2023 제 1회 HI-CON Solution

Official Solutions

홍익대학교 알고리즘 학회 HI-ARC

2023년 8월 12일

문제		의도한 난이도	출제자
A	HI-ARC=?	**Easy**	smjun04
B	알록달록 앵무새	**Easy**	dicohy27
C	볼링공 찾아주기	**Easy**	smjun04
D	마라탕 재료 고르기	**Medium**	swoon
E	광기의 PS	**Challenge**	smjun04
F	누가 이길까	**Hard**	swoon
G	컨벤 데드가 하고 싶어요	**Medium**	dicohy27
H	모기 킬러	**Hard**	swoon
I	탄막 게임	**Challenge**	tolelom

A. HI-ARC=?

`math`

출제진 의도 – **Easy**

- ✓ 제출 24번, 정답 19명 (정답률 79.167%)
- ✓ 처음 푼 사람: **xhdtlsid2**, 1분
- ✓ 출제자: `smjun04`

A. HI-ARC=?

- ✓ H, I, A, R, C 각각의 문자에 입력되는 수로 사칙연산을 하면 되는 문제입니다.
- ✓ 0솔 방지 목적으로 출제되었습니다.

B. 알록달록 앵무새

`set`

출제진 의도 – **Easy**

- ✓ 제출 26번, 정답 17명 (정답률 65.384%)
- ✓ 처음 푼 사람: **mastershim**, 3분
- ✓ 출제자: `dicohy27`

- ✓ 주어지는 4가지 색 중 서로 다른 색의 개수가 몇 개 인지에 따라서 조건을 나눌 수 있습니다.
- ✓ 예를 들어 서로 다른 색이 2가지인 경우 가능한 색 조합은 $(Color1, Color2), (Color2, Color1), (Color1, Color1), (Color2, Color2)$ 총 4가지 입니다.
- ✓ 이렇게 조건에 따라 가능한 색 조합을 구한 뒤 정렬해서 출력하면 됩니다.

- ✓ 하지만 set자료구조를 사용하면 굳이 조건을 나눌 필요가 없습니다.
- ✓ 왜냐하면 set에는 데이터가 중복되어 저장되지 않기 때문입니다. (수학에서 집합과 같습니다.)
- ✓ 또한 set에 데이터를 넣으면 특정 기준에 따라 자동으로 정렬이 됩니다. (C++의 경우 문자열은 사전 순으로 정렬됩니다.)
- ✓ 따라서 set에 16가지($_4C_2$)의 모든 색 조합을 집어넣고 순서대로 출력하면 문제에서 요구하는 정답이 됩니다.

C. 볼링공 찾아주기

`math`

출제진 의도 – **Easy**

- ✓ 제출 40번, 정답 16명 (정답률 40%)
- ✓ 처음 푼 사람: **mastershim**, 6분
- ✓ 출제자: `smjun04`

- ✓ w의 최대 크기가 10^9이기 때문에 10억개의 int 배열을 만들면 메모리 초과가 납니다.
- ✓ python의 dictionary나 C++의 map을 활용하면 문제를 해결할 수 있습니다.
- ✓ python의 dictionary를 활용하면 아래와 같이 풀 수 있습니다.
- ✓ 1번 요청이 들어왔을 때 dict[w] = x로 딕셔너리에 정보를 입력합니다.
- ✓ 2번 요청이 들어왔을 때 dict[w]를 출력해주기만 하면 됩니다.

D. 마라탕 재료 고르기

`dfs`

출제진 의도 – **Easy**

- ✓ 제출 32번, 정답 11명 (정답률 34.375%)
- ✓ 처음 푼 사람: **mastershim**, 10분
- ✓ 출제자: swoon

- ✓ N 개 중 K 개를 골라 고른 K 개 끼리의 궁합의 합의 최대를 구하는 문제입니다.
- ✓ N 개 중 K 개를 선택하는 것은 다음과 같이 dfs로 할 수 있습니다.
- ✓ 1 번 재료부터 N 번 재료까지 차례대로 넣는다고 한다면 함수는 다음과 같습니다.
  - $f(i, j) =$ 앞으로 j 번 더 재료를 선택할 수 있는데, i 번 재료를 선택할까 말까
  - 재료를 선택한다면 다음 재료를 선택하러 가며, 재료를 선택할 수 있는 횟수가 1회 줄어드니 $f(i+1, j-1)$ 를 호출하게 됩니다.
  - 재료를 선택하지 않는다면 다음 재료를 선택하러 가며, 재료를 선택할 수 있는 횟수는 그대로라서 $f(i+1, j)$ 를 호출하게 됩니다.

