Dynamic Programming 2
7회차 강의

서강대학교 전해성(seastar105)

DP와 분할정복의 차이점

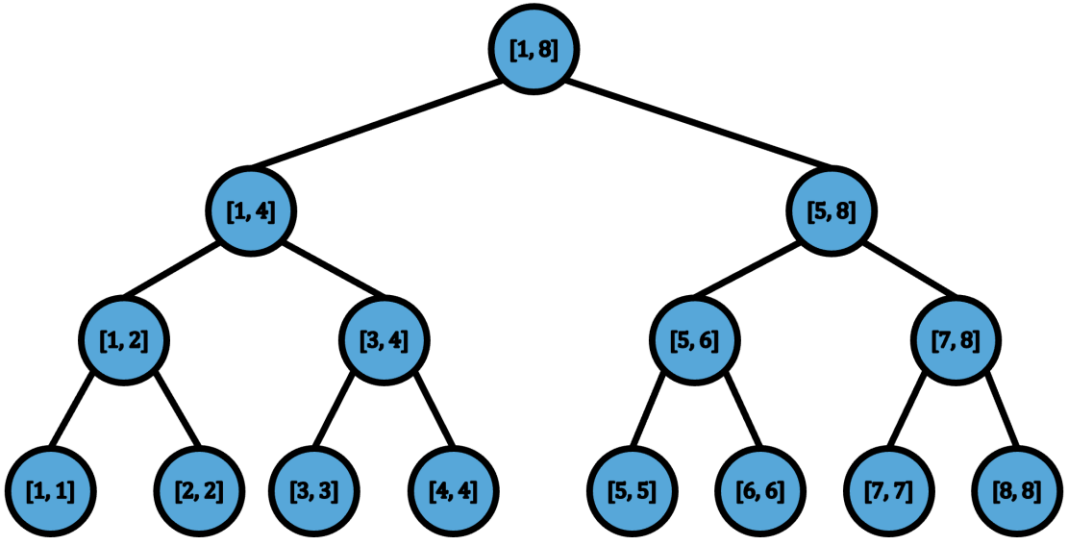
DP와 분할정복 모두 원래의 문제를 같은 구조의 부분 문제로 나누고 작은 문제부터 풀고 그를 통해 원래의 문제를 해결합니다.

하지만 차이점은 문제를 나누는 방식입니다.

DP와 분할정복의 차이점

분할정복의 경우 부분문제끼리 하위 부분문제를 공유하지 않도록 합니다.

Merge Sort에서 구간을 나눴던걸 생각해보면 서로 다른 부분문제끼리는 하위 부분 문제를 공유하지 않습니다.

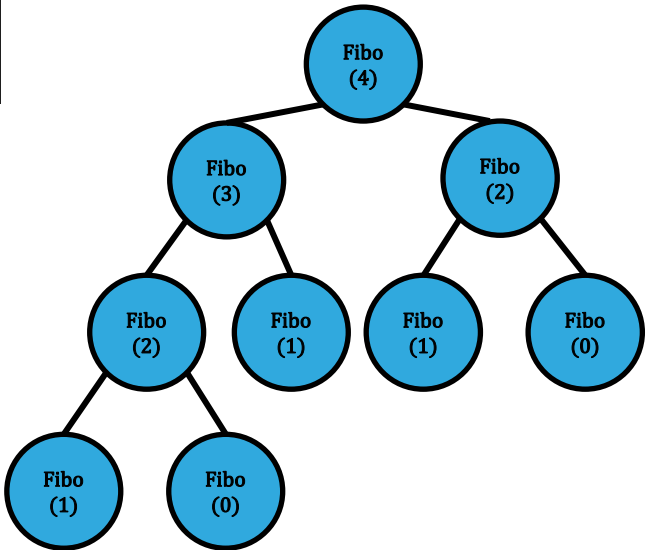


DP와 분할정복의 차이점

DP는 부분문제끼리 같은 하위 부분문제를 공유합니다.

재귀적으로 피보나치 수를 구한다고 해봅시다.

