제2회 곰곰컵 해설

2022년 11월 26일

문제		의도한 난이도	출제자
A	치킨댄스를 추는 곰곰이를 본 임스 2	**Easy**	lms0806
B	붙임성 좋은 총총이	**Easy**	pjshwa
C	곰곰이와 학식	**Medium**	jyheo98
D	오락실에 간 총총이	**Medium**	bnb2011
E	곰곰이와 시소	**Medium**	swoon
F	외로운 곰곰이는 친구가 있어요	**Medium**	r4pidstart

문제		의도한 난이도	출제자
G	곰곰이와 테트리스	**Hard**	jyheo98
H	곰곰아 선 넘지마	**Hard**	pichulia
I	곰곰이의 식단 관리 2	**Hard**	pjshwa
J	서커스 나이트	**Hard**	pjshwa
K	곰곰이와 토너먼트	**Hard**	jyheo98
L	곰곰이의 벼락치기	**Challenging**	chansol
M	곰곰이와 하카타	**Challenging**	chogahui05
N	곰곰이와 GGANALi	**Challenging**	pichulia

A. 치킨댄스를 추는 곰곰이를 본 임스 2

`implementation`
출제진 의도 – **Easy**

- 제출 743번, 정답 540명 (정답률 74.43%)
- 처음 푼 사람: **riroan**, 0분
- 출제자: `lms0806`

A. 치킨댄스를 추는 곰곰이를 본 임스 2

- 저번 제1회 곰곰컵의 “치킨댄스를 추는 곰곰이를 본 임스”와 이어지는 문제입니다.
- 치킨댄스를 추는 곰곰이를 본 임스가, 곰곰이한테 받은 기프티콘이 생각나 사용하고자 하는 내용으로 담았습니다.
- 문자열에서 “D-”를 떼어낸 후 남는 수가 90 이하인 경우가 얼마나 되는지 체크하면 됩니다.

B. 붙임성 좋은 총총이

`string`, `set`
출제진 의도 – **Easy**

- 제출 539번, 정답 345명 (정답률 66.42%)
- 처음 푼 사람: **kyo20111**, 1분
- 출제자: `pjshwa`

B. 붙임성 좋은 총총이

- 지난번 대회의 상징적인 문제 “인사성 밝은 곰곰이” 를 리메이크 해 보았습니다.
- 무지개 댄스를 추는 사람들의 이름을 담고 있는 문자열 set을 관리합시다. 이 set에는 최초에 “ChongChong”만 들어 있습니다.
- 이후 N 개의 기록을 순회하면서, 두 사람 중 적어도 한 사람의 이름이 set에 이미 포함되어 있었다면 둘의 이름을 모두 set에 넣는 방법으로 구현합니다. 이때 답은 마지막 기록을 처리한 뒤 set의 size입니다.

C. 곰곰이와 학식

`simulation`, `greedy`
출제진 의도 – **Medium**

- 제출 613번, 정답 191명 (정답률 32.30%)
- 처음 푼 사람: **xiaowuc1**, 3분
- 출제자: `jyheo98`

C. 곰곰이와 학식

- 어떤 음식을 먹고 싶은 곰곰이가 있는데 식권을 사용하지 않고 다음 식권으로 교환하는 것은 손해입니다.
- 따라서 현재 상태에서 식권으로 음식을 최대한 교환한 후, 남은 식권을 다음 식권으로 교환하는 것이 최적의 전략입니다.
- 잘 생각해보면, 위 시뮬레이션을 최대 3회만 반복하면 정답을 얻을 수 있음을 알 수 있습니다.
- 시간복잡도는 $\mathcal{O}(1)$ 입니다.

D. 오락실에 간 총총이

`case_work`
출제진 의도 – **Medium**

- 제출 579번, 정답 156명 (정답률 29.188%)
- 처음 푼 사람: **kyo20111**, 7분
- 출제자: bnb2011

D. 오락실에 간 총총이

- ✓ 총총이는 과연 곰곰 피규어를 얻을 수 있었을까요?
- ✓ 크게 세 가지 경우를 고려해줘야 합니다.
  1. 곰곰이가 한 마리 있는 경우
  2. 모든 곰곰이가 가로 또는 세로 일직선 상에 있는 경우
  3. 그 외의 경우

D. 오락실에 간 총총이

- 곰곰이가 한 마리 있는 경우는 처음부터 모든 곰곰이가 하나의 칸에 있기 때문에 버튼을 누를 필요가 없습니다.
- 따라서 0을 출력해야 합니다.