- ✓ $f(i, j)$에서 j가 0이면 어떻게 될까요?
  - 재료를 선택할 수 있는 모든 기회를 사용한거니, 선택한 재료끼리의 궁합의 합을 계산해주면 됩니다.
- ✓ $f(i, j)$에서 내가 앞으로 선택할 수 있는 재료의 개수가 j개보다 작으면 어떻게 될까요?
  - 그런 경우를 만들지 않기 위해 앞으로 선택 가능한 재료의 개수를 구해봅시다.
  - 재료의 총 개수 $= N$, 현재 고를까 하는 재료의 번호 $= i$, 현재 재료를 포함해 앞으로 고를 수 있는 재료의 개수 $= N - i + 1$
  - 따라서, $j == N - i + 1$이면 재료를 고르지 않고 넘어가는 선택지는 없어지게 됩니다.

- ✓ 위와 같은 방법으로 N 개 재료 중 K 개의 재료를 고르는 모든 조합을 찾은 뒤, 각 방법마다 마라탕의 맛을 구해주면 됩니다.
- ✓ 여기서, 마라탕의 맛의 최대가 음수일 수도 있다는 것을 유의해야합니다.
- ✓ 선택한 재료를 배열에 표기해둔다면 마라탕의 맛은 $\mathcal{O}\left(N^2\right)$ 으로 구할 수 있습니다.
- ✓ 시간복잡도는 $\mathcal{O}\left(N^2 \cdot {}_NC_K\right)$ 입니다.
- ✓ python은 itertools, c++은 next_permutation을 이용하여 보다 쉽게 조합을 찾을 수도 있습니다.
- ✓ 비슷한 문제로는 백준 N과 M 문제 시리즈가 있습니다.

E. 광기의 PS

`greedy`
출제진 의도 – **Challenge**

- ✓ 제출 16번, 정답 5명 (정답률 31.25%)
- ✓ 처음 푼 사람: **mastershim**, 33분
- ✓ 출제자: `smjun04`

- ✓ $K <= 5$인 문제는 풀었을 때 광기가 KT만큼 해소되므로 문제를 풀기 이전 상태와 광기가 동일합니다.
- ✓ 따라서 이러한 문제만을 해결하면 광기가 증가하지 않습니다.
- ✓ $K <= 5$인 문제는 광기가 0인 초기 상태에서 모두 해결해버립니다.

- ✓ $K > 5$인 나머지 문제는 해결했을 때 $K(T-5)$ 만큼의 광기가 쌓입니다.
- ✓ 현재 쌓인 광기를 G라고 하겠습니다.
- ✓ 만약 $(L-G)$가 해결해야 할 문제의 K_iT_i보다 작다면, 그 차만큼 휴식합니다.
- ✓ 이후 문제를 해결합니다.

- ✓ 총 휴식 시간은 모든 문제의 $K_i(T_i - 5)$ 합에 마지막 상태의 광기를 뺀 값입니다.
- ✓ 마지막으로 휴식한 시점 이후에 처음으로 푼 문제를 마지막 상태의 광기에 관여한 문제라고 하겠습니다.
- ✓ 마지막 상태의 광기에 관여한 문제의 K 가 작을수록 마지막 상태의 광기는 증가합니다.
- ✓ 즉, 휴식 시간이 줄어듭니다.

- ✓ K 가 큰 순서대로 정렬한 후 문제를 해결하면 항상 마지막 상태의 광기에 관여한 문제의 K 가 최소가 됩니다.
- ✓ 그리디하게 해결할 수 있습니다.

F. 누가 이길까

`sorting`, `binary_search`, `sweeping`
출제진 의도 – **Hard**

- ✓ 제출 50번, 정답 5명 (정답률 10%)
- ✓ 처음 푼 사람: **mastershim**, 22분
- ✓ 출제자: `swoon`

- ✓ HI팀과 ARC팀의 대결을 나이브하게 진행하면 NM 개의 대결을 진행하여 시간복잡도는 $\mathcal{O}(NM)$ 이 됩니다. 이 경우 시간 초과를 피할 수 없습니다.
- ✓ 먼저 HI팀의 참가자의 코딩실력을 나타내는 수열 a 를 정렬해봅시다.
- ✓ 정렬을 하면, 이분탐색을 사용하여 x 의 코딩실력을 가지고 있는 참가자가 몇 명까지 이길 수 있는지를 $\mathcal{O}(logN)$ 만에 알아낼 수 있습니다.