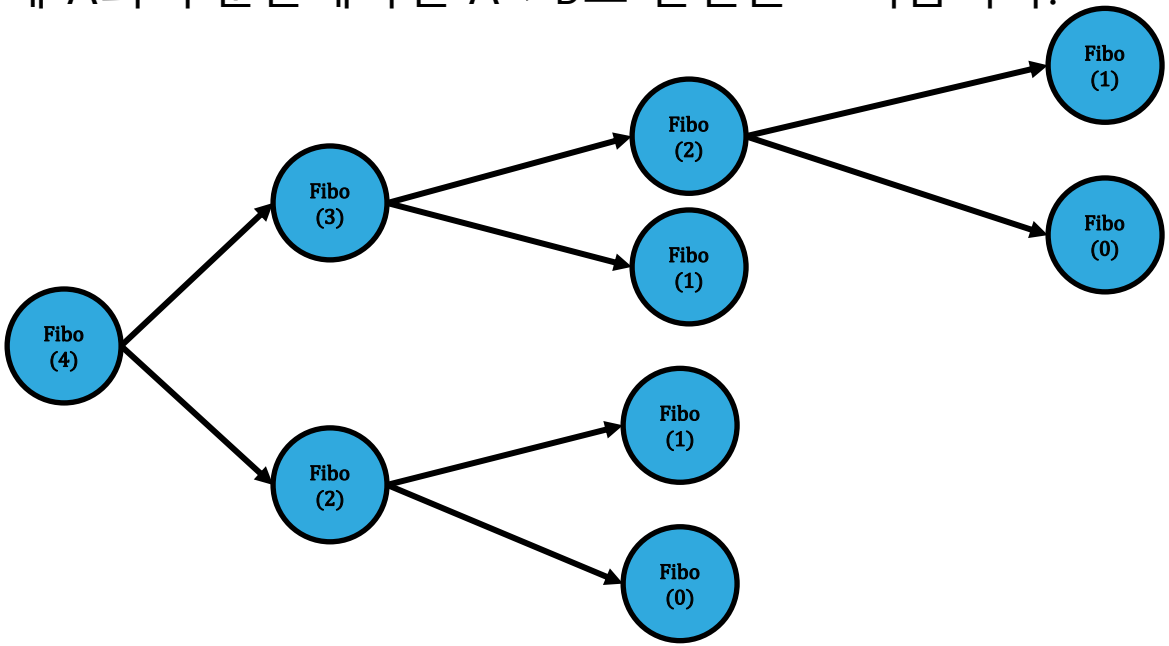
```
int Fibo(int n) {
    if (n == 0) return 1;
    if (n == 1) return 1;
    return Fibo(n-1) + Fibo(n-2);
}
```



DP와 DAG

문제들 간의 관계를 그래프로 표현해봅시다.

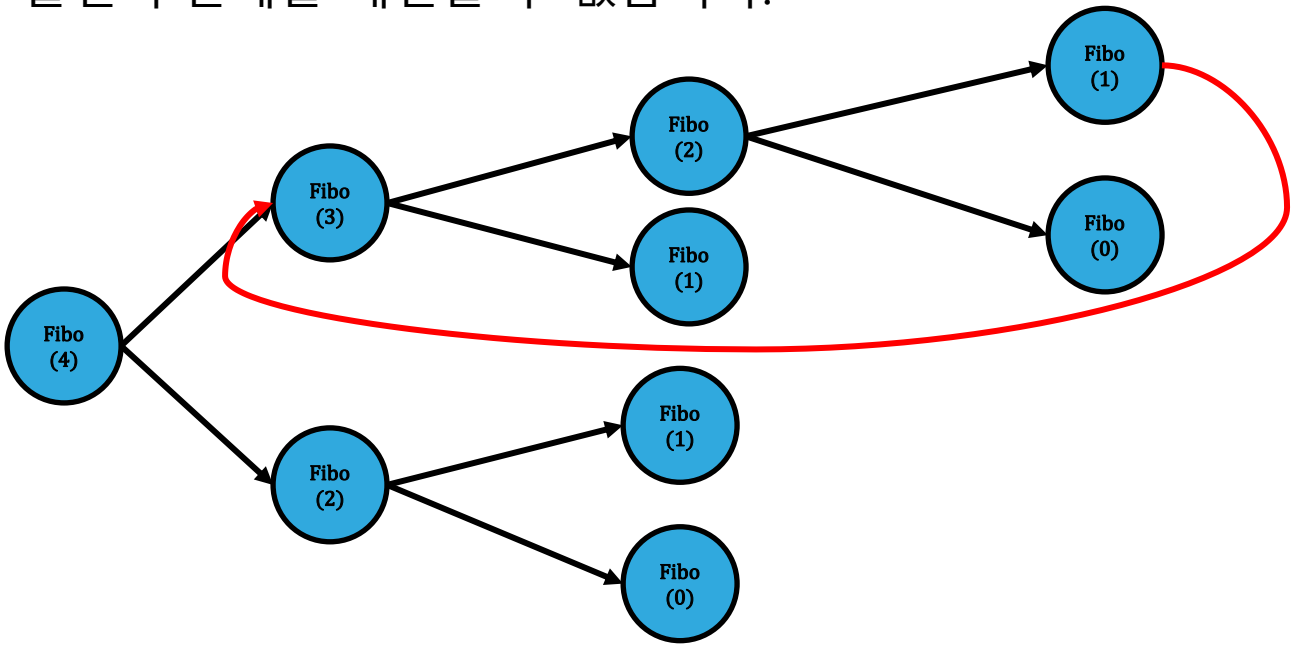
문제 B가 문제 A의 부분문제라면 A->B로 간선을 그어줍시다.



