2021 홍익대학교 프로그래밍 경진대회 풀이

Official Solutions

by

홍익대학교 프로그래밍 경진대회 출제진

2021년 10월 30일

문제		의도한 난이도	출제자
A	홍익 절대평가	**Easy**	swoon
B	홍익 댄스파티	**Easy**	906bc
C	초콜릿 뺏어 먹기	**Medium**	Green55
D	황소 다마고치	**Medium**	Green55
E	어려운 모든 쌍 최단거리	**Advanced**	Green55
F	마법천자문	**Advanced**	Green55
G	홍익 투어	**Hard**	swoon
H	리그전 오브 레전드	**Hard**	Green55
I	마을 구하기	**Challenging**	extra1563
J	구슬 보내기	**Challenging**	hwon233

A. 홍익 절대평가

`implementation`

출제진 의도 – **Easy**

- ✓ 제출 228번, 정답 79명 (정답률 34.65%)
- ✓ 처음 푼 참가자: **정X현**, 3분
- ✓ 출제자: swoon

A. 홍익 절대평가

- ✓ 상대평가 시 A학점을 받는 학생의 수는 $N \times X \div 100$ 입니다.
- ✓ 절대평가 시 A학점을 받는 학생의 수는 Y 점을 넘는 학생의 수입니다.
  - 반복문을 통해 Y 점 이상의 학생의 수를 구할 수 있습니다.
- ✓ 시간복잡도는 $\mathcal{O}(N)$ 입니다.

B. 홍익 댄스파티

`implementation`

출제진 의도 – **Easy**

- ✓ 제출 204번, 정답 35명 (정답률 17.16%)
- ✓ 처음 푼 참가자: **김X헌**, 14분
- ✓ 출제자: 906bc

B. 홍익 댄스파티

- ✓ 학생의 수를 n이라고 해봅시다.
- ✓ 먼저, 입력으로부터 각 학생의 상태를 확인해야 합니다.
  - 입력 i번째 줄에 나타난 j번째 학생의 모습을 $a_{i,j}$ 라고 해봅시다.
  - $a_{1,j}, a_{2,j}, a_{3,j}, a_{4,j}, a_{5,j}$ 를 모두 확인하여 j번째 학생의 상태를 확인할 수 있습니다.
  - 더 간단한 방법은 $a_{2,j}$ 만을 확인하는 것입니다. $a_{2,j}$ 가 o 라면 도약 준비, w 라면 도약 중, . 라면 착석임을 알 수 있습니다. 같은 방식으로 $a_{3,j}$ 또는 $a_{4,j}$ 로도 j번째 학생의 상태를 확인할 수 있습니다.

- ✓ 상태를 확인했으니, 다음 차례의 모습을 유추하여 출력해야 합니다.
  - 출력 i번째 줄에 나타날 j번째 학생의 모습을 $b_{i,j}$ 라고 해봅시다.
  - 이중 반복문을 이용하여 해결할 수 있습니다.
  - 첫번째 반복문은 i를 1부터 5까지, 두번째 반복문은 j를 1부터 n까지 반복하여 출력합니다.
- ✓ 입력과 출력의 시간복잡도는 각 $\mathcal{O}(n)$ 입니다.

C. 초콜릿 뺏어 먹기

ad-hoc

출제진 의도 – **Medium**

- ✓ 제출 181번, 정답 41명 (정답률 22.65%)
- ✓ 처음 푼 참가자: **정X현**, 21분
- ✓ 출제자: Green55

C. 초콜릿 뺏어 먹기

- ✓ 초기 배열에서 제일 작은 원소(즉, 초기의 a_1)가 m이라고 해봅시다.
- ✓ 우리가 매일 할 수 있는 행동은, 어떤 원소를 다른 원소와 같게 만드는 것입니다.
- ✓ 그러므로 어떤 원소도 m보다 작아질 수는 없습니다.
- ✓ 즉, 모든 원소를 m로 통일시키는 것 보다, 더 많은 초콜릿을 먹을 수는 없습니다.

C. 초콜릿 뺏어 먹기

- ✓ 만약 우리가 하루의 낭비도 없이, m이 아닌 원소를 m로 만들 수 있다면,
- ✓ 이것보다 더 빠르게 최대의 초콜릿을 먹을 방법은 없습니다.

- ✓ 그런데, 항상 위와 같은 방법이 가능합니다!
  - $K < i$이면서, $a_i \neq m$인 가장 작은 i를 골라서,
  - 모든 원소가 m이 될 때 까지 a_i를 a_{i-K}로 만들어 줍니다.
  - (곧 증명하겠지만, 무조건 $a_{i-K} = m$입니다.)

