# 2025 KSAAC Winter

Official Solutions

by

Automata



2025년 2월 15일

| 문제 | | 의도한 난이도 | 출제자 |
|---|---|---|---|
| **A** | 아름다운 순열 | 어려워요 | flakepowders |
| **B** | 저녁 태권도 | 어려워요 | eric00513 |
| **C** | 미술 수업 | 어려워요 | eric00513 |
| **D** | KSA 문자열 2 | 어려워요 | sooahn07 |
| **E** | 수열의 점수 | 어려워요 | mythofys |
| **F** | 멀티버스를 여행하는 한별이를 위한 안내서 | 어려워요 | davidkim0, flakepowders |
| **G** | 자습 째기 | 어려워요 | flakepowders |
| **H** | 쿠키 공장 | 어려워요 | mythofys |
| **I** | KSA 수열과 쿼리 | 어려워요 | mythofys |
| **J** | 애벌레와 트리 | 어려워요 | edward3378 |

## Sponsor









## Staff

- 운영진
  - davidkim0
  - edward3378
  - eric00513
  - flakepowders
  - mythofys
  - raehwan0418
  - sooahn07
- 검수진
  - aaron0526
  - ai4youej
  - dongwook7
  - gs22059
  - hjroh0315
  - kep1er07
  - kwoncycle
  - minsung05
  - parkky
  - runnie0427
  - sjhi00
  - utilforever
  - wizardrabbit

# A. 아름다운 수열

`ad_hoc, constructive`
출제진 의도 – **어려워요**

- 제출 245번, 정답 146명 (정답률 63.265%)
- 처음 푼 사람: **dabbler1**, 0분
- 출제자: `flakepowders`

## A. 아름다운 수열

- ✓ $1, 2, 3, \cdots, N$ 과 같이 구성하면 모든 두 원소 쌍에 대해 차이와 거리가 같아집니다.
- ✓ 따라서 항상 조건을 만족할 수 있습니다.

# B. 저녁 태권도

`constructive, implementation`
출제진 의도 – **어려워요**

- 제출 265번, 정답 99명 (정답률 37.358%)
- 처음 푼 사람: **dabbler1**, 7분
- 출제자: `eric00513`

- **서브태스크 2**에서는 각 학생마다 +X, -X, X+, X- 중 하나를 배정하여 모든 경우를 시도해보면 됩니다.
- **서브태스크 3**에서 우선 $A_j + B_j > N$인 $j$가 있는 경우 답이 존재하지 않음을 알 수 있습니다.
- 다음으로 $N$개의 X를 모두 배정할 수 있는지 확인해야 합니다.
- $j$번째 날에 필수적으로 $A_j$개의 +과 $B_j$개의 -가 필요하기 때문에 X가 들어갈 수 있는 곳은 $N - A_j - B_j$곳입니다.
- 만약 $j = 1, 2, \cdots, M$에 대해 $N - A_j - B_j$의 합이 $N$보다 작다면 이는 답이 존재하지 않음을 의미합니다.
- 이외의 경우에는 답이 항상 존재하고, 조건에 맞게 학생들을 배정하면 됩니다.

# C. 미술 수업

`binary_search`
출제진 의도 – **어려워요**

- 제출 278번, 정답 84명 (정답률 30.576%)
- 처음 푼 사람: **crispcofactor**, 4분
- 출제자: `eric00513`

- 구하는 교점의 개수를 $y$ 좌표가 $0$ 보다 큰 교점의 개수와 $x$ 축 위에 놓여있는 교점의 개수의 합으로 생각합시다.
- **서브태스크 1**에서는 서로 다른 $Y_i$ 의 개수를 $m$ 이라고 하면, 답은 $m^2 + 2m$ 입니다.
- 이제 일반적인 경우를 다뤄봅시다.
- 기울기가 같은 직선은 교점을 가지지 않으므로, $(X_i, Y_i)$ 를 지나고 기울기가 $-1$ 인 직선과 $(X_j, Y_j)$ 를 지나고 기울기가 $1$ 인 직선의 교점을 고려합니다.
- 두 직선이 $y$ 좌표가 $0$ 보다 큰 교점을 가지려면 $X_i + Y_i > X_j - Y_j$ 이 성립해야 합니다.
- 서로 다른 교점의 개수를 구해야 하므로 사전에 직선들의 중복 제거를 해두면 좋습니다.
- 이를 모든 $(i, j)$ 쌍에 대해 확인하면 $\mathcal{O}\left(N^2\right)$ 으로 **서브태스크 2**를 해결할 수 있습니다.

