2022 홍익대학교 HI-ARC 프로그래밍 경진대회 풀이

Official Solutions

by

홍익대학교 HI-ARC 프로그래밍 경진대회 출제진

2022년 11월 18일

문제		의도한 난이도	출제자
A	HI-ARC	**Easy**	swoon
B	나뭇잎 학회	**Easy**	xkdlaldfjtnl
C	K-Queen	**Medium**	xhdtlsid2
D	Codepowers	**Medium**	hwon233
E	해시 해킹	**Advanced**	Green55
F	험난한 등굣길	**Advanced**	xhdtlsid2
G	돈 피하지 않기 게임	**Hard**	Green55

A. HI-ARC

`implementation`

출제진 의도 – **Easy**

- ✓ 제출 66번, 정답 45명 (정답률 69.697%)
- ✓ 처음 푼 참가자: **김건회**, 1분
- ✓ 출제자: swoon

A. HI-ARC

- ✓ 문자열에 포함된 H, I, A, R, C 문자의 개수를 세어 줍니다.
- ✓ HIARC 이모지는 가장 적게 등장한 문자의 개수만큼만 만들 수 있습니다.
- ✓ 따라서 정답은 H, I, A, R, C 각 문자의 개수의 최솟값이 됩니다.
- ✓ 반복문을 한 번만 돌려 문자의 개수를 구할 수 있으므로, 시간복잡도 $\mathcal{O}(N)$에 해결할 수 있습니다.

B. 나뭇잎 학회

`math`

출제진 의도 – **Easy**

- ✓ 제출 72번, 정답 38명 (정답률 52.778%)
- ✓ 처음 푼 참가자: **한승준**, 5분
- ✓ 출제자: `xkdlaldfjtnl`

- ✓ 두 개의 스위치를 눌렀을 때 전구가 깜빡였다면 그 두 개의 스위치 중 하나가 연결된 것입니다.
- ✓ 전구가 깜빡이지 않았다면 두 개의 스위치 모두 전구와 연결되어 있지 않습니다.
- ✓ 너무나도 당연한 소리죠.
- ✓ 우리는 이 사실을 이용해서 문제에 접근할 겁니다.

B. 나뭇잎 학회

- A 스위치와 B 스위치를 눌렀는데 전구가 깜빡였다고 생각해봅시다.
- 우리는 추가 행동 한 번으로 전구와 연결된 스위치가 무엇인지 찾을 수 있습니다.
- A 스위치와 B 스위치 둘 중의 하나만 포함해 스위치를 누르면 연결된 스위치를 알 수 있습니다.
- 예를 들어서 A와 C를 눌렀는데 전구가 깜빡인다면 전구와 연결된 스위치는 A, 깜빡이지 않는다면 B가 되게 됩니다.
- 따라서 전구가 연결된 스위치 쌍을 찾을 때까지 이전 누른 것은 제외하고 눌러볼 겁니다.

B. 나뭇잎 학회

- ✓ 문제에서 말하는 필요한 나뭇잎 개수의 최대는 언제가 될까요?
- ✓ 운이 최대로 나빠서 마지막에 찾으면 필요한 나뭇잎 개수가 최대가 될 것 같습니다.
- ✓ 따라서 최대 개수는 모든 스위치를 누르는데 필요한 횟수 + 1이 됩니다.
- ✓ 맞을까요?

- ✓ 아닙니다.

- ✓ 마지막 두 개의 스위치만 남겨두고도 전구가 깜빡이지 않았다고 생각해봅시다.
- ✓ 우리는 굳이 눌러보지 않아도 두 개의 스위치 중 하나가 전구와 연결되었다는 것을 알 수 있습니다.
- ✓ 따라서 이 상황에서 한 번만 더 눌러보면 됩니다.

- ✓ N이 홀수인 경우는 살짝 다릅니다.
- ✓ 마찬가지로 마지막 하나의 스위치만 남겨두고도 전구가 깜빡이지 않았다고 생각해봅시다.
- ✓ 짝수인 상황과는 다르게 추가 행동 필요 없이 해당 스위치가 전구와 연결되어 있다는 사실을 알 수 있습니다.