- ✓ 예를 들면 다음과 같습니다.
  - $A = [1, 1, 2, 3, 4], K = 3$
  - $[1, 1, 2, \underline{3}, 4] \rightarrow [1, 1, 2, \underline{1}, 4] \rightarrow [1, 1, 1, 2, 4]$
  - $[1, 1, 1, \underline{2}, 4] \rightarrow [1, 1, 1, \underline{1}, 4]$ (이미 정렬 되어있음)
  - $[1, 1, 1, 1, \underline{4}] \rightarrow [1, 1, 1, 1, \underline{1}]$ (이미 정렬 되어있음)

C. 초콜릿 뺏어 먹기

- ✓ 이것이 왜 항상 가능할까요?
- ✓ 오늘 어떤 원소를 고를지 결정해봅시다.

- ✓ $a_{k+1} \neq m$ 이라면,
  - a_{k+1} 를 골라야합니다. a_{k+1} 이 $a_{(k+1)-k} = a_1 = m$ 이 됩니다.
  - a_1 은 처음부터 m 이었으니까요.

C. 초콜릿 뺏어 먹기

- ✓ $a_{k+1} = m$ 이라면,
  - A 는 정렬되어있으므로, $m = a_1 = a_2 = \cdots = a_{k+1}$ 입니다.
  - $a_{k+2} \neq m$ 이라면, a_{k+2} 를 $a_2 = m$ 으로 만들어 줍니다.
  - $a_{k+2} = m$ 이라면, $m = a_1 = a_2 = \cdots = a_{k+1} = a_{k+2}$ 입니다.
    - 이런식으로, m 이 아니면서 가장 작은 원소가 a_i 라면,
    - $m = a_1 = a_2 = \cdots = a_{i-1}$ 이기 때문에,
    - 결국 항상 m 과 같아집니다.

C. 초콜릿 뺏어 먹기

- 따라서 위의 방법을 시뮬레이션 해주면 해결 가능합니다.
  - 다음의 과정을 불가능 할 때까지 반복합니다
    - $K < i$ 이면서, $a_i \neq m$ 인 가장 작은 i 를 고릅니다.
    - a_i 를 삭제하고, A 를 정렬합니다.

- 위의 과정은 최대 $\mathcal{O}(N)$ 번 반복되고,
- 한번의 과정은 정렬의 $\mathcal{O}(N \log N)$ 이 지배하므로
- $\mathcal{O}(N^2 \log N)$ 이 소요됩니다.

C. 초콜릿 뺏어 먹기

- ✓ …하지만 조금만 더 생각해보면,
- ✓ 우리는 배열을 무조건 $[m, m, ..., m]$으로 만들 수 있습니다.
- ✓ 그러므로 먹을 수 있는 초콜릿은 $(a_1 - m) + (a_2 - m) + \cdots + (a_n - m)$개 이고,
- ✓ 이것을 ($a_i \neq m$인 원소의 개수)일 만에 가능합니다.
- ✓ $\mathcal{O}(N)$에 간단히 구현 가능합니다.

D. 황소 다마고치

`math, ad-hoc`

출제진 의도 – **Medium**

- ✓ 제출 210번, 정답 37명 (정답률 17.62%)
- ✓ 처음 푼 참가자: **김X헌**, 27분
- ✓ 출제자: `Green55`

D. 황소 다마고치

- ✓ 먼저 먹이의 효율에 대해 생각해봅시다.
- ✓ 황소의 체력은 매일 최소 1씩 줄어드므로
- ✓ x 개의 먹이로 황소의 수명을 x 일 넘게 늘릴 수는 없습니다.
- ✓ 따라서 1개의 먹이로 수명을 1일 늘릴 수 있다면, 최적의 전략일 것입니다.

D. 황소 다마고치

- 이제 현재 체력에 따른 생존 가능 날짜를 관찰해봅시다.

체력	1	2	3	4	5	6	7	8	9	10
생존 가능 날짜	1	2	2	3	3	3	3	4	4	4

- 체력은 계속 절반씩 줄어드므로, $1 + \lfloor \log_2 n \rfloor$ 일만큼 생존 가능합니다.

