제2회 흐즈로컵

Editorial

목차

- Credit
- A1. SAVE the World (Small)
  - 제약 설명
  - 운영진의 풀이
  - 참가자의 풀이
- A2. SAVE the World (Large)
  - 제약 설명
  - 부분점수를 받는 방법
  - 운영진의 풀이
  - 참가자의 풀이

Credit

- 검수진
  - @halin, @kiwiyou, @mingyu331, @tony9402, @utilforever, @wapas, @wider93
- 후원자
  - @utilforever
- Visualizer 제작자
  - @mingyu331
- 대회 운영 및 출제 지원
  - Startlink

A1. SAVE the World (Small)

- 의도한 난이도: **Gold**
- 점수 분포
  - 만점: 85명
  - 부분 점수: 14명
- 태그: implementation, constructive, ad_hoc

A1. SAVE the World (Small) – 제약

- $n \leq 500$
- $-15 \leq x, y \leq 15$
- 가능한 좌표의 수 $31 \times 31 - 1 = 960$을 500으로 나누면
  약 52%의 밀도를 가짐을 계산할 수 있으므로,
  상당히 dense한 무작위 데이터가 존재함을 추정할 수 있습니다.
- 본 문제를 만점으로 해결하기 위해서는,
  조건을 만족하는 해가 "항상 존재함"을 증명해야 합니다.

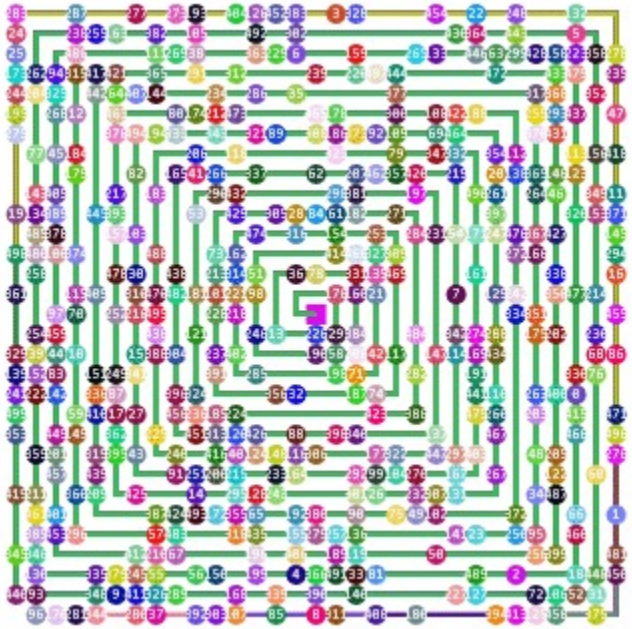
A1 – 운영진의 풀이 1

- $(0,0)$에서 출발하여 모든 정수 좌표를 한 번씩 지나가는 무한한 경로가 존재할까요?
- 다음과 같은 경로를 생각합시다.
- BOJ 1913: 달팽이에서 모든 k와 $k+1$을 연결하는 경로를 만들 수 있습니다.
- 모든 용사가 이 경로를 따라 $(0,0)$으로 이동한다면, 항상 정당한 지시가 됩니다. 왜 그럴까요?

A1 – 운영진의 풀이 1

- 각 좌표에 적힌 값이 k일 때, 용사는 k번째 순간에 도착합니다. k는 고유하므로, 모든 용사는 서로 다른 순간에 도착합니다.
- 경로에는 사이클이 없으므로 모든 용사는 각 칸을 최대 1번만 방문하게 됩니다.
- k가 적혀 있는 칸에서 출발한 용사는 t번째 순간에 $k-t$가 적힌 칸에 있게 됩니다. k는 고유하므로, 모든 용사는 매 순간에 서로 다른 칸에 있게 됩니다.
- 따라서, 모든 용사가 이 경로를 따른다면 정당한 지시가 됩니다.
- 많은 참가자께서 이 풀이로 문제를 해결해 주셨습니다.