D. 오락실에 간 총총이

- 곰곰이가 가로 또는 세로 일직선 상에 있는 경우에는 어떻게 해야 할까요?
- 편의상 모든 곰곰이가 가로 일직선 상에 있다고 합시다.
- 우선 모든 곰곰이의 y 좌표가 같으므로, 수직 방향의 버튼은 굳이 누를 필요가 없습니다.
- 이제 왼쪽 또는 오른쪽 버튼을 눌러야 하는데, 두 개의 버튼을 모두 누르는 것은 동선 낭비임을 어렵지 않게 알 수 있습니다.
- 따라서 모든 곰곰이가 왼쪽 또는 오른쪽 끝에 모일 때까지 두 버튼 중 하나를 계속해서 눌러주는 것이 최적 전략입니다.

D. 오락실에 간 총총이

- 곰곰이가 있는 위치의 x 좌표의 집합을 X 라 했을 때, 왼쪽과 오른쪽 버튼을 눌러야 하는 횟수는 각각 $max(X) - 1, N - min(X)$ 입니다. 버튼을 적게 눌러야 하니 둘 중 작은 값이 정답이 됩니다.
- 세로 일직선 상에 있는 경우도 동일하게 처리하면 됩니다.

D. 오락실에 간 총총이

- 그 외의 경우에는 어떻게 해야 할까요?
- 2번 경우를 x 좌표에 대해 한 번, y 좌표에 대해 한 번 적용해주면 됩니다.
- 또는 모든 곰곰이를 모을 수 있는 칸이 결국엔 게임판의 네 구석 중 하나밖에 없으므로, 해당 칸과의 맨해튼 거리를 구하는 식의 접근도 가능합니다.
- 구현은 unique한 x 좌표와 y 좌표의 개수를 세두면 편하게 할 수 있습니다.

E. 곰곰이와 시소

`math, parametric_search`
출제진 의도 – **Medium**

- 제출 284번, 정답 136명 (정답률 53.16%)
- 처음 푼 사람: **rustiebeats**, 8분
- 출제자: swoon

E. 곰곰이와 시소

- X 에 대한 매개 변수 탐색이 가장 직관적인 풀이일 것으로 생각됩니다.
- 이 해설에서는 식정리를 이용한 풀이를 소개해 보겠습니다.

E. 곰곰이와 시소

- 문제 노트에 언급된 $W_L = W_R$ 을 만족시키는 X 를 X_{ans} 라고 합시다. W_L 과 W_R 각각은
- $$W_L = \sum_{1 \le i \le N,\ x_i \le X_{ans}} w_i(X_{ans} - x_i)$$
- $$W_R = \sum_{1 \le i \le N,\ x_i > X_{ans}} w_i(x_i - X_{ans}) = - \sum_{1 \le i \le N,\ x_i > X_{ans}} w_i(X_{ans} - x_i)$$
- 이 됩니다. (수식 전개의 편의를 위해 $x_i = X_{ans}$ 인 경우에는 W_L 으로 포함시켰습니다.)
- 이제 $W_L = W_R$ 수식을 예쁘게 정리해봅시다.

E. 곰곰이와 시소

- $$\sum_{1 \le i \le N,\ x_i \le X_{ans}} w_i(X_{ans} - x_i) + \sum_{1 \le i \le N,\ x_i > X_{ans}} w_i(X_{ans} - x_i) = 0$$
- $$\sum_{1 \le i \le N} w_i(X_{ans} - x_i) = 0$$
- $$\sum_{1 \le i \le N} w_i X_{ans} = \sum_{1 \le i \le N} w_i x_i$$
- X_{ans} 의 값은 i 에 의존적이지 않기 때문에, $X_{ans} = \frac{\sum_{1 \le i \le N} w_i x_i}{\sum_{1 \le i \le N} w_i}$ 입니다.

