수를 구하는 방법과 RECURSIVE-MATRIX-CHAIN을 이용하는 방법 중에 어떤 것이 더 효율적인지 답하고 설명하라.

15.3-2

원소가 16개인 배열에 대해 2.3.1절의 MERGE-SORT에 대한 재귀 트리를 그려라. 그리고 이런 MERGE-SORT와 같은 좋은 분할정복 알고리즘에 대해 메모하는 방법이 속도를 더 빠르게 하지 못하는 이유를 설명하라.

15.3-3

행렬 곱 문제를 약간 변형해 스칼라 곱셈의 횟수를 최소가 아닌 최대가 되게 하려고 한다. 이 문제는 최적 부분 구조를 가지는가?

15.3-4

앞서 언급했듯이, 동적 프로그래밍에서는 일단 부분 문제를 풀고, 그것 중에서 어떤 것이 최적해에 사용될지를 선택한다. Capulet 교수는 최적해를 구하기 위해 모든 부분 문제를 언제나 다 풀 필요가 없다는 주장을 했다. 그녀는 행렬 곱 문제의 최적해는 부분 문제를 풀기도 *전에* 부분 곱 $A_i A_{i+1} \cdots A_j$를 나누는 행렬 A_k를 $p_{i-1} p_k p_j$를 최소로 만드는 k를 선택함에 의해 언제나 찾을 수 있다고 주장했다. 이런 그리디 방법이 최적해를 구하지 못하는 행렬 곱 문제의 예를 찾아라.

15.3-5

15.1절의 막대 자르기 문제에서 $i = 1, 2, \ldots, n$인 길이가 i인 조각들의 수가 l_i까지로 제한된다고 가정하자. 15.1절에서 설명한 최적 부분 구조 특징이 더 이상 성립하지 않음을 보여라.

15.3-6

어떤 화폐를 또 하나의 다른 화폐로 바꾸기를 원한다고 상상을 하라. 하나의 화폐를 다른 화폐로 직접 교환하는 대신 여러 화폐로의 교환을 통해 최종적으로 원하는 화폐로 교환하는 것이 더 좋을 수도 있다. $1, 2, \ldots, n$의 번호로 표시되는 n개의 다른 화폐로 바꿀 수가 있는데 1번 화폐로 시작해 n번 화폐로 바꾸기를 원한다고 하자. 각 화폐 i와 j에 대해 환율은 r_{ij}로 주어지는데 이는 화폐 i를 d만큼 가지고 있다면 화폐 j로 dr_{ij}만큼 바꿀수 있다는 의미다. 교환의 순서가 정해지면 교환의 개수에 따라서

연습문제

15.3-1

행렬 곱 문제에서 스칼라 곱의 최소한의 개수를 결정할 때, 가능한 괄호 묶는 경우를 모두 나열해서 최소한의 곱셈의 횟수를 구하는 방법과 RECURSIVE-MATRIX-CHAIN을 이용하는 방법 중에 어떤 것이 더 효율적인지 답하고 설명하라.

15.3-2

원소가 16개인 배열에 대해 2.3.1절의 MERGE-SORT에 대한 재귀 트리를 그려라. 그리고 이런 MERGE-SORT와 같은 좋은 분할정복 알고리즘에 대해 메모하는 방법이 속도를 더 빠르게 하지 못하는 이유를 설명하라.

15.3-3

행렬 곱 문제를 약간 변형해 스칼라 곱셈의 횟수를 최소가 아닌 최대가 되게 하려고 한다. 이 문제는 최적 부분 구조를 가지는가?

15.3-4

앞서 언급했듯이, 동적 프로그래밍에서는 일단 부분 문제를 풀고, 그것 중에서 어떤 것이 최적해에 사용될지를 선택한다. Capulet 교수는 최적해를 구하기 위해 모든 부분 문제를 언제나 다 풀 필요가 없다는 주장을 했다. 그녀는 행렬 곱 문제의 최적해는 부분 문제를 풀기도 *전에* 부분 곱 $A_i A_{i+1} \cdots A_j$를 나누는 행렬 A_k를 $p_{i-1} p_k p_j$를 최소로 만드는 k를 선택함에 의해 언제나 찾을 수 있다고 주장했다. 이런 그리디 방법이 최적해를 구하지 못하는 행렬 곱 문제의 예를 찾아라.

15.3-5

를 더 빠르게 하지 못하는 이유를 설명하라.

15.3-3

행렬 곱 문제를 약간 변형해 스칼라 곱셈의 횟수를 최소가 아닌 최대가 되게 하려고 한다. 이 문제는 최적 부분 구조를 가지는가?

