2017 KAIST RUN Spring Contest with HYEA Cup Solution (한국어)

HYEA (ho94949, hyea@kaist.ac.kr)

Sponsored By:

Startlink

Naver D^2

1 Card Game Contest

곱의 정리를 생각합시다. 어떤 일을 하는데 a가지가 가능하고, 다른 일을 하는데 b가지가 가능하다면, 총 가능한 가짓수는 $a \times b$가지 입니다. 답은 $max(A_i, 1)$을 모두 곱한 값입니다.

$M = 1$인 경우를, 특히 $N = 0$인 경우를 잘 생각해야 합니다. $N = 0$, $M = 1$인 경우에 답은 0입니다.

주석: $N = 0, M = 1$인 경우는 의도된 케이스이고, $M = 1$을 가지고 나머지 연산을 할 때는 조심하도록 합시다.

2 Mahjong

특정 패의 종류가 4장 이하 있다면, 대기패 일수 있습니다. 각각의 패를 추가해서 전체 14장의 패가 완성되었는지 확인하면 됩니다.

머리가 7개인 경우는 예외로 두고, 몸통이 4개이고 머리가 1개인 경우를 생각해 봅시다.

가능한 방법중 하나는 모든 $\binom{14}{2\ 3\ 3\ 3\ 3}$개의 조합을 생각해서, 538137600개의 가짓수를 각각 대기마다 확인하는 것입니다. 이 방법을, 같은 타일은 바꿔도 문제가 없다. 와 모든 몸통의 수의 합은 3으로 나누어 떨어진다. 라는 점을 이용해서 개선할 수 있습니다.

정식 해법에 쓰인 다른 방법은, 몸통 4개를 모두 확인하는 것입니다. 몸통은 16가지 밖에 없다는 사실을 이용해서 머리를 하나 정하고 몸통을 4개 정합니다. 이것은 최대 $9 \times 9 \times 16^4 = 5308416$가지만 생각하면 됩니다. 여기서 더 줄일 수도 있습니다.

아니면, 대기패를 직접 계산하는 방법으로 계산할 수 있습니다. 5가지의 대기 방법이 있습니다. (머리를 완성시키는 대기; 머리 2개가 있고, 둘 중 하나가 몸통이 되는 대기; 연속된 두 수가 있을때 양쪽을 기다리는 대기; 1, 2나 8, 9가 있을때 3, 7을 기다리는 대기; 2 차이나는 수가 있을 때 가운데를 완성시키는 대기.)

주석: 원래는 마작에 어떤 특정 패를 줬을때 점수 계산도 하라는 문제를 내려고 했지만... 헤아컵은 IPSC가 아니기 때문에 관뒀습니다. (2016 M - Mana troubles, 랜드를 탭해서 정해진 마나를 뽑을 수 있는지에 대한 문제.)

3 The Way

왼쪽으로 가는 것은, 세 행중 가운데 행에서만 가능합니다. S모양이나 ㄹ모양의 길을 생각해 보고, 파티션을 나누어 다이나믹 프로그래밍을 계산하면 됩니다.

좀 더 자세하게는, 모든 경로는, S모양이나 ㄹ모양의 길들이 있는 파트로 나눌 수 있고, 그것은 유일합니다. 그래서 $D_(i, j)$의 배열을 (i, j) 격자에서 끝나고 앞으로는 더 이상 뒤로 가지 않는 것들의 배열의 느낌으로 계산하면 됩니다.

주석: 처음에 $N = 10^6$이었지만, 문제 크기를 줄였습니다. 이 문제는 행렬 계산을 통해 $O(logN)$시간에도 해결 할 수 있습니다.

4 The Other Way

S에서 모든 마을까지의 최단경로를 다익스트라 알고리즘으로 계산합시다. 그리고, 최단경로에 사용된 배열들은 DAG를 이룰 것이고, 여기서 동적계획법을 사용하면 됩니다.

사실, 그래프를 명시적으로 만들기 보다는, 두 도시의 차이가 도로의 길이인지 여부를 계산하면 됩니다.