- **서브태스크 3**을 해결하기 위해서는 서로 다른 $X_i + Y_i$를 담은 정렬된 배열을 만들고 이분 탐색을 이용하여 $j = 1, 2, \cdots, N$에 대해 $X_j - Y_j$보다 큰 값의 개수를 세어주면 됩니다.
- 반대로 $X_j - Y_j$를 담은 배열을 만들고 $X_i + Y_i$보다 작은 값의 개수를 세어줘도 됩니다.
- 시간복잡도는 $\mathcal{O}(N \log N)$입니다.
- 이러한 과정을 투 포인터로 해결할 수도 있습니다.

# D. KSA 문자열 2

`string`, `greedy`
출제진 의도 – **어려워요**

- 제출 194번, 정답 40명 (정답률 20.619%)
- 처음 푼 사람: **flappybird**, 8분
- 출제자: sooahn07

## D. KSA 문자열 2

- KSAAC 운영진이 좋아하는 문자열을 **KSA 문자열**이라고 합시다.
- 문자 삭제의 시행 횟수는 쉽게 알 수 있으므로 문자 이동의 시행 횟수를 구하는 것이 핵심입니다.
- **서브태스크 1**는 K가 한 개이므로 S와 A의 배치에 따라 경우를 나눠 해결할 수 있습니다.
- **서브태스크 2**는 삭제를 시행하지 않는 경우이며 완성된 **KSA 문자열**이 A로 끝납니다.
- 따라서 $X$ 를 맨 뒤부터 살폈을 때 문자들은 **ASKASK...**와 같이 배열되어야 합니다.
- 단순히 $X$ 를 뒤에서부터 살피며 **ASKASK...** 규칙에 어긋난 알파벳을 세면 됩니다. 문자는 맨 앞에만 추가될 수 있기 때문입니다.

## D. KSA 문자열 2

- **서브태스크 3**은 **서브태스크 2**와 같이 해결할 수 있지만 추가적인 처리가 필요합니다.
- 삭제 연산을 몇 번 수행할지, **KSA 문자열**이 어느 알파벳으로 끝나야 할지 파악하기 위해 K, S, A의 개수를 세어 경우를 나눕니다.
- 또한 K가 맨 앞에 위치하지 않는 경우에 대한 예외처리가 필요합니다.

# E. 수열의 점수

`greedy`
출제진 의도 – **어려워요**

- 제출 161번, 정답 48명 (정답률 29.814%)
- 처음 푼 사람: **mathang08**, 28분
- 출제자: `mythofys`

## E. 수열의 점수

- **서브태스크 1**에서 $X = 1$이므로 어떤 $Y$를 선택하여도 $A_3 \leq 0$입니다. 따라서 답은 $3$입니다.
- **서브태스크 2**는 $A$의 점수가 대부분의 $x, y$에 대해 짧다는 것을 관찰한다면 $x \leq X, y \leq Y$의 모든 $x, y$에 대해서 점수를 구해 해결할 수 있습니다.
- **서브태스크 3**에서는 $l$을 $A$의 점수로 정의합시다.
- $A_1$과 $A_2$를 정한다면 $A_{l-1}, A_{l-2}$은 유일하고, 그 역도 성립합니다.
- 따라서 $A_{l-1} = a, A_{l-2} = b$로 두면 수열 $A$의 모든 항을 $k_1a + k_2b$꼴로 나타낼 수 있습니다.
- 이때, $i \leq l$에 대해 $A_i = k_{i1}a + k_{i2}b$로 나타내면 $k_{i1} \geq 0, k_{i2} \geq 0$을 만족합니다.
- 수열의 점수를 최대화하는 $a, b$가 $a = a_0, b = b_0$라고 가정하고, 이 수열을 $A$라고 합시다.
- 또한, $a = 1, b = 1$인 수열을 $B$라고 합시다.

## E. 수열의 점수

- 수열의 첫 두항은 $A_1 = k_{i1}a_0 + k_{i2}b_0 \geq B_1 = k_{i1} \times 1 + k_{i2} \times 1$, $A_2 = k_{j1}a_0 + k_{j2}b_0 \geq B_2 = k_{j1} \times 1 + k_{j2} \times 1$ 이므로 $a = 1, b = 1$ 또한 수열의 점수를 최대화 하는 $a, b$ 임을 알 수 있습니다.
- 따라서 $a = 1, b = 1$ 인 경우에 주어진 조건 내에서 가능한 $A$ 의 최대 길이를 구하여 문제를 해결할 수 있습니다.
- 또한, $a = 1, b = 1$ 인 경우 수열 $A$ 는 피보나치 수열을 역순으로 나열한 것임을 알 수 있습니다.
- 이를 이용해 피보나치 수열의 항을 DP 등을 이용하여 구할 수 있습니다.

