2025 SW-IT Contest 해설

Official Solutions

by

ANA

2025년 9월 27일

문제		의도한 난이도	출제자
A	Acquiring SW-IT Corn	**Easy**	안우진awj1052
B	Back to Origin	**Easy**	안우진awj1052
C	Copper Golem and Chests	**Medium**	조서현tjgus1668
D	Designing a Tree	**Hard**	조서현tjgus1668
E	Euler Tour Problem	**Challenging**	조서현tjgus1668
F	Form a Straight!	**Easy**	안우진awj1052
G	Grid Traveler	**Medium**	조서현tjgus1668
H	Hostile Cooperation	**Hard**	김승현gytjdttop
I	Island Cities	**Hard**	조서현tjgus1668

문제		의도한 난이도	출제자
J	Jumpring	**Medium**	황현석 hhs2003
K	Konpaku Youmu	**Challenging**	최준원 kaorin
L	Lunar Exploration	**Medium**	김승현 gytjdttop
M	Moving Formation	**Medium**	안우진 awj1052
N	Nimble Rendezvous	**Hard**	김승현 gytjdttop

A. Acquiring SW-IT Corn

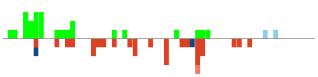
`arithmetic`

출제진 의도 – **Easy**

- 제출 32번, 정답 27팀 (정답률: 84.375%)
- 처음 푼 팀: **bumsoo0515**, 1분
- 출제자: 안우진`awj1052`

- $\frac{X \times U}{100} + \frac{Y \times V}{50} + \frac{Z \times W}{20}$
- 지문을 잘 읽어서 식을 세우고 답을 계산하면 됩니다. U,V,W 는 100 의 배수이므로 정답은 항상 정수입니다.

F. Form a Straight!



`bruteforcing`

출제진 의도 – **Easy**

- 제출 55번, 정답 26팀 (정답률: 47.273%)
- 처음 푼 팀: **AlLerGy**, 3분
- 출제자: 안우진`awj1052`

F. Form a Straight!

- **스트레이트**로 만들 수 있는 경우의 수 5가지에 대해 전부 시도하여 최소 횟수를 구할 수 있습니다.
- 가지고 있는 카드와 목표 스트레이트와의 다른 카드 개수를 세주면 됩니다.

B. Back to Origin

`math, simulation`

출제진 의도 – **Easy**

- 제출 98번, 정답 23팀 (정답률: 23.469%)
- 처음 푼 팀: **이태호**, 2분
- 출제자: 안우진`awj1052`

B. Back to Origin

- 크게 두 가지 풀이가 가능합니다.
- 1. 시뮬레이션 풀이
- $\theta = \frac{\pi d}{180}$ 일때, 매 $k+1$ 번째 이동마다 $(\cos k\theta, \sin k\theta)$ 를 더해주는 과정을 원점으로 돌아올 때까지 반복해주면 됩니다.
- 부동소수점 자료형을 사용하면 시뮬레이션 과정에서 실수 오차가 발생하므로, $|x|, |y| \leq \varepsilon$ (예: 10^{-9}) 이면 원점으로 돌아왔다고 판정하면 됩니다.

B. Back to Origin

- 2. 수학 풀이
- n번 이동 후 좌표는 $\sum_{k=0}^{n-1}(\cos k\theta, \sin k\theta)$ 입니다. 이게 $(0, 0)$이 되려면 $n \times d$가 360의 배수여야 합니다.
- 따라서 최소 횟수는 $\frac{360}{\gcd(360,\ d)}$ 입니다. 입력 범위 $1 \leq d \leq 359$에서는 항상 돌아오므로 -1를 출력할 경우는 없습니다.