15.3-4

앞서 언급했듯이, 동적 프로그래밍에서는 일단 부분 문제를 풀고, 그것 중에서 어떤 것이 최적해에 사용될지를 선택한다. Capulet 교수는 최적해를 구하기 위해 모든 부분 문제를 언제나 다 풀 필요가 없다는 주장을 했다. 그녀는 행렬 곱 문제의 최적해는 부분 문제를 풀기도 전에 부분 곱 $A_i A_{i+1} \cdots A_j$를 나누는 행렬 A_k를 $p_{i-1} p_k p_j$를 최소로 만드는 k를 선택함에 의해 언제나 찾을 수 있다고 주장했다. 이런 그리디 방법이 최적해를 구하지 못하는 행렬 곱 문제의 예를 찾아라.

15.3-5

15.1절의 막대 자르기 문제에서 $i = 1, 2, \ldots, n$인 길이가 i인 조각들의 수가 l_i까지로 제한된다고 가정하자. 15.1절에서 설명한 최적 부분 구조 특징이 더 이상 성립하지 않음을 보여라.

15.3-6

어떤 화폐를 또 하나의 다른 화폐로 바꾸기를 원한다고 상상을 하라. 하나의 화폐를 다른 화폐로 직접 교환하는 대신 여러 화폐로의 교환을 통해 최종적으로 원하는 화폐로 교환하는 것이 더 좋을 수도 있다. $1, 2, \ldots, n$의 번호로 표시되는 n개의 다른 화폐로 바꿀 수가 있는데 1번 화폐로 시작해 n번 화폐로 바꾸기를 원한다고 하자. 각 화폐 i와 j에 대해 환율은 r_{ij}로 주어지는데 이는 화폐 i를 d만큼 가지고 있다면 화폐 j로 dr_{ij}만큼 바꿀수 있다는 의미다. 교환의 순서가 정해지면 교환의 개수에 따라서

행렬 곱 문제에서 스칼라 곱의 최소한의 개수를 결정할 때, 가능한 괄호 묶는 경우를 모두 나열해서 최소한의 곱셈의 횟수를 구하는 방법과 RECURSIVE-MATRIX-CHAIN을 이용하는 방법 중에 어떤 것이 더 효율적인지 답하고 설명하라.

15.3-2

원소가 16개인 배열에 대해 2.3.1절의 MERGE-SORT에 대한 재귀 트리를 그려라. 그리고 이런 MERGE-SORT와 같은 좋은 분할정복 알고리즘에 대해 메모하는 방법이 속도를 더 빠르게 하지 못하는 이유를 설명하라.

15.3-3

행렬 곱 문제를 약간 변형해 스칼라 곱셈의 횟수를 최소가 아닌 최대가 되게 하려고 한다. 이 문제는 최적 부분 구조를 가지는가?

15.3-4

앞서 언급했듯이, 동적 프로그래밍에서는 일단 부분 문제를 풀고, 그것 중에서 어떤 것이 최적해에 사용될지를 선택한다. Capulet 교수는 최적해를 구하기 위해 모든 부분 문제를 언제나 다 풀 필요가 없다는 주장을 했다. 그녀는 행렬 곱 문제의 최적해는 부분 문제를 풀기도 *전에* 부분 곱 $A_i A_{i+1} \cdots A_j$를 나누는 행렬 A_k를 $p_{i-1} p_k p_j$를 최소로 만드는 k를 선택함에 의해 언제나 찾을 수 있다고 주장했다. 이런 그리디 방법이 최적해를 구하지 못하는 행렬 곱 문제의 예를 찾아라.

15.3-5

15.1절의 막대 자르기 문제에서 $i = 1, 2, \ldots, n$인 길이가 i인 조각들의 수가 l_i까지로 제한된다고 가정하자. 15.1절에서 설명한 최적 부분 구조 특징이 더 이상 성립하지 않음을 보여라.

15.3-6

어떤 화폐를 또 하나의 다른 화폐로 바꾸기를 원한다고 상상을 하라. 하나의 화폐를 다른 화폐로 직접 교환하는 대신 여러 화폐로의 교환을 통해 최종적으로 원하는 화폐로 교환하는 것이 더 좋을 수도 있다. $1, 2, \ldots, n$의 번호로 표시되는 n개의 다른 화폐로 바꿀 수가 있는데 1번 화폐로 시작해 n번 화폐로 바꾸기를 원한다고 하자. 각 화폐 i와 j에 대해 환율은 r_{ij}로 주어지는데 이는 화폐 i를 d만큼 가지고 있다면 화폐 j로 dr_{ij}만큼 바꿀수 있다는 의미다. 교환의 순서가 정해지면 교환의 개수에 따라서