DP와 DAG

만약 이런 관계로 나타냈을 때 사이클이 있다면 어떻게 될까요?

무한 루프가 돌면서 문제를 해결할 수 없습니다.



상태 공간과 상태 전이

DP를 할 때 결국 원래 문제를 부분 문제로 잘 쪼개야 합니다.

이 때 쪼개기 위해서는 문제의 특성을 잘 잡아야 합니다.

우리가 문제를 나타내는 특성으로 선택했다면 그 특성만으로 문제가 온전히 정의되어야 합니다.

상태 공간과 상태 전이

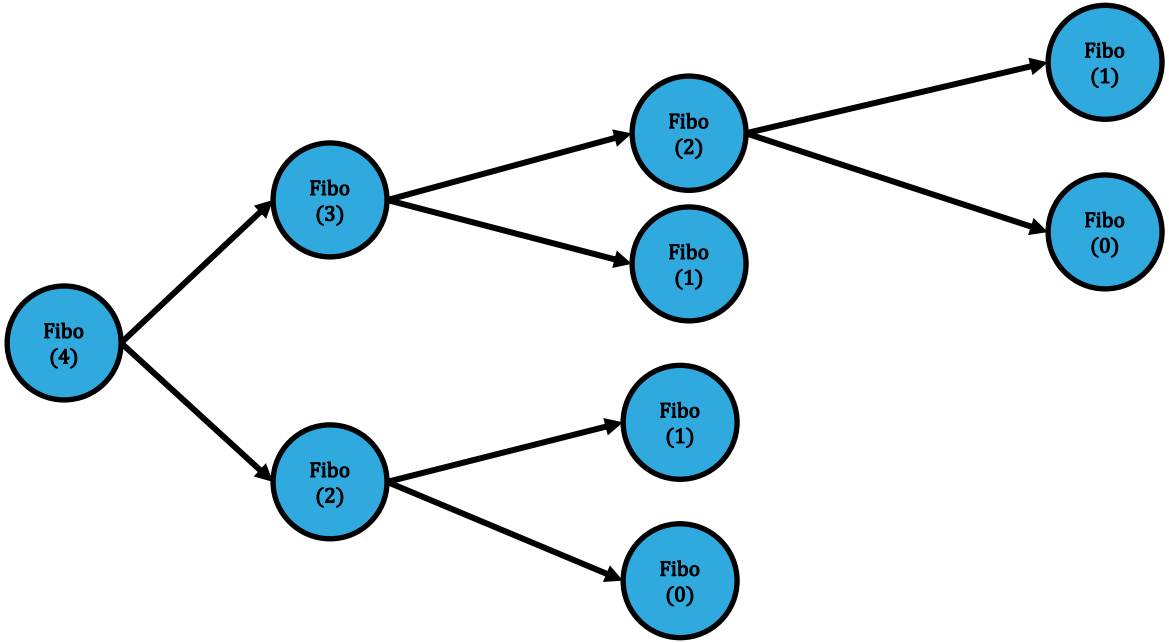
그리고 상태간의 전이 관계(A->B)가 잘 정의되어야 합니다.

이 때 잘 정의되어야 한다는 것은 상태를 정점, 전이를 간선으로 나타냈을 때 그 그래프가 DAG여야 한다는 것입니다.

DAG일 때 부분 문제를 해결할 때는 위상정렬 순으로 해결해야 합니다.

상태 공간과 상태 전이

아래와 같은 문제에서 위상정렬 상으로 뒤에 있는 문제들을 해결해야 그 전의 문제들에 대해서 답을 얻을 수 있습니다.



상태 공간과 상태 전이

문자열 A, B의 LCS에선 f(i, j)는 A의 i번째 문자까지, B의 j번째 문자까지 고려했을 때의 LCS로 정의했습니다.

