프로그래밍 언어 hw1

B711016 김길호

April 6, 2021

1 과제 1

리스트가 정렬된 상태라고 하였으므로 원소들이 항상 단조 증가하기에 리스트에서 n에 대해 이분 탐색을 진행합니다. 이분 탐색을 구현하는 방식은 사람마다 다양하겠지만 저의 경우 왼쪽 포인터 l이 오른쪽 포인터 r보다 항상 작거나 같은 채로 유지하도록 while문을 돌렸으며, 그 내부 구현방식은 l, r의 중간 값인 m을 인덱스로 갖는 $li[m]$과 n의 대소를 아래와 같이 비교했습니다.

1. $li[m] < n : l = m + 1$

2. $li[m] == n$: m번째 인덱스를 반환하는데 $0 - index$이므로 $m + 1$을 return

3. $li[m] > n : r = m - 1$

또한 while문을 빠져나오는 경우 n을 찾지 못한 것이므로 -1을 리턴했습니다.

2 과제 2

퀵소트 함수의 파라미터로 정렬할 구간을 넣어줬고, 함수 내부에선 시작 점을 가리키는 인덱스 변수 $p1$과 끝 점을 가리키는 인덱스 변수 $p2$를 사용했습니다. 또한 두 인덱스 변수가 가리키는 값과 비교할 피봇을 항상 구간의 중앙 값으로 잡아준 후, $p1 <= p2$를 만족하는 동안 $p1$은 시작 점부터 피봇보다 큰 원소를 찾고, $p2$는 끝 점부터 피봇보다 작은 원소를 찾아 두 인덱스가 가리키는 값들을 교환해줬습니다. 마지막으로 위의 과정을 반복하고 while문을 빠져나온 경우, 피봇보다 작은 값은 구간 $[s, p2]$에 피봇보다 큰 값은 구간$[p1, e]$에 존재하게 됩니다. 따라서 피봇을 제외한 나머지 두 구간에 대해 재귀적으로 함수를 호출하면 결과적으로 구간$[l, r]$에 대해서 정렬됨을 보장할 수 있습니다.

3 과제 3

머지소트는 정렬 할 구간을 분할 하는 함수와 분할 된 원소 조각들을 합쳐주는 함수로 구현방식에선 두 가지 함수를 분리했습니다. 우선 분할하는 함수에선 양 끝점l, r의 중점m에 대해 원래의 구간을 $[l, m]$, $[m+1, r]$으로 분리하여 같은 방식으로 구간을 분할하도록 재귀로 넘겨주었습니다. 구간을 나눈 두 분할 함수를 종료하였을 때 두 구간들의 원소들은 모두 분할되었으므로 이 때 이 둘을 다시 합쳐주는 머지함수를 사용합니다. 머지함수에선 정렬된 채로 합쳐질 $[l, r]$구간의 원소들을 담는 임시 배열 하나와 원소를 각각의 구간을 순회할 인덱스 변수$p1, p2$를 사용합니다. 두 인덱스는 임시 배열에 원소가 오름차순으로 담기도록 서로가 가리키는 값들의 대소를 비교하며 배열에 담아주고, 각 각 구간끝까지 도달했을 때 정렬된 임시 배열의 값을 원래 배열에 덮어씌움으로써 구간$[l, r]$의 원소들을 정렬된 채로 합칠 수 있습니다.

4 과제 4

트리를 구현하기 위해서 왼쪽 노드, 오른쪽 노드, 노드의 번호를 변수로 갖는 노드 클래스를 만들었습니다. 문제에서 주어진 노드간의 관계는 루트 노드를 만들어 루트 노드를 기준으로 자식 정보를 연결해주었고, 각 순회 방법은 자식 l, r를 순회할 때, 노드의 방문 순서를 달리하며 노드를 출력하도록 구현하였습니다.

5 과제 5

강의실을 최대로 배정하기위해서 아래와 같은 세 가지 가설을 세워보고 이를 간단한 예시들을 통해 반증하여 그리디하게 해결할 수 있음을 보이겠습니다.

1. 시작 시간이 빠른대로 배정하면 최대 개수를 만족한다.

2. 강의 시간이 짧은 순서대로 배정하면 최대 개수를 만족한다.

3. 종료 시간이 빠른대로 배정하면 최대 개수를 만족한다.

1번 가설의 경우, 10시까지 이용 가능할 때 $[0, 10], [2, 3], [3, 4]$...와 같이 시작 시간이 늦더라도 더 많은 강의실을 배정할 수 있는 간단한 반례가 존재하기에 최적의 해가 아닙니다.

2번 가설의 경우, 10시까지 이용 가능할 때 $[9, 10], [8, 10], [1, 8]$와 같이 강의 시간이 조금은 길더라도 더 많은 강의실을 배정할 수 있어서 최적의 해가 아닙니다.

3번 가설의 경우, 강의실 배정하는 방법 L의 최적 해중 종료 시간이 가장 빠른 Lmin을 포함하지 않는 답이 있다고 가정해봅시다. 이 때 이 최적해의 가장 첫번째 강의를 지운 후 Lmin을 포함하는 강의를 추가한다고 해도 최적해가 깨지지않습니다. 따라서 항상 Lmin을 포함하는 최적해가 존재함을 알 수 있습니다.

이에 주어진 리스트를 람다를 사용하여 1)종료 시간이 빠른 순, 2)시작 시간이 빠른 순으로 정렬한 후, 리스트를 현재 시간이 시작 시간보다 작거나 같은 경우에만 현재 시간을 종료 시간으로 갱신하며 스위핑하면 최대로 강의 배정을 할 수 있습니다.

6 과제 6

문제에서 설명하는 내용은 ~ 기호만 +로 바꾼다면 정규 표현식을 사용하여 쉽게 해결할 수 있습니다. 따라서 파이썬 라이브러리 re를 사용하여 주어진 패턴 객체를 만든 후, input으로 받는 문자열과 완전히 매칭되는지를 판별하였습니다.