알고리즘_06 3주차 수업 대체 과제

202111971 이재혁

알고리즘

특정한 입력에 대해 문제를 해결하는 방법

입력 크기에 따라 알고리즘의 실행시간이 달라지는데, 이것으로 알고리즘의 효율성을 판단할 수 있다.

→ 입력의 크기가 커져도 실행시간이 크게 증가하지 않고, 안정적으로 증가할 수 있어야한다.

현대에서 다루는 자료의 크기가 많아지고 있다.

ex) 페이스북 : 5억개 이상의 노드, 인간 genome 정보 10억개 이상, 미국 고속도로 차량 정보 등

대규모 입력(1조 이상)에 대한 문제를 처리의 효율성이 중요해지고있다.

앞으로 다루는 자료의 크기가 점점더 커질 것이기 때문에, 입력의 확장에도 안정적인 알고리즘 설계가 중요하다.

알고리즘적 사고

문제를 해결하는 여러 방법 중 가장 효율적인 방법을 찾고, 증명한다.

Peak finder : 극댓값 찾기

Case 1) 1차원 배열

배열 : .... a b c .... d(마지막) 일때

b ≥ a 이고 b ≥ c 이면 b는 극댓값이다.

d가 배열의 마지막 index 위치할 때 d는 배열의 왼쪽값보다만 크다면 극댓값이다.

극대값 정의는 등호가 포함되어 있기 때문에 무조건 존재하는 값이다.

straight foward algorithm

배열 원소의 처음 index부터 순서대로 극댓값인지 확인

확률적으로 n/2개의 원소의 값을 검사하면 극댓값을 찾을 수 있다.

최악의 경우 모든 값을 확인해야 하므로 최악의 시간 복잡도는 **Θ(n)이다.**

알고리즘적 사고 → 효율성 높이기

배열의 중간 원소를 기준으로 양옆을 비교한다.

중간원소가 양옆의 원소보다 크다 → 극댓값

중간원소보다 큰 값이 있다면, 큰 쪽의 나머지 배열을 부분 배열로 다시 반복한다.

→ 큰 값을 찾아가다보면 극댓값을 찾을 수 있다.

매 반복마다 배열의 절반씩 제거한다. 최악의 경우 배열의 길이가 1이 될때 까지 반복한다.

$n/2^k$ = 1 → k = $logn$

따라서 최악의 시간 복잡도는 **Θ($logn$) 으로 보다 효율적이다.**

Case 2) 2차원 배열

배열 : .... d ....

.... b a c ....

.... e .... 일 때

a ≥ b, a ≥ c, a ≥ d, a≥ e 이면 a는 극댓값이다.

greedy ascent algorithm

2차원 배열에서 극대값 찾기

특정 index 부터 인접한 값들 중 큰 방향으로 이동한다.

더이상 이동 할 수 없다는 것은 인접한 값들이 모두 현재 값보다 작기 때문에, 극대값임을 알 수 있다.

Θ(nm)의 시간복잡도를 가진다. n (행의 개수), m (열의 개수)

정사각행렬 → **Θ(n^2)**

알고리즘적 사고 → 효율성 높이기

시작 index 행렬의 중앙 열 중 최댓값

좌우 값을 비교해 큰 쪽으로 이동 후 반대쪽 행을 모두 버린다.

남은 행렬의 중앙 열을 찾아 위 과정을 반복한다.

이동한 열에서 최대값으로 이동 후 위 과정을 반복한다.

만약 좌우값보다 현재 값이 크다면 2차원 배열의 극대값이다.

2차원 배열 중 열의 개수를 절반 씩 줄여가기 때문에, m개의 열이 1개가 된다.

그 중 가장 큰 값을 찾는데 행의 개수만큼의 시간이 걸리기 때문에

최악의 시간 복잡도는 **Θ**$(nlogm)$로 보다 효율적이다.

중앙열 중 최대값을 고르는 이유

중앙열의 극대값만 고른다면, 이미 잘라낸 부분에서 자신의 값보다 큰 값이 있을 수 있다.

→ 2차원 극대값을 만족하지 않게 될 수도 있다.

의도한대로 작동하는 것이 효율적으로 구현하는 것보다 중요하다.

최대값만 고른다면, 현재값보다 큰 인접한 행으로 이동했을 때, 이동한 값은은 이전 행의 모든 값보다 큰 것을 보장할 수 있다.

→ 이동한 행중 가장 큰 값을 찾으면 주변 값보다 큰 2차원 극대값을 찾을 수 있다.

소감

알고리즘이 효율적이어야하는 이유를 문제를 빠르게 해결하기 위해서, 컴퓨터의 성능을 향상시키기 위해서 등 막연하게 생각하고 있었습니다.

영상 강의를 들으며 입력의 크기가 앞으로 더 커질 것이고 그럼에도 안정정으로 문제를 안정적으로 해결해야한다는 것이 크게 다가왔습니다.

전산학 분야가 인공지능에 초점을 두는 추세인 것 같습니다. 인공지능 모델학습을 위해 많은 데이터를 정제하고(행렬화) 학습시켜야한다고 알고 있습니다.

많은 데이터를 다루는 과정이 효율적이지 못했다면, 인공지능 학습에 시간이 오래 걸렸을 것이고, 기술의 발전이 뒤쳐졌을 수도 있었겠다는 생각이 들었습니다.

알고리즘이라는 과목이 굉장히 원론적인 과목이지만, 수업을 통해 알고리즘적 사고를 키우는 것이

문제해결, 기술구현능력에 필수적이라고 느꼈습니다.