2024 UDPC

Official Solutions

UDPC 2024 - 출제진 및 검수진

- ✓ 출제진
  - 김도훈 99asdfg
  - 김민서 mskim17
  - 박신욱 bnb2011
  - 우은규 ekwoo
  - 장래오 leo020630
  - 정병현 jbh0224
  - 최진우 andrew1010
  - 한동규 queued_q
- ✓ 검수진
  - hyperbolic
  - lky7674
  - mjhmjh1104
  - nick832
  - slah007
  - tony9402
  - utilforever

Junior Division	문제	의도한 난이도	출제자
A	핑크빈 레이드	**Easy**	정병현 jbh0224
B	행운을 빌어요	**Easy**	장래오 leo020630
C	지정좌석 배치하기 1	**Easy**	김도훈 99asdfg
D	Double Up	**Easy**	박신욱 bnb2011
E	UDP 스택	**Medium**	장래오 leo020630
F	현대모비스 자율 주행 테스팅 1	**Medium**	장래오 leo020630
G	Twitch Plays Pokemon	**Medium**	한동규 queued_q
H	산수화	**Hard**	장래오 leo020630
I	수강신청	**Hard**	김도훈 99asdfg

Senior Division	문제	의도한 난이도	출제자
A	핑크빈 레이드	**Easy**	정병현 jbh0224
B	행운을 빌어요	**Easy**	장래오 leo020630
C	Double Up	**Easy**	박신욱 bnb2011
D	현대모비스 자율 주행 테스팅 2	**Medium**	장래오 leo020630
E	지정좌석 배치하기 2	**Medium**	김도훈 99asdfg
F	무빙워크	**Hard**	장래오 leo020630
G	수강신청	**Hard**	김도훈 99asdfg
H	Hyper Tree Problem	**Hard**	박신욱 bnb2011
I	포닉스와 달구	**Hard**	박신욱 bnb2011

1A/2A. 핑크빈 레이드

`implementation`
출제진 의도 – **Easy**

- 통계:
  - Junior: 제출 29번, 정답 17번, 정답률 58.621%
  - Senior: 제출 46번, 정답 28번, 정답률 60.870%
- First Solve:
  - Junior: **김재환**POSTECH, 4분
  - Senior: **김경민**POSTECH, 2분
- 출제자: 정병현jbh0224

1A/2A. 핑크빈 레이드

- 각 마스코트가 사용하는 스킬의 사용 주기가 대미지보다 작거나 같으므로, 각 마스코트가 핑크빈에게 가할 수 있는 초당 대미지는 1 보다 크거나 같습니다.
- 따라서 핑크빈을 잡는 데에 걸리는 시간이 최대 약 $5000/3$ 이므로, 반복문을 이용하여 해결할 수 있습니다.
- 시간을 1 초씩 증가시키며, 유니, 달구, 포닉스 중 현재 시점에 스킬을 사용해야 하는 마스코트는 스킬을 사용하여 핑크빈의 체력을 감소시킵니다.
- 현재 시점에 스킬을 사용해야 하는지 여부는 현재 시각을 주기 C 로 나눈 나머지가 0 인지를 통해 확인할 수 있습니다.
- 핑크빈의 체력이 0 **보다 작거나 같아지면** 현재 시각을 출력하고 종료합니다.

1B/2B. 행운을 빌어요

`bruteforcing`
출제진 의도 – **Easy**

- ✓ 통계:
  - Junior: 제출 38번, 정답 14번, 정답률 36.842%
  - Senior: 제출 58번, 정답 23번, 정답률 39.655%
- ✓ First Solve:
  - Junior: **김재환**POSTECH, 11분
  - Senior: **이유담**POSTECH, 8분
- ✓ 출제자: 장래오`leo020630`

1B/2B. 행운을 빌어요

- 사용할 줄기 개수 X를 고정하고 생각합시다. X는 A 이상 1000 이하의 정수입니다.
- 고정한 X에 대해 더 사용해야 하는 잎의 개수 $f(X)$는 아래와 같습니다.
  - $3X \le B \le 4X$ 인 경우: $f(X) = 0$
  - $B < 3X$ 인 경우: $f(X) = 3X - B$
- $4X < B$인 경우는 불가능하니 고려하지 않습니다.
- 따라서 모든 $A \le X \le 1000$에 대해 더 사용해야 하는 줄기와 잎 개수의 합의 최솟값, $\min(X - A + f(X))$가 답이 됩니다.
- 시간복잡도는 $\mathcal{O}(1000T)$입니다.
- 경우를 잘 나누면 케이스당 $\mathcal{O}(1)$에도 해결할 수 있습니다.

