2019 연세대학교 컴퓨터과학과 프로그래밍 경진대회 솔루션

A. 아스키 아트

- 정답 : 22명 (전원 정답)
- First solved : 김강산 (0:03)

A. 아스키 아트

- 주어진 대로 구현하면 됩니다.

- cin, cout, endl 등의 느린 입출력을 사용할 경우
  시간 초과가 발생할 수 있습니다.

B. 나무 위의 빗물

- 정답 : 13 + ?명
- First solved : 박범준 (0:18)

B. 나무 위의 빗물

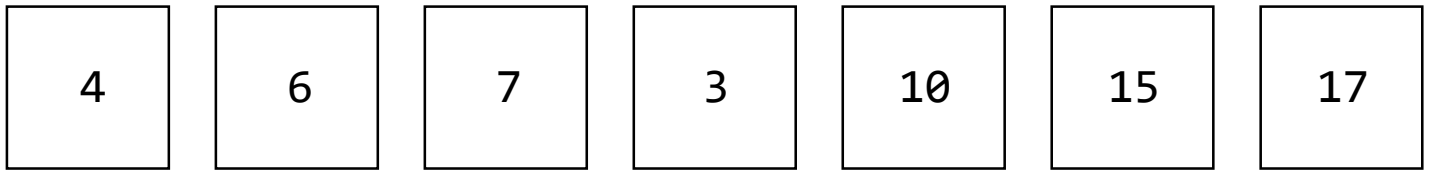
- 빗물은 가장 아래에 있는 정점에만 쌓이게 됩니다. 따라서 $P_i > 0$ 인 정점은 자식 정점이 없는 말단 정점들을 의미합니다. 이러한 정점들을 $leaf$라고 합니다.
- 빗물의 총량은 항상 W입니다.
- 따라서, $W/(number\ of\ leaf)$가 답이 됩니다.
- 이 때, 연결된 간선이 1개인 정점을 모두 $leaf$로 판정할 경우, 루트 정점이 세어질 수 있다는 점에 주의해야 합니다.

C. 정렬

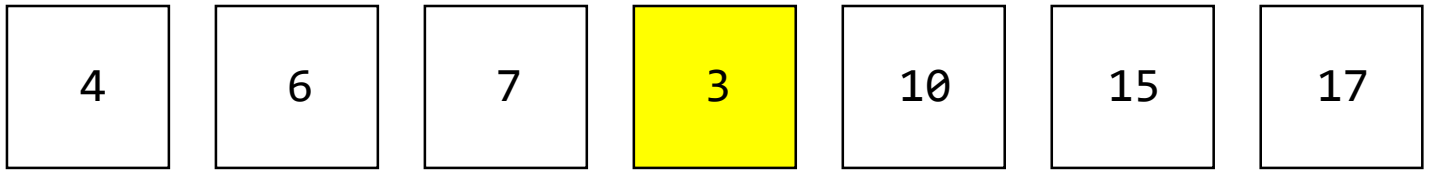
- 정답 : 14 + ?명
- First solved : 황준호 (0:11)

C. 정렬

- $O(N^2)$으로 직접 해 보기에는 시간이 부족합니다.
- 어떤 하나의 원소를 제거했을 때, 남은 원소들이 정렬되어 있기 위해 어떤 조건이 필요할까요?



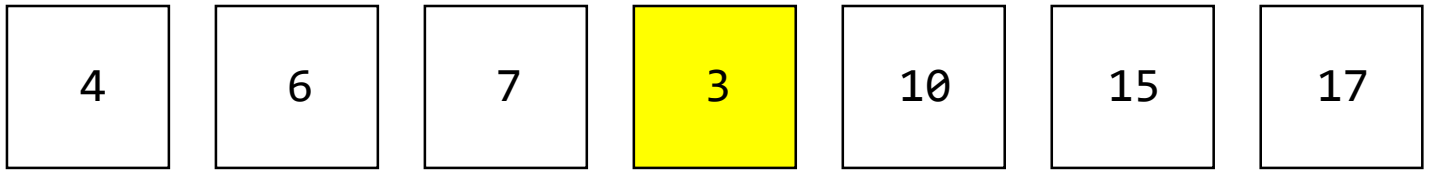
C. 정렬



- 가운데의 3을 제거하면 남은 배열은 정렬됩니다.
  $4 \leq 6 \leq 7 \leq 10 \leq 15 \leq 17$ 을 만족하기 때문입니다.
  이를 3의 위치를 기준으로 세 개의 조건으로 분할해 쓰면,
  $4 \leq 6 \leq 7,\ 7 \leq 10,\ 10 \leq 15 \leq 17$ 이 됩니다.