이렇게 정의해도 되는 이유는 원래 문제의 온전한 부분 문제면서 i,j로 설정한 것이 원래 문제를 온전히 정의할 수 있는 특성이기 때문입니다.

상태 공간과 상태 전이

플로이드 워셜 알고리즘을 생각해봅시다.

DP를 이용하여 모든 쌍의 최단 거리를 구해주는 알고리즘입니다.

이 때 dp(i,j)를 단순히 i에서 j로 가는 최단거리라고 정의하면 어떻게 될까요?

상태 공간과 상태 전이

문제를 분해해봅시다. 최단 경로의 특성 상 dp(i,j)는 아래처럼 분리 가능합니다.

$$dp(i,j) = dp(i,k) + dp(k,j)$$

i에서 j로 가는 최단 경로 상에 있는 정점 k에 대해서 전체 경로의 길이는 i부터 k까지 가는 경로의 길이와 k부터 j까지 가능 경로의 길이 합입니다.

부분문제로 온전히 분해가 가능하고 잃는 정보 없이 문제를 잘 나타냅니다.

상태 공간과 상태 전이

이런 분해방식에서 문제는 우린 최단 경로를 모른다는 점입니다.

그럼 부분문제 간의 순서를 어떻게 정해야 할까요?

dp(i,j)의 부분 문제로 모든 k에 대해서 dp(i, k) + dp(k, j)를 지정하는 것도 한 방법이겠습니다.

이러면 잘 돌까요?

상태 공간과 상태 전이

문제의 상태 간의 관계로 그래프를 만들었을 때 DAG여야 DP로 문제가 해결이 가능하다고 했습니다.

모든 k에 부분문제로 설정하면 사이클이 생깁니다.

플로이드 워셜은 출발지 i와 도착지 j 외에도 상태를 나타내기 위한 변수 하나를 추가해서 이를 해결합니다.

상태 공간과 상태 전이

dp(i,j,k)는 i에서 j로 갈 때 k번 정점까지만 썼을 때의 최단 경로입니다.

부분 문제로 쪼개서 점화식으로 나타내면 아래와 같습니다.

$$dp(i, j, k) = \min(dp(i,j,k-1), dp(i, k, k-1) + dp(k, j, k-1))$$

이렇게 상태를 정의하면 이제 DP로 해결이 가능한 형태가 됩니다.

DP의 설계

DP로 문제를 풀 때 고려해야할 것은 상태를 나타낼 특성을 잘 고르는 것이 첫 째입니다.

잘못 고르면 상태로 만든 그래프가 DAG가 아닐 수도 있고 혹은 풀어야 할 문제를 온전히 표현 못할 수도 있습니다.
혹은 상태가 너무 많아서 시간 내에 계산을 못할 수도 있습니다.

둘째로 고려할 것은 그 다음은 부분문제들과의 관계와 부분문제들을 통해서 원래 문제를 잘 푸는 것이 가능하냐입니다.

백준 22358번 스키장

스키장 성공 출처

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초 (추가 시간 없음)	1024 MB (추가 메모리 없음)	218	107	95	54.286%

문제

당신은 친구들과 함께 스키를 타러 스키장에 왔다. 스키장에는 일정 고도마다 중간 지점이 설치되어 있다. 중간 지점은 총 N개 있으며, 고도가 감소하는 순서대로 1번부터 N번까지 번호가 매겨져 있다. 즉 가장 높은 지점이 1번 지점, 가장 낮은 지점이 N번 지점이다.

현재 당신은 S번 지점에 친구들과 함께 있다. 당신의 친구들은 각자 자유롭게 스키를 탄 이후, 끝나면 T번 지점에 모이기로 약속했다.

스키장에는 M개의 코스가 있다. 각 코스는 a_i번 지점에서 b_i번 지점 방향으로 이어지며, 코스에 진입하면 t_i시간 동안 스키를 탈 수 있다. 코스는 항상 고도가 감소하는 방향으로 이어진다. 즉, $a_i < b_i$를 만족한다.

또한, 각 코스에는 스키 리프트가 있다. 스키 리프트는 코스와는 반대 방향으로, 고도가 증가하는 방향으로 이어진다. 즉, 스키 리프트를 타면 b_i번 지점에서 a_i번 지점으로 이동할 수 있다. 스키 리프트는 최대 K번 탑승할 수 있다.

당신은 스키 코스와 리프트만을 사용해서 T번 지점까지 가되, 스키를 타는 시간을 최대화하려고 한다. 리프트를 타는 시간은 스키를 타는 시간에 포함되지 않는다. 코스의 정보가 주어질 때, 최대 몇 시간 동안 스키를 탈 수 있을지 구하여라.