- ✓ x의 코딩실력을 가지고 있는 참가자가 HI팀의 참가자와 몇 번 무승부가 날 지를 알아내는 것은 수열 a에 x가 몇 개인지 알아내는 것과 같습니다.
- ✓ 수열 a를 맨 처음에 한 번 순회하면서 배열이나 map을 채워줍시다.
  - $arr[i] =$ HI팀에서 i의 코딩실력을 가진 사람의 수
- ✓ 이를 사용하여 x의 코딩실력을 가지고 있는 참가자가 몇 명과 비기는 지 $\mathcal{O}(1)$에 알아낼 수 있습니다.
- ✓ 마지막으로, 이기거나 비기는 것이 아니라면 지는 경우이므로 (HI팀 인원 수) $-$ (이기는 경기 수) $-$ (비기는 경기 수)를 통해 몇 명에게 지는 지 알아낼 수 있습니다.

- ✓ 수열 a를 정렬하는 것이 $\mathcal{O}\left(NlogN\right)$
- ✓ 수열 a의 각 원소의 개수를 저장하는 배열을 만드는 것이 $\mathcal{O}\left(N\right)$
- ✓ x의 코딩실력을 가진 사람이 몇 명을 이기는 지 찾는 것이 실행당 $\mathcal{O}\left(logN\right)$, 총 M번 반복하니 $\mathcal{O}\left(MlogN\right)$
- ✓ 따라서, 총 시간복잡도는 $\mathcal{O}\left(NlogN+MlogN\right)$이 됩니다.
- ✓ 답의 최댓값은 $1e10$이므로, 32비트 자료형(int)를 사용할 경우 오버플로우가 발생할 수 있음을 유의해야합니다.
- ✓ sweeping을 사용할 경우 $\mathcal{O}\left(N+M\right)$으로 해결할 수 있습니다.

G. 컨벤 데드가 하고 싶어요

`dp`
출제진 의도 – **Medium**

- ✓ 제출 14번, 정답 4명 (정답률 28.571%)
- ✓ 처음 푼 사람: **mastershim**, 48분
- ✓ 출제자: `dicohy27`

- ✓ 가장 간단한 풀이를 생각해 봅시다.
- ✓ 처음 위치인 $(1, 1)$에서 출발하여 (N, M) 위치에 도착하는 모든 경우에 대해 마주치는 헬창 형님들 눈치력의 총 합과 은신력을 비교하면 됩니다.
- ✓ 이동 방향은 오른쪽 또는 아래쪽 이므로 총 2^{NM} 가지 경우의 수를 탐색해보면 됩니다.
- ✓ 하지만 N 과 M 이 1 이상 1000 이하 이므로 경우의 수가 최대 $2^{1,000,000}$ 가지가 되어 제한 시간 안에 돌아가는 것은 불가능합니다.

- ✓ 이제 이 풀이를 개선해 봅시다.
- ✓ 우선 $dp[i][j]$ 를 $(1,1)$ 에서 (i,j) 에 도착할 때 마주치는 헬창 형님들 눈치력의 총 합의 최소값이라고 정의합시다.
- ✓ 현재 위치가 (i,j) 라면 이전 위치는 $(i-1,j)$ 또는 $(i,j-1)$ 입니다.
- ✓ 다시 말해 (i,j) 에 도착하기 위해서는 $(i-1,j)$ 또는 $(i,j-1)$ 를 거쳐야 합니다.
- ✓ 따라서 $dp[i][j]$ 를 구하려면 $(1,1)$ 에서 출발하는 $2^{i \times j}$ 가지의 모든 경로를 탐색할 필요 없이 $dp[i-1][j]$ 와 $dp[i][j-1]$ 만 알고 있으면 됩니다.

- ✓ dp 점화식은 다음과 같습니다.

$$dp[i][j] = min(dp[i-1][j], dp[i][j-1]) + board[i][j]$$

  - $board[i][j]$ 는 (i, j) 에 위치하는 헬창 형님의 눈치력을 뜻합니다.
- ✓ dp 테이블의 크기가 $N \times M$ 이고 테이블의 한 칸을 계산하는데 드는 시간복잡도는 $O(1)$ 이므로 총 시간 복잡도는 $O(NM)$ 입니다.