L. Lunar Exploration

`sort, greedy`
출제진 의도 – **Medium**

- 제출 1번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 김승현`gytjdttop`

L. Lunar Exploration

- 회수선이 X 축에 평행하게 배치되어 있다면 좌석 배정을 어떻게 하든간에 동일한 크기의 Y 좌표를 움직여야 합니다. 반대도 마찬가지입니다.
- 따라서 시간을 최소화하기 위해서는 회수선이 놓여있는 방향과 반대 좌표축을 기준으로 잘 조작해야 함을 짐작할 수 있습니다.
- 모든 로봇이 회수선에 탑승하는 시간은 로봇 좌표와 좌석 좌표 간 맨해튼 거리의 합과 같습니다.
- 맨해튼 거리의 합을 최소화하기 위해 회수선과 평행한 축을 기준으로 로봇 좌표를 정렬합니다.
- 이후 각 좌석과 순서대로 매칭하면 최소 시간을 구할 수 있습니다.
- 정렬 $O(N \log N)$, 거리 합 계산 $O(N)$ 이므로 전체 시간 복잡도는 $O(N \log N)$ 입니다.

G. Grid Traveler

`constructive, adhoc`
출제진 의도 – **Medium**

- 제출 6번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 조서현`tjgus1668`

- i가 홀수면 i에서 맨헤튼 거리가 i 이하며 홀수인 아무 위치에 $i+1$를 배치해도 됩니다.
- i가 짝수면 i에서 맨헤튼 거리가 i 이하며 짝수인 아무 위치에 $i+1$를 배치해도 됩니다.
- 왜냐하면 방문하지 않은 다른 두 칸을 방문하여, 최단 거리 경로에서 거리를 2씩 늘릴 수 있기 때문입니다.

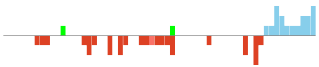
G. Grid Traveler

- 앞서 말한 내용을 쉽게 구현하는 방법 중 하나는 다음과 같습니다.

$$\begin{matrix} 1 & 2 & 4 \\ 6 & 5 & 3 \\ 8 & 7 & 9 \end{matrix} \quad (1)$$

- 격자를 ㄹ 모양(뱀 모양)을 따라서 1차원으로 피면 $[1, 2, 4, 3, 5, 6, 8, 7, 9]$ 입니다.
- N^2 길이의 1차원 수열에 수를 채워 넣고 출력할 때만 2차원으로 바꿔서 출력합니다.
- i가 홀수일 때는 $i+1$을 i 바로 왼쪽 혹은 오른쪽에 배치하고, i가 짝수일 때는 $i+1$을 두 칸 오른쪽에 배치하면 됩니다. 총 시간 복잡도는 $O(N^2)$ 입니다.
- 항상 가능한 방법이 존재하므로 NO를 출력할 일은 없습니다.

M. Moving Formation



`implementation`
출제진 의도 – **Medium**

- 제출 34번, 정답 2팀 (정답률: 5.882%)
- 처음 푼 팀: **bumsoo0515**, 34분
- 출제자: 안우진`awj1052`

- 여러 가지 풀이가 있지만 대표적인 2가지를 소개합니다.
- 1. 드론 사이의 거리 **최소화**
- 처음 A를 기준으로 드론이 1×1 사각형 모양이 되도록 합니다.
- 각 드론들을 한 칸씩 움직이며 A를 목표 좌표로 옮깁니다.
- 마지막으로, 드론이 $N \times N$ 사각형 모양이 되도록 합니다.

- 2. 드론 사이의 거리 **최대화**
- 최초 상태에서 각 드론쌍의 거리가 줄어들도록 움직일 수 밖에 없습니다.
- 각 드론 사이의 거리를 줄였다가 늘리는 방식을 반복할 수 있습니다.
- 이동 횟수를 최소화할 필요는 없으므로 한 칸씩 움직여 구현을 최소화할 수 있습니다.
- 각 이동에서 각 드론쌍의 거리가 N 을 넘지 않는 것은 수학적으로 증명할 수 있습니다.

J. Jumpring

dp

출제진 의도 – **Medium**

- 제출 48번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 황현석 hhs2003

J. Jumpring

- 문자열 S에서 인접하지 않은 문자를 제거하여 U를 만들 수 있는지 판별하는 문제
- 핵심 아이디어: DP 정의 및 탐색

- S에서 그리디하게 문자열을 선택하거나 고를 수 있는 방법이 없습니다.
- 따라서 dp상태 배열을 정의합니다.

$$dp[i][j] = \begin{cases} 1 & S\text{의 } i\text{번째 까지 고려하여, } U\text{의 } j\text{번째까지의 문자열을 만들 수 있으면} \\ 0 & \text{그렇지 않으면} \end{cases}$$

- 문자열 S의 처음 i글자와 U의 처음 j글자를 사용했을 때
- 도달 가능한 상태를 2차원 배열로 관리합니다.