입력

첫 번째 줄에 다섯 개의 정수 N, M, K, S, T ($1 \le N, M \le 10^5$, $0 \le K \le 10$, $1 \le S, T \le N$) 가 주어진다.

이후 M 개의 줄에 각 코스의 정보가 세 개의 정수 a_i, b_i, t_i ($1 \le a_i < b_i \le N$, $1 \le t_i \le 10^9$) 로 주어진다.

서로 다른 두 지점을 잇는 코스는 최대 하나이다.

백준 22358번 스키장

각 지점을 정점으로 생각해봅시다.

문제의 조건에서 항상 번호가 낮은 지점에서 높은 지점으로만 코스가 있다고 합니다.

이 조건은 코스를 간선으로 봤을 때 주어지는 그래프가 DAG라는 뜻입니다.

심지어 1, … , N이 위상정렬 순이 됩니다.

백준 22358번 스키장

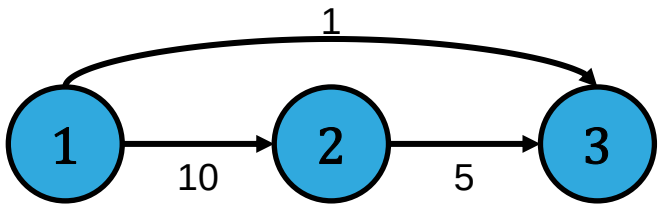
해당 문제는 DP로 해결이 가능합니다.

dp(u, k)를 S부터 u번 정점까지 가되 리프트를 k번 사용했을 때 스키를 탄 최대시간이라고 합시다.

백준 22358번 스키장

k가 0일 때부터 해결합시다.

0일 때의 문제는 S부터 T로 가는 최장경로를 찾는 것과 동일한 문제입니다.



백준 22358번 스키장

일반적인 그래프에서 최장경로는 NP-hard입니다.

그러나 DAG에선 DP로 해결이 가능합니다.

dp(u)를 u에서 끝나는 최장 경로의 길이라고 합시다.

u에서 v로 나가는 간선이 있는 정점 v에 대해서 아래처럼 점화식이 나옵니다.

$$dp(v) = \max(dp(u) + w(u, v))$$

백준 22358번 스키장

여기서 dp를 채울 때 위상 정렬 상으로 앞에 있는 정점들이 부분문제가 됩니다.

따라서 이 문제에서는 1부터 N까지의 순서로 dp 값들을 채우면 됩니다.

다만 시작점이 정해져 있기 때문에 초기에 dp[S]만 정의되어 있고 dp값이 정의가 되어 있지 않은 점은 계산에서 빼줘야 합니다.

```
memset(dp, -1, sizeof(dp));
dp[S] = 0;
for(int u=1;u<=N;++u) {
    if(dp[u] == -1) continue;
    for(auto p:G[u]) {
        int v = p.first;
        ll w = p.second;
        dp[v] = max(dp[v], dp[u] + w);
    }
}
```

백준 22358번 스키장

원래 문제로 돌아옵시다.

리프트를 최대 K번 탈 수 있습니다. 리프트는 코스가 있는 곳을 역으로 가는 것이기 때문에 간선 (u, v)가 있으면 v에서 u로 리프트를 탈 수 있습니다.

이를 통해 dp(u, k)를 리프트를 k번 타고 u에 도착 했을 때 최대 스키 시간이라고 합시다.

백준 22358번 스키장

dp(u, 0)은 DAG에서의 최장경로 문제기 때문에 값을 구할 수 있습니다.

이제 dp(u, k) 들을 가지고 dp(u, k+1)을 구할 수 있다면 문제를 해결할 수 있습니다.

리프트들로 이루어진 그래프는 정확히 역방향 간선들만 존재하는 그래프입니다.

백준 22358번 스키장

정점 u에서 리프트를 타고 갈 수 있는 정점들에 대해서 아래처럼 계산할 수 있습니다.

$$dp(v, k+1) = \max(dp(v, k+1), dp(u, k))$$

이 때도 채우는 순서는 위상정렬 순이어야 하며, 이는 N부터 1까지의 순서로 채우면 만족합니다.

백준 22358번 스키장

리프트를 타고 올라가서 다시 스키를 타는 과정은 dp(u, 0)을 채울 때의 과정을 다시 반복하면 됩니다.

$$dp(v, k) = \max(dp(v, k), dp(u, k) + w(u, v))$$

이걸로 문제를 해결가능합니다.