- ✓ 아직 안 누른 스위치가 세 개가 남았고, 두 개의 스위치를 눌렀을 때 전구가 깜빡인 경우가 위의 경우에서 추가 행동이 필요한 것을 알 수 있고 이 경우가 최악입니다.

- ✓ 따라서 N이 짝수일 때는 $\frac{N^2}{2}$가 정답이고,
- ✓ 1이 아닌 홀수일 때는 $\frac{N^2-1}{2}+1$가 정답이 됩니다.

C. K-Queen

`case_work`

출제진 의도 – **Medium**

- ✓ 제출 81번, 정답 18명 (정답률 22.222%)
- ✓ 처음 푼 참가자: **김범수**, 20분
- ✓ 출제자: `xhdtlsid2`

C. K-Queen

- ✓ a 행 b 열에 위치한 퀸이 c 행 d 열을 공격할 수 있는 경우를 식으로 나타내면 다음과 같습니다.
  1. $a = c$ (같은 행에 위치한 경우)
  2. $b = d$ (같은 열에 위치한 경우)
  3. $|a - c| = |b - d|$ (같은 대각선에 위치한 경우)
- ✓ 어떤 특정한 칸을 공격할 수 있는 퀸이 존재하는지 확인하려면, 입력으로 주어진 각 퀸에 대하여 위 세 가지 조건을 체크해 주면 됩니다.
- ✓ 즉, 어떤 특정한 칸이 퀸에 의해 공격받고 있는지 여부를 $\mathcal{O}(K)$ 에 알 수 있습니다.

C. K-Queen

- 이제, 두 조건 X, Y 를 다음과 같이 정의해 봅시다.
  - $X :=$ 킹이 위치한 칸이 공격받고 있다.
  - $Y :=$ 킹과 8방향으로 인접한 칸들 중 공격받고 있지 않은 칸이 존재한다.
- 문제에서 설명한 각 상태를 X, Y 를 이용하여 나타내면 다음과 같습니다.
  - 체크 $\Leftrightarrow X$ 가 참이고, Y 도 참
  - 체크메이트 $\Leftrightarrow X$ 가 참이고, Y 는 거짓
  - 스테일메이트 $\Leftrightarrow X$ 가 거짓이고, Y 도 거짓
- X 와 Y 의 값을 알아내기 위해 공격받고 있는지 확인해야 할 칸은 $\mathcal{O}(1)$ 개이므로, 전체 시간복잡도 $\mathcal{O}(K)$ 에 문제를 해결할 수 있습니다.

D. Codepowers

`prefix_sum`

출제진 의도 – **Medium**

- ✓ 제출 88번, 정답 22명 (정답률 25%)
- ✓ 처음 푼 참가자: **김혁진**, 23분
- ✓ 출제자: hwon233

D. Codepowers

- 쿼리를 계산하려면, 먼저 각 라운드에 참여한 후 레이팅 점수를 알아야 합니다.
- 문제에서는 초기 점수 X 와 증감 수열 A 만 주어지므로, 이를 사용하여 레이팅 점수 수열 B 를 정의해봅시다.
- 수열 B 의 원소 B_i 는 i 번째 라운드에 참여한 직후의 레이팅 점수입니다.
- 다음과 같이 초기 점수에 증감을 누적하여 만들 수 있습니다.

$$B_i = \begin{cases} X + A_i, & \text{if } i = 1 \\ B_{i-1} + A_i, & \text{if } 2 \le i \le N \end{cases}$$

- ✓ 이렇게 구한 수열 B 를 사용하여 M 개의 쿼리를 구해봅시다.
- ✓ 각 쿼리마다 직접 B_{l_j} 에서부터 B_{r_j-1} 까지 모든 원소를 방문하여 K 와 비교하면 어떨까요?
- ✓ 구간의 길이는 최대 N 이고 쿼리가 M 개이므로 $\mathcal{O}(NM)$ 이 걸릴 것입니다.
- ✓ 이는 제한 시간 안에 수행할 수 없습니다. 더 빠른 방법이 필요합니다.