주석: SPFA의 반례 데이터를 넣었기 때문에 잘 동작하지 않을것입니다. 만들어준 alex9801에게 감사합니다.

5 Just as Tic Tac Toe

처음에, 모든 바구니의 Grundy Number를 계산할 수 있습니다. ($M+1$로 나눈 나머지) Grundy Number의 신기한 성질은 두 게임을 동시에 진행할 때, 새로운 게임의 Grundy Number는 두 수의 xor이라는 점이죠. 그래서 문제는 3XOR문제로 바뀝니다. 이것은 a xor $b = c$를 빨리 계산하는 문제고, 모든 n개의 비트에 대해 푸리에 변환을 사용해서 n차원 푸리에 변환을 계산하면 됩니다. 이것을 Hadamard 변환 이라고 부릅니다.

Fast Walsh-Hadamard Transform 행렬은 다음과 같은 모양입니다.:

$$H_N = \frac{1}{\sqrt{2}} \begin{pmatrix} H_{N-1} & H_{N-1} \\ H_{N-1} & -H_{N-1} \end{pmatrix}$$

주석: 원래는, 여러개의 성질이 있는 Set게임을 내려고 했습니다. 그러면 식이 더 복잡해졌지만, 그 문제만 풀 수 있을것 같아서 문제를 좀 더 쉽게 만들었습니다.

6 Balance

문제에 대한 답을 $A(N)$이라고 합시다. 마지막 단계에서는 N개의 추 중 $N-1$개의 가벼운 추 들은 아무쪽에나, 무게 2^N의 추는 오른쪽에만 놓을 수 있습니다. 총 $2N-1$가지 경우의 수가 있기 때문에, $A(N) = (2N-1)A(N-1)$ 입니다. (나머지 추에 대해서는 어차피 무거운 추의 무게의 합은 가벼운 추를 넘지 않습니다.)

그래서 답은 $1 \times 3 \times \cdots \times (2N-1)$ 입니다.

주석: $O(N^2)$의 동적 계획법 풀이도 있습니다. 잘 최적화를 해도 TLE가 나지만, 이것을 시간복잡도 측면에서 최적화 하는 등의 다른 여러 방법을 열어놓기 위해 $N = 50000$으로 정했습니다.

7 Memory

가장 적은 행동 수는 $RC/2$입니다. 왜냐하면 모든 카드가 적어도 한번은 뒤집어져야 하기 때문입니다. 다음 방법으로 뒤집는것을 생각합시다. 처음에, 뒤집은 두 카드의 모양이 A, B고, 두번째에 뒤집은 첫 카드 모양이 C고 두번째는 A였습니다. 그래서 A두개를 제거했고, 모양 A, C가 남았습니다. 이것은 최악의 경우중 하나입니다. 이 경우에, 두 카드를 없애는데 두 스텝이 걸리고, 이것을 반복하면 (4개일 경우 3번 뒤집어야 한다는 점을 이용해서) 답이 $RC-1$입니다. 이것이 최적인것은, 어떤 경우에도 $RC-1$번 을 넘지 않게 뒤집는 방법이 있습니다. $RC-2$개의 카드를 $RC/2-1$ 행동을 하여 뒤집고, 나머지 두 카드중 하나를 뒤집어서 맞추고, 다른 카드를 뒤집어서 맞추고, 나머지 $RC-4$개의 카드에 대해서는 짝이 맞는걸 찾아가면 됩니다.

주석: 이 문제는, $D_{i,j}$를, i개의 카드가 존재하고 j개의 서로 다른 카드 정보를 알 때 최대 뒤집는 횟수라는 동적계획법 문제로 변활 할 수 있습니다.

8 Too Many Traps

답은, 이불이 보물에 있는 곳을 제외하면 항상 가능합니다. 이 증명에 대해서 수학적 귀납법 풀이가 Constructive 하게 존재하고, 이것을 구현하는것이 문제에 대한 답입니다. (이불의 수는 보물의 수 보다 "작아야" 합니다.)