F. 외로운 곰곰이는 친구가 있어요

`math`, `euclidean`, `number_theory`
출제진 의도 – **Medium**

- 제출 347번, 정답 73명 (정답률 24.78%)
- 처음 푼 사람: **xiaowuc1**, 13분
- 출제자: `r4pidstart`

F. 외로운 곰곰이는 친구가 있어요

- ‘베주 항등식’을 아십니까?
- 만약 0이 아닌 정수 a, b가 있고 이 둘의 최대공약수가 d일때 항상 $ax + by = d$를 만족하는 정수 x, y가 존재하며, d는 이런 형태로 표현할 수 있는 가장 작은 자연수라는 명제가 성립합니다.
- 이 항등식을 이용하면, 이 문제에서는, A_{ij} 들의 최대공약수가 친구들이 이동할 수 있는 최소 단위라는 것을 알 수 있습니다.
- 그리고, 이 문제에서는 별 다른 제약이 없기 때문에, x 축과 y 축을 따로 생각해봐도 괜찮습니다.
- 그러므로 각 i번째 친구는 $\gcd_{j=1}^{K_i}(A_{ij})|X_i$ 와 $\gcd_{j=1}^{K_i}(A_{ij})|Y_i$ 를 모두 만족한다면 곰곰이의 집 좌표인 $(0, 0)$에 정확히 도착할 수 있습니다.

G. 곰곰이와 테트리스

`ad_hoc`, `game_theory`
출제진 의도 – **Hard**

- 제출 60번, 정답 20명 (정답률 35.00%)
- 처음 푼 사람: **alex9801**, 9분
- 출제자: `jyheo98`

G. 곰곰이와 테트리스

- 주어진 제한 $1 \le N, M \le 20$은 함정입니다. 실제로는 N, M이 매우 커도 풀 수 있습니다.
- $1 \times 2, 2 \times 1$ 게임판은 자명하게 곰곰이 $P_8 - 0.5$점, 총총이 P_8점으로 총총이가 승리합니다.
- 이외의 경우에는 곰곰이가 언제나 이길 수 있습니다.
  - N, M이 모두 홀수이면 곰곰이는 가운데에 ■ 블록을 놓습니다.
  - N, M이 모두 짝수이면 곰곰이는 가운데에 블록을 놓습니다.
  - N, M 중 하나만 짝수이면
    - N, M 중 하나가 1이라면 곰곰이는 가운데에 블록을 놓습니다.
    - 그렇지 않다면 가운데에  블록이나  블록을 적절히 놓습니다.
- 이후 곰곰이는 총총이가 놓는 블록을 점대칭하여 그대로 놓기만 하면, 언제나 0.5점 이상 앞서 게임을 승리할 수 있습니다.

H. 곰곰아 선 넘지마

`greedy`, `ad_hoc`
출제진 의도 – **Hard**

- 제출 118번, 정답 72명 (정답률 62.71%)
- 처음 푼 사람: **jhnah917**, 17분
- 출제자: `pichulia`

H. 곰곰아 선 넘지마

- 몇 가지 관찰이 필요한 문제입니다.
- 우선 **S** 를 고정시킨 채로 **T** 에서만 연산을 수행하는 상황을 상상해봅시다.
- **S** 의 가장 왼쪽에 위치한 숫자 1 자리에 **T** 도 숫자 1 이 위치되어야 합니다.
- 해당 위치로 1 을 이동시킬 때, 가장 작은 연산횟수가 필요한 **T** 에서의 숫자 1 은, 가장 왼쪽에 위치한 숫자 1 이 될 것입니다.
- 마찬가지로 **S** 의 두 번째에 위치한 숫자 1 도 **T** 의 두 번째에 위치한 숫자 1 을 이동시킬 때 비용이 최소가 됩니다.

H. 곰곰아 선 넘지마

- 이를 일반화하면, **S** 를 고정시켰을 때 필요한 연산 횟수 최소값 Z 는 대략 아래와 같은 식으로 계산이 됩니다.
- $Z = \sum |(\text{S 에서 } \texttt{i} \text{ 번째 숫자 } \texttt{1} \text{ 의 위치}) - (\text{T 에서 } \texttt{i} \text{ 번째 숫자 } \texttt{1} \text{ 의 위치})|$
- 위 계산 결과 값은 숫자 1 대신 숫자 0 에 대해서 계산해도 동일한 값이 나오게 됩니다.
- 증명은 어렵지 않지만, 간단하게 묘사하자면, **S** 에서 연속한 문자열 01 을 10 으로 교환되었을 때, Z 값을 숫자 1 로 계산한 결과값과, 숫자 0 으로 계산한 결과값들이 동시에 1씩 증가하거나 동시에 1씩 감소하게 됩니다.
- 설명의 편의를 위해, 앞으로 계산은 모두 숫자 1 을 기준으로 계산하겠습니다.