# F. 멀티버스를 여행하는 한별이를 위한 안내서

`binary_search`, `number_theory`
출제진 의도 – **어려워요**

- 제출 115번, 정답 17명 (정답률 15.652%)
- 처음 푼 사람: **urd05**, 62분
- 출제자: `davidkim0`, `flakepowders`

- ✓ $\gcd(r, 10) = 1$ 이 성립하는 임의의 정수를 $r$ 라 합시다.
- ✓ 오일러 정리에 의해 아래 두 사실이 성립합니다.
  - $\gcd(r, 2) = 1$ 이므로 $r^{2^7} \equiv 1 \pmod{2^8}$
  - $\gcd(r, 5) = 1$ 이므로 $r^{2^2 \times 5^7} \equiv 1 \pmod{5^8}$
- ✓ 이때 $\operatorname{lcm} 2^7, 2^2 \times 5^7 = 10^7$ 이므로 $r^{10^7} \equiv 1 \pmod{10^8}$ 이 성립합니다.
- ✓ 이제 임의의 양의 정수 $s$ 가 $s = 2^p \times 5^q \times r (p, q \geq 0;\ \gcd(r, 10) = 1)$ 의 형태를 갖는다고 합시다.
- ✓ $s^{10^7+8} = 2^{p(10^7+8)} \times 5^{q(10^7+8)} \times r^{10^7+8}$ 는 $2^8$ 과 $5^8$ 로 모두 나누어떨어집니다.
- ✓ 따라서 $b^{10^7+8} \equiv b^8 \pmod{10^8}$ 이 성립합니다.

## F. 멀티버스를 여행하는 한별이를 위한 안내서

- ✓ 곱셈 사이클이 $2^a \times 5^b$ 의 형태임을 알았으니, 각 지수를 구해봅시다.
- ✓ 가능한 후보는 $0$ 부터 $7$ 까지의 정수입니다.
- ✓ $N$ 의 값에 관계없이, $N \times K^{10^8}$ 은 이미 사이클 안에 포함되었을 것입니다.
- ✓ 따라서 "? $10^8$ $10^8 +$ 사이클 길이의 후보" 쿼리를 사용하면, 후보가 사이클의 길이보다 크거나 같은지 확인할 수 있습니다.
- ✓ $a$ 의 값을 구할 때는 $2^a \times 5^7$ 을 시도해 보고, $b$ 의 값을 구할 때는 $2^7 \times 5^b$ 을 시도해 봅시다.
- ✓ 각 값은 이분 탐색으로 $3$ 번의 쿼리로 구할 수 있으므로, 사이클의 길이를 확정하는 데는 총 $6$ 번의 쿼리가 필요합니다.
- ✓ 마찬가지로 사이클의 시작 위치를 찾을 때도 이분 탐색을 이용하며, $4$ 번의 쿼리가 필요합니다.
- ✓ 따라서 총 $10$ 번의 쿼리로 문제를 해결할 수 있습니다.

# G. 자습 째기

`binary_search`, `constructive`, `knapsack`
출제진 의도 – **어려워요**

- 제출 92번, 정답 1명 (정답률 1.087%)
- 처음 푼 사람: **flappybird**, 294분
- 출제자: `flakepowders`

- 먼저 **서브태스크 1**을 풀어봅시다.
- $x$ 번째 자습까지 참석한다고 가정해봅시다. 그렇다면, 모든 선생님에 대해 아래 둘 중 하나는 성립시켜야만 합니다.
  - I. $x$ 번째 이후에 자습 감독을 맡지 않으신다.
  - II. $x$ 번째 이후의 자습에서 감독을 맡더라도 돌지 않으신다.
- 이제 두 선택지를 택했을 때 각각의 경우를 생각해봅시다.

## G. 자습 깨기



- 선생님 $y$에 대해서 I.를 성립하게 하려면, $x$번째 이후 $y$의 등장 횟수만큼 수열 $t$를 수정해야 합니다.
- 그 대신, $y$가 $t$의 $x$번째 항과 그 이전에는 몇 번 등장하던 상관없습니다. 따라서, $x$번째와 그 이전에서의 등장 횟수만큼 **자유 공간**을 얻었습니다.
- 자유 공간이란, 다른 조건을 망치지 않고도 다른 값으로 변경할 수 있는 $t$의 원소의 수입니다.
- 선생님 $y$에 대해서 II.를 성립하게 하려면, $x$번째 자습 직후에 남은 감독 횟수만큼 앞에 $y$를 추가로 등장시켜줘야 합니다. 따라서 그 수만큼 수열 $t$를 수정해야 하고, 또 자유 공간을 소모합니다.
- 초기 상태에서도 미리 남은 감독 횟수가 $0$이 되는 선생님이 계신다면, 남은 감독일은 자유 공간으로 간주할 수 있습니다.