시간복잡도는 O(K(N+M))입니다.

백준 22358번 스키장

이 문제와 같이 그래프가 주어졌을 때 그래프 상에서 DP를 하는 문제도 꽤 많습니다.

그리고 DAG가 아닐 때 SCC로 정점들을 묶어주면 DAG로 만들어서 문제를 해결하기도 합니다.

백준 1086번 박성원

박성원 성공

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	128 MB	4735	1154	790	23.106%

문제

박성원은 이 문제를 풀지 못했다.

서로 다른 정수로 이루어진 집합이 있다. 이 집합의 순열을 합치면 큰 정수 하나를 만들 수 있다. 예를 들어, {5221,40,1,58,9}로 5221401589를 만들 수 있다. 합친수가 정수 K로 나누어 떨어지는 순열을 구하는 프로그램을 작성하시오.

하지만, 박성원은 이 문제를 풀지 못했다.

따라서 박성원은 그냥 랜덤하게 순열 하나를 정답이라고 출력하려고 한다. 이 문제에는 정답이 여러 개 있을 수도 있고, 박성원이 우연히 문제의 정답을 맞출 수도 있다.

박성원이 우연히 정답을 맞출 확률을 분수로 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 집합의 수의 개수 N이 주어진다. N은 15보다 작거나 같은 자연수이다. 둘째 줄부터 N개의 줄에는 집합에 포함된 수가 주어진다. 각 수의 길이는 길어야 50인 자연수이다. 마지막 줄에는 K가 주어진다. K는 100보다 작거나 같은 자연수이다.

출력

첫째 줄에 정답을 기약분수 형태로 출력한다. p/q꼴로 출력하며, p는 분자, q는 분모이다. 정답이 0인 경우는 0/1로, 1인 경우는 1/1로 출력한다.

백준 1086번 박성원

눈 여겨 볼 점은 주어지는 숫자가 최대 15개라는 점입니다.

그렇다고 모든 경우의 수인 15! 가지를 일일이 해보기엔 시간 초과를 받습니다.

여기서 경우의 수를 줄여보는 것을 목표로 합시다.

백준 1086번 박성원

순열을 정한 다음 계산하는게 아니라 수를 하나씩 붙인다고 생각해봅시다.

현재 만든 수가 x일 때, L자리수 y를 붙이면 결과는 다음과 같습니다.

$$xy_1y_2\cdots y_L = x \times 10^L + y$$

문제는 주어지는 수들이 최대 50자리라는 점입니다. 수를 온전히 저장하는 것은 힘듭니다.

백준 1086번 박성원

문제에서 원하는 점이 K로 나눈 나머지가 0일 경우입니다. 주어지는 수들의 K로 나눈 나머지로만 연산을 진행해도 됩니다.

$$xy_1y_2\cdots y_L \equiv (x \bmod K) \times (10^L \bmod K) + (y \bmod K) \pmod K$$

자리수가 너무 크다는 문제는 해결됐습니다.

백준 1086번 박성원

현재 수가 x일 때, 이를 나눈 나머지가 a라고 합시다.

그러면 x를 만드는 데 사용된 수를 제외한 수들 중에서 y를 골라서 뒤에 붙여줘야 합니다.

우리가 갖고 있는 수는 x를 K로 나눈 나머지 뿐입니다. y를 고르기엔 정보가 부족합니다.

백준 1086번 박성원

어떤 수를 사용해서 x를 만들었다고 기록을 하고 싶습니다.

이런 상황에 잘 쓰이는 것이 비트마스킹입니다.

수가 N개 있을 때 고르는 조합의 수는 2^N입니다.

이는 N자리 이진수가 나타낼 수 있는 경우의 수와 동일합니다.

백준 1086번 박성원

만일 i번째 수를 고른 상태라면 i번째 비트가 1을 가집니다.

i번째 수를 고르지 않았다면 i번째 비트는 0을 가집니다.

이런식으로 bit를 이용해서 정수를 set처럼 사용이 가능합니다.

백준 1086번 박성원

이제 현재 만든 수를 K로 나눈 나머지가 a이고 수를 만드는 데 사용된 수들을 기록해둔 것을 mask라고 합시다.

dp(mask, a)를 위와 같은 상태에서 남은 수들을 끝까지 붙였을 때 K로 나눈 나머지가 0인 경우의 수로 정의합시다.

이걸로 점화식을 세울 수 있습니다.

$$dp(mask, a) = \sum_{i \notin mask} dp(mask \cup i, a \times 10^L + a_i \bmod K)$$

모든 수를 사용하고 나머지가 0일 때는 dp값이 1이고 모든 수를 사용했지만 나머지가 0이 아니면 dp값은 0입니다.