D. Codepowers

- ✓ 수열 B의 각 원소들과 K를 미리 비교해서 수열 C에 저장해둡시다.
- ✓ 다음과 같이 수열 C를 만들 수 있습니다.

$$C_i = \begin{cases} 1, & \text{if } B_i < K \\ 0, & \text{if } B_i \geq K \end{cases}$$

- ✓ C_{l_j}에서부터 C_{r_j-1}까지 합을 구하면, B_{l_j}에서부터 B_{r_j-1}까지 원소들 중 K보다 작은 원소의 개수를 구할 수 있습니다.
- ✓ 즉, 원래 문제를 수열 C에서 구간 합을 구하는 문제로 변환한 것입니다.

D. Codepowers

- ✓ 누적 합(prefix sum)을 사용하면 구간 합을 빠르게 구할 수 있습니다.
- ✓ 누적 합을 사용할 수 있는 조건은
  - 수열의 원소가 변하지 않아야 하며,
  - 연산이 결합 법칙이 성립해야하고,
  - 연산의 역연산이 존재해야 합니다.

  수열 C의 경우 세 가지 모두 만족합니다.

D. Codepowers

- 수열 C에 대한 누적 합 S를 다음과 같이 정의할 수 있습니다.

$$S_i = \begin{cases} 0, & \text{if } i = 1 \\ S_{i-1} + C_{i-1}, & \text{if } 2 \le i \le N+1 \end{cases}$$

- 예를 들어, C_3에서 C_7까지 합을 구하고자 한다면,
  - $S_8 = C_1 + C_2 + C_3 + C_4 + C_5 + C_6 + C_7$
  - $S_3 = C_1 + C_2$
  - $S_8 - S_3 = C_3 + C_4 + C_5 + C_6 + C_7$

  이렇게 두 누적 합의 차를 사용해서 구할 수 있습니다.

- 문제에 적용해보면,

$$(C_{l_j}\text{에서부터 } C_{r_j-1}\text{까지 합}) = S_{r_j} - S_{l_j}$$

  입니다.
- 쿼리를 계산하기 전에 미리 누적 합 S를 구해두고, 각 쿼리마다 한 번만 연산하면 쿼리의 답을 구할 수 있습니다.
- 이 방법을 사용하면 $\mathcal{O}(N+M)$으로 제한 시간 안에 수행할 수 있습니다.

E. 해시 해킹

`math`

출제진 의도 – **Advanced**

- ✓ 제출 9번, 정답 1명 (정답률 11.111%)
- ✓ 처음 푼 참가자: **김범수**, 54분
- ✓ 출제자: `Green55`

- ✓ P_0 이 아닌 P_i 는 A^i 가 곱해져서 값을 예측하기 힘들지만
- ✓ P_0 는 $A^0 = 1$ 이 곱해지기 때문에 $h(P)$ 에 그대로 더해진다는 것을 주목해봅시다.

- ✓ P_0 가 $h(P)$ 의 값을 컨트롤 하기 위한 수단이라는 느낌입니다.

E. 해시 해킹

- ✓ P_0 만 제외한 해시값, $(P_1 \cdot A^1 + ... + P_{N-1} \cdot A^{N-1}) \mod M$ 을 x 라고 해봅시다.
- ✓ $h(P) = (P_0 + x) \mod M$ 입니다.