2C. 지정좌석 배치하기 1

`greedy`, `sorting`
출제진 의도 – **Easy**

- 통계:
  - Junior: 제출 31번, 정답 17번, 정답률 54.839%
- First Solve:
  - Junior: **김재환**[POSTECH], 21분
- 출제자: 김도훈[99asdfg]

2C. 지정좌석 배치하기 1

- ✓ 문제에서 중요한 부분은 (좌석의 높이 + 키) 뿐이므로, i번 행을 희망한 모든 학생의 키를 $D \times i$ 만큼 크다고 생각해도 좋습니다.
- ✓ i번 행을 희망한 학생들의 키를 $D \times i$만큼 늘려준 뒤 각 행의 학생들의 키를 오름차순 정렬하여, i행에서 j번째로 작은 학생의 키를 $H_{i,j}$라 합시다.
- ✓ 이때, 반드시 $H_{1,1} < H_{2,1} < \cdots < H_{N,1}$을 만족해야 모든 학생들을 배치할 수 있습니다.
  - 만약 $H_{i,1} \geq H_{i+1,1}$이라면, 정의에 의해 모든 j에 대해 $H_{i,j} \geq H_{i+1,1}$이 되므로 키가 $H_{i+1,1}$인 학생은 항상 앞줄의 학생보다 키가 작거나 같기 때문입니다.

2C. 지정좌석 배치하기 1

- 또한, $H_{1,2} < H_{2,2} < \cdots < H_{N,2}$ 도 만족해야 합니다.
  - 만약 $H_{i,2} \geq H_{i+1,2}$ 라면, 정의에 의해 2 이상의 모든 j 에 대해 $H_{i,j} \geq H_{i,2} \geq H_{i+1,2} \geq H_{i+1,1}$ 이므로, 키가 $H_{i+1,2}$ 인 학생과 $H_{i+1,1}$ 인 학생 중 적어도 한 명은 앞줄의 학생보다 키가 작거나 같기 때문입니다.
- 위의 아이디어를 키가 3 번째로 작은 학생, $\cdots$, M 번째로 작은 학생 으로 확장해 나갈 수 있고, 따라서 모든 j 에 대해 $H_{1,j} < H_{2,j} < \cdots < H_{N,j}$ 를 만족하면 YES, 아니면 NO를 출력하면 됩니다.
- 모든 행을 정렬한 뒤 조건을 체크하는 방식으로, 시간복잡도 $\mathcal{O}\left(NM \log M\right)$ 으로 문제를 해결할 수 있습니다.

1E. 지정좌석 배치하기 2

`greedy`, `sorting`, `combinatorics`
출제진 의도 – **Medium**

- 통계:
  - Senior: 제출 20번, 정답 8번, 정답률 40.000%
- First Solve:
  - Senior: **권찬우**POSTECH, 53분
- 출제자: 김도훈`99asdfg`

1E. 지정좌석 배치하기 2

- 지정좌석 배치하기 1 문제와 똑같이, i 행을 희망한 모든 학생의 키를 $D \times i$ 만큼 늘린 뒤 각 행의 오름차순으로 정렬하여 $H_{i,j}$ 라 합시다.
- 지정좌석 배치하기 1 문제에서 본 것과 같이, 만약 $H_{1,j} < H_{2,j} < \cdots < H_{N,j}$ 를 만족하지 않는다면 단순히 0 을 출력하고 프로그램을 종료하면 됩니다.
- 만약 해당 조건을 만족한다면, 바로 앞 행의 학생만 자신보다 키가 작더라도 그 앞에 있는 다른 모든 학생들의 키는 항상 자신의 키보다 작음을 확인할 수 있습니다.
- 따라서, 앞에 앉은 **모든** 학생이 아니라 바로 앞 행에 앉은 학생의 키만을 고려해도 모든 경우의 수를 셀 수 있습니다.