H. 곰곰아 선 넘지마

- 아무튼 이렇게 계산된 Z 값을 이용해 정답을 구해야 합니다.
- 먼저, 다음과 같은 부등식을 생각해봅시다.

- $Z \leq X + Y$

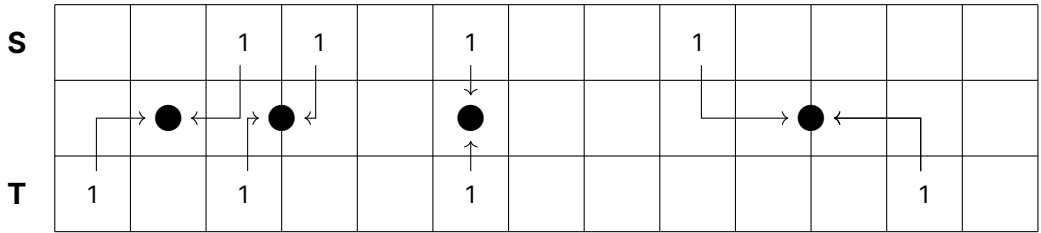
- Z 를 계산할 때 문자열 **T** 에서 소모시간 연산의 일부를 **S** 에서 수행하도록 하면, 최종 숫자 1 의 위치가 조금 달라질 뿐 필요한 연산의 합은 계속해서 Z 가 됩니다.
- 만약 최종 숫자 위치가 **S** 에서 숫자 1 의 위치나 **T** 에서 숫자 1 의 위치 사이 범위를 벗어난다면, 무의미하게 연산 횟수만 늘어나게 되고, 이 경우 $Z < X + Y$ 가 됩니다.

H. 곰곰아 선 넘지마

- 우리의 최종 목표는 $X^2 + Y^2$ 의 최솟값을 찾는 것입니다.
- 위 식은 $\frac{(X+Y)^2 + (X-Y)^2}{2}$ 으로 정리할 수도 있습니다.

- $Z \leq X + Y$ 식과 결합하면, 결국 $Z = X + Y$ **를 만족시키면서,** X **와** Y **의 차이가 최소가 되도록** 해야함을 알 수 있습니다.

H. 곰곰아 선 넘지마



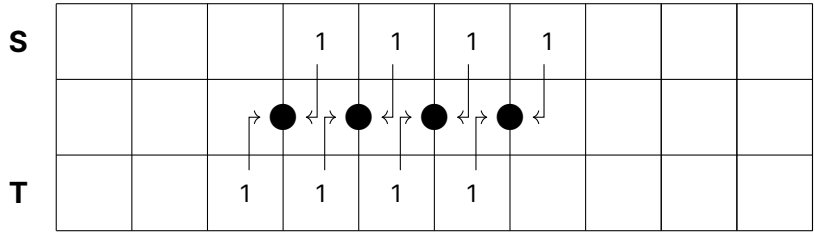
- 설명의 편의상 연산 횟수를 0.5 도 가능하다고 가정합니다.
- **S** 에서 숫자 1 의 위치와 **T** 에서 숫자 1 의 위치의 정확히 사이인 지점에 오도록 연산을 수행하면 X 와 Y 연산 횟수가 동일해집니다.

H. 곰곰아 선 넘지마

- 만약 **S** 와 **T** 에 있는 두 숫자 1 의 위치 차이가 홀수인 경우 X 와 Y 둘 중 하나는 연산을 한번 더 수행할 수 밖에 없습니다.
- 하지만 연산을 두 문자열에 **적절히** 분배하면, $|X - Y|$ 값이 항상 (Z 가 짝수인 경우) 0 또는 (Z 가 홀수인 경우) 1 이 되도록 할 수 있습니다.

- 숫자 1 이 중간지점이 위치해 있는 경우 X 쪽이나 Y 쪽 둘 중 한군데로 원하는 개수만큼 지정을 시켜줄 수 있기 때문입니다.

H. 곰곰아 선 넘지마



- 자유롭게 지정을 못하는 케이스는 위와 같이 중간지점이 '연속'해서 위치한 케이스 뿐입니다.
- 하지만 이 경우에도 아무 위치를 기준으로 왼쪽은 왼쪽방향으로, 오른쪽은 오른쪽 방향으로 연산을 1회 덜 수행하도록 지정을 시켜줄 수 있습니다.

