3주차 문제 풀이

모든 코드는 www.github.com/jjwdi0/ 에 있습니다~

문제 목록-(1)

- 1로 만들기
- RGB 거리
- 계단 오르기
- 숫자삼각형
- 포도주 시식
- 연속합
- 스티커
- 가장 긴 증가하는 부분 수열
- LCS
- 동전

문제 목록-(2)

- 돌다리 건너기
- 1학년
- 트리의 독립집합
- 환상의 듀엣
- 우수 마을
- 개근상
- 외판원 순회
- 무한 수열
- 박성원
- 고속도로
- 브리징 시그널
- 건포도
- 경찰차

이론적 배경 - 동적계획법

- 이번 주 문제가 많은데, 그만큼 동적계획법은 중요하면서도 매우 많이 나오는 개념이다.
- 우선 동적계획법(Dynamic Programming)은…
- "복잡한 문제를 간단한 여러 개의 문제로 나누어 푸는 방법"
- 즉 어떤 문제를 풀 때 그 문제와 유사한 부분 문제의 해법을 통해 전체 문제를 해결하는 방법이다.

- 뭔 소리인지 모를 것이다. 무시하자.

이론적 배경 - 동적계획법

- 예를 들면 nCr을 계산한다고 하자. F(n, r) = nCr을 반환하는 함수 라 정의하면,
- 점화식은 F(n, r) = F(n-1, r-1) + F(n-1, r) 이다.

- 즉 F(n, r)이라는 문제를 해결하기 위해 F(n-1, r-1)과 F(n-1, r)이라는 부분 문제로 해결하는 것이다.
- 이러한 부분 문제가 이미 풀려 있을 때, 전체 문제를 간단하게 풀 수 있겠지??
- 이걸 잘하면 확통을 잘한다. (= 대학을 잘간다.)

1. 1로 만들기

- 동적계획법 모른다고 못푼다고 하지 말자.
- BFS로도 해결할 수 있는 문제.
- D(i) = i를 1로 만들기 위한 최소 연산 횟수 라 정의하자.
- D(1) = 0일 거고...
- D(x) = min(D(x / 3) + 1, D(x / 2) + 1, D(x - 1) + 1)
- 이를 1 <= x <= N에 대해서 계산만 해주면 O(N)에 해결 가능하다.

2. RGB거리

- 이 문제를 처음 본 초보들은 그리디적인 방법을 생각한다. (그리고 틀리고 망한다.)
- D(i, j) = i번째 집까지 칠했고 j의 색으로 칠했을 때, 최소 비용

- D(i, j) = A(i, j) + min(D(i-1, k)) (0 <= k <= 2, k != j)

- 이거 못풀면 초등학생

3. 계단 오르기

- D(i, j) = i번 계단까지 오르는 데 j칸 연속 올랐을 때 최대 점수
- D(i, j) = D(i-1, j-1) (if j > 0)
- D(i, j) = max(D(k, 1), D(k, 2)) (for 1 <= k < i)

- 마지막 계단은 반드시 밟아야 하므로 max(D(N, 1), D(N, 2))가 답이다. 시간복잡도 O(N).

4. 숫자삼각형

- D(i, j) = A(i, j)까지 올 때 최대 점수
- D(i, j) = max(D(i-1, j-1), D(i-1, j)) + A(i, j)

- 시간복잡도 O(N^2).

- IOI 기출문제이다ㄷㄷ

5. 포도주 시식

- 3. 계단 오르기와 동일
- 단 마지막 포도주를 반드시 먹어야 하는 것은 아니다.

6. 연속합

- 유명한 문제 (Maximum Subarray Problem 이라고도 불림)

- Naïve 한 접근 : O(N^3)
  - N^2개의 [i, j]에 대해 O(N)으로 합 구하기.
- 개선된 접근 : O(N^2)
  - 합을 구할 때 Prefix Sum (sum(i, j) = sum(1, j) – sum(1, i-1)) 이용.