H. 모기 킬러

`greedy`
출제진 의도 – **Hard**

- ✓ 제출 1번, 정답 1명 (정답률 100%)
- ✓ 처음 푼 사람: **mastershim**, 135분
- ✓ 출제자: swoon

- ✓ 먼저 모기에 대해 생각해보면, 같은 칸에 있는 모기는 가장 체력이 많은 모기만 고려하면 됩니다.
- ✓ 그 뒤로 하얗이가 할 수 있는 행동을 하나씩 분석해봅시다.
- ✓ R동을 향해 1 만큼 이동한다.
  - 하얗이가 R동으로 1 만큼 이동하면, 모기는 하얗이를 향해 1 만큼 가까워집니다.
  - 하얗이와 모기 간의 거리는 변한 것이 없는데, 하얗이는 T동과의 거리만 멀어지고, 시간만 썼습니다.
  - 따라서 모기를 피하기 위함이라도 R동으로 이동하는 것은 도움이 되지 않으므로 선택하지 않아야 합니다.

- ✓ 모기 스프레이를 분사한다.
  - 하얗이의 현재 위치를 now, i번 모기의 위치를 d_i라고 했을 때 하얗이가 i번 모기에게 줄 수 있는 피해의 최대는 $min(A, d_i - now) \cdot B$입니다.
  - 하얗이가 줄 수 있는 피해가 해당 모기의 체력 h_i보다 작다면 하얗이는 도달할 수 없게 될 것입니다.

- ✓ T동을 향해 1 만큼 이동한다.
  - 하얗이가 T동을 향해 1 만큼 이동하면, 모기는 그 뒤에 하얗이를 향해 1 만큼 가까워집니다. 따라서 모기와의 거리는 2 만큼 줄어들게 됩니다.
  - 만약, 하얗이와 모기의 거리가 2 인데, 스프레이의 범위 A 가 1 이면 어떻게 해야 할까요?
    - ► 하얗이가 T동을 향해 이동하게 된다면 모기와 만나게 되고, R동을 향해 이동하게 된다면 시간만 흐를 뿐 상황은 바뀌지 않습니다.
    - ► 따라서, 제자리에서 모기 스프레이를 분사하며 모기와의 거리가 1 이 되도록 기다려줍니다.
  - 위와 같이 생각해보면 T동을 향해 이동하거나, 모기 스프레이를 분사하는 행동만 진행하면 된다는 것을 알 수 있습니다.

- ✓ 그리고, 모기 스프레이는 모기를 잡을 수 있는 선에서 최대한 모기와 가까운 상태에서 분사해야, 많은 모기에게 피해를 줄 수 있습니다.
- ✓ 하얗이가 T동을 향해 이동해도 모든 모기를 처리할 수 있는지를 알아내 봅시다.
  - 하얗이의 현재 위치를 now, i번 모기의 위치를 d_i라고 했을 때 하얗이가 i번 모기에게 줄 수 있는 피해의 최대는 $min(A, d_i - now) \cdot B$입니다.
  - 하얗이가 T동으로 1만큼, 모기가 하얗이 방향으로 1만큼 다가온다면 해당 식은 $min(A, d_i - now - 2) \cdot B$가 됩니다.
  - 그럼 $[now + 1, now + A + 2]$에 있는 모기들에 대해 하얗이가 T동으로 1만큼 이동해도 해당 범위의 모기를 다 잡을 수 있는지 위 수식으로 계산해주면 됩니다.
- ✓ 만약, 이동했다가는 잡을 수 없다면 스프레이를 분사해줍니다.

- ✓ 하얗이가 도달하지 못하는 경우가 아니라면 최대 $2L$ 시간이 걸립니다.
  - 하얗이가 가야할 거리는 L, 모기가 하얗이에게 다가오며 이동하는 최대 거리도 L 입니다.
- ✓ 매 시간마다 행동을 하는데 걸리는 시간복잡도는 $\mathcal{O}(A)$ 입니다.
  - 하얗이가 T동을 향해 가도 되는지 검사하는 것은 $\mathcal{O}(A)$ 만에 할 수 있습니다.
  - 하얗이가 스프레이를 분사하여 $[now+1, now+A]$ 위치의 모기에게 피해를 입히는 것은 $\mathcal{O}(A)$ 만에 할 수 있습니다.
- ✓ 따라서 총 시간복잡도는 $\mathcal{O}(AL)$ 입니다.
- ✓ 우선순위 큐를 사용한다면 A 값에 종속되지 않고, $\mathcal{O}(L + NlogN)$ 으로 해결할 수 있습니다.

