Hands-on Assignment 4-1: FindMaxMin

2271056 정은채

Step 1: Implement Two Types of FindMaxMin Algorithm

First Type: 2n-2 Comparisons
첫 번째 유형의 findMaxMin 알고리즘은 배열을 한 번 순회하면서 최대값과 최소값을 찾습니다. 이를 위해 각 요소를 순회하면서 현재 요소가 최대값보다 크면 최대값을 갱신하고, 현재 요소가 최소값보다 작으면 최소값을 갱신합니다. 이 알고리즘은 총 2(n-1) 번의 비교를 수행하며, 이는 2n - 2 비교로 표현됩니다.

Second Type: 3n/2 - 2 Comparisons
두 번째 유형의 findMaxMin 알고리즘은 분할 정복 방식을 사용합니다. 배열을 두 개의 하위 배열로 나눈 다음, 각각의 하위 배열에서 최대값과 최소값을 찾습니다. 그런 다음 두 하위 배열의 결과를 결합하여 전체 배열의 최대값과 최소값을 구합니다. 이 알고리즘은 총 3n/2 - 2 번의 비교를 수행합니다.

Step 2: Analyze the Time Complexities of Your Implementations₩

First Type (2n-2 Comparisons):
이 알고리즘은 배열을 한 번 순회하며, 각 요소마다 최대값과 최소값을 비교합니다. 따라서 총 비교 횟수는 2(n-1) = 2n - 2입니다.

Time Complexity: **O(n)**

Second Type: 3n/2 - 2 Comparisons
이 알고리즘은 분할 정복 방법을 사용합니다. 각 단계에서 배열을 두 개의 하위 배열로 나누며, 비교 횟수는 다음과 같이 계산됩니다:
두 개의 요소를 가진 세그먼트에서는 1번의 비교가 필요합니다.
더 큰 세그먼트에서는 세그먼트를 나누고 결과를 결합합니다.
총 비교 횟수는 대략 3n/2 - 2가 됩니다.

Time Complexity: **O(n)**

Step 3: Explain Why One Type Performs More Efficiently Than the Other

.

첫 번째 유형의 알고리즘은 구현이 간단하고 직관적이지만, 비교 횟수가 2n - 2로 더 많습니다. 이는 특히 큰 배열에서 비효율적일 수 있습니다. 두 번째 유형의 알고리즘은 분할 정복 전략을 사용하여 비교 횟수를 3n/2 - 2로 줄입니다. 이는 첫 번째 유형에 비해 비교 횟수가 적어 더 효율적입니다. 특히 배열의 크기가 클수록 성능 차이가 두드러집니다.
두 번째 유형의 알고리즘이 더 효율적임을 확인할 수 있습니다.