1E. 지정좌석 배치하기 2

- 가장 앞 행부터 차례대로 학생을 배치해 봅시다.
- 첫 행에는 아무렇게나 학생들을 배치해도 상관 없으므로, $M!$ 개의 경우의 수로 학생을 배치할 수 있습니다.
- $H_{2,k}$ 인 학생이 배치될 수 있는 좌석의 개수는 $H_{1,j} < H_{2,k}$ 인 학생의 수와 동일합니다.
- 이때, $H_{2,k-1} \leq H_{2,k}$ 이므로 $H_{2,k}$ 인 학생이 배치될 수 있는 좌석은 항상 $H_{2,k-1}$ 인 학생이 배치될 수 있는 좌석을 포함합니다.
- 따라서, 2 행에 배치하는 경우의 수는 $H_{2,1}$ 부터 $H_{2,M}$ 까지 순서대로 좌석을 배치할 때 $(H_{1,j} < H_{2,k}$ 인 학생의 수 $) - (k-1)$ 를 순서대로 곱하여 구해줄 수 있습니다.

1E. 지정좌석 배치하기 2

- 위에서 언급한 바와 같이 학생을 배치하는 경우의 수는 오로지 앞 행의 학생들에 의해서만 결정되므로, 위의 시행을 2 행부터 N 행까지 반복한 뒤 마지막에 $M!$ 을 곱해 주면 됩니다.
- $H_{i,j} < H_{i+1,k}$ 인 개수는 투 포인터 혹은 이분 탐색으로 구할 수 있으므로, 총 시간복잡도 $\mathcal{O}(NM)$ 또는 $\mathcal{O}(NM \log NM)$ 으로 문제를 해결할 수 있습니다.

1C/2D. Double Up

`math, ad_hoc`

출제진 의도 – **Easy**

- ✓ 통계:
  - Junior: 제출 39번, 정답 14번, 정답률 35.897%
  - Senior: 제출 42번, 정답 24번, 정답률 57.143%
- ✓ First Solve:
  - Junior: **김재환**POSTECH, 41분
  - Senior: **이유담**POSTECH, 10분
- ✓ 출제자: 박신욱bnb2011

1C/2D. Double Up

- 주어지는 모든 수는 $A_i = 2^{a_i} \times b_i$ (b_i 는 홀수) 형태로 나타낼 수 있습니다.
- 달구가 어떤 수에 2를 곱하게 되면 b_i의 값은 변하지 않고, a_i만 1 증가합니다.
- 따라서 어떤 두 수 $A_i, A_j (A_i < A_j)$가 $b_i = b_j$ 라면 A_i에 연산을 $a_j - a_i$번 적용해서 두 수를 같게 만들 수 있습니다. 반대로 $b_i \neq b_j$ 어떻게 연산을 적용해도 두 수를 같게 만들 수 없습니다.
- 이제 문제에서 요구하는 **가장 많이 등장하는 수**의 **최대 등장 횟수**는 곧 b_i의 최대 등장 횟수와 같다는 것을 알 수 있습니다.
- 모든 A_i를 2로 더 이상 나누어 떨어지지 않을 때까지 나눠서 b_i를 구한 뒤, 가장 많이 등장하는 b_i의 값을 출력하는 것이 답이 됩니다.
- 구현 방식에 따라 $\mathcal{O}(N \log \max A_i)$ 또는 $\mathcal{O}(N \log N \log \max A_i)$ 시간에 문제를 해결할 수 있습니다.

2E. UDP 스택

`greedy`, `stack`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 61번, 정답 5번, 정답률 8.197%
- First Solve:
  - Junior: **곽민성**POSTECH, 59분
- 출제자: 장래오`leo020630`

2E. UDP 스택

- ✓ 관찰 1.
  - 당연하게도 비어 있는 스택이 많아야 유리합니다.
- ✓ 관찰 2.
  - 원소가 삽입된 스택의 바닥을 열면 저장된 모든 원소가 함께 떨어집니다.
  - 따라서 만약 스택에 원소를 삽입해야 한다면, 이 원소들은 연속한 구간을 이루어야 합니다.