A1 – 운영진의 풀이 1

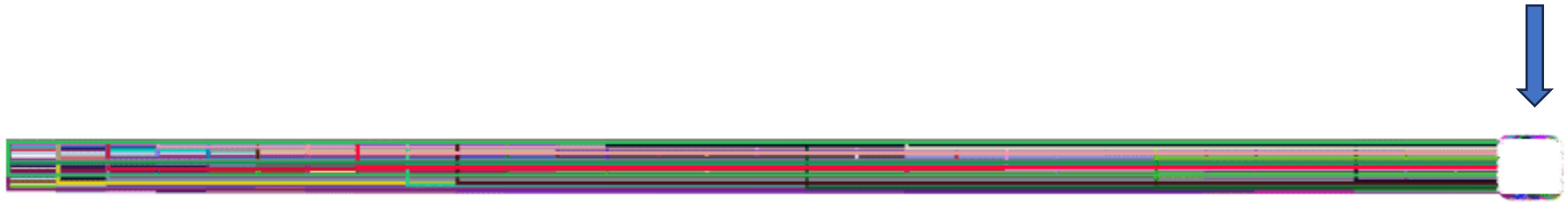


A1 – 운영진의 풀이 2

- @wapas님은 다음과 같은 풀이를 사용했습니다.
- 모든 용사가 동시에 왼쪽으로 이동하기 시작합니다.
- 이때, 일정 시간 후에 $(0,0)$의 오른쪽에 있었던 용사는 모두 $(0,0)$에 도착한 상태가 됩니다.
- 이 시점부터 순서대로 대열 오른쪽 끝에서 한 줄씩 대열의 반대 방향으로 돌아서 $(0,0)$으로 진행하도록 합니다.
- $(0,0)$으로 진입할 때에는 기존의 경로와 만나지 않도록 $(0,0)$의 오른쪽으로 진입하도록 합니다.
- 이렇게 하면 조건을 만족하는 정당한 지시가 됩니다.

A1 – 운영진의 풀이 2

(0,0)은 여기에 있습니다. (분홍색)



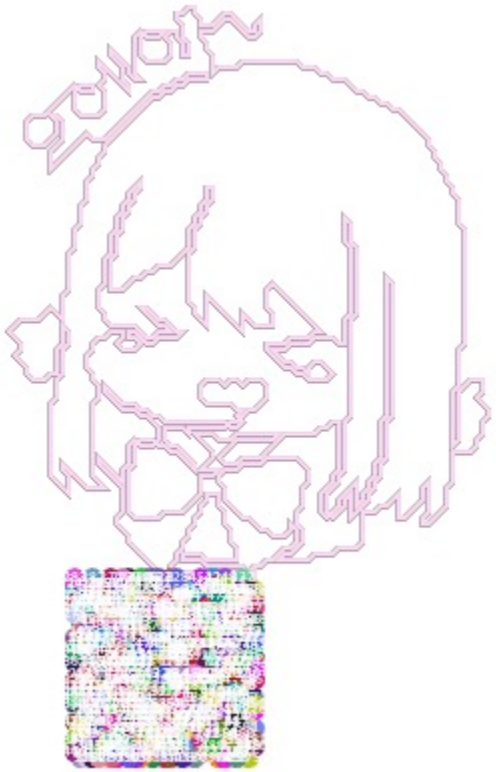
A1 – 참가자의 풀이 1

- 모든 용사가 같은 달팽이 모양 경로를 따르게 하지 않고, 모든 용사의 배치를 달팽이 모양 경로를 따라 “평행이동”하는 풀이가 존재합니다.
- 이렇게 하면 모든 용사는 서로 만나지 않으며, 유한 시간 내에 모든 용사가 $(0,0)$에 도착하며, 정당한 지시가 됩니다.

A1 – 참가자의 풀이 2

- @ruykun님의 풀이입니다.
- 정해에서 설명한 “경로”는, 반드시 달팽이 모양이 되어야 하는 것은 아닙니다.
- $(0,0)$에서 시작하면서 범위 내의 모든 좌표를 한 번씩 지난다면, 어떤 경로든 같은 방식으로 문제를 해결할 수 있습니다.
- @ruykun님이 사용하신 경로는 다음 슬라이드와 같습니다.

A1 – 참가자의 풀이 2



A1 – 참가자의 풀이 3

- 그 외에, 무작위화를 통해 용사 간의 거리가 커지도록 용사들을 후진시킨 다음 다시 $(0,0)$을 향하는 무작위 경로를 따라서 이동시키는 풀이가 통과하였습니다.
- 본 대회에서 거의 유일하게 휴리스틱을 사용하여 만점을 받은 솔루션입니다.
- 이 풀이가 왜 작동하는지는 운영진 또한 증명하지 못했습니다.