I. 탄막 게임

`math`

출제진 의도 – **Challenge**

- ✓ 제출 5번, 정답 1명 (정답률 20%)
- ✓ 처음 푼 사람: **mastershim**, 99분
- ✓ 출제자: `tolelom`

- ✓ 이 문제의 핵심은 캐릭터가 총알을 피해 달아나지 못한다는 것입니다.
- ✓ 캐릭터가 총알을 피해 달아나지 못한다면 캐릭터는 최대한 총알을 오래 버티는 위치로 이동하는 것이 최선입니다.
- ✓ 캐릭터가 최대한 오래 버티는 위치를 찾아봅시다.

- ✓ 캐릭터가 최대한 오래 버티는 위치를 찾기 위해 한 가지 사실을 알아야합니다.
- ✓ 먼저 캐릭터의 T 초에서의 위치를 K 라고 합시다.
- ✓ 캐릭터의 이동 경로와 상관 없이 K 와 총알의 초기 위치 사이의 체비셰르 거리가 T 이하라면 T 초에서 총알은 캐릭터와 같은 위치에 있습니다.
- ✓ 체비셰프 거리는 두 점의 x 좌표 차이와 y 좌표 차이 중 큰 값을 갖는 거리입니다.
- ✓ 이를 증명해봅시다.

- ✓ 증명에 앞서 한 가지만 생각해봅시다.
- ✓ 총알의 x 좌표나 캐릭터의 x 좌표가 총알의 y 좌표 상에서의 이동에는 어떠한 영향도 주지 않습니다.
- ✓ 반대로 y 좌표도 마찬가지입니다. 따라서 각 좌표에 대해서 생각해봅시다.

- ✓ 우선 x 좌표만 고려해봅시다. 총알의 초기 좌표를 x_1, 캐릭터의 초기 좌표를 x_2, 캐릭터의 T 초에서의 좌표를 x_3 이라 합시다.
- ✓ 만약 T 초에서 총알의 좌표가 x_3 이 아니라면 그 전의 시간들에서도 캐릭터와 총알이 겹치는 경우는 없습니다. (한 번 겹치면 계속 겹치기 때문)
- ✓ 즉, 특정 지점 x_3 에서 가장 빨리 총알과 캐릭터가 만나는 시점 이후에 x_3 으로 이동한다면 총알과 만나게 됩니다.
- ✓ 따라서 캐릭터가 특정 지점까지 이동하는 최선은 최단 거리로 이동하는 것입니다.

- ✓ 다시 x 좌표만 고려해봅시다. 총알의 초기 좌표를 x_1, 캐릭터의 초기 좌표를 x_2, 캐릭터의 T 초에서의 좌표를 x_3 이라 합시다.
- ✓ $|x_3 - x_1| < |x_3 - x_2|$ 이라면 캐릭터가 총알을 만나지 않고 도달할 수 있는 지점임을 알 수 있습니다.
- ✓ 또한 x_3 에 총알이 도달하는 데 걸리는 시간은 $|x_3 - x_1|$ 입니다.
- ✓ 즉, 캐릭터는 x_3 에서 $|x_3 - x_1| - 1$ 초 동안 버틸 수 있습니다.

- ✓ y 좌표에 대해서도 동일하니 이제 원래 문제로 돌아가봅시다.
- ✓ 총알의 초기 좌표 (x_2, y_2), 캐릭터의 초기 좌표 (x_1, y_1), 캐릭터의 T 초에서의 좌표를 (x_3, y_3) 이라 합시다.
- ✓ $max(|x_3 - x_1|, |y_3 - y_1|) < max(|x_3 - x_2|, |y_3 - y_2|)$ 이면 캐릭터가 총알을 만나지 않고 도달할 수 있는 지점임을 알 수 있습니다.
- ✓ 이 때, 총알이 도달하는데 걸리는 시간은 $max(|x_3 - x_2|, |y_3 - y_2|)$ 입니다.

- ✓ 이제 각 좌표에 캐릭터가 도달하는데 걸리는 시간(t_c)과 가장 가까운 총알이 도달하는데 걸리는 시간(t_b)을 계산합니다.
- ✓ 이 때 $t_c < t_b$라면 캐릭터가 위치에 도달할 수 있고 이 때 버틸 수 있는 시간은 $t_b - 1$입니다.
- ✓ 즉, $t_c < t_b$ & $T < t_b$인 위치가 존재하면 게임에서 승리할 수 있습니다.
- ✓ 이를 계산하는데 맵의 크기 $O(NM)$와 총알의 개수, 캐릭터 $K + 1$을 곱한 $O(KNM)$에 해결할 수 있습니다.