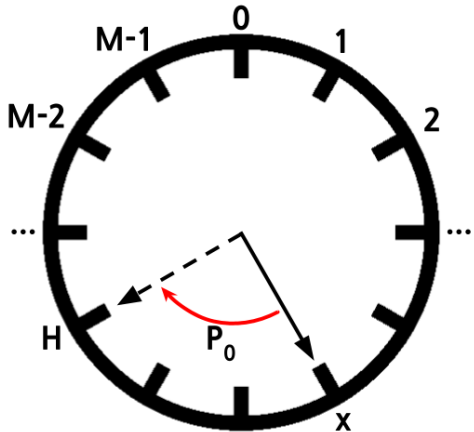
- ✓ 그런데 $0 \leq P_0 < M$ 이므로, P_0 만 잘 설정해주면
- ✓ x 의 값에 상관 없이 $h(P)$ 를 원하는 값으로 만들 수 있습니다!

- ✓ 즉, $h(P) = (P_0 + x) \mod M = H$ 로 만들기 위해서는 $(0 \leq H < M)$
- ✓ $P_0 = (H - x + M) \mod M$ 으로 정하면 됩니다.

- ✓ $0 \leq x, H, P_0 < M$ 이라는 제한 범위를 생각해보면
- ✓ x 와 H 가 정해졌을 때, 이를 만족하는 P_0 의 값이 정확히 한 가지라는 것을 알 수 있습니다.

E. 해시 해킹

- 그림으로 표현하면 훨씬 직관적입니다.
- 눈금이 M 칸 있는 시계에서
- 현재 침이 x 를 가르키고 있습니다.
- 딱 한번 침을 $0 \leq P_0 < m$ 칸 돌릴 수 있으면
- 원하는 칸(H)으로 침을 옮길 수 있습니다.
- H 로 침을 옮기는 P_0 는 딱 1 개 밖에 없습니다.



E. 해시 해킹

- 결론적으로, $P_1, P_2, \cdots, P_N - 1$ 이 어떤 값을 갖든 상관 없이
- $h(P) = H$ 로만들어 주는 P_0 의 값은 정확히 1 개입니다.
- 따라서 답은 '아무' $P_1, P_2, \cdots, P_N - 1$ 의 개수입니다
- $0 \leq P_i < M$ 이므로, M 개의 정수 중 하나를 고르는 것을 $N - 1$ 번 하는 경우의 수 = M^{N-1}
- Python에서는 `pow(M,N-1,10**9+7)` 한 줄로 쉽게 구현 가능합니다.

- ✓ 수식이 비교적 간단하므로, 백트래킹이나 DP를 짜서 규칙을 찾아도 풀 수 있습니다.
- ✓ 가능한 모든 P의 M^N 가지의 해시값이 0 이상 M 미만의 정수에 $\frac{1}{M}$씩 균등하게 분포한다는 것만 알아내면 풀 수 있는 문제였습니다.

F. 험난한 등굣길

`ad_hoc`, `bfs`

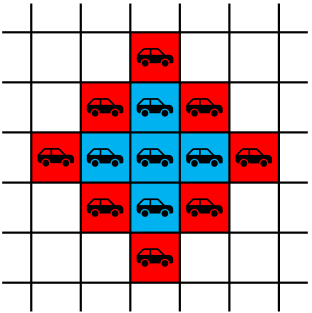
출제진 의도 – **Advanced**

- ✓ 제출 47번, 정답 1명 (정답률 2.128%)
- ✓ 처음 푼 참가자: **김범수**, 87분
- ✓ 출제자: `xhdtlsid2`

- ✓ 교통 정체가 일어나고 있는 칸들을 모두 마킹해 놓는다면, 너비 우선 탐색을 이용하여 쉽게 답을 구할 수 있습니다.
- ✓ 단순하게 각 정체 구역에 대해서, 해당 정체 구역에 속하는 칸들을 하나씩 전부 마킹하는 방법을 생각해 봅시다.
- ✓ i번째 정체 구역을 처리할 때는 $\mathcal{O}({D_i}^2)$ 개의 칸을 마킹하게 됩니다.
- ✓ 따라서 모든 정체 구역을 처리하려면 총 $\mathcal{O}(K{D_{\max}}^2)$ 번 마킹해야 하는데, 마킹 횟수가 너무 많으므로 제한 시간 안에 수행할 수 없습니다.