- 이를 일반화하면, a[i]를 제거했을 때 배열이 정렬되어 있는지 확인하려면, 아래의 세 조건을 체크하면 됩니다.
  - `a[1] <= a[2] <= … <= a[i-1]`
  - `a[i-1] <= a[i+1]`
  - `a[i+1] <= a[i+2] <= … <= a[n]`

C. 정렬



- `a[1] <= a[2] <= … <= a[i-1]`
  - 배열의 앞 부분, 즉 prefix가 정렬되어 있는지 확인하면 됩니다.
- `a[i-1] <= a[i+1]`
  - 직접 비교하면 됩니다.
- `a[i+1] <= a[i+2] <= … <= a[n]`
  - 배열의 뒷 부분, 즉 suffix가 정렬되어 있는지 확인하면 됩니다.

- prefix와 suffix의 정렬 여부는 맨 처음에 $O(N)$ 전처리를 한 번만 수행하여 모두 구해 둘 수 있습니다.

C. 정렬

- i까지의 prefix가 정렬되어 있으려면, i-1까지의 prefix가 정렬되어 있으며, a[i-1] <= a[i] 를 만족하면 됩니다.
- 이를 pseudo code로 표현하면 아래와 같습니다.

```
for i in range(1,n+1):
    pre[i] = pre[i-1] and a[i-1] <= a[i]
```

- suffix에 대해서도 거꾸로 똑같은 작업을 수행해 주면 됩니다.

- 그 이후에는, 모든 i에 대해,

  `pre[i-1] and suf[i+1] and a[i] <= a[i+1]`

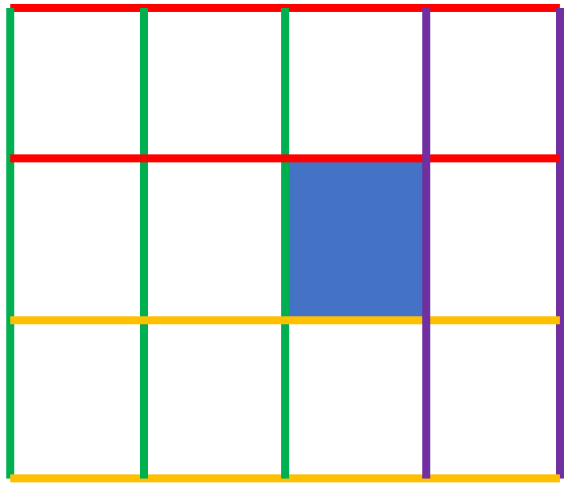
  인 경우를 모두 세어 주면 됩니다.

D. 유물 복원

- 정답 : 7 + ?명
- First solved : 황준호 (0:34)

D. 유물 복원

- 모든 직사각형 내부의 합을 구하지 말고, 각 칸마다 해당 칸을 포함하는 직사각형의 개수를 세는 쪽으로 생각해 봅시다.



- 파란 칸을 포함하는 직사각형은 파란 칸의 위에서 가로줄 하나,
아래에서 가로줄 하나,
왼쪽에서 세로줄 하나,
오른쪽에서 세로줄 하나를 뽑아 만들 수 있습니다.

- 각 칸에 대해, 해당 칸을 포함하는 직사각형의 수를 처음에 $O(NM)$ 전처리를 통해 모두 구해 둡니다. i행 j열 칸을 포함하는 직사각형의 개수를 $C_{i,j}$라고 합시다.

D. 유물 복원

- 이제 dynamic programming 풀이를 설계할 수 있습니다. NM개의 물건이 있고, 그 중 일부는 이미 구매했으며(a[i][j]==1), 일부는 구매하지 않음이 확정된 상황에서(a[i][j]==0), 나머지 물건을 적절히 구매해 구매한 총액을 K의 배수로 만드는 knapsack 문제가 됩니다.

- 아래와 같은 점화식을 설계할 수 있습니다.

```
//a[i][j]를 결정하는 중이고, 현재까지 지나온 물건들의 구매액 총합을
  K로 나눈 나머지가 mod인 상황이 가능하면 1, 불가능하면 0
d[i][j][mod] = d[i][j-1][mod-C_{i,j}] or d[i][j-1][mod]
```

- a[i][j]가 0 또는 1로 고정되어 있을 경우엔, or 양변 중 하나를 제외해 주면 됩니다. mod와 관련된 연산은 모두 modular K에서 합니다.

D. 유물 복원

- j가 1일 경우, d[i][j-1] 대신 d[i-1][m]을 확인해야 하는 점에 주의하도록 합시다.

- 가능/불가능 여부의 판정은 위와 같이 할 수 있으며, 복원은 추가적인 배열 하나를 더 만들어 각 d[i][j][mod]가 참조한 두 개의 값 중 어느 쪽이 1이었는지를 따로 저장하도록 하면 됩니다.