D. 황소 다마고치

- ✓ 따라서, 황소의 수명을 늘리기 위해선, $\lfloor \log_2 n \rfloor$ 의 값을 늘려야 합니다.
  - 예를 들어, 체력이 6 일 때 먹이를 2 줘서 8 로 만든다면
  - $\lfloor \log_2 n \rfloor$ 의 값이 2 에서 3 로 바뀌었기 때문에, 황소가 1 일 더 생존 가능합니다.
- ✓ 따라서 체력이 $2^k - 1$ 일 때, 먹이를 1 줘서 체력을 2^k 로 바꿔주면
- ✓ 1개의 먹이로 수명을 1일 늘릴 수 있고,
- ✓ 이것을 m 번 반복 할 수 있다면 최적의 전략입니다.

D. 황소 다마고치

- ✓ 그런데, 황소의 체력은 언젠가 1이 되고,
  - $\lfloor x/2 \rfloor = 0$을 만족하는 양수 x는 1밖에 없기 때문입니다.
- ✓ 1은 $2^1 - 1$입니다.

D. 황소 다마고치

- ✓ 따라서, 황소의 체력이 1이 됐을때 먹이를 1씩 주는 것을 반복하면
- ✓ m 개의 먹이로 m 일 수명을 늘릴 수 있고, 이것이 최적입니다.
  - $7 \to 3 \to 1 \xrightarrow{\text{먹이 1}} 2 \to 1 \xrightarrow{\text{먹이 1}} 2 \to 1 \xrightarrow{\text{먹이 1}} 2 \ldots$
- ✓ 따라서, 답은 $m + 1 + \lfloor \log_2 n \rfloor$ 입니다.
- ✓ $\lfloor \log_2 n \rfloor$ 의 값은 직접 2로 나눠가며 $\mathcal{O}(\log_2 n)$ 에 구할 수 있습니다

D. 황소 다마고치

- 또는, n이 충분히 작기 때문에 C++의 `<math.h>`나 Python의 `math`에서 제공하는 `log2` 함수를 이용 할 수도 있습니다.
- 하지만, n이 조금만 더 커져도 실수 오차 때문에 틀린 값이 나온다는 것을 주의하세요.
- 두 언어 모두 `log2`는 실수를 사용하여 계산하므로, $\log_2(2^{49}-1)=49$로 계산합니다.
- 앞쪽 문제라는 것을 고려하여 `log2`를 써도 괜찮게끔 n을 작게 설정했습니다.
- 정수 연산으로만 가능하다면 실수는 쓰지 않는 것을 권해드립니다.

E. 어려운 모든 쌍 최단거리

`graphs, graph_traversal`

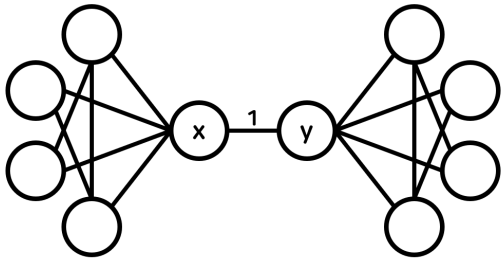
출제진 의도 – **Advanced**

- ✓ 제출 65번, 정답 12명 (정답률 18.46%)
- ✓ 처음 푼 참가자: **김X헌**, 36분
- ✓ 출제자: `Green55`

E. 어려운 모든 쌍 최단거리

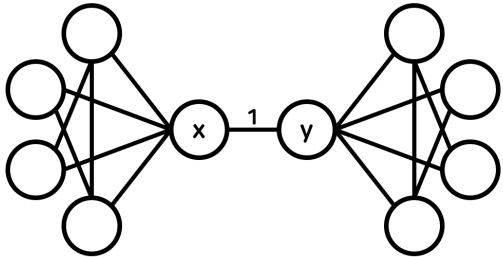
- ✓ 가중치가 있는 간선을 $x - y$라고 해봅시다.
- ✓ 두 정점 u와 v의 최단거리는
  - u에서 v에 도달하기 위해 반드시 $x - y$를 걸쳐야한다면, 1
  - u에서 v에 도달하기 위해 $x - y$를 거칠 필요가 없다면, 0
- ✓ 이것을 달리 말하면
  - 간선 $x - y$를 삭제했을 때, u에서 v에 도달 할 수 없으면, 1
  - 간선 $x - y$를 삭제했을 때, u에서 v에 도달 할 수 있으면, 0

E. 어려운 모든 쌍 최단거리



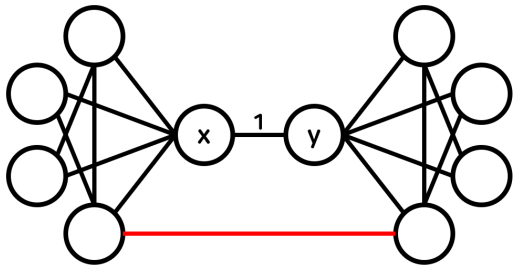
- ✓ $x - y$를 삭제했을 때, 그래프가 두 덩어리로 분리된다면,
  분리된 덩어리끼리 경로가 존재하지 않습니다.

