안녕하세요, 동계 대학생 S/W 알고리즘 특강의 세번째 시간인 오늘은 DP 에 대해 다루어보도록 하겠습니다.

1. 기초 강의

동영상 강의 컨텐츠 확인 > 5. DP

Link :
https://swexpertacademy.com/main/learn/course/subjectDetail.do?courseId=CONTENTS_REVIEW&subjectId=AYVXakhKQSsDFARs

※ 출석은 강의 수강 내역으로 확인합니다.

DP 7차시 TSP 강의 오류

11:34부터 시작하는 시간 복잡도 분석에서 바로 이전 슬라이드의 for문 3개만 계산했지만, 3번 for 문 아래에서 minimum을 구하는 과정이 k번 반복되므로 계산에 넣어야 합니다.

11:34부터 시작하는 슬라이드에서 시간 복잡도 계산을 아래처럼 정정합니다.

$$T(n) = \sum_{k=1}^{n-1} \binom{n-1}{k} (n-k-1)(k) \in \theta(n^2 2^n)$$

2. 실전 강의

2-1. DP

DP를 설명하기에 앞서 간단하게 수학적 귀납법에 대해 알아보겠습니다. 어떤 등식이 모든 n에 대하여 성립함을 보일 때 가능한 모든 n을 등식에 대입해서 증명할 수는 없습니다. 이때 사용하는 방법이 수학적 귀납법입니다. 수학적 귀납법은 주어진 등식이 n = 1 일 때 성립함(귀납 기본, induction base)을 증명하고, n일 때 성립한다고 가정(귀납 가정, induction hypothesis)한 뒤 n+1일 때 성립함을 증명(귀납 단계, induction step)하는 것입니다. 이러한 증명이 완료되면 모든 n에 대해 등식이 성립함을 보일 수 있습니다.

DP는 위와 같이 귀납법을 만족하는 점화식을 찾아서 recursive와 memoization을 활용해 문제를 해결하는 방식입니다. Memoization은 컴퓨터 프로그램을 실행할 때 이전에 계산한 값을 메모리에 저장해서 중복 계산을 줄여 전체적인 실행 속도를 빠르게 하는 기술입니다.

DP 알고리즘은 그리디 알고리즘과 같이 최적화 문제를 해결하는 알고리즘입니다. 문제를 해결할 때 먼저 작은 부분 문제들의 해를 구하고 이를 이용하여 보다 큰 크기의 부분 문제를 해결해 나가며 최종적으로 원래 주어진 문제를 해결하는 방법을 사용합니다.

DP를 적용할 수 있는 문제는 두가지 요건을 필수로 가지고 있어야 합니다.
1) 최적 부분 문제 구조(Optimal substructure): DP를 효율적으로 적용하기 위해선 문제가 최적화

의 원칙(Principle of Optimality)을 만족해야 합니다. 최적화의 원칙이란 어떤 문제에 대한 해가 최적일 때 그 해를 구성하는 작은 문제들의 해 역시 최적이어야 한다는 것입니다. DP 방법이 작은 해의 최적 해를 통해 큰 문제의 최적화를 구하기 때문에 최적화의 원칙을 만족하는 문제가 아니라면 DP를 적용할 수 없습니다.

2) 중복 부분 문제 구조(Overlapping sbproblems): DP는 큰 문제를 이루는 작은 문제들을 먼저 해결하고 작은 문제들의 최적 해를 이용하여 순환적으로 큰 문제를 해결합니다. 이때 순환적인 관계를 명시적으로 표현하기 위해 점화식을 사용합니다. 이러한 순환적인 성질 때문에 이전에 계산된 작은 문제의 해가 다른 문제에서 필요하게 되는데(Overlapping subproblems) 이를 위해 이미 해결된 작은 문제들의 해들을 어떤 저장 공간에 저장합니다. 저장된 작은 문제들의 해는 필요할 때마다 다시 계산하는 과정 없이 참조할 수 있기 때문에 중복된 계산을 피하게 만들어줍니다.

3. 기본 문제

· 최장 공통 부분 수열

· 0/1 Knapsack