백준 1086번 박성원

비트마스킹을 할 때 첫번째 수의 인덱스를 0이라고 합시다.

i번째 수가 mask에 포함되었는지 확인하려면 i번째 비트가 1인지 확인해야 합니다.

if(mask & (1<<i)) 로 확인이 가능합니다.

i번째 비트를 1로 만드는 것은 mask | (1<<i)로 가능합니다.

i번째 비트가 1일 때 0으로 만드는 것은 mask ^ (1<<i)로 가능합니다.

백준 1086번 박성원

현재 상태가 mask일 때 포함되지 않은 수를 확인하는 것은 아래 for문으로 가능합니다.

```
for(int i=0;i<N;++i) {
    if(!(bit & (1<<i))) {
```

이렇게 비트마스킹과 나머지를 이용해 DP를 하면 총 상태의 수는 $2^N K$개 있고 DP값을 계산하는 데에는 O(N)만큼 걸립니다.

따라서 시간 복잡도는 $O(2^N NK)$입니다.

백준 13283번 Daruma Otoshi

문제가 워낙 길어서 https://www.acmicpc.net/problem/13283을 읽어주세요.

요약하면 수가 N개 주어지는데 인접한 두 수의 차가 2보다 작다면 두 수를 없앨 수 있습니다.

없애면 없앤 두 수의 양 옆의 수가 인접합니다. 이런 과정을 반복했을 때 없앨 수 있는 최대의 수를 구하시오.

백준 13283번 Daruma Otoshi

배열에서 구간을 통해 상태를 정의해서 DP로 문제를 해결하는 경우도 많습니다.

dp(i,j)를 [i...j]에서 없앨 수 있는 최대 수로 정의해봅시다.

$$dp(i,j) = \max(dp(i,k) + dp(k+1,j))$$

로 점화식을 하나 세워볼 수 있습니다.

백준 13283번 Daruma Otoshi

한 가지 더 고려할 상황이 있습니다.

[i...j]에서 [i+1...j-1]을 전부 없앨 수 있고 i번째 수와 j번째 수의 차이가 2보다 작다면 사이의 수들을 전부 없애고 i번째 수와 j번째 수도 없앨 수 있습니다.

그러면 [i,j] 구간의 수들을 전부 없앨 수 있는 경우고 이 경우가 dp(i,j)의 최대가 됩니다.

백준 13283번 Daruma Otoshi

시간복잡도를 계산해봅시다. 모든 구간에 대해서 dp값을 계산한다고 생각합시다.

구간의 개수는 $O(N^2)$개이며 값을 하나 구하는 데에 $O(N)$이므로 $O(N^3)$입니다.

게임 DP

두 사람이 최적으로 게임을 할 때 승패를 결정하시오.

이런 문제도 꽤 자주 나오는 종류입니다.

여기서 최적으로 게임을 한다는 게 어떤 의미일까요?

배스킨라빈스 31

숫자를 1부터 세는데 한 번에 셀 수 있는 수는 최대 3개인 게임입니다.

승패는 31을 세는 사람이 지게 됩니다.

이 게임을 둘이서 할 때 이기는 방법은 항상 2 + (4의 배수)로 끝내게 하면 됩니다.

이는 항상 먼저 시작하는 사람이 이길 수 있다는 뜻입니다.

배스킨라빈스 31

2 + (4의 배수)로 끝내게 하면 이긴다고 했습니다.

이는 반대로 말하면 3 + (4의 배수)부터 말하게 되는 포지션의 사람이 항상 지게 된다는 뜻도 됩니다.

이런 식으로 어떤 게임은 어떤 상태가 항상 지게 되는 게임의 상태가 존재합니다.

게임 DP

이를 토대로 최적으로 게임을 한다는 것을 정의합시다.

만약 내가 할 수 있는 행동으로 상대방에 넘길 수 있는 게임 상황 중에 질 수 밖에 없는 상황이 있다면 무조건 그 선택을 한다는 것입니다.

만약 그런 포지션이 없다면 내가 있는 상황은 필패 포지션입니다.

게임 DP

따라서 게임의 승패를 결정하라는 뜻은 게임의 시작 포지션이 필승인지 필패인지를 판단하라는 뜻이 됩니다.

배스킨라빈스 31이라면 게임의 시작은 1부터 말하는 것이고 상대에게 넘겨줄 수 있는 상황은 2부터 말하는 상황, 3부터 말하는 상황, 4부터 말하는 상황입니다.