E. 어려운 모든 쌍 최단거리



- 즉, 왼쪽 덩어리에서 임의의 정점을 하나,
  오른쪽 덩어리에서 임의의 정점을 하나 뽑으면 둘의 최단거리가 1이고,
- 최단거리의 합은 결국 두 덩어리에서 정점을 하나씩 뽑는 경우의 수
- 즉, (왼쪽 덩어리에 있는 정점의 개수)×(오른쪽 덩어리에 있는 정점의 개수)가 됩니다.

E. 어려운 모든 쌍 최단거리



- ✓ $x - y$를 삭제해도 그래프가 연결되어 있다면, 모든 정점 쌍의 최단거리가 0입니다.
- ✓ 위 그래프처럼, $x - y$가 삭제되어도 빨간색 간선을 이용하여 모든 정점들이 연결됩니다.

- ✓ 이렇게 서로 연결된 정점의 덩어리를 **컴포넌트**라고 부르고,
- ✓ 어떤 간선을 삭제했을 때, 하나의 컴포넌트가 두개로 분리된다면 그 간선을 **단절선(bridge)**라고 부릅니다.

- 이제 우리는
  - $x - y$ 가 단절선인지와, 단절선이라면
  - x 가 속해있는 컴포넌트의 크기와
  - y 가 속해있는 컴포넌트의 크기

  를 알아낼 필요가 있습니다.

E. 어려운 모든 쌍 최단거리

- ✓ 이것은 실제로 간선 $x - y$를 삭제하고, 그래프 순회(DFS 혹은 BFS)를 이용하면 됩니다.
- ✓ x에서 시작해 그래프 순회를 했을 때 방문한 정점들,
  즉 x에서 도달 가능한 정점들의 집합을 V라고 해봅시다.
  - V에 y가 속해있다면, x와 y가 연결되어 있으므로 단절선이 아닙니다.
  - 그렇지 않다면, 간선 $x - y$가 단절선이고,
    분리된 두 컴포넌트의 크기는 $|V|$와 $n - |V|$입니다.

- ✓ 시간복잡도는 그래프 순회 한 번에 걸리는 $\mathcal{O}(N+M)$ 입니다.
- ✓ 또는 컴포넌트의 크기를 셀 수 있는 다른 자료구조인, 유니온 파인드 등을 사용해도 됩니다.

F. 마법천자문

dp

출제진 의도 – **Advanced**

- ✓ 제출 9번, 정답 2명 (정답률 22.22%)
- ✓ 처음 푼 참가자: **김X헌**, 81분
- ✓ 출제자: Green55

F. 마법천자문

- ✓ 주어진 문자열을 S라고 하고, $S[..i]$를 S의 앞 i 글자를 뗀 문자열이라고 합시다.
- ✓ 대략적으로 다음과 같은 형태의 DP를 생각해볼 수 있습니다.
  - $dp[i] = S[..i]$의 올바른 해석 중 가장 계산 결과가 큰 것.
    (단, 올바른 해석이 존재하지 않으면 $dp[i] = -\infty$)
- ✓ $dp[i]$를 '이용'할 것이라면 i 이후를 어떻게 해석하든,
  $S[..i]$에서 가장 큰 해석을 취하는 것이 이득이기 때문에 가능합니다.

F. 마법천자문

- ✓ 구체적은 점화식은 다양한 형태로 가능하지만, 가장 간단한 풀이 중 하나는 다음과 같습니다.
- ✓ 다음과 같은 6개의 '덩어리'와 각 덩어리에 해당하는 값을 정의합니다.
  - `++-` : $+11$
  - `++` : $+10$
  - `+-` : $+1$
  - `-+-` : -11
  - `-+` : -10
  - `--` : -1

F. 마법천자문

- ✓ S 앞에 +를 덧붙인 문자열을 S' 라고 합시다.
- ✓ 그렇다면 S' 를 여러개의 덩어리로 쪼개고,
  각 덩어리에 해당하는 값의 합을 최대화 시키는 문제가 됩니다.
  - 예를 들어, $S =$ +----+-+-라면, $S' =$ +----+-+-입니다.
  - S' 를 ++, --, --, +-, +-로 쪼개면,
  - 대응하는 값의 합은 $(+10) + (-1) + (-1) + (+1) + (+1) = 10$ 으로 최대입니다.