E. 망가진 데이터

- 정답 : 3 + ?명
- First solved : 황준호 (1:14)

E. 망가진 데이터

- M을 주어진 수 중 하나로 고정해 봅시다.
- N은 앞에 있는 수 중 하나입니다.
- u_i, v_i 는 뒤에 있는 수 중 $2M$개입니다.
- N은 클 수록 좋으며, u_i, v_i는 작을 수록 좋습니다.
- 따라서 a[i]를 M으로 고정하면 N은
  max(a[1], a[2], …, a[i-1])
  를 선택하는 것이 최적입니다.

E. 망가진 데이터

- u_i, v_i 는 a[i+1], a[i+2], …, a[k] 중 가장 작은 $2M$개를 선택하면 최적입니다.

- 이 때, $N \geq u_i, N \geq v_i$ 조건을 확인하기 위해, 선택한 수들 중 가장 큰 수가 얼마인지 알아낼 필요가 있습니다.

- 수의 범위에 주목합시다. 주어지는 모든 수가 20만 이하이므로, 각 수의 개수를 담을 수 있는, 20만개의 leaf를 가진 segment tree를 만들고, M의 값을 이동하는 과정에서 적절히 업데이트하고 쿼리를 진행하면 i번 원소 뒤에 있는 수들 중 $2M$번째로 작은 수를 알아낼 수 있습니다.

F. 고양이 우선 탐색

- 정답 : 0 + ?명
- First solved : ?

F. 고양이 우선 탐색

- 우선, 필요한 고양이의 최소 수가 0마리인지 판정할 필요가 있습니다.
- 한 번 시뮬레이션하며, 고양이가 없더라도 수열을 완성할 수 있다면 1을 출력하고 끝냅니다. 가능한 집합이 공집합 하나이기 때문입니다.
- 이제 고양이가 최소한 한 마리 필요한 상황만 남았습니다.
- 사실 한 마리만 있어도 항상 가능합니다. 고양이가 택희와 같은 정점에서 출발하여, 매번 다음 정점을 가이드해주는 방식으로 이동하면 원하는 수열을 만들 수 있기 때문입니다.

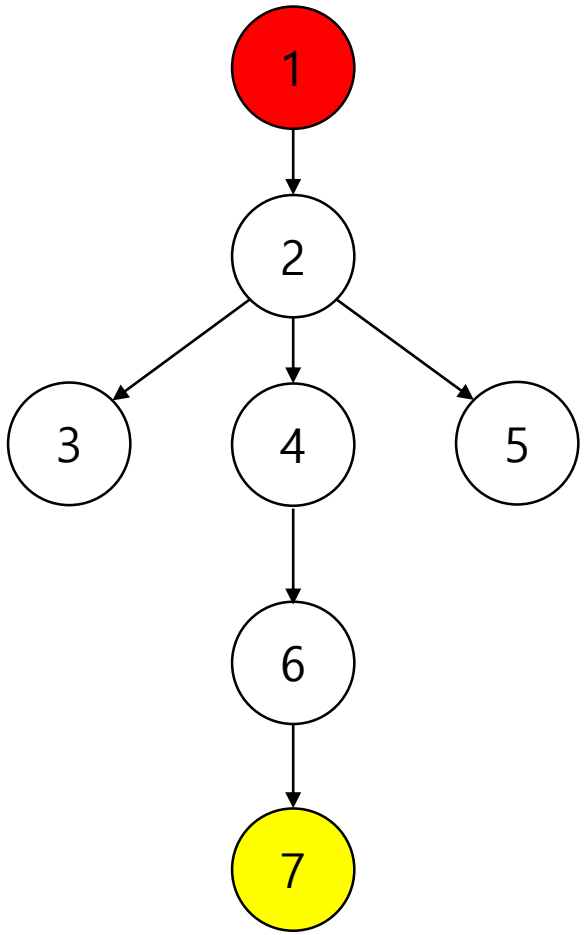
F. 고양이 우선 탐색

- 최소 고양이 수가 1마리이므로, 그 한 마리의 고양이가 출발할 수 있는 정점의 개수를 세는 문제로 바뀌었습니다.
- 우선, 택희가 탐색하는 과정에서 처음으로 고양이가 필요해지는 시각과 그 때 택희가 어디로 가야 하는 지를 구해 둡니다. 그 시각을 T라고 하고, 목표 지점을 V라고 합시다.
- 시각 T 이내에 V에 도달할 수 없는 정점들은 답이 될 수 없음이 자명합니다.
- 시각 T 이내에 V에 도달할 수 있다면, 고양이는 시각 T에 V에서 택희를 만나 쭉 가이드를 해 주기만 하면 됩니다.