- 따라서, 자유 공간 총 증감량이 $0$ 이상이어야 하고, 그러면서도 $t$를 최소한 수정하게끔 하면 됩니다.
- 둘을 각각 무게와 가치로 생각하면, 전형적인 $\mathcal{O}(NM)$ 냅색 DP로 해결할 수 있습니다.
- II.에 해당되는 선생님이 적어도 한 분은 계시도록 해야 하므로, 따로 처리해줘야 함에 유의해주세요.
- 이제, $x$번째 자습 직후부터 째기 시작하기 위해서 필요한 가장 적은 수정 횟수를 계산할 수 있습니다.
- 따라서, 이분 탐색으로 조건을 만족할 수 있는 최소 $x$를 찾아주면 됩니다. 시간 복잡도는 $\mathcal{O}(NM \log N)$ 입니다.

## G. 자습 때기



- 이제 전체 문제를 풀어봅시다.
- $B$ 회까지 결석 처리가 되어도 괜찮다는 것은 직관적으로는 $B$ 회의 수정 기회가 더 주어진다는 것과 매우 비슷합니다.
- 결석 처리가 될 날들을 결석 처리가 되지 않게 수정하는 대신 수정 기회를 $1$ 회 아낄 수 있기 때문입니다.
- 그런데, 결석 처리가 되는 날들은 누가 감독하셔도 상관 없습니다. 따라서, 결석 처리가 되는 날들은 자유 공간으로 간주할 수 있습니다.
- 따라서, $B$ 회 결석 처리가 되어도 괜찮다는 것은 수정 기회와 자유 공간을 $B$ 씩 더 받는다는 것과 같습니다.
- $A + B$ 개의 값이 수정된 수열로부터 출력해야 할 $t'$ 를 어떻게 구성하느냐 하는 아이디어는 if 방향의 증명 중에 얻을 수 있습니다.

## G. 자습 때기

- ”배정된 선생님이 없는 날(None)”을 수열 $t$ 앞에 $B$ 개 붙인 뒤에 $A+ = B;\ B := 0$ 을 적용한 버전의 변형 문제를 만든 뒤 두 문제의 답이 일대일 대응되는 논리로 증명합니다.
- 먼저 if 방향을 증명해봅시다.
- 변형된 문제의 답을 수열 $S$ 라 하고, 이때 $x$ 회차부터 쨀 수 있다고 합시다. (이번에도 앞에 $B$ 개를 이어붙였지만 인덱싱은 바꾸지 않고, 즉 $1 - B$ 회차부터 $N$ 회차까지 있습니다.)
- $S$ 에서 수정되었던 $A + B$ 개의 값들 중 $B$ 개를 돌려놓는다고 했을 때, 하나의 값을 되돌릴 때마다 결석 처리는 최대 1번씩 더 됩니다.
- 음이 아닌 정수 인덱스가 총 $B$ 개 있으므로, 이들을 우선적으로 None으로 되돌린다면 $x$ 회차부터 째도 결석 처리를 $B$ 회 이하로 되고도 전부 되돌릴 수 있습니다.
- 이제 수열의 앞 $B$ 개를 제거하여 양의 정수 인덱스만 남깁니다. 이게 원본 문제의 답이 됩니다.

## G. 자습 때기



- ✓ 반대로 원본 문제의 답을 수열 $S$라 하고, 이때 $x$ 회차부터 쨀 수 있다고 합시다.
- ✓ $x$ 회차 이후에 결석 처리가 되는 날들을 모아봅시다. 이들을 $S_{a_1}, S_{a_2}, \cdots, S_{a_B}$ 라고 합시다.
- ✓ 새로운 $N+B$ 길이의 수열 $S'$를 아래와 같이 정의합니다. 인덱스는 $1-B$ 부터 $N$ 까지입니다.
- ✓ $$\begin{cases} S'_{-i+1} = S_{a_i} (1 \le i \le B) \\ S'_i = (x\text{회차부터 감독 안 도시는 임의의 선생님}) (1 \le i \le N, i \in \{a_1, a_2, ..., a_B\}) \\ S'_i = S_i (1 \le i \le N, i \notin \{a_1, a_2, ..., a_B\}) \end{cases}$$
- ✓ 이렇게 구성하면, $S'$는 $S$에서 최대 $B$번을 더 수정하면서 도달할 수 있는, $x$ 회차부터 쨀 수 있는 변형 문제의 답이 됩니다.
- ✓ 따라서 두 문제의 답은 같고, 변형판을 풀기 위해 DP의 두 초기값만 조금 수정해주면, 전체 문제도 $\mathcal{O}(NM \log N)$에 해결할 수 있습니다.