A2 – SAVE the World (Large)

- 의도한 난이도: **Platinum ~ Diamond**
- 점수 분포
  - 만점: 16명
  - $[200\times10^6, 250\times10^6)$: 2명
  - $[150\times10^6, 200\times10^6)$: 1명
  - $[100\times10^6, 150\times10^6)$: 1명
  - $[50\times10^6, 100\times10^6)$: 1명
  - $(0, 50\times10^6)$: 7명
- 태그: ad_hoc, constructive, sorting, greedy

A2 – 제약 설명

- $n \leq 500$
- $-1000 \leq x, y \leq 1000$
- 특수한 성질을 만족하도록 생성된 데이터가 50개씩 총 200개 존재합니다.
- C_n은 주어진 용사의 배치에 따라서 계산할 수 있는 L의 자명한 하한입니다.
- 이 사실은 각 용사가 독립적으로 존재할 때 $(0,0)$에 도달하는데 걸리는 시간이 L^∞-norm과 같다는 점과, 매 순간 한 명의 용사만 $(0,0)$에 도달할 수 있다는 사실에 근거합니다.

A2 – 제약 설명

- 점수는 C_n과 L의 비에 따라서 일차적으로 계산되며, 최대 실행 시간이 1초를 초과하는 경우 실행 시간에 따라 지수적으로 감점 됩니다.
- 시간에 따른 페널티를 둔 이유는, 실행 시간을 최대한 사용하여 연산의 양을 늘리는 기존의 휴리스틱 대회 특유의 전략을 막기 위함입니다.
- 모든 실행에서 $C_n = L$이며 실행 시간이 1초를 초과하지 않으면 만점을 받을 수 있습니다.

A2 – 부분점수를 받는 방법 1

- 서로 다른 x, y, $x+y$, $x-y$의 값의 종류가 최대 10가지인 경우 어떻게 점수를 얻을 수 있을지 알아봅시다.
- 이 경우는 모든 용사들이 최대 10개의 평행한 직선 위에 배치된 상태입니다.
- 점수를 얻기 위해 이 정보를 어떻게 활용할 수 있을까요?

A2 – 부분점수를 받는 방법 1

- 각 직선을 위치 순서대로 $a_1, a_2, \cdots, a_k$로 부릅시다.
- $(0,0)$과 가장 가까운 직선을 고른 뒤, 이 직선을 중심으로 직선들을 한 경로로 연결해 줍시다.
- 예를 들어, $k = 10$이고 $(0,0)$과 가장 가까운 직선이 a_3이었다고 생각해 봅시다.
- 이 경우, $a_3, a_2, a_4, a_1, a_5, a_6, a_7, a_8, a_9, a_{10}$ 순으로 연결해 줄 수 있을 것입니다.
- 이 순서를 최적화함으로써 점수를 얻어내는 전략을 구성해 볼 수 있습니다.

A2 – 부분점수를 받는 방법 1

- 이 순서들은 최대 $9! = 362880$가지이므로, 모든 경우의 수를 따지기에는 시간이 충분하지 않습니다. 하지만 무작위 순서를 여러 개 탐색하여 그 중 최적인 경우를 고르거나, 탐욕적 방법을 사용하는 등의 방법이 존재합니다.
- 이 방법으로 이 경우에 만점을 받을 수 있음이 보장되지 않지만, "충분히 높은" 점수를 얻어내기 위해 이 방법을 사용할 수 있습니다.
- 또한 직선 위의 용사들의 $(0,0)$까지의 L^∞-norm이 좌표에 따라 낮아졌다가 높아지는 형태의 "끊어진 일차함수" 형태라는 점도 응용할 수 있습니다.

A2 – 부분점수를 받는 방법 2

- 서로 다른 L^∞-norm의 종류가 최대 10가지인 경우 어떻게 점수를 얻을 수 있을지 알아봅시다.
- 이 경우는 모든 용사들이 최대 10개의 동심원 위에 배치된 상태입니다.
- 점수를 얻기 위해 이 정보를 어떻게 활용할 수 있을까요?

A2 – 부분점수를 받는 방법 2

- 같은 동심원 위에 k명의 용사가 존재한다고 가정해 봅시다.
- 이 용사 k명의 C 값은 서로 다르며, 모두 연속합니다.
- 이 용사들이 모두 동심원 위를 도는 동안 C의 최댓값은 변화하지 않습니다.
- 우선 용사들의 C 값을 모두 전처리해줍시다. 그런 뒤, 각 용사들이 $C - P$에 따라 동심원 위를 돌다가 $(0,0)$으로 접근하는 전략을 생각합시다.