F. 고양이 우선 탐색

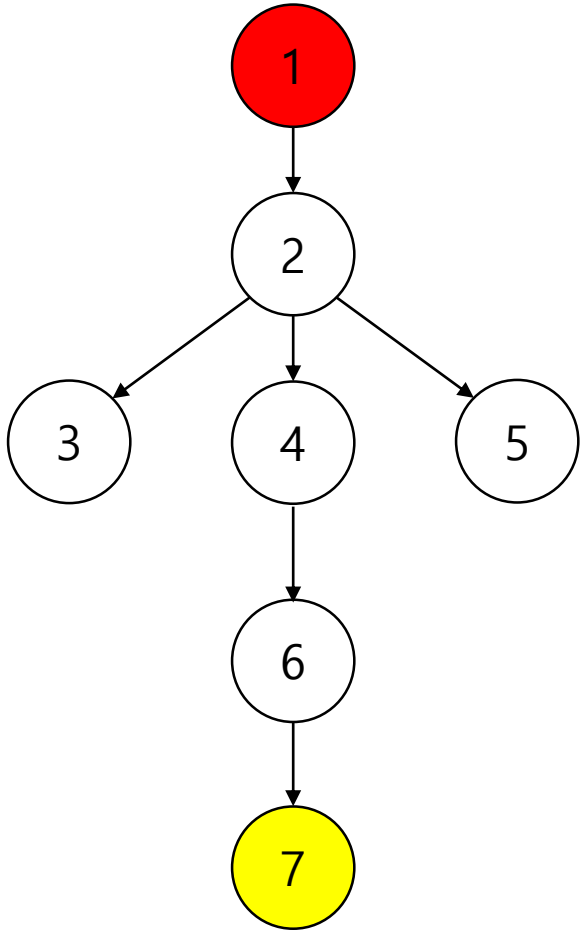
- 그렇다면, 시각 T에 정점 V로 올 수 있는 출발점은 몇 개일까요?
- 정점 V에서 T 이하의 시간이 걸려 도달할 수 있는 다른 정점들이 후보가 됩니다. 이 후보들은 $O(N)$ BFS로 계산할 수 있습니다.
- 여기까지 설계한 뒤 제출하면 `틀렸습니다` 를 받습니다.
- 왜 틀리냐 하면..

F. 고양이 우선 탐색



- 왼쪽과 같은 트리에서, 원하는 탐색 결과가
  1 2 3 5 4 6 7
  이라고 합시다.
- 처음으로 어긋나는 시간은 4(1->2->3->2->5), 목표 정점은 5입니다.
- 7번 정점은 5번 정점에서 4초만에 도달할 수 있는 정점입니다.
- 하지만 7번 정점은 답이 될 수 없습니다.

F. 고양이 우선 탐색



- 7번 정점에서 출발한 고양이가 5번 정점에 4초만에 도달하기 위해서는, 단 한 순간도 쉬지 않고 7->6->4->2->5로 이동해야 합니다.
- 하지만, 2초에 정점 4에 있게 된다면, 1초에 정점 2에서 출발한 택희가 고양이를 보고 정점 4로 내려와버리게 됩니다!
- 이러한 정점들은 답이 될 수 없으므로, 적절히 제거할 방법을 생각해보아야 합니다.

F. 고양이 우선 탐색

- 이를 해결하기 위해, 각 시각마다 어떤 정점에 있어야 하는지, 각 정점들의 부모 정점은 무엇인지, 각 정점을 처음 방문하는 시각이 언제인지를 모두 전처리해 둡니다.

- 이 때, 탐색에 걸리는 총 시간은 N보다 더 커질 수 있음에 주의합니다. 예를 들어, 앞선 트리에서는 1->2->3->2->5->2->4->6->7 로 탐색하며, 총 시간이 8초 걸리게 됩니다.

- 필요한 정보가 모두 전처리 되어 있다면, 시각 t에 어떤 정점 u에 고양이가 있어도 되는 지를 $O(1)$에 케이스 분류로 판정할 수 있습니다.

F. 고양이 우선 탐색

- 불가능할 조건은, 시각 t에 택희가 u의 부모 정점에 있으며, u를 방문한 적이 없고, u를 방문할 차례가 아니어야 합니다. 하나라도 만족하지 않는다면 택희는 u를 시각 t에 방문할 수 있습니다.

- 이를 이용해 정점 V에서 시작해 탐색을 하는데, 만약 들어갈 수 없는 정점이 생긴다면, 1초간 기다린 뒤 들어가는 것으로 합니다. 매 초마다 택희는 항상 이동하기 때문에, 1초만 기다리면 해당 정점이 열릴 것임을 알 수 있기 때문입니다.

- 실수 없이 모든 것을 잘 구현하면 AC를 받을 수 있습니다.

- p.s) 이번 대회 최고 난이도 문제예요!

G. 원 위의 개미

- 정답 : 3 + ?명
- First solved : 박범준 (1:34)

G. 원 위의 개미

- 두 개미가 만났을 때, 서로 되돌아가지 않고 그냥 무시하고 가던 길을 가는 문제와 정확히 동치임을 파악해야 합니다.