- ✓ 따라서 점화식은 다음과 같이 정의됩니다.
  - $i < 0$일 때, $dp[i] = -\infty$
  - $i = 0$일 때, $dp[i] = 0$
  - $i > 0$일 때, S'가 x라는 덩어리로 끝난다고 해보자.
    - ▶ 그런 x들 중, $dp[i] = \max\left(dp[i - |x|] + (x\text{에 대응하는 값})\right)$.
    - ▶ 그런 x가 존재하지 않으면, $dp[i] = -\infty$
- ✓ S의 길이를 n이라고 하면, 답은 $dp[n]$입니다.
- ✓ $\mathcal{O}(n)$만에 계산할 수 있습니다.

G. 홍익 투어리스트

`set`, `data_structure`

출제진 의도 – **Hard**

- ✓ 제출 53번, 정답 3명 (정답률 5.66%)
- ✓ 처음 푼 참가자: **정X현**, 85분
- ✓ 출제자: swoon

G. 홍익 투어리스트

- ✓ N과 Q의 범위를 고려하였을 때, 매 쿼리를 $\mathcal{O}(N)$ 미만으로 처리해야 합니다.
- ✓ set을 이용하여 모든 쿼리를 $\mathcal{O}(\log N)$ 이하로 처리할 수 있습니다.

- ✓ 1번 쿼리로 i번 구역이 명소로 지정되면 set에 삽입해주고, 해제된다면 지워줍니다.
  - set의 insert와 erase를 통하여 $\mathcal{O}(\log N)$에 가능합니다.
- ✓ 2번 쿼리로 x가 들어온다면 $idx = (idx + x)\%N$을 통해 idx가 N이 넘지 않도록 갱신해줍니다. 특히 N번째 구역으로 이동할 때 조심해야 합니다.
  - 단순 연산이므로 $\mathcal{O}(1)$입니다.

- ✓ 3번 쿼리는 현재 위치를 idx라 가정할 때, 다음과 같이 간단화할 수 있습니다.
  - idx 이상인 가장 작은 원소를 구하기
  - 존재하지 않는다면, 가장 작은 원소를 구하기
  - 존재하지 않는다면, -1
- ✓ 특정 원소 이상인 가장 작은 원소는 set의 lowerbound를 통하여 $\mathcal{O}(\log N)$에 구할 수 있습니다.

- ✓ 모든 쿼리가 $\mathcal{O}(\log N)$ 이하이므로 총 시간복잡도는 $\mathcal{O}(Q \log N)$ 입니다.
- ✓ BBST 외에도 구간 내 가장 작은 원소를 리턴하는 segment tree를 짜는 풀이도 존재합니다.

H. 리그전 오브 레전드

`math`, `prefix_sum`

출제진 의도 – **Hard**

- ✓ 제출 50번, 정답 3명 (정답률 6.00%)
- ✓ 처음 푼 참가자: **정X현**, 69분
- ✓ 출제자: `Green55`

H. 리그전 오브 레전드

- ✓ 편의상, 쿼리 `l r`에 대한 답을 $fun(l, r)$ 이라고 합시다.

$$fun(l, r) = \sum_{(i,j)}^{l \le i < j \le r} a_i \times a_j$$

- ✓ 일단 $fun(l, r)$ 을 딱 한번만 구하는 것도 쉽지 않습니다.
- ✓ $l = 1, r = n$ 이라면 naive한 방법은 $\mathcal{O}(n^2)$ 이 소요되기 때문입니다.

- ✓ 여기서 두 다항식의 곱은, 모든 항의 쌍을 곱한 것의 합이라는 것을 이용합시다.
- ✓ 우리가 알고 싶은 '모든 쌍 곱의 합'과 상당히 유사하기 때문입니다.

$$(a_l + a_{l+1} + \cdots + a_r)^2 = (a_l^2 + a_{l+1}^2 + \cdots + a_r^2) + 2 \times \sum_{i,j}^{l \le i < j \le r} a_i \times a_j$$

$$fun(l, r) = \frac{(a_l + a_{l+1} + \cdots + a_r)^2 - (a_l^2 + a_{l+1}^2 + \cdots + a_r^2)}{2}$$

- ✓ 따라서, $fun(l, r)$ 을 구하기 위해서는
  - 구간 $l \leq i \leq r$ 에서의 모든 원소의 합과
  - 모든 원소의 제곱의 합

  을 알아야 합니다. 이것은 $\mathcal{O}(n)$ 에 가능합니다.
- ✓ 하지만 우리는 $fun(l, r)$ 을 Q 번 구해야 하므로, 더 빠른 방법이 필요합니다.
- ✓ 어떤 구간의 합을 $\mathcal{O}(1)$ 에 구하기 위해서, **누적합**이라는 테크닉을 사용할 수 있습니다.

