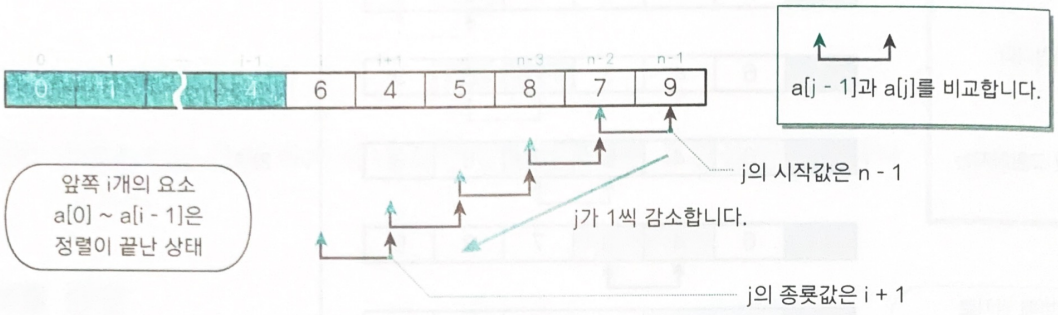
니다. 이때 두 요소(a[j - 1], a[j])의 값을 비교하여 앞쪽이 크면 교환합니다. 그 이후의 비교, 교환 과정은 바로 앞쪽에서 수행해야 하므로 j의 값은 1씩 감소합니다.



[그림 6-5] 버블 정렬의 i번째 패스

각 패스에서 앞쪽 i개의 요소는 정렬이 끝난 상태라고 가정합니다(정렬하지 않은 부분은 a[i] ~ a[n - 1]라고 가정합니다). 따라서 한 번의 패스에서는 j의 값이 i + 1이 될 때까지 비교, 교환을 수행하면 됩니다.

i가 0인 첫 번째 패스는 j값이 1이 될 때까지 반복하고(그림 6-3), i가 1인 두 번째 패스는 j값이 2가 될 때까지 반복(그림 6-4)합니다.

그리고 비교하는 두 요소 중에서 오른쪽 요소의 인덱스는 i + 1이 될 때까지 감소하고 왼쪽 요소의 인덱스는 i가 될 때까지 감소합니다. 서로 한 칸 이상 떨어져 있는 요소를 교환하는 것이 아니라 서로 이웃한 요소에 대해서만 교환하므로 이 정렬 알고리즘은 안정적이라고 할 수 있습니다. 비교 횟수는 첫 번째 패스는 n - 1회, 두 번째 패스는 n - 2회, … 이므로 그 합계는 다음과 같습니다.

$$(n - 1) + (n - 2) + \cdots + 1 = n(n - 1) / 2$$

그러나 실제 요소를 교환하는 횟수는 배열의 요솟값에 더 많이 영향을 받기 때문에 교환 횟수의 평균값은 비교 횟수의 절반인 n(n - 1) / 4회입니다. 또한 swap 함수 안에서 값의 이동이 3회 발생하므로 이동 횟수의 평균은 3n(n - 1) / 4회입니다.

실습 6-1

• 완성 파일 chap06/bubble1.c

```
01  /* 버블 정렬 */
02  #include <stdio.h>
03  #include <stdlib.h>
04  #define swap (type, x, y) do { type t = x; x = y; y = t; } while(0)
05
```