- 위는 충돌한 뒤 서로 되돌아가는 상황입니다.

- 위는 충돌하지 않고 서로 지나쳐가는 상황입니다. 이 때 만약 두 개미가 서로의 ID를 교환하며 지나친다면, 첫 상황과 정확히 동일한 상태가 됩니다.

G. 원 위의 개미

- 따라서 개미끼리 충돌하지 않는 것으로 생각합시다.
- 그렇다면, 모든 개미들은 다른 개미와 독립적이므로, 각 지점이 밟힌 횟수는 각 개미가 혼자 원을 돌며 해당 지점을 밟은 횟수들을 모두 합해 주면 됩니다.
- 모든 개미는 N초마다 원을 정확히 한 바퀴 돌기 때문에, $floor\left(\frac{T}{N}\right)$회 만큼 모든 지점을 밟은 뒤, 바라보는 방향과 초기 위치에 따라 $T\ mod\ N$개 정도의 지점을 추가로 1회씩 밟게 됩니다.

G. 원 위의 개미

- $Q = 1$일 경우, 답이 될 시간 T를 결정하고, 각 개미가 해당 지점을 몇 번 밟는 지를 계산하여 합산해주는 방식으로 binary search를 통해 $O(NlogT)$ 정도에 정답을 구할 수 있습니다.

- 하지만 Q가 크기 때문에, 각 쿼리마다 더 효율적인 방법으로 답을 구할 수 있어야 합니다.

- 같은 방향을 보는 개미들을 초기 위치 순으로 정렬해 놓으면, 어떤 시각에 한 지점을 추가로 밟는 개미들은 정렬된 배열 상에서 연속한 몇 마리임에 주목합시다. 즉, 연속된 몇 마리 개미는 C+1회, 다른 개미들은 C회 밟게 될 것입니다.

G. 원 위의 개미

- 개미들의 초기 방향에 따라, prefix sum 배열을 두 개 만듭시다.
  `pre[i] = 1,2, …, i번째 지점에 있는 개미의 총 수`
  가 되도록 정의하면 됩니다.

- 이제, 지점의 번호와 시간이 주어지면, 간단한 수식을 통해 +1회 밟는 개미들이 존재하는 구간이 결정되고, prefix sum 배열을 이용하여 $O(1)$에 해당 구간 내에 있는 개미의 수를 셀 수 있습니다.

- 따라서, 쿼리마다 $O(logT)$ 정도의 시간에 답을 계산할 수 있습니다.

H. 달콤새콤

- 정답 : 0 + ?명
- First solved : ?

H. 달콤새콤

- 우선, 각 선수가 발생시킬 승점의 기댓값을 따로 구해 합산해도 전체 경기에서 얻을 수 있는 기댓값을 얻을 수 있음에 주목합시다.

- 또한, 달콤 경기에서 얻을 수 있는 승점과, 새콤 경기에서 얻을 수 있는 승점을 따로 계산해 합한 뒤 반으로 나누어 주면 전체 승점을 얻을 수 있습니다. 따라서, 경기가 항상 달콤이라고 가정합시다.

- 능력치가 a_i인 선수는 전체 선수들 중 달콤함이 a_i이하인 선수와 매칭되었을 때 승점을 얻습니다. 이러한 선수들의 수는 미리

  `pre[i] = (달콤함이 i 이하인 사람의 수)`

  으로 전처리해 두면 $O(1)$에 계산할 수 있습니다.

H. 달콤새콤

- 물약을 먹는 경우에는 달콤함이 $[L_a, R_a]$ 범위 중 하나의 값으로 변하게 됩니다. 물약을 먹은 선수가 얻는 승점의 기댓값은 선수의 이전 능력치와 연관이 없으므로 한 번만 계산하면 됩니다. 모든 $[L_a, R_a]$ 범위의 능력치에 대해, 얻는 승점의 기댓값을 구해 둡시다. 1000^2 정도의 연산이 필요합니다.

- 이제, 원래 얻을 수 있는 승점의 기댓값보다 물약을 먹었을 때의 승점의 기댓값이 더 높은 선수에게만 물약을 주면 됩니다.

- 대소 비교를 위해, 이제까지의 내용을 정확히 수식으로 정리해 봅시다.

H. 달콤새콤

- 어떤 선수 i가 능력치 a_i를 갖고 있을 때, 이 선수가 치를 수 있는 서로 다른 조합의 라운드는 쿠키 나라와 초콜릿 나라에서 한 명씩을 뽑는 경우의 수인 N^2가지 경우가 있으며, 이 경기들에서 얻는 승점의 총합은 그 중 능력치가 a_i 이하인 두 선수를 뽑는 경우의 수가 됩니다. 즉, (초콜릿 나라 선수 중 능력치가 a_i 이하인 선수의 수)×(쿠키 나라 선수 중 능력치가 a_i 이하인 선수의 수) / N^2 이 됩니다. 지금부터는 이렇게 계산한 값을 일괄적으로 C_{a_i}라고 하겠습니다.