2E. UDP 스택

- 관찰 3.
  - 만약 2번 연산을 사용해 닫혀 있던 스택을 열었다면 2번 연산을 다시 사용해 기존에 열려 있던 스택을 다시 열 수 있습니다.
  - 따라서 바닥이 열려 있는 스택이 항상 같다고 가정해도 좋습니다. 편의상 이를 U스택이라 합시다.
- 이러한 사실들을 기반으로 그리디한 전략을 세워볼 수 있습니다.

2E. UDP 스택

- 현재 x 까지의 수를 스택 바닥 아래로 떨어뜨려 정렬을 완료했다고 가정합시다. x 의 초깃값은 0 입니다.
- 또한 순열 A 의 가장 앞 원소를 a, D스택의 가장 위 원소를 d, P스택의 가장 위 원소를 p 라 합시다. 해당 스택이 비어 있을 경우에 d 나 p 의 값은 $\emptyset$ 으로 정의합니다.

2E. UDP 스택

- 1) $a = x + 1$
  - 이 경우 a를 U스택으로 삽입해 바로 떨어뜨리는 것이 최선입니다.
- 2) $a = d + 1$ or $a = p + 1$ $(d, p \neq \emptyset)$
  - 이 경우 a를 해당하는 스택에 저장하는 것이 최선입니다.
- 3) $d = \emptyset$ or $p = \emptyset$
  - 이 경우 a를 빈 스택에 저장하는 것이 최선입니다.

- 위 세 경우를 모두 만족하지 못해 a를 삽입할 수 없는 경우 순열을 오름차순으로 재배열하는 것이 불가능합니다.
- 이렇게 선택을 반복하며, 도중에 D스택이나 P스택의 가장 아래 원소가 $x+1$일 경우 해당하는 스택을 열어 원소를 떨어뜨려주면 됩니다.
- x의 값 역시 연산에 맞도록 적절히 관리해줍시다.
- 시간복잡도는 테스트 케이스당 $\mathcal{O}(N)$입니다.

2F. 현대모비스 자율 주행 테스팅 1

`greedy`, `implementation`, `case_work`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 37번, 정답 2번, 정답률 5.405%
- First Solve:
  - Junior: **곽민성**POSTECH, 91분
- 출제자: 장래오`leo020630`

2F. 현대모비스 자율 주행 테스팅 1

- 완주가 불가능한 경우를 먼저 처리합시다.
- 한 열의 두 칸이 모두 장애물이거나, 대각선으로 배치된 장애물이 있는 경우 완주가 불가능합니다.
- 이 때 $K > 1$인 경우 샘플 주행 트랙의 N번째 열과 1번째 열이 대각선으로 연결되는 경우가 있을 수 있음에 유의합시다.

2F. 현대모비스 자율 주행 테스팅 1

- 각 연산에 대한 사용 횟수를 나누어 구해봅시다.
- 직진은 자명하게 $NK - 1$ 번 이루어집니다. 차선 변경 횟수를 구하는 데에 집중합시다.
- 완주가 가능한 상황이라면 매번 장애물이 나오기 직전까지 직진한 후 차선 변경을 하는 것이 최선입니다.

2F. 현대모비스 자율 주행 테스팅 1

- 완성된 샘플 주행 트랙에 장애물이 T 개 있다고 가정한 후, 장애물의 행 번호를 나타낸 수열을 $R_1, \cdots, R_T$ 라 합시다.
- $R_i \neq R_{i+1}$ 인 i 의 개수가 차선 변경의 횟수가 됩니다. K 가 크기 때문에 수열을 직접 저장할 수는 없다는 데에 유의합시다.
- 샘플 주행 트랙에 대한 수열 R 에서의 답을 X 라 하면 $X \times K$ 가 답이 됩니다. 이때 $R_T \neq R_1$ 이라면 여기에 $K - 1$ 을 더해주어야 합니다.
- 시간복잡도는 $\mathcal{O}(N)$ 입니다.