A2 – 부분점수를 받는 방법 2

- 각 용사의 C값을 정하는 방법은 유일하지 않습니다. P값이 같은 용사의 순서를 정하는 방법을 지문에서 언급하지 않았기 때문입니다. P값이 같은 용사의 순서를 매개변수로 최적화한다면, 용사가 동심원 위에서 돌다가 $(0,0)$으로 접근하는 전략을 유효하게 활용할 수 있을 것입니다.
- 이러한 관측은 만점을 받는 풀이를 도출하는데 일부 응용할 수도 있습니다.

A2 – 부분점수를 받는 방법 3

- 정확히 한 용사가 $(0,0)$과 인접하며, 이 용사는 다른 용사 한 명과 인접하며, 나머지 모든 용사는 최대 2명의 용사와 인접할 때, 어떻게 점수를 얻을 수 있을지 알아봅시다.
- 이 경우는 용사의 인접 여부에 따른 그래프가 하나의 경로(path graph)를 형성합니다.
- 점수를 얻기 위해 이 정보를 어떻게 활용할 수 있을까요?

A2 – 부분점수를 받는 방법 3

- 용사의 좌표에 따라 배열 P를 구하면, $P_1 = 1$이며, $P_i \leq P_{i-1} + 1$ 임을 확인할 수 있습니다.
- 이 사실에 따라, $C_n = n$이 조건 하에 항상 성립합니다.
- 그런데, C_n뿐만 아니라 경로의 길이 또한 n이라는 점을 관찰할 수 있습니다.
- 그러므로, DFS를 사용해 경로 그래프를 탐색하여 용사가 이동할 경로를 전처리해 줍시다.
- 이 경로를 따라서 모든 용사가 함께 이동하면 $L = n$이 됩니다.

A2 – 부분점수를 받는 방법 4

- 용사의 위치가 무작위로 선정된 경우를 생각해 봅시다.
- 용사의 위치, 용사 간의 거리, L^∞-norm 등의 수치는 무작위적인 분포의 성질에 따라 결정됩니다.
- 점수를 얻기 위해 이 정보를 어떻게 활용할 수 있을까요?

A2 – 부분점수를 받는 방법 4

- 용사의 x좌표와 y좌표를 균등분포로 근사할 수 있습니다.
- 서로 다른 용사의 x좌표와 차가 평균적으로 $2000/n$이라고 가정하면, 이 값을 4 정도로 전제하고 문제를 풀 수 있습니다.
- 탐욕적 방법에 따라 $(0,0)$에 가까운 용사부터 지시를 정합시다.
- 탐욕적으로 L^∞-norm을 최소화하는 방향을 우선시하여 이동하면서, 이미 그 시점에 그 위치에 있는 용사와 충돌하지 않도록 백트래킹을 합니다.
- 이렇게 하면 용사끼리 충돌할 가능성은 매우 작게 됩니다.