H. 리그전 오브 레전드

- 다음과 같은 배열 p_1, p_2를 정의합시다.
  - $p_1[i] = a_1 + a_2 + \cdots + a_i$
  - $p_2[i] = a_1^2 + a_2^2 + \cdots + a_i^2$
- 다음의 방법을 사용하면 p_1, p_2를 $\mathcal{O}(N)$ 한번으로 채울 수 있습니다.
  - $p_1[0] = 0, p_1[i] = p_1[i-1] + a_i$
  - $p_2[0] = 0, p_2[i] = p_2[i-1] + a_i^2$
- 이제 우리는 원하는 값을 $\mathcal{O}(1)$에 구할 수 있습니다.
  - $a_l + a_{l+1} + \cdots + a_r = p_1[r] - p_1[l-1]$
  - $a_l^2 + a_{l+1}^2 + \cdots + a_r^2 = p_2[r] - p_2[l-1]$

- ✓ 누적합 배열의 전처리에 $\mathcal{O}(N)$ 이 걸리고
- ✓ Q 개의 쿼리가 한 번에 $\mathcal{O}(1)$ 이 소요되므로
- ✓ 따라서 $\mathcal{O}(N+Q)$ 에 전체 문제를 해결 할 수 있습니다.

I. 마을 구하기

`sort, case_work`

출제진 의도 – **Challenging**

- ✓ 제출 11번, 정답 1명 (정답률 9.09%)
- ✓ 처음 푼 참가자: **김X헌**, 165분
- ✓ 출제자: `extra1563`

- ✓ 터지는 폭탄의 종류를 B, 폭탄 쉴드를 b라 할 때, 최적의 배치를 찾는 과정은 아래와 같습니다.
- ✓ 양쪽을 터뜨리는 폭탄의 특성을 생각해보면 폭탄의 피해를 받는 지역을 최대한 겹치게 하여야 피해를 줄일 수 있음을 알 수 있고, 이에 폭탄(B)들을 서로 인접하게 모아두어야 합니다.
  - 쉴드를 폭탄 사이에 두는 배치도 생각해볼 수 있으나 쉴드(b)는 폭탄(B)에 사전 순으로 뒤에 있기에 항상 폭탄을 모아두는 게 최적입니다.

- ✓ 모아둔 폭탄들의 위치만 고려해 볼 때 가능한 배치는 “…BB…”, “…BB”, “BB…” 입니다.
- ✓ 세 배치는 각각 어떠한 상황에서 최적의 배치가 될 수 있을까요?
- ✓ 사전 순으로 가장 앞서는 배치를 찾아야 한다는 점을 유의하며 케이스를 나눠봅시다.

- ✓ Case 1. “…BB…” 의 경우, B보다 사전 순으로 앞선 대문자 알파벳이 존재해야 하며 b가 2개 이상 존재해야 합니다.
  - B보다 사전 순으로 앞선 대문자 알파벳이 존재하지 않는다면 “BB…” 와 같은 배치에 사전 순으로 밀리게 됩니다.
  - b가 1개라면 “…BBb…” 배치일 텐데 이는 “…bBB” 와 같은 배치에 피해량에서 밀리게 됩니다.
  - b가 0개라면 “…BB…” 배치일 텐데 이는 “…BB” 와 같은 배치에 피해량에서 밀리게 됩니다.
  - 따라서 위의 조건을 만족하는 경우엔 “…bBBb…” 와 같이 배치해야 합니다.

- ✓ Case 2. “…BB” 의 경우, B보다 사전 순으로 앞선 대문자 알파벳이 존재해야 하며 b가 2개 미만 존재해야 합니다.
  - B보다 사전 순으로 앞선 대문자 알파벳이 존재하지 않는다면 “BB…” 와 같은 배치에 사전 순으로 밀리게 됩니다.
  - b가 2개 이상 존재한다면 앞서 나눈 1번째 케이스의 조건과 정확히 일치하므로 1번째 케이스에 속하게 됩니다.
  - b가 1개라면 가장 좌측 B의 좌측에 b를 배치하여 피해량을 줄일 수 있습니다.