일반성을 잃지 않고, $A_1 < A_2 < \cdots < A_N$ 라고 하고, 수학적 귀납법을 사용합시다. $N = 0, 1$인 경우에 이불이 존재하지 않습니다.

1. A_N 에 이불이 위치 해 있고, $A_1, \cdots, A_{N-1}$중 어디에도 이불이 위치하지 않은 경우:

   A_N으로 점프를 시작해서, 나머지에 대해 문제를 풀고, 앞의 첫 두 점프를 바꾸면 A_N 위치에 있는 이불도 밟지 않을 수 있습니다.

2. A_N 에 이불이 위치해 있지 않고. $A_1, \cdots, A_{N-1}$ 중 한 곳에 이불이 위치한 경우:

   A_N으로 점프를 시작해서, 나머지에 대해 문제를 풀고, 끝입니다. A_N이전에 이불 하나를 이미 밟지 않아서, 고려할 필요가 없습니다.

3. A_N 에 이불이 위치 해 있고, $A_1, \cdots, A_{N-1}$ 중 한 곳에 이불이 위치한 경우:

   $N-1$개의 쌍 $(A_1, A_1+A_N), \cdots, (A_{N-1}, A_{N-1}+A_N)$을 고려합시다. 비둘기집의 원리에 의해, A_N을 제외한 $N-2$개의 이불이 있고, 적어도 한 쌍은 둘 다 이불이 위치해 있지 않습니다. 그래서 그 i에 대해 A_i, A_N으로 점프를 하면 두개의 이불을 회피 할 수 있고, 나머지 $N-2$개의 수에 대해 문제를 해결하면 됩니다.

4. $A_1, \cdots, A_N$에 모두 이불이 위치하지 않은 경

   A_N으로 시작해서 $A_1, \cdots, A_{N-1}$ 점프를 하는 대신, 가장 가까운 이불에 대해서는 신경을 쓰지 맙시다. 만약에 가장 가까운 이불 위로 점프를 하게 된다면, 그 다음 점프와 A_N을 바꿉시다. 그러면, A_N 이전의 모든 점프는 가장 가까운 이불이라는 점 때문에, 이불 위로 착지하지 않을 거고, A_N점프는 합이 유지되기 때문에, 그리고 앞으로는 기존 점프에 대한 가정 때문에 다른 이불에 착지하지 않습니다.

이걸 적절히 구현하면 됩니다. 가장 시간이 많이 걸리는 부분은 3번째 부분이고 시간복잡도가 $O(NlogN)$ 이겠지만, 평균 시간 복잡도는 $O(logN)$에 구현할 수 있다는 것을 증명 할 수 있습니다.

그래서 이것을 구현하면 시간 복잡도가 $O(N^2)$ 이 나옵니다.

이것을 개선해 봅시다. 두 부분을 개선해야 합니다. 한 부분은 $A_2, \cdots, A_N$ 이 이불이 위치하는지 하고, 4번째 경우에 대해서는 이불 위로 착지하는 경우가 있는지를 확인하는 것입니다.

첫번째는, 처음 몇개의 이불 위치만 확인하는것으로 개선 할 수 있습니다. 증명에서 가장 중요한 부분은, 이불 위에 착지하는지에 여부가 아니라 이불을 스킵해서 앞으로 남은 이불의 갯수를 줄이는 것이기 때문입니다.

두번째는, BIT같은 누적합을 구하고, 갱신을 할 수 있는 자료구조를 사용해서 (swap 하는 부분 때문에 갱신이 필요합니다.), 첫번째 이불에 착지하는 지 여부를 이진 탐색을 하면 됩니다.

이것을 구현하면 $O(Nlog^2N)$시간 복잡도가 나올것이고, 정해가 이 방법으로 구현 되었습니다. 두번째 부분을 더 향상시켜서 $O(NlogN)$으로 풀 수 있습니다.

