<알고리즘>

6강 요약문

# Branch and Bound

Branch and Bound 알고리즘은 BackTracking과 매우 비슷한 형태를 갖는다. 형태 뿐만 아니라, 동작원리나 구조도 상당히 비슷하다. 그렇다면 굳이 왜 BackTracking 만이 아니라, Branch and Bound 알고리즘이 필요한 것인가. Branch and Bound 알고리즘은 최적화 문제를 위한 것이기 때문이다. Bound라는 단어에서부터 상한선 혹은 하한선을 만들어 두고, BackTracking처럼 가지치기를 한다는 것이다. State Space Tree에서 bound 값을 이용하여 최적해를 찾고, 그 최적해로의 과정도 bound 값을 이용하여 최적인 것을 찾았기 때문에 최적의 원칙이 적용되고 최적화 문제에 적용할 수 있는 것이다.

BAB 알고리즘이 BT와 구체적으로 어떻게 다른 지 먼저 알아보자. BT은 깊이우선탐색, DFS를 한다. DFS는 재귀호출로 간단하게 구현이 가능하기 때문에 별도의 자료구조가 필요 없다. 하지만 DFS가 별도의 정보가 없이 그저 깊이를 우선하여 탐색하는 알고리즘이기 때문에 가지치기의 효율은 상대적으로 적다고 할 수 있다.

BAB 알고리즘은 BT에서 좀 더 나은 가지치기를 위해 BEST FIRST SEARCH를 한다. Bound 값을 이용하여 가능성이 큰 순으로 먼저 탐색을 할 것이기 때문에, DFS처럼 순차적인 탐색보다는 임의 순서의 탐색을 할 것이고, 이를 위해서 HEAP이라는 별도의 자료구조가 필요하다. 또한 bound값을 이용하기 위해서 그 값을 계산하는 절차가 필요한데, 이때 소요되는 시간이 너무 길어지면 원래의 목적인 가지치기로 인한 시간 줄이기가 무용지물이 된다. 따라서 문제와 instance에 따라서 적절한 타협이 필요하다.

동적 프로그래밍 단원에서 배웠던 TSP를 보면 잘 알 수 있듯이 단순히 모든 경우의 수를 계산하면 (n-1)! 개를 탐색해야 하지만, BAB를 사용하면 탐색해야 하는 노드의 수가 17개 밖에 되지 않는다. 최적화 문제에 한해서지만, 엄청난 효율을 갖고 있다고 생각한다.

지난 시간에 배운 BT에 이어서 같은 범주지만 알아 두면 두고두고 써먹을 수 있는 효자 같은 알고리즘이다. 골자가 거의 대부분의 알고리즘에서 공용으로 쓰이기 때문에 기본 틀이라도 기억해두는 것이 좋을 것이다.