- 비슷한 방식으로, 물약을 먹은 선수의 승점 기댓값도 계산해 보면,
  $\sum_{i=L_a}^{R_a} C_i \,/(R_a - L_a + 1)$ 이 됩니다.

H. 달콤새콤

- 이제 달콤 경기와 새콤 경기를 합칩시다. 그냥 두 값을 더하고 2로 나누어 주면 됩니다.

- 양쪽의 분모를 모두 양변에 곱해도 C++의 `long long` 범위 안에 들어가는 값이기 때문에, 대소 비교를 바로 할 수 있습니다. 물약을 먹을 선수를 결정하고, 기댓값을 모두 더해 줍시다.

- 기댓값이 증가하는 선수에게만 물약을 주었기 때문에, 최소 인원임이 보장됩니다.

I. 결함 게임

- 정답 : 9 + ?명
- First solved : 최재혁 (0:58)

I. 결함 게임

- 선공이 할 수 있는 유효한 전략이 하나 보입니다. 두 번째로 작은 돌을 보드 위에 올리는 것입니다.
- 그럴 경우, 후공은 가장 작은 돌을 보드 위에 올리는 것밖에 할 수 없고, 돌탑이 하나 늘어나며 N은 2 줄어듭니다.
- 이 행동을 두 번 반복하면, 선공은 아무런 페널티 없이 N을 4 줄일 수 있습니다.

I. 결함 게임

- 만약 N을 4로 나눈 나머지가 0이라면, N이 4가 될 때까지 N을 줄입시다. 그 이후, 두 번째로 작은 돌을 고르고, 그 다음 턴에는 남은 두 개의 돌 중 작은 쪽을 고르면 승리합니다.

- N을 4로 나눈 나머지가 1이라면 N이 1이 될 때까지 N을 줄이면 되고, N을 4로 나눈 나머지가 2라면 N이 2가 될 때까지 N을 줄인 뒤 남은 두 돌 중 큰 돌을 사용하면 승리합니다.

- 따라서, N을 4로 나눈 나머지가 0, 1, 2 중 하나라면 선공이 이깁니다.

I. 결함 게임

- N을 4로 나눈 나머지가 3이라면 어떻게 될까요?
- 현재 $N = 4K + 3$개의 돌이 있다고 합시다. 여기서 선공이 세 번째로 작은 돌 혹은 그보다 큰 돌을 고르게 된다면, 후공은 두 번째로 작은 돌을 골라버린 뒤, 선공이 가장 작은 돌을 고르게 하여 N을 3 줄입니다.
- 방금 돌탑이 1개 완성되었으므로 후공은 $4K$개의 돌을 이용하여 돌탑을 홀수 개 만들면 이기는 상태로 갈 수 있습니다.
- 이것은 앞서 보았듯이, 필승인 상태입니다. 따라서, 선공은 세 번째 혹은 그보다 크거나 같은 돌을 고를 수 없습니다.

I. 결함 게임

- 선공이 가장 작은 돌을 고르고 시작하면, 후공은 $4K + 2$개의 돌이 있으며, 돌탑을 홀수 개 만들면 이기는 게임을 받게 됩니다. 이 역시 후공이 이기게 됩니다.

- 따라서, 선공은 두 번째로 작은 돌을 고르는 행동밖에 할 수 없습니다.

- 이 경우, 후공은 가장 작은 돌을 골라야 하며, 선공은 $4K + 1$개의 돌을 이용해 짝수 개의 돌탑을 만들어야 하는 상황이 됩니다.

I. 결함 게임

- 만약 이 상태에서 선공이 가장 작은 돌을 고르면, 후공은 $4K$개의 돌을 이용해 짝수 개의 돌탑을 만들어야 하는 상태가 되고, 이는 계속해서 두 번째로 작은 돌을 고르는 행동을 반복하면 가능하므로 후공이 승리합니다.

- 선공이 다른 돌을 고르면, 후공은 가장 작은 돌을 고르고, 선공은 $4K + 3$개의 돌을 이용해 홀수 개의 돌탑을 만들어야 하는, 초기 상태로 돌아오게 됩니다. 이 상태에서 선공이 할 수 있는 행동은 계속해서 이것을 반복하는 것밖에 없습니다. 다른 행동은 바로 필패로 이어지기 때문입니다.