주석: 처음 해법은 $O(N^2)$ 이고, zlzmsrhak이 $O(NlogN)$으로 시간을 줄이는 법에 대한 아이디어를 줬습니다. $N = 100000$인 이유는, 가장 먼 점프를 아껴두는 등의 랜덤 셔플 알고리즘을 막기 위해서 입니다. (가장 긴 점프는 이 문제에서 굉장히 핵심적입니다.) 이 문제는 대회에서 가장 어려운 문제 입니다. 풀었다면 자랑스러워 해도 좋습니다.

9 Protocol

$112345^{167} \equiv 1 (\text{mod } 1000000009)$.

그래서, 선 용량에 대해 167으로 나눈 나머지만 고려하면 됩니다. 행렬 곱셈과 고속으로 지수를 계산하는 알고리즘을 쓰면 됩니다.

좀 더 정확하게는, $A_{(ax^2+bx+c,x)} = 1$ and $A_{(dx^2+ex+f,x)} = 1$ (두 수가 같으면 2) 이고, 나머지가 0인 행렬을 사용하고, A^M은 M년 후에 선 용량의 변화를 나타냅니다.

주석: 대회를 준비한 이유 중 하나입니다. 상수 112345는 멋지거든요!

10 Communism

배열을 절반으로 나누어서 $\pm(A_1 \text{ or } 0) \pm (A_2 \text{ or } 0) \pm \cdots$ 을 계산합시다. 두개의 배열을 병합정렬과 비슷한 아이디어로 정렬할 수 있습니다. 그냥 모든 원소를 넣어서 정렬하는 방법은 $O(N \times 3^{(N/2)})$으로 시간 초과가 날 것입니다. 만약 배열의 한 원소가 골라졌다면, 다른 배열에서 D차이 이하로 나기 위한 구간을 계산합시다. Sliding Window / 2 Pointer 방법을 사용할 수 있고, 총 시간 복잡도는 $O(3^{(N/2)})$입니다.

주석: $O(N \times 3^{(N/2)})$으로 시간을 최대한 줄여서 구현한 풀이에 대한 시간은 1.2초 였습니다. 좀 더 확장성 있게, C++ STL을 사용한 코딩 방법은 0.6초 였습니다. $O(3^{(N/2)})$에 돌지 않는다고 생각할 수 있지만 컴퓨터는 빠르더라고요!

11 Thanks

대회를 위해 준비한 모두에게 감사드립니다.

문제에 이름을 사용할 수 있게 해주신 메이지, 리샤, 알렉스, 여우, 토끼, 키르, 리유나, 모래두지, 혜아, 카흐, 린카루, 아드 그리고 래리님 감사합니다.

문제 셋을 준비하고 작업한 제 자신에게 감사합니다.

문제들을 풀고, 문제 디스크립션과 Mahjong, The Other Way, Balance, Memory 문제에 대해 조언을 주신 alex9801님 감사합니다.

RUN@KAIST의 회장으로서, 스폰서와 말하고, 제 이름을 걸고 문제를 낼 수 있게 해주신 wwiiiii님 감사합니다.

문제들을 풀고, 다양한 코너케이스를 처리할 수 있는 틀린 풀이들과, Communism 문제에 대해 조언을 주신 koosaga님 감사합니다.

문제들을 풀고, Too Many Traps문제에 대해서 개선된 백트래킹 알고리즘을 제시하고, 문제 난이도에 대한 평가를 주신 functionx님 감사합니다.

문제들을 풀고, Too Many Traps문제의 개선 아이디어를 주고, Card Game Contest, Just as Tic Tac Toe, Balance, Too Many Traps에 대해 조언을 주시고 문제 난이도에 대한 평가를 해주신 zlzmsrhak님 감사합니다.

대회를 진행할 수 있게 대회 운영비와 상품비를 후원 해 주신 Naver D^2, 감사합니다.

acmicpc.net 온라인 저지 서비스를 대회에 이용할 수 있게 도와주신 최백준님과 Startlink, 감사합니다.

그리고 누구보다 이 대회에 참여해 주시고, 이 대회 솔루션을 보고 계신 당신에게도 감사드립니다.