1D. 현대모비스 자율 주행 테스팅 2

`greedy`, `implementation`, `case_work`
출제진 의도 – **Medium**

- 통계:
  - Senior: 제출 93번, 정답 10번, 정답률 10.753%
- First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 41분
- 출제자: 장래오[leo020630]

1D. 현대모비스 자율 주행 테스팅 2

- 기본적으로는 현대모비스 자율 주행 테스팅 1 문제의 풀이를 따라갑니다.
- 단, 샘플 주행 트랙의 종류가 여럿이기 때문에 유의해서 처리해야 하는 상황이 늘어납니다.
- 샘플 주행 트랙을 이어붙이는 과정에서 장애물이 없는 샘플 주행 트랙이 있거나, 아예 사용하지 않는 샘플 주행 트랙이 있을 수 있음에 유의합시다.
- 시간복잡도는 $\mathcal{O}\left(K + \sum N_i\right)$ 입니다.

2G. Twitch Plays Pokemon

`dp`, `graph_traversal`
출제진 의도 – **Medium**

- 통계:
  - Junior: 제출 2번, 정답 1번, 정답률 50.000%
- First Solve:
  - Junior: **곽민성**POSTECH, 124분
- 출제자: 한동규`queued_q`

2G. Twitch Plays Pokemon

- 달구의 채팅 중 가장 앞 a개와 포닉스의 채팅 중 가장 앞 b개에 적힌 명령을 사용했을 때, 캐릭터카 최종적으로 x 행 y 열에 위치하는 상황을 가정해 봅시다.
- 이러한 상태를 (a, b, x, y)라고 표현하겠습니다.
- 캐릭터는 다음으로 달구의 $a+1$번째 명령 또는 포닉스의 $b+1$번째 명령을 따를 수 있습니다. 각각에 대해 캐릭터의 다음 좌표를 (x_D, y_D), (x_P, y_P)라고 합시다.
- 현재 상태 (a, b, x, y)에서 이동할 수 있는 상태는 $(a+1, b, x_D, y_D)$ 또는 $(a, b+1, x_P, y_P)$입니다.

2G. Twitch Plays Pokemon

- ✓ 상태 사이의 관계를 그래프로 생각해서 BFS 탐색을 진행합니다.
- ✓ 이때, a, b, x, y에 대한 4차원 방문 배열을 사용해야 함에 유의합니다. 이유는 다음과 같습니다.
  - 채팅을 재배열하는 순서에 따라 a, b개의 명령을 사용해서 (x, y)에 도달하는 방법이 다양할 수 있으므로, 같은 상태를 중복해서 탐색하지 않기 위해 방문 배열이 필요합니다.
  - 탈출 경로가 동일한 (x, y) 좌표를 여러 번 방문할 수 있으므로 x, y에 대한 2차원 방문 배열로는 문제를 해결하기 어렵습니다.
- ✓ 가장 먼저 목적지에 도달한 상태의 a, b 값에 대해 $a + b$가 답이 됩니다.
- ✓ 문자열의 최대 길이를 L이라고 하면, 상태의 수가 N^2L^2 이므로 총 시간복잡도는 $\mathcal{O}\left(N^2L^2\right)$ 입니다.

2H. 산수화

`dp`, `prefix_sum`
출제진 의도 – **Hard**

- 통계:
  - Junior: 제출 1번, 정답 0번, 정답률 0.000%
- First Solve:
  - Junior: -
- 출제자: 장래오 `leo020630`

2H. 산수화

- (i, j) 를 중심으로 하는 산의 최대 크기를 $M_{i,j}$, (i, j) 를 왼쪽 위 꼭짓점으로 하는 호수의 최대 크기를 $L_{i,j}$ 라 합시다.
- 산의 형태가 가지는 특성으로, (i, j) 를 중심으로 하는 크기 $1, 2, \cdots, M_{i,j}$ 의 산이 모두 존재합니다. 호수의 경우에도 동일합니다.
- 따라서 $M_{i,j}$ 와 $L_{i,j}$ 를 모두 구하면 정답은 imos 법이라 불리는 누적합 테크닉을 이용해 계산할 수 있습니다.