# H. 쿠키 공장

`segment_tree`, `offline_query`
출제진 의도 – **어려워요**

- 제출 77번, 정답 16명 (정답률 20.779%)
- 처음 푼 사람: **kiki7000**, 83분
- 출제자: `mythofys`

## H. 쿠키 공장



- **서브태스크 1**에서는 쿠키를 공장에 납품할 때, 각 공장에 대해 납품을 최대한 빨리 끝낼 때, 그 기간을 공장 $i$의 최적 납품 기간이라고 정의합시다.
- 최적 납품 기간이 겹친다면 적어도 겹치는 일수만큼 납품 일정이 밀린다는 사실을 관찰할 수 있습니다.
- 이를 활용하면 직사각형의 넓이의 합집합을 구하는 문제처럼 $s_i$를 기준으로 정렬하여 스위핑을 통해서 $\mathcal{O}\left(NQ \log N\right)$에 문제를 해결할 수 있습니다.
- 정렬 대신 오프라인 쿼리를 활용하여 $\mathcal{O}\left(N(Q + \log N)\right)$에 문제를 해결할 수도 있습니다.
- **서브태스크 2**는 하나의 관찰을 하면 문제를 해결할 수 있습니다.
- 각 공장이 쿠키를 납품받기 시작하여 그 공장에 우선적으로 쿠키를 배급하였을 때 마지막으로 쿠키를 납품하는 날을 $e_i$, 납품을 시작하는 날을 $s_i$로 둡시다.

## H. 쿠키 공장

- $s_i$ 가 작은 순으로 정렬을 합시다. 이때, 답은 $\max(s_i + \sum_{k=i}^{n}(e_k - s_k))$ 임을 증명할 수 있습니다.
- 위 공장들의 요청 $i, j$ 에 대해 $s_j < e_i$ 라면 $(s_i, e_i), (s_j, e_j)$ 의 두 요청을 최적으로 해결할 때 $(s_i, e_i + e_j - s_j)$ 를 최적으로 해결하는 것과 같습니다.
- 이렇게 가능한 모든 요청을 최대한 적은 개수의 요청으로 합쳐 **유일하게** 나타낼 수 있습니다.
- 합친 이후의 요청들 중 $\max(e_i)$ 가 답임을 알 수 있습니다.
- 이때, 합친 이후의 $\max(e_i)$ 는 기존 마지막 구간에 몇 개의 구간이 합쳐졌는지에 의해 결정됩니다.
- 두 구간 $(i, j)$ 만 존재하는 경우에 대해 구간이 이어지지 않았다면 $e_i + s_j - e_j < e_j$ 일 것이고, 그렇지 않다면 $e_i + s_j - e_j \geq e_j$ 일 것입니다.
- 또한, 두 구간이 이어졌다면 답은 $e_i + s_j - e_j$ 이고, 그렇지 않다면 답은 $e_j$ 일 것입니다.

## H. 쿠키 공장

- 따라서 답은 $\max(e_i + s_j - e_j, e_j)$ 임을 알 수 있습니다.
- 이를 여러 개의 구간에 확장하여 적용시켜 $\max$ 값을 구하면 답을 구할 수 있습니다.
- 따라서 위에서 제시한 답을 빠르게 구하면 문제를 해결할 수 있습니다.
- 좌표압축과 세그먼트 트리를 활용하여 1번 노드부터 $n$ 번째 노드까지 기본값을 $i$ 로 초기화한 이후 모든 $i$ 에 대해 1번 노드부터 $i$ 번째 노드까지 $e_i - s_i$ 를 더해준 후 $\max$ 값을 취하는 방식으로 최적화 할 수 있습니다.
- 매 쿼리가 반영되어도 $s_i$ 의 값이 일정하여 노드의 순서가 바뀌지 않으므로 각 노드의 값을 갱신하여 $\mathcal{O}(N \log N + Q \log N)$ 에 문제를 해결할 수 있습니다.
- **서브태스크 3**에서는 세그먼트 트리에 $e_i - s_i = 0$ 인 경우 이 값을 반영하지 않게 하는 갱신 방식을 추가하여 문제를 해결할 수 있습니다.

- ✓ 미리 바뀔 $s_i, e_i$에 대해 구간을 전처리하여 세그먼트 트리의 노드를 $N+Q$개 만들어 놓고, 기존 $s_i, e_i$에 해당하는 노드의 값을 $e_i = s_i$로 갱신하여 반영하지 않게 하고, 새로운 구간의 노드를 $e_i = s_i$에서 갱신된 값으로 반영하게 하는 방식을 사용하여 문제를 해결할 수 있습니다.
- ✓ 따라서 $\mathcal{O}\left((N+Q)\log(N+Q)\right)$에 문제를 해결할 수 있습니다.

