<알고리즘>

4장 요약문 작성

동적프로그래밍을 배우면서 다루었던 최적화 문제에 대한 연장선으로 Greedy Approach 알고리즘을 배웠다. 작은 문제에 대한 답을 구하고 나중에 결합하여 전체의 문제를 만드는 동적 프로그래밍과는 다르게 Greedy Approach 알고리즘은 각 단계마다 일정한 규칙으로 답을 구하고, 결국 전체 답의 일부가 되는 것이다. 따라서 해당하는 답이 진정한 최적해인지, Global solution인지 검증하는 correctness 증명과정이 필요하다. Greedy 알고리즘도 동적 알고리즘처럼 반복문으로 작성되는 경우가 많고, 때문에 적은 시간복잡도로 문제 해답을 구할 수 있다. Correctness 만 제대로 입증하여 올바른 답을 구할 수 있다면 매우 효율적인 해결방법이다.

동적프로그램과의 차이점을 제대로 알고 넘어가기 위해 Greedy 단원에서 배운 알고리즘 중 single-source shortest path 문제를 다루고 넘어가겠다. 해당 문제는 다익스트라 알고리즘으로도 알려져 있다. 동적 프로그래밍 단원에서 플로이드 알고리즘으로 배웠던 shortest path 문제가 모든 노드 쌍에 대한 최소 경로를 구하는 것이라면, 다익스트라 알고리즘은 시작 노드가 주어진 채로 다른 모든 노드에 대한 최소 경로를 구하는 것이다. 따라서 처음 시작으로 주어진 노드에서부터 가장 최소 길이로 도달할 수 있는 노드를 찾아 path를 구성하는 선택 기준으로 계속해서 최적 경로를 탐색한다. 각 단계에서 일정한 규칙으로 해답을 만드는 것이니 greedy 알고리즘의 아주 좋은 예라고 할 수 있다. 다익스트라 알고리즘과 매우 유사한 해결 방법으로 다른 문제를 풀 수도 있다. Minimize spanning tree 문제이다. 해당 문제를 푸는 방법은 prim과 kruskal이 있는데, 두 방법 모두 다익스트라 알고리즘처럼 기본적인 해결 접근법이 Greedy Approach이다.

코딩 수업을 할 때 Greedy 알고리즘으로 구현을 하면 항상 원하는 정답을 내지 못해 답답했었는데 그 이유를 알 수 있게 되어 속이 시원하고, 애초에 구할 수 없던 방식으로 머리를 감싸고 덤벼들었던 작년 생각이 난다. 직접 피부로 느꼈기에 기억도 더 오래 남을 것 같다.