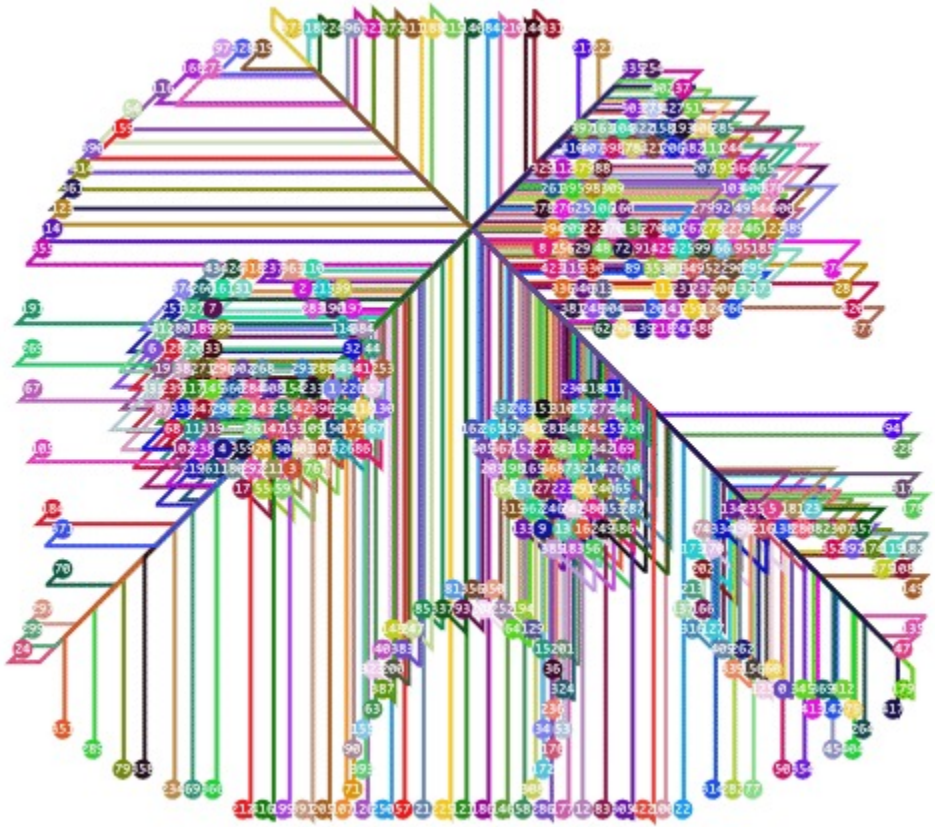
A2 – 부분점수를 받는 방법 4

- 실제로 이러한 알고리즘은 무작위로 생성된 50개의 경우 모두에서 만점을 받음을 확인할 수 있습니다.
- 이 휴리스틱은 용사의 밀도가 매우 낮다는 점을 전제로 문제를 접근하기 때문에, 이 전제가 성립하지 않는 테스트 케이스에는 적용하기 어려울 수 있습니다. 그러나 이 전제가 확실히 성립하는 50개의 테스트 케이스에서 점수를 얻으려는 경우, 이 전략은 확실히 유효합니다.

A2 – 운영진의 풀이

- 두 배열 C와 P를 매 순간 갱신하면서 다음 규칙에 따라 용사를 이동시킵니다.
- $C - P = 1$이라면, 동심원을 따라 용사를 반시계 방향으로 이동시킵니다.
- $C - P > 1$이라면, L^{∞}-norm이 커지는 반시계 방향 대각선으로 이동시킵니다.
- $C = P$라면, 용사를 그대로 안쪽으로 이동시킵니다.
- 이렇게 하면, 매 순간 C_n이 1씩 감소하며, 지시는 정당한 지시가 됩니다. 증명은 생략합니다.