# I. KSA 수열과 쿼리

`segment_tree, implementation`
출제진 의도 – **어려워요**

- 제출 178번, 정답 6명 (정답률 3.371%)
- 처음 푼 사람: **kiki7000**, 154분
- 출제자: `mythofys`

- **서브태스크 1**에서는 모든 쿼리를 배열에 직접 수행하면 각 쿼리를 $\mathcal{O}(N)$에 해결할 수 있으므로 이를 구현하면 됩니다.
- **서브태스크 2**는 수열의 각 원소가 $0, 1$ 중 하나의 상태를 유지하므로 $1$번 쿼리가 입력되는 구간의 $0$과 $1$의 개수를 바꾸는 것을 알 수 있습니다.
- 이를 lazy propagation을 이용하여 구현하여 주면 해결할 수 있습니다.
- **서브태스크 3**에서는 서브태스크 2를 확장하여 해결할 수 있습니다.
- 세그먼트 트리의 각 노드에 해당하는 구간에 $0$부터 $k-1$까지의 수가 각각 몇 개 있는지를 노드에 저장하면 $1$번 쿼리는 서브태스크 2와 비슷하게, $2$번 쿼리는 (해당하는 구간의 $i$의 개수) $\times\, i$를 더해 주는 것으로 해결할 수 있습니다.
- 이때 시간복잡도 $\mathcal{O}(QK \log N)$에 문제를 해결할 수 있습니다.

- **서브태스크 4**는 $\mathrm{mod}$ 연산에 주목합시다. $\mathrm{mod}$ 연산에서 각 수가 $k$를 넘길 때만 잘 처리해 준다면 문제를 해결하는데에 어려움이 사라집니다.
- 어떤 구간에 해당하는 수와 그 개수를 저장한 후, linked list로 관리하는 상황을 생각하여 봅시다. 작은 수부터 정렬하여 자신 앞의 수와 뒤 수를 가리켜줍시다.
- 만약, 해당 구간에 업데이트가 필요하다면 두 가지 케이스로 나누어 볼 수 있습니다.
- 먼저, 모든 수가 $k-1$보다 작다면 해당 구간의 합은 $2k$가 증가하는 것을 알 수 있습니다.
- 그렇지 않다면, 해당 구간의 합은 $2k - (\text{해당하는 수}) \times (\text{수의 개수})$만큼 증가함을 알 수 있습니다.
- 이를 통해 실제로 계산하지 않고 레이지 형태로 해당 구간에 값이 얼마 더해졌는지를 저장합시다.

- 위와 같이 구간 전체에 수를 합하는 연산은 실제로 수를 합하지 않고 총 얼마가 더해졌는지를 저장하여 여태까지 더해진 수 + linked list의 마지막 수의 값이 $k$가 넘는지를 체크하는 것으로 대신할 수 있습니다.
- 이제 전체를 $\frac{\sqrt{N}}{\sqrt{\log N}}$ 개의 구간으로 나누어 관리합시다. 쿼리가 구간을 포함한다면 포함되는 구간은 위와 같이 연산하여 $\mathcal{O}(1)$에 처리할 수 있고, 포함하지 않는다면 레이지를 반영하고, 직접 수를 더해준 뒤 다시 정렬합시다.
- 업데이트 쿼리는 최악의 경우 $\sqrt{N \log N}$ 개의 구간을 갱신하여야 하고, $\frac{2\sqrt{N}}{\sqrt{\log N}}$ 개의 수를 직접 갱신하여 정렬하여야 하므로 $\mathcal{O}\left(\sqrt{N \log N} + \frac{2\sqrt{N}}{\sqrt{\log N}} \log \frac{2\sqrt{N}}{\sqrt{\log N}}\right)$에 각 쿼리를 해결할 수 있습니다.

## I. KSA 수열과 쿼리



- 출력 쿼리는 구간을 포함하는 경우 위에서 구해놓았던 합을 더하고, 그렇지 않은 경우 레이지를 전파하고 각 값을 더하여 $\mathcal{O}\left(\sqrt{N\log N}+\frac{2\sqrt{N}}{\sqrt{\log N}}\log\frac{2\sqrt{N}}{\sqrt{\log N}}\right)$에 해결할 수 있습니다.
- $\sqrt{N\log N}$와 $\frac{2\sqrt{N}}{\sqrt{\log N}}\log\frac{2\sqrt{N}}{\sqrt{\log N}}$를 각각 제곱하여 비교하면 $\sqrt{N\log N}$가 더 큽니다.
- 따라서 시간복잡도 $\mathcal{O}\left(N+Q\sqrt{N\log N}\right)$에 문제를 해결할 수 있습니다.
- 버킷 크기를 그냥 $\sqrt{N}$으로 잡을 경우 $\mathcal{O}\left(N+Q\sqrt{N}\log N\right)$이 되므로 유의해야 합니다.
- 이외에도 $k<\sqrt{N}, k>\sqrt{N}$인 경우에 나눠서 비재귀 레이지 세그먼트 트리를 활용한 $\mathcal{O}\left(Q\sqrt{N}\log N\right)$ 풀이가 존재합니다.
- 또한, 버킷 크기를 조금 더 크게 잡으면 거의 $\mathcal{O}\left(N+Q\sqrt{N}\right)$에 동작하게 할 수 있습니다.