- ✓ Case 3. “BB…” 의 경우, B가 맨 앞에 있으므로 B보다 사전 순으로 앞선 대문자 알파벳이 존재하지 않습니다.
  - b가 1개 이상 존재한다면 가장 우측 B의 우측에 b 1개를 배치하여 피해량을 줄일 수 있습니다.
- ✓ 따라서 우리는 “B보다 사전 순으로 앞선 대문자 알파벳이 존재하는가?” 를 기준으로 1, 2번째 케이스와 3번째 케이스를 구분할 수 있고, “b의 개수가 2개 이상 존재하는가?” 를 기준으로 1, 2번째 케이스를 구분할 수 있습니다.

- ✓ 이제 "..." 에 해당하는 부분에 먼저 배치해놓은 B, b를 제외한 나머지 문자들을 오름차 순으로 정렬한 채로 배치하면 됩니다.
  - 1번째 케이스에서 나눠진 두 부분에 배치할 때는 먼저 배치한 부분의 양 끝이 b라는 것을 고려해 (정렬한 대문자)(b보다 작은 정렬된 소문자) "bBBb" (b보다 크거나 같은 정렬된 소문자)의 형태로 배치해야 합니다.
- ✓ 모든 B와 필요한 b들을 먼저 배치해놓고 나머지를 배치하기에 카운팅 정렬을 이용하여 알파벳 개수를 관리한다면 $\mathcal{O}(N)$에 해결할 수 있습니다.

J. 구슬 보내기

`sort, graphs, dijkstra`

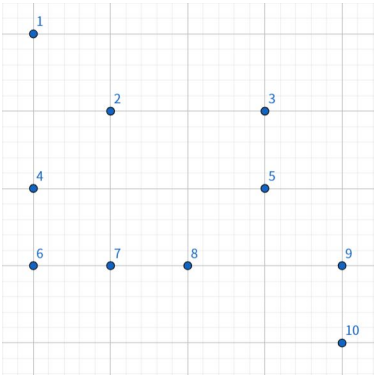
출제진 의도 – **Challenging**

- ✓ 제출 0번, 정답 0명 (정답률 00.00%)
- ✓ 처음 푼 참가자:
- ✓ 출제자: hwon233

- ✓ 구슬 발사기를 정점으로 하는 그래프로 모델링할 수 있습니다.
- ✓ 먼저, 문자열로 표기된 방향을 관리하기 쉽도록 정수로 바꿔줍시다.
- ✓ 방향을 북쪽부터 시계 방향으로 0부터 대응하면 i번 발사기를 d_{target}로 회전하는데 필요한 비용은 $c_i \times ((d_{target} - d_i + 8) \mod 8)$입니다.

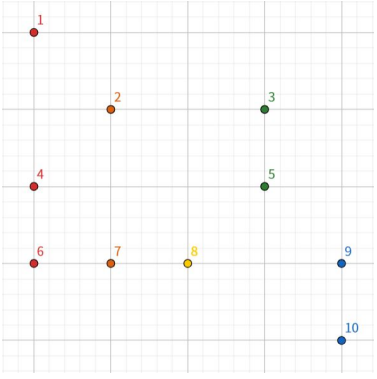
J. 구슬 보내기

✓ 예를 들어, 그림과 같이 발사기가 위치해 있다고 해봅시다.



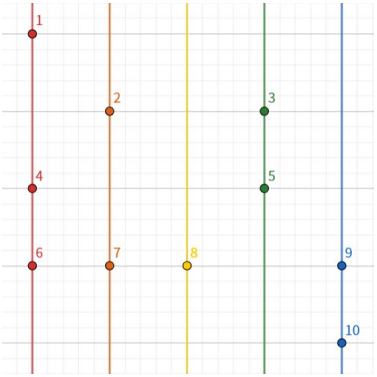
J. 구슬 보내기

- ✓ 어떤 발사기에서 도달 가능한 다른 발사기의 위치를 관찰해봅시다.
  - 구슬을 남, 북 방향으로 발사했을 경우, x 좌표가 같은 발사기끼리 도달할 수 있습니다.



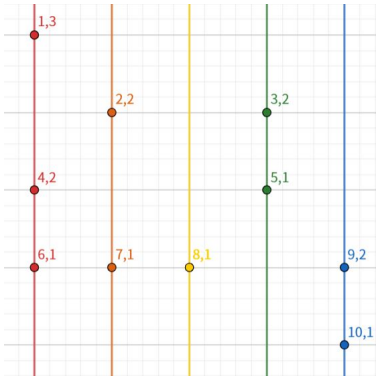
J. 구슬 보내기

- ✓ 서로 도달 가능한 발사기들을 그룹지을 수 있습니다.
  - 남, 북 방향에 대하여 그룹지었을 때, 각 그룹을 y_i 로 정렬합니다. 그룹 내의 발사기들이 남쪽에서 북쪽으로 정렬됩니다.