2H. 산수화

- $M_{i,j}$ 와 $L_{i,j}$ 는 다음과 같은 DP를 통해 구할 수 있습니다.
- $$M_{i,j} = \begin{cases} \min(M_{i+1,j-1}, M_{i+1,j}, M_{i+1,j+1}) + 1 & \text{if } S_{i,j} = black \\ 0 & \text{if } S_{i,j} = white \end{cases}$$
- $$L_{i,j} = \begin{cases} \min(L_{i,j-1}, L_{i,j-1}, L_{i-1,j-1}) + 1 & \text{if } S_{i,j} = white \\ 0 & \text{if } S_{i,j} = black \end{cases}$$
- 이외에도 다양한 방법으로 DP를 정의해 해결할 수 있습니다.
- 시간복잡도는 $\mathcal{O}(NM)$ 입니다.

1F. 무빙워크

`dijkstra`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 19번, 정답 8번, 정답률 42.105%
- First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 67분
- 출제자: 장래오[leo020630]

1F. 무빙워크

- ✓ 1번 빌딩과 연결된 무빙워크들을 살펴봅시다.
- ✓ $u = 1$인 경우 (1번 빌딩에서 나가는 방향)
  - 이때는 무빙워크의 전원을 키는 것이 항상 이득입니다.
- ✓ $v = 1$인 경우 (1번 빌딩으로 들어오는 방향)
  - 이때는 무빙워크의 전원을 끄는 것이 항상 이득입니다. 1번 빌딩으로 돌아오는 것이 이득이 되지 않기 때문입니다.

1F. 무빙워크

- 이를 확장해 봅시다.
- 다익스트라 알고리즘을 사용합니다. 현재 탐색 중인 빌딩의 번호를 x라 합시다.
- 1번 빌딩의 경우와 마찬가지로 x번에서 나가는 방향의 무빙워크는 전원을 키는 것이, 들어오는 방향의 무빙워크는 전원을 끄는 것이 항상 이득입니다.
- x번 빌딩은 남은 빌딩 중 최단 도달 시간이 가장 작은 빌딩입니다. 따라서 x번 빌딩으로 다시 들어올 수 있는 방향의 발전은 항상 이득이 아님을 보일 수 있습니다.

- ✓ 최종적으로 구한 i번 빌딩의 최단 도달 시간을 D_i라 합시다.
- ✓ 가정에 의해, 무빙워크의 전원을 어떻게 조작하더라도 i번 정점에 대한 도달 시간이 D_i보다 짧아질 수 없습니다.
- ✓ 따라서 이 알고리즘은 모든 빌딩에 대한 최단 도달 시간의 합을 최소화합니다.

1F. 무빙워크

- 각 무빙워크에 대해 단방향 간선 (u, v, d)와 $(v, u, 2d)$를 그래프에 추가하고 다익스트라 알고리즘을 적용하면 위와 같이 동작하게 됩니다.
- 각 무빙워크의 전원 여부를 구하는 것은 $D_{u_i} < D_{v_i}$ 일 경우 1, 그렇지 않을 경우 0입니다.
- 시간복잡도는 $\mathcal{O}(MlogN)$ 입니다.

1G/2I. 수강신청

`greedy`, `sorting`
출제진 의도 – **Hard**

- ✓ 통계:
  - Junior: 제출 0번, 정답 0번, 정답률 0.000%
  - Senior: 제출 19번, 정답 2번, 정답률 10.526%
- ✓ First Solve:
  - Junior: -
  - Senior: **Mansur Mukimbekov**[UNIST], 82분
- ✓ 출제자: 김도훈[99asdfg]

1G/2I. 수강신청

- ✓ 우선, $a_i \geq b_i$ 인 과목들은 자명하게 달구가 수강신청에 성공합니다.
- ✓ 정답에 그러한 경우의 수만큼 더해준 뒤, $a_i < b_i$ 인 경우만을 고려합시다.
- ✓ 또한, 항상 최악의 경우만을 상정할 것이므로, 달구의 우선순위에 맞춰 다른 학생들이 수강신청을 한다고 생각해도 좋습니다. 즉, 달구와는 다르게 다른 학생들은 상황에 맞춰 우선순위를 변경하는 것이 허용된다고 생각해도 좋습니다.
- ✓ 이후 작은 관찰을 통해, $a_i < b_i$ 인 경우에는 b_i 는 더 이상 고려하지 않아도 된다는 사실을 알 수 있습니다.
  - 항상 달구가 수강신청을 최소로 하도록 하는 최악의 경우에는 다른 학생들이 수강신청을 실패하는 경우가 없어야 하기 때문입니다.
- ✓ 따라서, 이 이후의 관찰은 b_i 와는 독립적으로 진행할 수 있습니다.