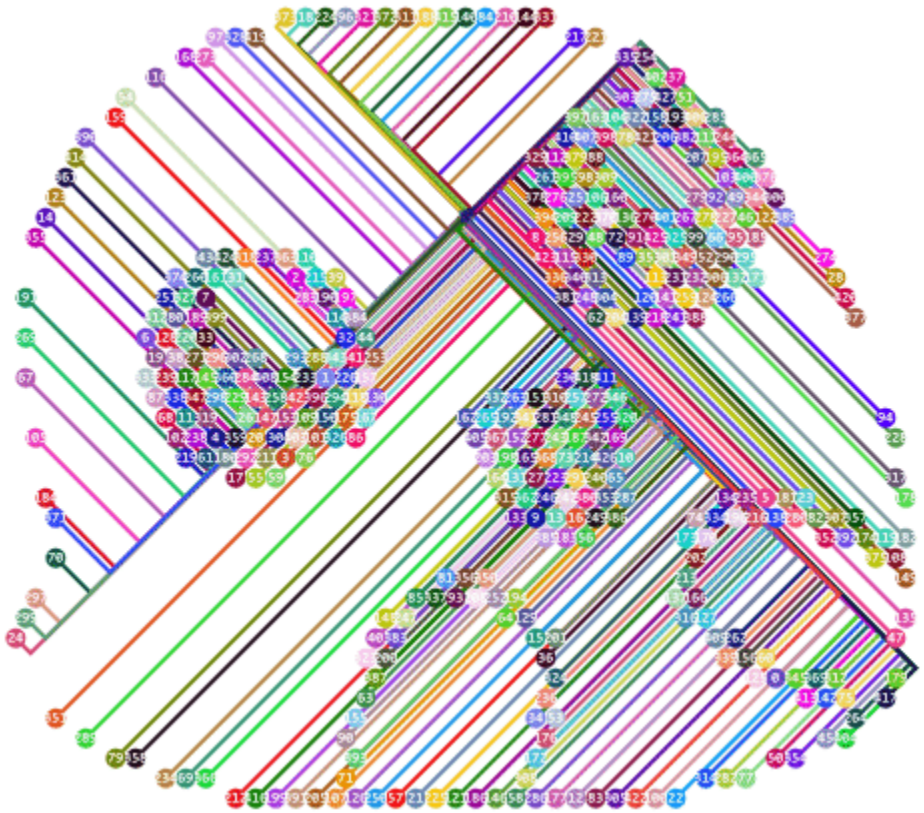
A2 – 운영진의 풀이



A2 – 참가자의 풀이

- BOJ 1913: 달팽이의 배열을 전처리합시다.
- 매 순간, k번째 용사가 서 있는 좌표의 달팽이 배열 값을 a_k라고 합시다.
- 우선, a_k가 작은 순서대로 용사들을 정렬합니다.
- 정렬한 순서대로 용사가 다음으로 이동할 위치를 정하되, 다음 순간에 용사가 존재하지 않으면서 달팽이 배열 값이 가장 작은 위치로 이동하도록 합니다.
- 이렇게 하면, a_k의 대소관계는 C배열의 대소관계와 일치하며, 매 순간 각 용사의 a_k가 감소합니다. 이에 따라 C_n의 값이 감소함을 보이는 것이 가능합니다.
- 많은 참가자께서 이 풀이를 통해 문제를 해결해 주셨습니다.

A2 – 참가자의 풀이



A2 – 참가자의 풀이

- 각 위치의 값은 전처리가 없이도 $O(1)$에 구하는 것이 가능하므로, 시간 복잡도 분석에서 전처리 시간은 생략합시다.
- 매 순간 용사들의 배열을 정렬해야 하므로, L번의 순간에 대해 시간 복잡도는 $O(Ln \log n)$이 됩니다.
- L의 값은 최대 1500이므로, 이 알고리즘의 실행 시간은 세부적인 구현에 민감할 수 있고, 그대로 구현했을 때 1초를 넘게 될 수도 있습니다. 이에 다음 슬라이드부터는 이 풀이의 실행 시간을 최적화하는 방법을 제시합니다.

A2 – 참가자의 풀이 – 최적화

- 용사들을 a_k에 따라 정렬할 때, 첫 지시를 시행하기 전에 한 번만 $O(n \log n)$ 시간 복잡도로 정렬합시다. 그 이외의 경우에는, 정렬을 할 때 삽입 정렬을 사용하는 것이 오히려 병합 정렬 등의 $O(n \log n)$ 시간 복잡도의 알고리즘보다 더 나을 수 있습니다.
- 왜 그럴까요?

A2 – 참가자의 풀이 – 최적화

- 삽입 정렬은 adaptive한 성질을 가지는 정렬 알고리즘입니다. 즉, 정렬해야 하는 배열이 이미 정렬된 상태와 가깝다면, 최악 시간 복잡도보다 더 빠른 시간에 정렬할 수도 있습니다.
- 삽입 정렬의 경우, 배열의 반전수가 I일 때, 시간 복잡도는 $O(n+I)$라고 할 수 있습니다.
- 그런데, 우리는 이미 a_k가 낮은 용사에게 더 낮은 a_k를 배정해 주고 있으므로, 순서가 바뀐 용사는 많지 않다고 전제할 수 있습니다. 즉, $I = O(n)$이라고 전제할 수 있습니다.
- 이 전제가 성립할 때, 정렬의 시간 복잡도는 $O(n)$이 됩니다.

A2 – 참가자의 풀이 – 최적화

- 또한, 용사가 다음 순간에 어떤 칸을 차지하는지의 여부를 확인하기 위해서 $O(\log n)$ 시간 복잡도의 자료구조를 사용하지 않아도 됩니다.
- 용사의 좌표는 배열의 인덱스로 저장 가능한 크기이기 때문에, 용사의 존재 여부 자체를 2차원 배열로 표현할 수 있습니다.
- 단, 이 경우 용사의 존재 여부를 초기화하는 것이 다소 난감할 수 있습니다.

A2 – 참가자의 풀이 – 최적화

- 순서와 상관없이, 용사가 존재하는 위치를 모두 담는 자료구조를 함께 관리합시다.
- 이 자료구조가 존재한다면, 현재 n명의 용사가 존재할 때, 다음 순간의 초기화도 $O(n)$에 처리할 수 있습니다.
- 이러한 자료구조로는 스택이나 큐를 사용할 수 있습니다.
- 이로써 시간 복잡도에서 $\log n$ 계수를 제거할 수 있으며, 새로운 시간 복잡도는 $O(n \log n + Ln)$입니다.
- 이러한 최적화에 따라, 실행 시간은 C++ 기준 576ms → 84ms로 감소합니다.

감사합니다.

Thank you for participating!