고급 소프트웨어 실습

분반: 4분반

학번: 20191583

이름: 김태곤

문제1. 실습 3이 Greedy algorithm과 Dynamic programming 중 어디에 해당된다고 할 수 있는가? 이유를 설명하시오.

Greedy alorithm이란 매 순간 최선의 선택을 통해 결과를 도출하는 알고리즘이다. Dynamic programming은 복잡한 문제를 여러 문제로 나누어 푸는 방식으로, 하위 문제의 결과를 저장하고 상위 문제에 적용해 해결하는 알고리즘이다. 이러한 특징으로 DP는 하위 문제들이 여러번 반복되어 비효율적인 계산이 되는 것을 막기 위해 자주 사용된다.

실습3은 Dynamic programming에 해당된다. DP를 사용하기 위해서는 2가지 조건을 만족해야 한다. 첫번째로는 겹치는 부분 문제이다. DP는 하위 문제로 나누고, 그 값을 사용하여 전체의 답을 구한다. 따라서 DP를 사용하기 위해서는 하위 문제들이 반복적으로 나타나야 한다. 3번 문제에서는 i번째까지의 수열 중 가장 큰 값이 반복적으로 등장한다. 또한 이를 이용하여 i+1번째 수열까지의 가장 큰 합을 도출해낸다. 두번째 조건은 최적 부분 구조이다. 하위 문제의 처적 결과 값을 사용해 전체의 최적의 결과를 도출해 내야한다. 본 문제에서 최대 합을 구하는 식은 sum(i+1) = max(max(a[i+1],sum[i]+a[i+1]))으로 i번째 문제의 값이 최적이면 i+1번째도 최적의 결과를 가져온다. 따라서 실습3은 DP에 해당한다.

Greedy algorithm은 매 순간 최선의 선택을 하지만, 최종 결과가 항상 최적화 되지 않는다는 단점으로 실습3에서는 Greedy algorithm보다 DP가 더 적합하다.

문제2. 앞에서 다루지 않은 문제 중에 Dynamic programming을 사용했을 때 더 효율적으로 풀 수 있는 문제를 2개 들고, 알고리즘을 설명하시오. brute force 방식에 비하여 DP를 사용하는 경우 어느 정도 빨라지는지도 설명하시오.

1로 만들기는 DP를 사용했을 때 더 효율적으로 풀 수 있다. 1로 만들기 문제는 주어진 X가 3으로 나누어 떨어지면 3으로 나누고, 2로 나누어 떨어지면 2로 나누고, 아니면 1을 빼는 3가지 과정을 통해 1을 만드는 문제이다. 이 문제를 DP로 풀면 다음과 같다. d[i] = d[i-1]+1을 통해 1을 빼는 연산과, i가 2로 나누어 떨어질 때 2로 나누는 연산 d[i]=min(d[i],d[i//2]+1), i가 3일 때 3으로 나누는 연산 d[i]=min(d[i],d[i//3]+1)을 2부터 x+1까지 반복해준다. 이러한 알고리즘으로 해결하면 O(n)의 시간복잡도를 가지게 된다. brute force 방식으로 해결하면 시간은 이보다 더 오래 걸리게 된다. brute force는 조합 가능한 모든 문자열을 하나씩 대입해 보는 방식이다. 1로 만들기 문제는 3가지 과정을 통해 1을 만들어야 한다. 백트래킹을 통해 3가지

연산 결과를 모두 탐색하도록 구현하면 시간 복잡도는 O(3^n)이 된다. 따라서 이 문제는 DP를 사용하는 것이 더 효율적이다.

LCS(최장 공통 부분 수열) 문제는 두 문자열이 주어졌을 때, 모두의 부분 수열이 되는 수열 중 가장 긴 것을 찾는 문제이다. 이 문제는 이차원 배열을 사용한 동적 계획법으로 해결할 수 있다. 두 문자열의 길이만큼 2중 반복문을 통해 해당 위치의 글자가 같으면 전의 LCS 값보다 1을 더해 저장(array[i][j]=array[i-1][j-1]+1), 다르면 이전까지 비교한 값중 최대값을 저장하면 된다.(array[i][j]=max(array[i][j-1],array[i-1][j])) LCS의 길이는 배열의 가장 마지막 인덱스인 array[n][m]이 된다. 이러한 DP를 이용한 알고리즘을 통해 구현하면 시간복잡도는 O(nm)이 된다. DP를 사용하지 않고 Brute force 방식으로 단순 재귀적 해결을 하면 다음과 같다. 함수 lcs를 해당 위치의 글자가 같으면 1+lcs(x,y,m-1,n-1), 다르면 max(lcs(x,y,m,n-1),lcs(x,y,m-1,n))으로 재귀적으로 선언한다. 이렇게 선언된 lcs는 최악의 경우 O(2^n)의 시간복잡도를 가지게 된다. 따라서 이 문제도 DP를 사용하여 해결하는 것이 효율적이다.