- 따라서 최종 정답은 $\lceil \frac{Z^2}{2} \rceil$ 이 됩니다.

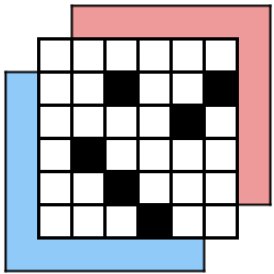
I. 곰곰이의 식단 관리 2

`ad_hoc`, `graph_theory`
출제진 의도 – **Hard**

- 제출 130번, 정답 29명 (정답률 22.30%)
- 처음 푼 사람: **jhnah917**, 22분
- 출제자: `pjshwa`

I. 곰곰이의 식단 관리 2

- 곰곰이는 $(1, 1)$ 에서 출발하고 격자판 밖으로 나갈 수 없기 때문에, 처음에 움직일 수 있는 다음 칸의 개수는 최대 2 개입니다. 이 두 곳만 막으면 곰곰이의 이동을 막을 수 있기 때문에 답의 상한은 2 입니다.
- 그렇다면 격자판이 주어졌을 때 확인해야 할 것은 답이 0 혹은 1 이 될 수 있는지입니다.
- 답이 0 이 되는 경우는, 주어진 격자판의 상태에서 곰곰이가 이미 (N, M) 으로 이동할 수 없는 경우입니다. 이는 그래프 탐색 등으로 간단하게 확인할 수 있습니다.
- 다음으로 답이 1 이 되는 경우를 알아봅시다. 격자판에 현재 장애물이 놓여있지 않은 칸 중, 해당 칸에 장애물을 놓았을 때 곰곰이의 이동을 막을 수 있다면 답이 1 일 것입니다.
- 칸마다 매번 그래프 탐색으로 목적지 도달 가능 여부를 시뮬레이션 하는 방법은, $\mathcal{O}\left((NM)^2\right)$ 로 시간 초과를 받게 될 것입니다.



- 곰곰이의 이동을 막을 수 있으려면, 위 그림에서 빨간색으로 칠해진 부분 (이하 구역 1)과 파란색으로 칠해진 부분 (이하 구역 2)이 연속적으로 장애물로 연결되어 있어야 한다는 사실을 관찰합시다. 이 장애물들은 상하좌우 뿐만 아니라 대각선으로 연결되어 있기만 해도 곰곰이의 이동을 막을 수 있습니다.

- 그렇다면 답이 1 이 되는 경우는, 현재는 구역 1과 구역 2가 장애물로 연결되어 있지 않지만 어떠한 칸에 장애물을 놓을 경우 구역 1과 구역 2가 연결되는 경우입니다. 모든 장애물에 대해서 해당 장애물이 구역 1과 연결되어 있는지, 혹은 구역 2와 연결되어 있는지 multi-source BFS / Union-Find 등의 방법을 통해 기록해둘 수 있습니다.
- 어떤 칸에 장애물을 놓았을 때 구역 1과 구역 2가 연결되려면, 해당 칸의 주변 8칸에 구역 1과 구역 2와 연결된 장애물이 각각 적어도 하나씩은 있어야 합니다.
- 새로운 장애물을 놓기 전에 이미 구역 1과 구역 2가 연결되어 있는 경우 답이 0 이므로, 위 내용을 구현하면 두 가지를 모두 체크할 수 있습니다.

J. 서커스 나이트

`number_theory`, `union_find`
출제진 의도 – **Hard**

- 제출 119번, 정답 26명 (정답률 22.69%)
- 처음 푼 사람: **xiaowuc1**, 25분
- 출제자: `pjshwa`

J. 서커스 나이트

- 어떤 돌고래는 자신의 ID와의 gcd가 2 이상인 ID를 가진 다른 돌고래에게만 메시지를 보낼 수 있고, 메시지를 받는 입장에서도 마찬가지 입니다.
- 메시지를 다른 돌고래를 통해서 전달하는 것도 가능하기 때문에, 돌고래들간 연결 상태를 Union-Find 자료구조를 통해서 관리하면 될 것입니다.
- ID가 될 수 있는 모든 수에 대해서 집합을 하나씩 두며, 각 수의 쌍마다 gcd가 2 이상인 경우 union을 수행하는 풀이를 생각해 봅시다.
  - 각 ID의 초기 set size는 해당 수가 주어진 배열 A에 등장하는 빈도수입니다.