6. 연속합

- 풀이 1. 동적계획법

- D(i) = A(i)를 포함하는 최대 구간합
- D(i) = max(D(i-1) + A(i), A(i))
- 1 <= i <= N에 대해 D(i)의 최대가 답.
- 시간복잡도 O(N).

6. 연속합

- 풀이 2. 분할 정복

- 합이 최대가 되는 구간은 전체를 절반으로 나눴을 때
  - 1. 왼쪽 구간에 있다.
  - 2. 오른쪽 구간에 있다.
  - 3. 양쪽 구간에 걸쳐 있다.
- 이 때 왼쪽과 오른쪽을 재귀호출로 답을 구했다 하고, 양쪽 구간에 걸친 답을 구하자. 어떻게 구할 수 있을까??

6. 연속합

- 풀이 2. 분할 정복

- 양쪽 구간에서 가장 왼쪽 원소를 포함한 최대합, 가장 오른쪽 원소를 포함한 최대합, 전체의 최대합을 구하자.
- 각각 l, r, s라 하면, 전체 구간에서의 답은
- max(l.s, r.s, l.r + r.l) 이다.
- 이유는 다음에 생각하자.
- 시간복잡도는 O(N log N).
- 나중에 자료구조 배울 때 다시 한 번 나올 풀이.

7. 스티커

- 앞의 문제들을 풀었으면 이걸 풀 수 있을 것이다.
- 한번 풀어보자.

8. 가장 긴 증가하는 부분 수열

- LIS (Longest Increasing Subsequence) 문제라고도 함.

- D(i) = A(i)가 증가 부분 수열의 마지막 원소가 될 수 있는 최대 증가 수열의 길이.
- D(i) = max(D(j) + 1) (for j < i and A(j) < A(i))

- 시간복잡도 O(N^2).
- 이진탐색 또는 세그먼트 트리로 시간복잡도 O(N log N)에도 구할 수 있음.

9. LCS

- Longest Common Subsequence 문제.

- D(i, j) = A(1~i)와 B(1~j)까지의 부분문자열에서의 LCS의 길이.
- D(i, j) = max(D(i-1, j), D(i, j-1))
- if A(i) = A(j) then D(i, j) = D(i-1, j-1) + 1
- 시간복잡도 O(NM)
- Toggling이라는 기법을 이용하면 공간복잡도(메모리)를 O(N+M)으로 줄일 수 있음.

10. 동전

- D(i) = i원을 만들 수 있는 가지수
- D(i) = sum(D(i-A(j)))
- 시간복잡도 O(N).

- 동전 문제는 여러 가지 변형이 많고 배낭 문제(knapsack problem)과도 연관이 많다.

11. 돌다리 건너기

- D(i, j, k) = A(i, j)를 목표 문자열의 k번째 문자로 건넜을 때
- D(i, j, k) = sum(D(i^1, j', k-1)) (for j' < j)
- 시간복잡도 O(NM).

- 흔히 쓰는 배열은 2차원이 최대일것이다. 이번 기회에 3차원까지 써보자.

12. 1학년

- D(i, j) = A(i)까지 썼을 때 j를 만들 수 있는 경우의 수
- D(i, j) = D(i-1, j+A(i)) + D(i-1, j-A(i))
- 시간복잡도 O(N).

13. 트리의 독립집합

- 일반적으로 최대 독립 집합 문제는 NP-Hard이다.
- 트리에서는 동적계획법을 이용해 O(N)에 해결할 수 있다.

- 어떤 정점을 선택했을 때, 다른 인접한 정점을 선택하면 안 된다.
- 즉 어떤 부분 트리에서 루트가 선택되었을 때, 인접한 자식들을 선택하면 안 된다는 것을 이용하자.