1G/2I. 수강신청

- $a_i \geq b_i$ 인 경우를 제외하고 남은 과목들의 개수가 K 라고 합시다.
- 달구의 우선순위가 주어졌을 때, 최악의 경우를 구하는 문제를 생각해 봅시다.
- 다른 학생들의 입장에서 생각했을 때, 달구가 신청에 성공할 수 있는 과목에는 아무도 수강신청을 하지 않는 것이 이득입니다.
- 따라서, 달구가 정한 우선순위에 따라 달구가 수강할 수 있는 x 개의 강의를 제외한 $K - x$ 개의 강의의 정원을 채워버리는 것이 최악의 경우입니다.
- 즉, 최악의 경우는 항상 달구가 수강하지 못하는 $K - x$ 개 과목의 수강 정원의 합이 $(M - 1) \times x$ 이하가 되는 최소 x 를 찾는 문제로 환원할 수 있습니다.

1G/2I. 수강신청

- ✓ 이를 달구의 입장에서 보면, 달구의 최적의 전략은 달구가 수강할 수 있는 과목의 수강 정원의 합을 최소로 하는 것임을 알 수 있습니다.
- ✓ 수강 정원을 오름차순으로 정렬한 것을 순서대로 $a_1, a_2, \cdots, a_K$ 라 합시다.
- ✓ 달구는 항상 수강 정원의 오름차순으로 수강신청을 하는 것이 최적입니다.
  - 만약 달구가 $i+1$ 번 과목을 i 번 과목보다 우선적으로 신청한다면, 다른 학생의 입장에선 달구가 수강할 수 없는 과목의 수강 정원이 줄어든 것과 동일하기 때문입니다.
- ✓ 따라서, 달구는 항상 수강 정원이 적은 과목 먼저 수강신청하고, 다른 학생들은 달구가 신청하지 않은 과목들 중 수강 정원이 가장 적은 것을 우선적으로 신청하는 것이 서로의 최적의 전략임을 알 수 있습니다.

1G/2I. 수강신청

- 구현에 주의할 점은 다음과 같습니다.
  - 달구 또한 주어진 M 명에 포함되므로, 다른 학생들은 매초 최대 $M-1$ 개의 정원만 채울 수 있음에 유의합시다.
  - 다른 학생들이 매초 수강 정원이 적은 과목에 정원을 채우는 과정에서, 현재 시각에 정원을 전부 채울 수 없다면 다음 과목을 먼저 채워넣어야 합니다.
  - 문제 조건 상, 연속하여 두 개의 과목의 정원을 채우지 못하는 일은 없으므로 편한 방식으로 구현해 줄 수 있습니다.
- 이를 구현해 주면, $\mathcal{O}(N \log N)$ 으로 문제를 해결할 수 있습니다.

1H. Hyper Tree Problem

`trees`, `disjoint_set`, `bitmask`
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 7번, 정답 1번, 정답률 14.286%
- First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 133분
- 출제자: 박신욱[bnb2011]

1H. Hyper Tree Problem

- 각 간선의 가중치 범위가 $[0, 1]$ 이라고 생각해 봅시다.
- 1번 쿼리가 주어지면 bitwise AND 연산의 특징에 따라 가중치가 1 에서 0 으로만 변할 수 있습니다.
- 가중치 범위를 $[0, 1]$ 로 제한했으므로, $dist(i, j)$ 로 가능한 값 또한 $0, 1$ 중 하나입니다.
- $dist(i, j)$ 가 0 이 되기 위해선 i 번 정점과 j 번 정점 사이의 모든 간선 가중치가 0 이어야 하고, $dist(i, j)$ 가 1 이 되기 위해선 반대로 i 번 정점과 j 번 정점 사이의 간선 가중치 중 1 인 것이 하나라도 있어야 합니다.
- $dist(i, j) = 0$ 이 되는 경우의 조건이 더 다루기 쉽다는 점에 집중합니다.