I. 결함 게임

- 따라서 선공은 최종적으로, 3개의 돌을 이용해 홀수 개의 돌탑을 만들어야 하는 상황이 됩니다.
  - 선공이 가장 작은 돌을 고를 경우, 후공은 남은 두 돌 중 큰 돌을 고르고, 돌탑은 2개 만들어집니다.
  - 선공이 두 번째로 작은 돌을 고를 경우, 돌탑이 반드시 2개 만들어지게 됩니다.
  - 선공이 가장 큰 돌을 고를 경우, 후공이 가장 작은 돌을 고르면 돌탑이 2개 만들어집니다.
- 이에 따라, 선공이 항상 패배합니다.

- 결론적으로, N을 4로 나눈 나머지가 0, 1, 2이면 선공, 3이면 후공이 이깁니다.

J. RPG Extreme

- 정답 : 0 + ?명
- First solved : ?

J. RPG Extreme

- 제약 조건이 모두 작으므로, 문제에서 요구하는 내용을 그대로 충실히 구현하면 맞을 수 있습니다.
- 구현 외에 필요한 것은 아무 것도 없으므로 풀이를 생략합니다.
- 그런데 이 문제가 왜 나왔을까요?
- 처음 발단은, 최고 난이도 그룹의 문제를 구현 문제로 만들면 어떨까에 대한 토의였습니다.
- 학부 저학년 학생들 혹은 PS를 많이 접해보지 못한 참가자들도 시도해 볼 수 있는 고난이도 문제를 만드는 것이 목적이었습니다.

J. RPG Extreme

- 그러나 첫 prototype은 알고리즘/자료구조를 하나도 사용하지 않다 보니 생각보다 너무 쉽고 빠르게 짤 수가 있었습니다.
- 그러던 중, 객체지향프로그래밍 과목의 마지막 과제인 *rabbit tiger hunter* 에 1학년 학생들이 고생하고 있다는 소문을 들었습니다.
- 해당 과제에서 착안해 이런 문제가 설계되었고, PS에서 잘 다루지 않는 개념인 OOP를 사용하면 비교적 편하게 짤 수 있는 구현 문제가 탄생하게 되었다고 생각해 출제하게 되었습니다.
- 출제진의 코드는 대충 4000byte 내외입니다.

K. 쿼리와 쿼리

- 정답 : 1 + ?명
- First solved : 황준호 (2:44)

K. 쿼리와 쿼리

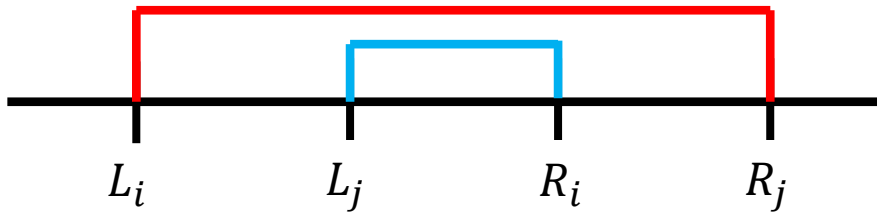
- 놀이 과정에서 등장한 $[L, R]$ 쿼리의 $[L, R]$ 구간들을 모두 합집합해 얻은 구간들의 집합을 V라고 합시다.

- 만약 어떤 놀이 i와 놀이 j에 대해, $V_i \subseteq V_j$라면 놀이 i를 하는 것이 놀이 j를 하는 것보다 **배열의 상태와 관계없이** 항상 같거나 더 좋은 결과를 낼 것임을 알 수 있습니다.

- 그리고 이 놀이에는, 모든 다른 V_j에 대해, 항상 $V_i \subseteq V_j$를 만족하는 놀이 i가 존재합니다.

K. 쿼리와 쿼리

- 만약 어떤 놀이에서, $L_i \le L_j$, $R_i \le R_j$인 네 값 L_i, L_j, R_i, R_j 에 대해, $[L_i, R_j]$와 $[L_j, R_i]$ 를 매칭시킨 경우가 있다고 합시다.
  - 만약 $L_i > R_j$ 또는 $L_j > R_i$라면, 이미 답이 10^9이므로 최악입니다.
  - 그렇지 않다면, 항상 아래처럼 매칭됩니다.



- 두 경우 모두, 매칭을 $[L_i, R_i]$와 $[L_j, R_j]$로 변경해도 손해보는 것이 없습니다. 오히려, $L_i > R_j$ 또는 $L_j > R_i$였을 경우, 해당 상태를 풀어줄 수도 있는 효과가 있습니다.

K. 쿼리와 쿼리

- 따라서 $L_i \leq L_j$, $R_i \leq R_j$인 네 값 L_i, L_j, R_i, R_j 가 있다면, 매칭을 $[L_i, R_i], [L_j, R_j]$로 하는 것이 최적입니다.

- 이를 연쇄적으로 적용하면, 집합 L과 R을 각각 정렬한 뒤 순서대로 매칭시키는 것이 모든 놀이에서 최적임을 알 수 있습니다.