13. 트리의 독립집합

- 각 정점의 상태를 선택하거나 / 선택하지 않는다 로 나누자.
- D(i, j) = i번 정점의 선택 여부가 j일 때 i번 정점을 부트리로 하는 트리의 독립집합의 최대 크기
- 만약 j = 1일 때(선택했을 때)
  - D(i, j) = sum(D(k, j^1)) (for i is parent of k)
- j = 0일 때(선택하지 않았을 때)
  - D(i, j) = sum(max(D(k, 0), D(k, 1))) (for i is parent of k)
- 루트를 임의로 잡고 DFS를 통해 아주 쉽게 구할 수 있다.
- 시간복잡도 O(N).

14. 환상의 듀엣

- D(i, j) = 두 사람이 A(i), A(j)를 마지막으로 불렀을 때 최소 힘듬의 정도
- D(i, j) = min(D(i, i-1), D(i, j-1)) (if i < j)
- D(i, j) = min(D(j-1, j), D(i-1, j)) if(i > j)
- 시간복잡도 O(N^2).

- 이렇게 점화식을 세우는 동적계획법을 바이토닉 DP라 한다.

15. 우수 마을

- 트리의 독립집합 문제와 동일.

16. 개근상

- 점화식이 매우 다양한 문제.
- 알아서 풀어 봅시다. 이정도는 풀어야지ㅎㅎ

17. 외판원 순회

- 최 모씨 졸업논문 주제.
- N이 매우 작으면 모든 순열에 대해 경로를 탐색해주면 O(N!)으로 해결할 수 있다.
- 그런데 이렇게 애매하게(?) 작으면 어떻게 할까?

17. 외판원 순회

- 만약 지금까지 거쳐온 도시들의 정보와 지금 있는 도시가 상태로 주어진다면, 지금까지의 경로의 가중치의 합의 최소는 일정한가?
- 이를 가지고 동적계획법을 수행할 수 있다.
- D(i, j) = i의 비트에 해당하는 도시를 들렀고 j번 도시에 있을 때, 그때까지 가능한 최소 가중치 합
- D(i, j) = A(j) + min(f(i^(1<<j), k))

17. 외판원 순회

- 자세한 구현을 설명하자면...
- 만약 1번 도시, 2번 도시, 5번 도시를 방문했다면
- 10011(2) 처럼 나타내자.(2진수)
- 그러면 2^N의 가짓수를 모두 나타낼 수 있다.
- 자세한 건 코드 참고ㅎㅎ

- 시간복잡도 O(N*2^N)

18. 무한 수열

- N의 범위가 너무 크다.
- 메모이제이션 + std::map을 이용해 원하는 메모리만 잡아서 해결하면 어떨까?

- 이를 알아채면 구현은 넘나 쉽다.

19. 박성원

- 외판원 순회처럼 비트 DP를 하는 문제다.
- 코딩도 적당히 난이도가 있고 생각해야 할 것도 많다.
- 풀면 좋은 문제. 설명은 생략.

20. 고속도로

- 이 문제는 2016 KOI 고등부 2번 주유소 문제와 매우 유사하다.
- D(i, j) = i번 도시에 있을 때 돈이 j원 남았을 때 최소 시간
- 위 정의를 가지고 BFS를 돌리면 된다.
- 시간복잡도는 O(K*N^2)

- 이를 좀 더 개선하려면 우선순위 큐를 이용해 Dijkstra 알고리즘을 이용하면 된다.
- 시간복잡도는 O(NK log N).

21. 브리징 시그널

- KOI 전깃줄 문제와 유사.
- LIS 문제와 동치라는 것을 눈치챘는가??

- 그렇다면 O(N log N)으로 LIS를 찾으면 된다.

22. 건포도

- IOI 2009 기출문제.
- IOI 겨울학교 실습문제.

- 개쉽다.
- 힌트를 주자면... 4차원 배열?

23. 경찰차

- 환상의 듀엣 문제와 비슷. 바이토닉 DP로 해결 가능하다.
- DP 후 답을 출력하는 문제도 연습해 두자.

- 이번주는 문제도 많고, 바빠서 참여율이 저조하다.bb
- 다음주는 더 쉽고 간단한 그리디를 하자