J. 서커스 나이트

- 수의 범위가 크기 때문에 모든 쌍에 대해서 union을 수행하는 $\mathcal{O}\left(N^2\right)$ 풀이는 시간 초과를 받게 될 것입니다. 더 빨리 할 수 있는 방법이 없을까요?
- 어떤 양의 정수 x와 y의 gcd가 2 이상이라는 것은, 두 수를 소인수분해 했을 때 적어도 1개 이상의 소인수를 공유한다는 뜻이기도 합니다.
- 그렇다면 A에 등장하는 수들을 각각 소인수분해 한 뒤 그 소인수들과 union하기만 해도, gcd가 2 이상인 모든 수 쌍은 이 과정에서 간접적으로 합쳐질 것입니다.
- 1부터 100만 사이의 모든 수에 대해 해당 수가 가지는 가장 작은 소인수를 에라토스테네스의 체를 이용하여 저장해 두면, 여러 개 수를 빠르게 소인수분해 하는 것이 가능합니다.
  - BOJ 16563번 “어려운 소인수분해” 문제를 참고하세요.

K. 곰곰이와 토너먼트

`combinatorics, linearity_of_expectation`
출제진 의도 – **Hard**

- 제출 49번, 정답 20명 (정답률 40.81%)
- 처음 푼 사람: **xiaowuc1**, 36분
- 출제자: `jyheo98`

K. 곰곰이와 토너먼트

- x를 곰곰이(1번 참가자)보다 실력이 낮은 참가자 수라 정의합니다.
- 일관성을 잃지 않고, 곰곰이를 토너먼트 대진표 맨 왼쪽에 놓습니다.
- i번째 라운드의 상금을 획득하려면, 대진표에서 곰곰이 오른쪽 $2^i - 1$명의 참가자보다 곰곰이의 실력이 높아야 합니다.
- 이는 가능한 대진표 조합 $(2^k - 1)!$가지 중 $\binom{x}{2^i - 1} \times (2^i - 1)! \times (2^k - 2^i)!$가지에 해당됩니다.
- 기댓값의 선형성을 이용하여 각 라운드마다 독립적으로 상금의 기댓값을 구한 후 더해주면 답을 구할 수 있습니다.
- 분수의 modulo 값은 페르마 소정리를 이용해 구할 수 있습니다.

L. 곰곰이의 벼락치기

`tree`, `combinatorics`, `dfs`, `fermat_little_theorem`, `exponentiation_by_squaring`, `dp`

출제진 의도 – **Challenging**

- 제출 36번, 정답 25명 (정답률 69.44%)
- 처음 푼 사람: **jhnah917**, 38분
- 출제자: `chansol`

L. 곰곰이의 벼락치기

- 강의 a, b에 대한 관계 $a \to b$를 그래프의 간선으로 생각한다면, 입력으로 주어지는 그래프는 포레스트입니다.
- 트리 하나만 주어지면 어떻게 풀 수 있을까요?
- 정점 u를 루트로 하는 서브트리에 대하여 D_u를 서브트리에 속하는 모든 강의를 보는 방법의 수, 그리고 S_u를 서브트리의 크기로 정의합니다.
- S_u는 DFS로 트리를 한번 순회하면서 모두 구할 수 있습니다.

L. 곰곰이의 벼락치기

- 정점 u의 자식들을 $v_1, v_2, \cdots, v_k$라고 할 때,
- $D_u = D_{v_1} \times D_{v_2} \times \cdots \times D_{v_k} \times \dfrac{(S_{v_1} + S_{v_2} + \cdots + S_{v_k})!}{(S_{v_1})! \times (S_{v_2})! \times \cdots \times (S_{v_k})!}$
- 자식 정점들에서 나오는 각각의 방법(D_{v_i})을 독립적으로 선택할 수 있으므로 $D_{v_1} \times D_{v_2} \times \cdots \times D_{v_k}$를 곱합니다.
- 자식 정점들에서 나오는 강의 순서를 유지하면서 모든 강의를 배열하는 방법의 수는 $S_{v_1} + S_{v_2} + \cdots + S_{v_k}$를 $S_{v_1}, S_{v_2}, \cdots, S_{v_k}$로 분할하는 방법의 수와 동일합니다.
- D_u도 DFS로 트리를 한번 순회하면서 모두 구할 수 있습니다.