# J. 애벌레와 트리

`lca`, `tree_dp`
출제진 의도 – **어려워요**

- ✓ 제출 6번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: **todo**, q분
- ✓ 출제자: `edward3378`

- 편의상 애벌레의 길이(애벌레가 차지하는 정점의 개수)를 $k$로 합시다.
- 먼저, $N$이 작을 때를 풀어봅시다.
- 애벌레를 나타내는 순서쌍 (머리 번호, 꼬리 번호)를 정점으로 하는 그래프를 생각해봅니다.
- 만약 두 순서쌍이 한번에 이동이 가능하다면, 두 애벌레를 나타내는 정점을 간선으로 잇습니다.
- 이 때, 이 그래프의 정점 개수는 최대 $N^2$개임을 알 수 있고, 간선 개수를 주목해봅시다.

- 머리가 $u$인 애벌레가 머리로 이동해서 다음 상태로 갈 수 있는 가짓수는 $\deg(u) - 1$개이면서 이러한 애벌레의 개수는 많아야 $N$개입니다.
- 꼬리가 특정 번호인 애벌레도 마찬가지이므로 각 애벌레에 대해 다음 상태로 갈 수 있는 가짓수를 모두 합하면 최대 $O\left(N\sum_{i=1}^{N}\deg(i)\right)$ 개입니다.
- 즉, 간선의 개수 또한 $O(N^2)$개 이하입니다.
- 이러한 간선들은 naive하게 $O(N^3)$에 찾으면 서브태스크 1번 문제를 풀 수 있습니다.
- 더 빠르게 찾기 위해 정점 $a$와 인접해 있으면서 $a$와 $b$를 잇는 경로에 포함되는 정점을 전처리 $O(N^2)$을 통해 $O(1)$만에 찾을 수 있습니다.
- 따라서, $O(N^2 + Q)$로 서브태스크 2번 문제를 풀 수 있습니다.

- 이제, 전체 문제를 해결해봅시다.
- 머리가 $h$이고, 꼬리가 $t$인 애벌레가 차지하는 간선들을 주목합니다.
- 이 애벌레가 차지하는 간선 중 아무리 이동해도 변하지 않는 간선을 봅시다.
- 이러한 간선들을 각 애벌레 별로 어떻게 빠르게 구할 수 있을까요?

- 아래와 같은 알고리즘 $Z$를 생각해봅시다.
  - 머리를 현재 꼬리와 멀어지게 꼬리의 반대방향으로 최대한 이동한다.
  - 꼬리를 현재 머리와 멀어지게 머리의 반대방향으로 최대한 이동한다.
  - 이 두 과정을 반복하다가 초기 애벌레와 현재 애벌레가 공유하는 간선 집합이 공집합이거나 같은 순간이 다시 온다면 이 간선 집합을 반환하고 알고리즘을 멈춘다.
- 이 알고리즘이 최악의 경우에는 최대 몇 번 반복되고 멈출 지를 생각해봅시다.

- $Z$는 머리를 더이상 꼬리로부터 멀리 이동시키지 못하면서 꼬리도 마찬가지로 머리로부터 기존보다 멀리 이동시키지 못할 때 종료됩니다.
- 즉, $Z$가 종료되지 않는다면 기존보다 더 멀리 갈 수 있는 경로가 생긴다는 말과 같습니다.
- 각 스텝마다 항상 최대로 이동하기 때문에 새로 더 갈 수 있는 경로는 그 전 경로와 겹치지 않습니다.(겹친다면, 기존 경로에서 그 겹치는 점으로부터 더 멀리 갈 수 있었는데 가지 않았다는 것에 모순)
- 따라서, 알고리즘의 스텝 수를 $t$라 하면, 정점의 개수는 O$(1+2+\cdots+t)$개이므로 $t = O(\sqrt{N})$이 됩니다.
- 각 알고리즘 스텝마다 빠르게 처리하는 방법을 알아봅시다.