F. 험난한 등굣길

- ✓ 정체 구역의 모양을 관찰해 봅시다.
- ✓ 아래 $D = 2$인 정체 구역에서, 파란색 영역에 도달하기 위해서는 반드시 빨간색 칸을 지나야 함을 알 수 있습니다.



- ✓ 즉, 어떤 정체 구역에 속한 칸들을 마킹할 때는 위의 빨간색 칸과 같이 테두리에 해당하는 칸들만 마킹해도 충분합니다.
- ✓ i번째 정체 구역의 테두리에는 $\mathcal{O}(D_i)$개의 칸이 있으므로, 모든 정체 구역을 처리하는 데 총 $\mathcal{O}(KD_{\max})$번의 마킹을 수행하게 됩니다.
- ✓ 마킹이 끝난 뒤 너비 우선 탐색을 하면 전체 시간복잡도 $\mathcal{O}(KD_{\max}+NM)$에 문제를 해결할 수 있습니다.

- ✓ 우선순위 큐를 이용한 multi-source BFS나 difference array를 이용하여 풀 수도 있습니다.

G. 돈 피하지 않기 게임

dp

출제진 의도 – **Hard**

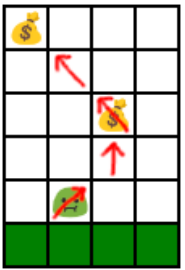
- ✓ 제출 0번, 정답 0명 (정답률 0%)
- ✓ 출제자: Green55

G. 돈 피하지 않기 게임

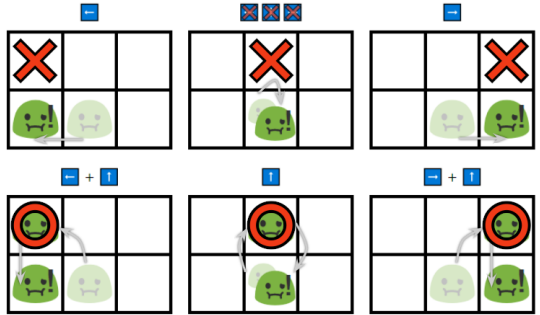
- ✓ 문제가 복잡하므로, 점프가 없는 경우부터 생각해봅시다.
- ✓ 같은 층에 (y가 동일한) 돈이 2개 이상 있다면, 모든 돈을 모으는 것은 **불가능**합니다.
- ✓ 그렇지 않다면, 즉 모든 돈의 y가 다르다면
  - 돈을 모으게되는 순서는 y의 오름차순입니다.
  - 현재 (x_1, y_1)에 있고, 다음에 먹을 돈이 (x_2, y_2)라면
  - $y_2 - y_1$초 안에 x_1에서 x_2로 이동해야합니다.

G. 돈 피하지 않기 게임

- ✓ 각 칸마다 $\nwarrow, \nearrow, \uparrow$를 배치해서 모든 돈을 잇는 경로를 만듭니다.
- ✓ $\nwarrow, \nearrow$는 하나당 P_{lr}의 비용이 필요합니다.



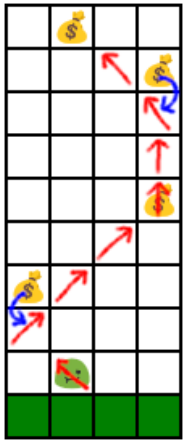
G. 돈 피하지 않기 게임



- ✓ 이제 점프에 대해 생각해봅시다. 점프를 했을 때와 하지 않았을 때의 차이를 살펴보면
- ✓ 점프를 하면 도착한 칸 바로 윗칸의 돈을 '추가로' 먹습니다.
- ✓ 그렇다면 아예 점프를 '돈을 한 칸 내리는 연산'으로 생각합시다.

G. 돈 피하지 않기 게임

- ✓ 각 돈마다 최대 한번, P_j 의 비용을 사용해서
- ✓ 돈을 바로 아래($(x, y) \to (x, y-1)$)로 내릴 수 있습니다.