L. 곰곰이의 벼락치기

- 포레스트를 트리 하나로 바꿀 수 있을까요?
- 가상의 정점을 만들고, 이 정점에서 포레스트의 모든 트리의 루트로 가는 간선을 만들어줍니다.
- 이제, 새롭게 만들어진 트리에서 전체 문제를 해결할 수 있습니다.

- 팩토리얼 값은 미리 구해두고 사용합니다.
- 분할 식에서 분모의 팩토리얼 값은 페르마 소정리로 구하면 됩니다.
- 분할 정복을 이용한 거듭제곱을 사용하면 빠르게 계산할 수 있습니다.

M. 곰곰이와 하카타

`math`, `dynamic_programming`, `combinatorics`
출제진 의도 – **Challenging**

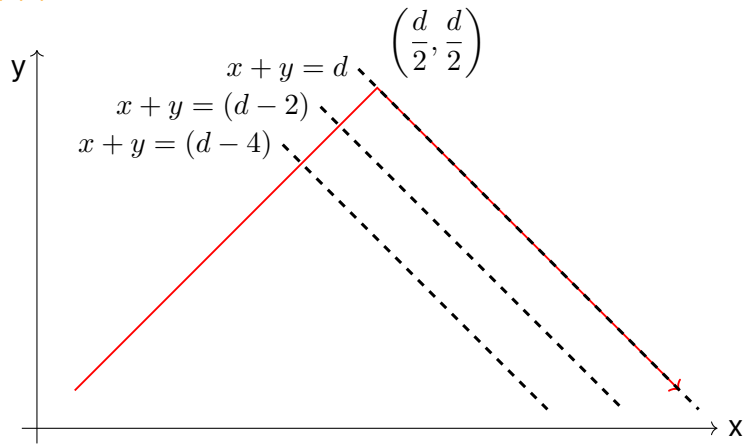
- 제출 66번, 정답 4명 (정답률 6.06%)
- 처음 푼 사람: **xiaowuc1**, 50분
- 출제자: chogahui05

- 사전순 k 번째를 구하는 것이 어려워 보입니다.
  - $d = 6$ 이고 $k = 2$ 라고 해 보겠습니다.
    - UUU??? 는 1가지입니다. 2보단 작습니다.
    - UUD???는 1가지입니다.
    - 따라서 사전순 2번째는 UUD??? 꼴로 가는 것임을 알 수 있습니다.
- $dp[y][x]$ 를 (x, y) 에서 $(d, 0)$ 에 있는 **하카타까지 갈 수 있는 가짓수**로 정의하면 되겠군요.

- 그런데 d의 최댓값이 **20만**이라 모든 좌표를 다 저장하기는 힘들어 보입니다.
- n이 매우 크다면, $\binom{n}{k}$은 k가 7 정도만 되어도 매우 큰 수가 된다는 것을 알 수 있습니다.
- $\binom{2000}{7} > 10^{18}$ 입니다.

- 이 사실을 이용하면 어떤 아이디어를 세울 수 있을까요?
  - 직선 $x + y = d$를 기준선 1로 잡겠습니다.
  - 직선 $x + y = (d - 2)$를 기준선 2로 잡겠습니다.
  - 직선 $x + y = (d - 4)$를 기준선 3으로 잡겠습니다.
- 이런 식으로 기준선 8까지 잘 그려보면, 어디선가 한 번쯤 본 듯한 그림을 볼 수 있습니다.

M. 곰곰이와 하카타



- $(\frac{d}{2}, \frac{d}{2})$ 를 기준으로 보면 어떨까요?

M. 곰곰이와 하카타

- 기준점에서 좌측으로 1만큼 하강하거나, 우측으로 1만큼 하강 할 수 있다 해 봅시다.
- $(\frac{d}{2}, \frac{d}{2})$ 에서 $(d-15, 1)$ 까지 이동해 봅시다.
  - $(d-15, 1)$ 은 직선 $x+y=(d-14)$ 위에 있습니다.
  - $x-y=0$ 과 $x+y=(d-14)$ 가 만나는 지점은 $(\frac{d}{2}-7, \frac{d}{2}-7)$ 입니다.
  - $(\frac{d}{2}-8)$ 회 우측으로, 7회 좌측으로 하강하면 $(d-15, 1)$ 로 이동 가능합니다.
- 이 상황은 가로 길이가 $(\frac{d}{2}-8)$, 세로 길이가 7, 격자 1칸당 가로와 세로 크기가 1일 때
  - $(0, 0)$ 에서 $(\frac{d}{2}-8, 7)$ 로 이동하는 가짓수와 같습니다.

