202235070 안수빈

동적 계획법의 핵심은 ‘문제의 일부분을 풀고, 그 결과를 재활용하는 것’인 반면 탐욕 알고리즘의 핵심은 ‘현재 상황에서 당장 좋은 것만 고르는 것에 있다.

백준 1931번 코드 비교

greedy 알고리즘을 선택했을 때

* 풀이 과정

정렬

회의 종료 시간을 기준으로 오름차순 정렬. (종료 시간이 같으면 시작 시간을 기준으로 정렬)

탐욕적 선택

현재 회의가 이전에 선택한 회의의 종료 시간 이후에 시작한다면 선택.

선택된 회의의 종료 시간을 갱신하며 반복.

* 총 시간 복잡도: O(N log N)

정렬: O(N log N)

회의 탐색: O(N)

* 한 번 정렬 후 순차적으로 탐색하므로 간단하고 효율적
* 문제의 최적 부분 구조의 탐욕적 선택 속성을 모두 만족하기 때문에 최적해를 보장

dp 알고리즘을 선택했을 때

* 풀이 과정 :
* 시간 복잡도 :
* 최적 부분 구조 정의:

결론

Greedy가 더 효율적.

이 문제는 탐욕적 선택 속성과 최적 부분 구조를 만족하므로 Greedy 알고리즘이 적합

DP는 비효율적으로 메모리를 사용하며 불필요한 계산을 포함함 -> 과하게 복잡

DP는 중복 하위 문제가 많거나, 탐욕적 접근이 최적해를 보장하지 못할 때 적합하지만, 이 문제에서는 Greedy로 최적해를 간단히 구할 수 있음.

알고리즘 비교

greedy가 효율적인 경우

* 문제의 최적 부분 구조가 단순할 때 : 현재 단계에서의 최선의 선택이 전체 최적해에 영향을 주지 않는 경우
* 해결 과정이 순차적일 때 : 한번 결정한 선택을 되돌리지 않아도 되는 경우
* 문제를 직관적으로 풀 수 있을 때 : 최적해를 도출하기 위해 복잡한 상태 추적이나 중복 계산이 필요 없을 때

dp가 효율적인 경우

* 중복된 하위 문제가 존재할 때 : 동일한 계산을 반복하는 경우
* 최적 부분 구조가 복잡할 때: 최적해를 구하기 위해 여러 가지 경우를 모두 탐색해야 할 때
* 경우의 수가 많고, 과거 상태를 참조해야 할 때: 현재 상태에서 이전 여러 단계의 결과를 종합해서 최적해를 구해야 할 때

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

결론

탐욕적 선택이 이후 과정에도 항상 최적임을 보장할 수 있다면 → Greedy.

현재 상태가 과거의 결정에 의존하거나, 가능한 모든 경우를 계산해야 한다면 → DP.

문제를 해결한 후, 결과가 틀릴 가능성이 있다면 Greedy를 의심하고 DP로 변경