1H. Hyper Tree Problem

- 2번 쿼리가 들어올 때마다 $dist(n, i) = 0$에 해당하는 i번 정점의 개수를 빠르게 셀 수 있다면 $\sum_{i=1}^{N} dist(n, i)$ 또한 구할 수 있습니다.
- 이는 가중치가 0인 간선으로만 이루어진 연결 컴포넌트의 크기를 관리한다면 $\mathcal{O}(1)$ 시간에 답할 수 있습니다.
- 1번 쿼리에 의해서 가중치가 0인 간선의 개수는 항상 증가합니다. 따라서 분리 집합을 이용하면 $\mathcal{O}(1)$ 시간에 컴포넌트의 연결 정보를 관리할 수 있습니다.

1H. Hyper Tree Problem

- 원래 가중치 범위에서도 위 내용을 확장해서 적용할 수 있습니다. 모든 연산이 비트 독립적이라는 사실을 이용합니다.
- 따라서 각 비트에 대해 답을 계산한 뒤 합쳐주면 $\mathcal{O}\left(logW\right)$ 시간에 쿼리를 답할 수 있습니다.
- 시간복잡도는 $\mathcal{O}\left(N + Q\log W\right)$ 입니다.

11. 포닉스와 달구

dp
출제진 의도 – **Hard**

- 통계:
  - Senior: 제출 15번, 정답 2번, 정답률 13.333%
- First Solve:
  - Senior: **Mansur Mukimbekov**[UNIST], 158분
- 출제자: 박신욱[bnb2011]

- ✓ $LU[r][c]$를 $(1,1)$에서 (r,c)로 이동하는 경로에 포함된 최대 가중치 합으로 정의합니다.
- ✓ $RD[r][c]$를 (r,c)에서 (N,N)으로 이동하는 경로에 포함된 최대 가중치 합으로 정의합니다.
- ✓ 위 두 값을 이용하면 $(1,1)$에서 (r,c)를 거쳐 (N,N)으로 이동하는 경로에 포함된 최대 가중치 합을 구할 수 있습니다.
- ✓ 이는 $dp[r][c] = LU[r][c] + RD[r][c] - A_{r,c}$처럼 계산 가능합니다.

- ✓ 달구가 $[i, i+K-1)$ 행과 $[j, j+K-1)$ 열에 해당하는 칸을 윤이에게 선물로 준 경우를 생각해 봅시다.
- ✓ 포닉스가 해당 영역을 지나지 않는다면, 반드시 $(i+K, 1), (i+K, 2), \cdots, (i+K, j-1)$ 또는 $(i-1, j+K), (i-1, j+K+1), \cdots, (i-1, N)$ 칸들 중 하나를 지나야만 합니다.
- ✓ 따라서 이 경우에 포닉스가 얻을 수 있는 최대 점수는 $dp[i+K][1], dp[i+K][2], \cdots, dp[i+K][j-1]$ 과 $dp[i-1][j+K], dp[i-1][j+K+1], \cdots, dp[i-1][N]$ 중 최댓값입니다.
- ✓ 전자는 dp 의 Row Prefix Max 를, 후자는 dp 의 Row Suffix Max 를 전처리해두면 $\mathcal{O}(1)$ 에 계산할 수 있습니다.

- ✓ 달구가 선물로 줄 영역을 고르는 경우의 수는 $\mathcal{O}\left(N^2\right)$ 이고, 각 경우에서 포닉스가 얻을 수 있는 최대 점수를 $\mathcal{O}\left(1\right)$ 시간에 계산할 수 있음을 보였습니다.
- ✓ 달구는 포닉스의 점수를 최소로 하길 원하므로, $\mathcal{O}\left(N^2\right)$ 개의 영역 중 포닉스가 얻을 수 있는 점수가 가장 작아지는 영역을 선택합니다.
- ✓ 따라서 $\mathcal{O}\left(N^2\right)$ 시간에 문제를 해결할 수 있습니다.