Assignment 2 – selection

|  |  |
| --- | --- |
| Student ID | 2021270660 |
| Name: | 이지원 |

1. Design your algorithm with Pseudo code for selection in linear time even in the worst case. (4 points)

|  |
| --- |
| SELECT(A, n, k):  input:  A: n개의 원소가 있는 배열  k: 찾고자 하는 원소의 순위 (1 ≤ k ≤ n)  output:  A에서 k번째로 작은 원소  1. n이 작은 경우 (예: n ≤ 5), 배열 A를 정렬한 후 k번째 원소를 반환한다.  2. A를 ⌈n / 5⌉개의 그룹으로 나누고, 각 그룹에 최대 5개의 원소를 넣는다.  3. 각 그룹을 정렬하고, 그룹의 중앙값을 찾는다.  4. 그룹의 중앙값 배열에 대해 재귀적으로 SELECT를 호출하여, 중앙값들의 중앙값(M)을 찾는다.  5. 배열 A를 세 부분으로 나눈다:  - L: M보다 작은 원소들의 배열  - E: M과 같은 원소들의 배열  - G: M보다 큰 원소들의 배열  6. 만약 k가 L의 크기 이하라면:  - SELECT(L, |L|, k)를 재귀적으로 호출한다.  7. 만약 k가 L의 크기보다 크고 L과 E의 크기 합보다 작거나 같다면:  - M을 반환한다. (M이 k번째 원소임)  8. 그렇지 않다면 (즉, k > |L| + |E|):  - SELECT(G, |G|, k - |L| - |E|)를 재귀적으로 호출한다. |

1. Implement your algorithm in Python and explain your algorithm. (Include code comments 코드에 주석을 작성하시오) (4 points)

|  |
| --- |
| 1. *# 배열을 주어진 기준 값(pivot)을 기준으로 분할하는 함수* 2. def partition(arr, low, high, pivot): 3. *# 기준 값(pivot)의 인덱스를 찾음* 4. pivot\_index = arr.index(pivot) 5. *# 기준 값을 배열의 마지막 요소와 교환* 6. arr[pivot\_index], arr[high] = arr[high], arr[pivot\_index] 8. i = low 9. *# 배열을 순회하면서 기준 값보다 작은 값들은 왼쪽으로 이동* 10. for j in range(low, high): 11. if arr[j] < pivot: 12. arr[i], arr[j] = arr[j], arr[i] 13. i += 1 14. *# 기준 값을 올바른 위치에 배치* 15. arr[i], arr[high] = arr[high], arr[i] 16. return i 17. *# 중간값의 중간값을 찾는 함수* 18. def median\_of\_medians(arr, low, high): 19. *# 배열의 크기가 5 이하일 경우, 배열을 정렬한 후 중간값 반환* 20. if high - low < 5: 21. return sorted(arr[low:high+1])[(high - low) // 2] 23. *# 배열을 5개의 원소로 이루어진 그룹으로 나누고 각 그룹의 중간값을 찾음* 24. medians = [] 25. i = low 26. while i <= high: 27. group = arr[i:min(i + 5, high + 1)] 28. medians.append(sorted(group)[len(group) // 2]) 29. i += 5 31. *# 재귀적으로 중간값들의 중간값을 찾음* 32. return median\_of\_medians(medians, 0, len(medians) - 1) 33. *# Median of Medians 알고리즘을 사용하여 k번째 작은 원소를 찾는 함수* 34. def select(arr, low, high, k): 35. *# 기저 사례: 배열에 원소가 하나밖에 없을 경우 그 원소를 반환* 36. if low == high: 37. return arr[low] 39. *# 중간값의 중간값을 찾아 기준 값(pivot)으로 사용* 40. pivot = median\_of\_medians(arr, low, high) 42. *# 배열을 기준 값(pivot)으로 분할* 43. pivot\_index = partition(arr, low, high, pivot) 45. *# 기준 값의 순위를 계산* 46. rank = pivot\_index - low + 1 48. *# k번째 원소가 기준 값일 경우, 해당 값을 반환* 49. if rank == k: 50. return arr[pivot\_index] 51. *# k번째 원소가 왼쪽 부분에 있을 경우, 왼쪽 부분에서 재귀적으로 탐색* 52. elif k < rank: 53. return select(arr, low, pivot\_index - 1, k) 54. *# k번째 원소가 오른쪽 부분에 있을 경우, 오른쪽 부분에서 재귀적으로 탐색* 55. else: 56. return select(arr, pivot\_index + 1, high, k - rank) 57. *# k번째 작은 원소를 찾는 메인 함수* 58. def kth\_smallest(arr, k): 59. *# 입력 값 유효성 검사* 60. if k < 1 or k > len(arr): 61. raise ValueError("k의 값이 배열 크기를 벗어났습니다.") 62. return select(arr, 0, len(arr) - 1, k) 63. *# 사용 예시* 64. arr = [12, 3, 5, 7, 4, 19, 26] 65. k = 3 66. result = kth\_smallest(arr, k) 67. print(f"배열에서 {k}번째 작은 원소는 {result}입니다.") |

1. Test your **linear selection** with random test inputs for verification of correctness. (2 points)

|  |
| --- |
| 스크린샷, 텍스트, 멀티미디어 소프트웨어, 소프트웨어이(가) 표시된 사진  자동 생성된 설명 |