- ✓ 점프 연산이 생겼어도, 여전히 돈을 모으게되는 순서는 y의 오름차순입니다.
  (점프 연산을 쓰기 전의 초기 y값 기준)
  - 예를 들어 초기에 $y = 9, 10$에 돈이 하나씩 있을 때
  - $y = 10$에 있는 돈을 아래로 내렸으면
  - 최종적으로 같은 층에 돈은 하나씩 있어야 하기 때문에
  - $y = 9$에 있는 돈도 반드시 아래로 내려야합니다.

- ✓ 물론 이제는 같은 층에 돈이 2개 있어도 불가능이 아닐 수 있으며,
- ✓ 이 경우 둘 중 어느 것을 먼저 모을지는 확실하지 않습니다.

- ✓ 하지만 같은 층에 돈이 3개 이상 있다면 확실히 불가능합니다.
- ✓ 따라서 같은 층에 돈이 2개 이하 있는 경우만 해결하면 됩니다.

- ✓ 지금까지의 내용들로 2개의 핵심적인 관찰이 나옵니다.
- ✓ 관찰 1. 돈을 모을 수 있는 좌표의 후보는 최대 $2N$ 개입니다.
  - 초기에 (x, y) 에 있는 돈은, (x, y) 에서 모으거나
  - 점프 연산을 사용하여 $(x, y-1)$ 에서만 모을 수 있습니다

- ✓ 관찰 2. 어떤 돈을 모은 직후, 다음에 모을 돈의 후보는 최대 2개입니다.
  - 마지막으로 모은 돈의 초기 위치가 (x', y') 라고 할 때
  - y 좌표가 y' 와 동일한 돈이 있고, 그 돈을 아직 모으지 않았다면..
    - ▶ 다음 모을 돈은 반드시 그 돈입니다. (y 의 오름차순으로 돈을 모아야 함)
  - 그 이외의 경우에는..
    - ▶ '바로 윗 층'에 있는 돈 (최대 2개) 중 하나를 모아야 합니다.
    - ▶ 바로 윗 층은 y보다 크면서 가장 작은 위치에 존재하는 돈을 의미합니다.

- ✓ 이 관찰들을 이용해 다이나믹 프로그래밍으로 문제를 해결할 수 있습니다.
- ✓ 다음과 같은 정보를 하나의 '상태'로 정의합시다.
  - 마지막으로 모은 돈은 몇 번째 돈?
  - 마지막으로 모은 돈을 점프를 이용해서 아래로 내려서 모았는가?
  - 마지막으로 모은 돈과 초기 y가 동일한 돈이 있다면, 그 돈은 모았는가?
- ✓ 마지막으로 모은 돈보다 y가 작은 돈은 이미 전부 모았을 것이므로
  이 정보들만으로도 N개의 돈 중 어떤 돈을 모았고, 어떤 돈을 모으지 않았는지 알 수 있습니다.
- ✓ 관찰 1에 의하여, 상태의 개수는 $\mathcal{O}(N)$개입니다.

G. 돈 피하지 않기 게임

- ✓ 상태 전이는 관찰 2에 언급한 가능한 모든 다음 돈의 후보를 전부 고려해주면 됩니다.
- ✓ 물론 각 돈을 점프를 해서 모으는 것과 점프를 하지 않고 모으는 것 모두 고려해야 합니다.
- ✓ 상태 전이가 상수 개이고, 각 상태 전이를 계산하는 것도 상수 시간에 가능합니다.
- ✓ 따라서 전체 문제를 $\mathcal{O}(N)$에 해결할 수 있습니다.

- ✓ 설명한 방법 외에도 풀이의 디테일에서 다양한 방법이 가능하며,
  검수진분들의 풀이도 전부 상이했습니다.

- ✓ 편의상 미리 언급하지 않은 하나의 예외
  - $y = 1$에 있는 돈은 점프로 아래로 내려 먹을 수 **없습니다**. (예제 4)
  - 이 경우에만 점프를 하지 않도록 따로 처리해주면 됩니다.