- 당연하게도, a 와 b가 큰 수이고, $a < b$이면 $\binom{a}{7} < \binom{b}{7}$ 입니다.
- 따라서 d가 매우 커진다면 의미있는 $x + y = k$ 꼴의 직선 수는 8보다 작아집니다. (그렇게 많지 않습니다.)
- 즉, $dp[u][u] > 10^{18}$ 이라면, $x + y = 2u$ 꼴의 직선은 의미가 없다는 것을 이용해서 의미 있는 상태만 관리하면 됩니다.

요약

- ✓ 의미가 있는 $x + y = 2u$ 꼴의 직선은 몇 개인가?
- ✓ 어떤 상태를 저장해야 사전순 k 번째를 찾을 수 있을까?

N. 곰곰이와 GGANALi

`simulation`
출제진 의도 – **Challenging**

- 제출 22번, 정답 2명 (정답률 9.091%)
- 처음 푼 사람: **jhnah917**, 175분
- 출제자: `pichulia`

- “쉽고 재밌는 애드혹 대회” 목표에 맞춰서, 구현 원툴로 풀 수 있는 문제가 필요해서 제작한 문제입니다.
- **문제에서 주어진 정보를 토대로 그대로 구현하면 됩니다.**
- 문제 제목처럼, 각 연산을 $\mathcal{O}(N)$ 에 돌도록, Naive 하게 구현해도 통과하도록 제한을 정했습니다.
- 이게 풀이의 전부입니다.
- 농담이 아니라 진짜입니다...

- 조금 더 부연설명을 하겠습니다.
- 각 Property를 멤버변수로 가진 Actor class 를 구현합니다.
- 이 Actor class 에는 Actor* parent; std::vector<Actor*> children; 같은 형태로, 자식 Actor 를 관리할 수 있도록 구현합니다.
- A->Add(B) 연산을 수행할 때, Add 함수 내부에서 B->Unparent(); 를 내부적으로 먼저 수행한 다음에 A->children.push_back(B); 를 해주면 됩니다.

- SCREEN_POSITION 계산은 조금 머리를 써야합니다.
- 현재 Actor를 A, A 의 부모 Actor를 P 라고 지칭하겠습니다.
- A.SCREEN_POSITION = P.SCREEN_POSITION + (A.PARENT_ORIGIN - P.ANCHOR_POINT) * (P.SIZE) + A.POSITION 이 됩니다.
- 부모 Actor 의 정보를 이용해 기준점 위치를 먼저 계산하고, 그 기준점으로 부터의 상대위치 (=A.POSITION 속성) 을 더해서 A.ANCHOR_POINT 위치를 계산하는 로직입니다.
- 현재 Actor A 가 Window 가 될 때까지 위 연산을 재귀적으로 반복하면 됩니다. 물론 루트 Actor 가 Window 가 아닌 경우 적절한 예외처리가 필요합니다.

- SCREEN_POSITION 계산에 성공했으면, Actor 가 현재 화면에 어느 영역을 차지하는지를 나타내는 사각형 정보도 같이 알아낼 수 있습니다.

- 트리의 Pre-order 순서대로, 사각형 영역만큼 색을 칠해주면 됩니다.
- Actor 의 사이즈가 최대 $100\,000 \times 100\,000$ 이므로 모든 사각형 영역에 대해서 전부 색을 칠해선 안되고, 적당히 Window 영역과 겹치는 부분에 대해서만 순회를 해야합니다.
- **(이 문제에서 최적화가 필요한 유일한 부분입니다.)**

- 일련의 구현과정에서, 함수 입력 인자로 Actor 그 자체를 집어넣는 경우가 종종 생깁니다.
- Actor 에는 자식 Actor 들에 대한 정보들을 나타내는 리스트들이 모두 포함되어 있어야 하므로, std::vector<Actor> 와 같은 형태로 저장하는 것은 메모리를 폭발적으로 증가시키게 됩니다.
- 웬만해서는 Actor* 처럼, 포인터의 형태로 데이터들을 계속 관리하는 것이 좋습니다.
- 혹은 memory pool 개념을 이용해서, Actor* 대신 memory pool 의 index 를 대신 저장 / 사용 하는 방식으로 구현해도 됩니다.