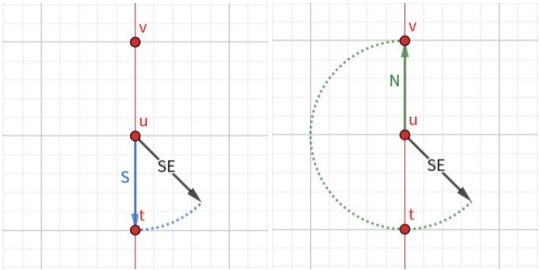
J. 구슬 보내기

- ✓ 구슬이 이동 중에 발사기를 만나면 무조건 그 발사기를 방문합니다.
- ✓ 즉, 그룹 내에서 j 번째인 발사기를 발사한다면 다음에 방문하는 발사기는 $j-1$ 번째 발사기나 $j+1$ 번째 발사기입니다.



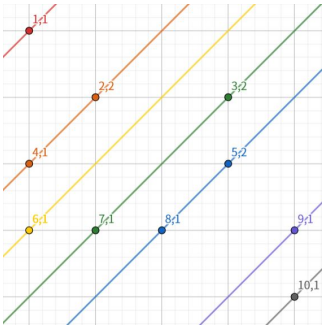
J. 구슬 보내기

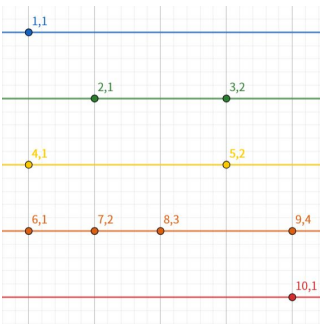
- ✓ j 번째 발사기의 초기 방향을 알고, $j-1$ 번째 발사기로의 방향 또는 $j+1$ 번째 발사기로의 방향이 결정되어 있으므로, 회전하는데 필요한 비용을 계산할 수 있습니다.
  - j 번째 발사기의 번호가 u, $j-1$ 번째 발사기의 번호가 v, $j+1$ 번째 발사기의 번호가 t 일 때, 그래프에 u 에서 v 로 향하는 간선과 u 에서 t 로 향하는 간선을 추가합니다.
  - 이 때, 각 간선의 가중치는 회전하는 데 필요한 비용입니다.

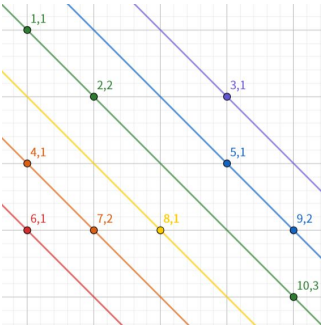


J. 구슬 보내기

- ✓ 나머지 3가지 방향에 대해서도 같은 과정을 진행합니다.
  - 남서, 북동 방향으로는 $x_i - y_i$ 가 같은 발사기끼리 도달할 수 있습니다.
  - 동, 서 방향으로는 y_i 가 같은 발사기끼리 도달할 수 있습니다.
  - 남동, 북서 방향으로는 $x_i + y_i$ 가 같은 발사기끼리 도달할 수 있습니다.







J. 구슬 보내기

- ✓ 만들어진 그래프에서 다익스트라 알고리즘을 사용하여 최단 경로의 길이를 구합니다.
- ✓ 이 때, 최단거리를 저장하는 배열을 업데이트를 할 때 더 짧은 거리를 가진 이웃 정점도 함께 업데이트하면 경로도 구할 수 있습니다.

- ✓ 적절한 자료구조를 사용한다면 발사기들을 그룹짓는데 $\mathcal{O}(n \log n)$ 이 소요됩니다.
- ✓ 각 그룹을 정렬하는데에도 $\mathcal{O}(n \log n)$ 이 소요됩니다.
- ✓ 다익스트라 알고리즘은 $\mathcal{O}(V \log E)$ 이 소요되는데 정점의 개수 $V = n$, 간선의 개수 $E \leq 8n$ 이므로 $\mathcal{O}(n \log n)$ 으로 표현할 수 있습니다.
- ✓ 따라서 모든 과정을 $\mathcal{O}(n \log n)$ 이내에 해결할 수 있습니다.