- 각 스텝마다 현재까지 모든 애벌레가 차지했던 간선의 교집합을 저장합니다.
- 이러한 교집합은 트리 상에서 두 점 간의 경로로 나타나므로, 경로의 양 끝점만 저장해도 충분합니다.
- 각 스텝을 빠르게 처리하기 위해, 아래 두 쿼리를 각각 tree dp로 $O(1)$, LCA로 $O(\log N)$에 처리하면, 알고리즘 $Z$를 $O(\sqrt{N}\log N)$의 시간복잡도로 시행할 수 있습니다.
  - 정점 $a$에서 시작하여 $a$와 인접한 정점 $b$를 지나지 않는 최장 경로의 길이
  - 정점 $a$에서 $b$로의 경로 중 $x$번째 정점의 번호

- 이렇게 구한 알고리즘 $Z$ 의 반환값이 우리가 찾고자 한 애벌레의 변하지 않는 모든 간선이라는 것을 알 수 있습니다.
- 애벌레 $C$ 의 간선 집합을 $S_C$ 로 정의합시다.
- 이제 본격적으로 애벌레 $C_1$ 이 애벌레 $C_2$ 와 같은 상태로 이동할 수 있는지를 간선 집합에 따라 경우를 나누어 판별합시다.
- $S_{C_1}$ 과 $S_{C_2}$ 가 다른 경우에는 자명하게 답은 NO입니다.
- 따라서 $S_{C_1}$ 과 $S_{C_2}$ 가 같은 경우만 생각합시다.

- Case 1. $S_{C_1}$ 과 $S_{C_2}$ 가 공집합이 아닌 경우
- 이 경우에는 두 애벌레의 방향이 같은지만 고려해주면 됩니다.
- 만약 $S_{C_1}$ 를 이루는 간선들을 차지하는 애벌레의 방향이 다르다면, 이 부분이 고정되었기 때문에 답은 NO입니다.
- 방향이 같다면, 이 부분만 똑같이 고정되었고 나머지 부분은 이동이 가능하므로 $C_1$ 에서 $S_{C_1}$ 이 이루는 경로의 양 끝점에서 잘 이동하면 $C_2$ 로 갈 수 있어 답은 YES입니다.

- Case 2. $S_{C_1}$ 과 $S_{C_2}$ 모두 공집합인 경우
- 일단 머리와 꼬리 방향을 고려하지 말고 $C_1$ 에서 $C_2$ 로 도달 가능한지만 생각해봅시다. 즉, ($C_1$ 의 머리,$C_1$ 의 꼬리) = ($C_2$ 의 머리,$C_2$ 의 꼬리) 또는 ($C_1$ 의 머리,$C_1$ 의 꼬리) = ($C_2$ 의 꼬리,$C_2$ 의 머리)가 될 수 있는지를 고려해봅시다.
- 애벌레 $C_1$ 과 $C_2$ 모두 트리 상에서 임의의 한 점이 머리 또는 꼬리가 되도록 이동할 수 있다는 사실을 알 수 있습니다.
- $C_1$ 에서 알고리즘 $Z$ 를 시행했을 때 종료 직전 애벌레의 위치를 이용하면 이 애벌레가 차지하는 경로를 포함한 길이 $2(k-1)$ 의 애벌레가 지나갈 수 있는 경로를 찾을 수 있습니다.
- $C_2$ 의 머리 또는 꼬리를 이러한 길이 $2(k-1)$ 의 경로의 중간 정점으로 위치시키면, $C_1$ 과 $C_2$ 는 방향을 신경쓰지 않는다면 일치시킬 수 있음을 알 수 있습니다.

- 이렇게 일치시킨 두 애벌레의 방향을 확인하여 같으면 답은 YES입니다.
- 만약 다르다면, 머리와 꼬리가 뒤바뀐 형태가 될 것입니다.
- 이제 이 애벌레를 reverse할 수 있는지만 판별하면 됩니다.
- 아래 명제 $P$와 이 애벌레가 reversible하다는 것이 동치임을 보이면 됩니다.
- $P$ : 어떤 정점 $u$가 존재하여 $u$에서 시작하고 $v_i (1 \leq i \leq 3)$를 지나는 가장 긴 경로의 길이가 $k-1$ 이상인 $u$와 인접한 서로 다른 세 정점 $v_1, v_2, v_3$이 존재한다.

- ✓ 따라서 각 점에 대해 $P$에서 $u$ 역할이 가능한지 알기 위해 "정점 $a$에서 시작하여 $a$와 인접한 정점 $b$를 지나는 최장 경로의 길이"를 tree dp로 전처리 후 $O(1)$에 구하면 됩니다.
- ✓ 이를 구현하면 총 시간복잡도 $O(N \log N + Q\sqrt{N} \log N)$에 문제를 해결할 수 있습니다.
- ✓ LCA와 어떤 정점의 k번째 조상을 쿼리 당 $O(1)$에 처리하면 $O(N \log N + Q\sqrt{N})$에 풀 수 있지만 요구되진 않습니다.