이 중에 3부터 말하는 상황은 필패 포지션이므로 선공이 이기는 것입니다.

백준 1519번 부분 문자열 게임

부분 문자열 뽑기 게임 성공

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
2 초	256 MB	96	54	49	56.977%

문제

게임 판에 어떤 자연수 N이 쓰여 있을 때, 두 명의 플레이어가 턴을 번갈아가면서 이 게임을 하려고 한다.

매 턴이 돌아올때마다, 플레이어는 현재 게임 판에 쓰여 있는 수의 진 부분 문자열인 양의 정수 M을 고를 수 있다. 그리고 나서 원래 수에서 M을 뺀다. 진 부분 문자열이란 자기 자신을 제외한 모든 연속된 부분 문자열을 말한다.

예를 들어, 현재 게임 판에 2309가 써있을 때, 플레이어는 2, 3, 9, 23, 30, 230, 309를 고를 수 있다. 2를 고르면, 현재 게임 판에 쓰여 있는 수는 2307이 되고, 3은 2306, ..., 309는 2000이 된다.

만약에 플레이어가 부분 문자열을 고를 수 없게되면, 게임에서 지게된다.

입력으로 현재 게임 판에 쓰여 있는 수 N이 주어졌을 때, 플레이어 1(첫 턴을 가지는 플레이어)이 이기기 위해서 골라야 하는 수를 출력하는 프로그램을 작성하시오. 만약 여러 가지 경우가 있다면, 가장 작은 것을 출력하고, 이길 수 없다면 -1을 출력한다.

입력

첫째 줄에 N이 주어진다. N은 1,000,000보다 작거나 같은 자연수이다.

백준 1519번 부분 문자열 게임

숫자 X가 주어졌을 때 할 수 있는 행동은 X의 부분문자열에 해당하는 수를 고르고 X에서 그 수를 빼준 다음에 상대방에게 넘기는 것입니다.

이 때 X는 부분 문자열이 될 수 없습니다.

플레이어에게 주어진 X에서 부분문자열을 고를 수 없다면 지게 됩니다.

백준 1519번 부분 문자열 게임

239가 처음에 주어졌다고 합시다.

고를 수 있는 부분 문자열은 2, 3, 9, 23, 39입니다.

각 문자열을 골랐을 때 상대에게 넘겨줄 수 있는 수는 237, 236, 230, 216, 200입니다.

이런 수들 중에서 필패 포지션이 하나라도 있다면 이길 수 있습니다.

백준 1519번 부분 문자열 게임

숫자 N이 주어지면 모든 경우의 수에 대해서 해볼 수도 있습니다.

239가 주어지면 2를 빼서 237을 만들고 또 2를 빼서 235를 만들고 쭉 빼다가 9가 되면 지는 거죠.

다시 돌아와서 다른 수를 빼고 이런 식으로 하기엔 너무 많습니다.

백준 1519번 부분 문자열 게임

하지만, 수 X가 주어졌을 때 내가 필승일지 아닐지는 X가 어떤 수인지에만 의존합니다.

Memoizaition을 이용할 수 있는 것이죠.

백준 1519번 부분 문자열 게임

다행히도 주어지는 N은 최대 100만입니다.

그리고 플레이어가 어떤 행동을 한다면 넘겨주는 수는 무조건 작아집니다.

게임을 진행하면서 수는 작아지고 언젠가는 끝난다는 뜻이고 게임에서 있을 수 있는 상태수는 최대 100만 개입니다.

백준 1519번 부분 문자열 게임

X의 자릿수가 L이라고 합시다. 그러면 나올 수 있는 부분 문자열은 $O(L^2)$개가 있고 이러한 부분 문자열 중에서 필패포지션을 만들 수 있는 것이 하나라도 있다면 X는 필승 포지션입니다.

dp(X)를 X가 필승인지 필패인지를 나타내는 값으로 생각합시다. 필승이라면 True, 필패라면 False 입니다.

백준 1519번 부분 문자열 게임

점화식은 아래와 같습니다.

$$dp(X) = \bigvee_{x\text{는 }X\text{의 부분문자열}} \sim dp(X - x)$$

X가 한자릿수라면 dp(X)는 False입니다.

제일 긴 자릿수가 L이라고 했을 때 시간복잡도는 $O(NL^2)$이 됩니다.

Problem Set

- 필수 문제
  - 22358 스키장
  - 1086 박성원
  - 13283 Daruma Otoshi
  - 1519 부분 문자열 게임

“Any Question?”