- 이제 매칭을 찾았으니, 놀이에서 건드리는 정수와 아닌 정수를 set 등으로 구분해 관리합시다. 쿼리당 $O(logN)$ 정도에, 건드리는 정수들 중 최대값을 알 수 있는 아무 자료구조나 사용하면 됩니다.

L. 그리드랜드

- 정답 : 1 + ?명
- First solved : 김강산 (1:33)

L. 그리드랜드

- 두 파벌 중 한 쪽만 있다면 항상 3색으로 지붕을 모두 색칠할 수 있습니다. 행우선 또는 열우선으로 차례대로 탐색하면서, 지금까지 색칠된 지붕 중 인접한 두 지붕과 다른 색을 넣어 주면 됩니다.

- 5색으로 두 파벌을 색칠하려면, 3색 / 2색 또는 2색 / 3색을 사용하게 될 것입니다. 이 때, 3색을 사용하는 쪽은 항상 색칠 가능합니다.

- 그래프가 2색으로 색칠 가능한지는, 그래프가 Bipartite graph(이분 그래프)인지 판정하면 되며, 이를 해결하는 간단한 알고리즘이 존재합니다. BFS를 실행하며, 색칠하지 않은 정점이 나올 때마다 greedy하게 색칠해 주면 됩니다.

L. 그리드랜드

- 이제 파벌 1 또는 2 둘 중 하나가 이분 그래프인지 판정하고, 만약 이분 그래프라면 해당 파벌을 2색으로, 다른 쪽 파벌을 3색으로 색칠한 뒤 출력하면 됩니다.

- 양쪽 모두 이분 그래프가 아니라면 최소 3+3=6색이 필요하므로 불가능합니다.

M. f(k, n)

- 정답 : 3 + ?명
- First solved : 김규상 (2:05)

M. f(k, n)

- 우선, Fibonacci 수열은 $n \leq 0$일 때도 정의할 수 있습니다.
  $fib(n+2) = fib(n+1) + fib(n)$이므로,
  $fib(n) = fib(n+2) - fib(n+1)$이 됩니다.
- $f(2,i) = f(1,i+1) - f(1,i) = fib(i+1) - fib(i) = fib(i-1)$
- $f(3,i)$는 같은 방식으로 $fib(i-2)$
- …
- 따라서, 첫 줄은 피보나치 수열, 그 이후로는 윗 줄을 오른쪽으로 한 칸 밀고 왼쪽은 음수 n으로 확장한 수열이 순서대로 등장하게 됩니다.

M. f(k, n)

- 앞 내용을 식으로 정리하면, $f(i,j) = f(i+1,j+1)$가 됩니다.
- 따라서 어떤 부분배열의 왼쪽 꼭지점이 (r,c)이고, $r > 1\ and\ c > 1$이라면, $(r-1,c-1)$에서부터 찾아도 동일한 부분배열을 찾을 수 있습니다. 즉, 찾을 부분배열은 첫 행 또는 첫 열에서만 찾고, 각각이 등장한 횟수를 더해 주기만 하면 됩니다. 부분배열은 오른쪽 아래로 내려가며 계속 동일하게 등장하므로, 등장 횟수는 $N - \max(r,c) - p + 2$입니다.
- 또한, 피보나치 수열은 index가 88 이상이 될 경우 10^{18}을 초과하는 값이 됩니다. 따라서, $2N-1$개의 후보를 모두 검사할 필요는 없습니다. 행이나 열이 수백 단위를 넘어가면 이미 첫 값부터 틀리기 때문입니다.

M. f(k, n)

- 따라서 첫 행 또는 첫 열에 왼쪽 위 꼭지점을 고정하고 적당한 범위 내에서 brute force를 통해 주어진 배열을 찾아 본 뒤, 등장 횟수를 앞서 설명한 수식에 따라 모두 더해 주면 답이 됩니다.

- p.s) 전개가 잘 안 되더라도, 직접 $f(i,j)$를 한 번 적당한 범위 내에서 출력해 보면 규칙을 쉽게 찾을 수 있습니다. 이러한 풀이도 정해에서 상정하고 있으며, 이를 위한 힌트로 본문에 잡담을 넣어 두었습니다.

fin

- 수고하셨습니다!
- 잠시 휴식한 뒤 시상식 및 경품 추첨이 진행될 예정입니다.
- 서로 토의하셔도 되고, 추가적인 질문이 있다면 출제진 중 아무에게나 질문해 주세요~

+Bonus

		A	B	C	D	E	F	G	H	I	J	K	L	M	
kdh9949		+1				+1	+6		+2	+1	-1		+2		1765 min
doju								+2				+1	+1		1523 min.
jwvg0425					+2					+2	+5			-5	811 min