J. Jumpring

- $dp[0][0] = dp[1][0] = 1$ 입니다.
- $S[i] = U[j]$ 인 경우들에 한에 상태들을 다음으로 전이 할 수 있습니다.

$$dp[i+1][j+1] \leftarrow dp[i][j]$$

이때, S 의 i 번째 문자열을 지우지 않고 사용했으므로, S 의 다음 문자열은 임의로 지울 수 있습니다.

- 따라서, 아래의 경우도 전이 시켜 주어야 합니다.

$$dp[i+2][j+1] \leftarrow dp[i][j]$$

- 시간 복잡도: $O(NM)$
- 최종 상태 $dp[N][M]$ 이 0이 아니면 YES, 아니면 NO

C. Copper Golem and Chests

`permutation_cycle_decomposition`, `binary_search`, `tree_set`, `greedy`

출제진 의도 – **Medium**

- 제출 23번, 정답 3팀 (정답률: 13.043%)
- 처음 푼 팀: **좋아너가좋아할만한자연스럽고예의는갖추면서도너무딱딱하지않게팀명을정해봤어어떤종류의대회인지알려주**, 42분
- 출제자: 조서현 `tjgus1668`

C. Copper Golem and Chests

- B 가 순열로 주어진다는 것은, 구리 골렘이 같은 아이템에 대해 작업을 반복하게 되면 처음 상자로 돌아오게 된다는 것을 의미합니다. 즉, 서로 이동할 수 있는 상자끼리 사이클을 형성합니다.
- 우리는 한 원소가 속해있는 사이클 안에서 원하는 원소를 선택할 수 있습니다.

C. Copper Golem and Chests

- 아이템을 잘 정리한다는 것은, A를 비내림차순으로 정렬한다는 것과 동치입니다.
- A_1은 A_1이 속한 사이클 내에서 가장 작은 값을 선택하는 것이 최선입니다.
- A_i는 A_i가 속한 사이클 내에서 A_{i-1}보다 크거나 같은 가장 작은 값을 선택하는 것이 최선입니다.
- 각 사이클을 정렬해서 이분탐색을 수행하거나 트리 집합을 사용하면 이 작업을 로그 시간에 수행할 수 있습니다.
- B를 사이클끼리 모으는데 $O(M)$, 각 사이클을 정렬하는데 총 $O(M \log M)$, 각 A에 대해 이분탐색을 수행하는데 $O(N \log M)$이므로 총 $O((N+M) \log M)$에 해결할 수 있습니다.

H. Hostile Cooperation

`two_pointers`, `game_theory`, `sort`

출제진 의도 – **Hard**

- 제출 12번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 김승현`gytjdttop`

H. HostileCooperation

- 우진이는 점수를 최대화하기 위해 항상 자신의 카드 중 가장 작은 값 m 과 가장 큰 값 M 을 마지막까지 남기는 것이 최선의 선택입니다.
- 마지막 선택권은 우진이에게 있으므로, 그는 m 과 M 중 K 와 더 멀어지는 쪽을 선택합니다.
- 따라서 영우와 민우의 최적 전략은 두 사람이 고른 카드 합을 $K - \frac{m+M}{2}$ 에 최대한 가깝게 만드는 것입니다.
- 영우와 민우의 카드 배열을 각각 정렬한 뒤, 투 포인터로 두 배열 합이 $K - \frac{m+M}{2}$ 에 가장 가까운 값을 찾습니다.
- 정렬 $O(N \log N)$, 탐색 $O(N)$ 이므로 전체 시간 복잡도는 $O(N \log N)$ 입니다.

N. Nimble Rendezvous

`math`, `mitm`

출제진 의도 – **Hard**

- 제출 12번, 정답 3팀 (정답률: 25%)
- 처음 푼 팀: **좋아너가좋아할만한자연스럽고예의는갖추면서도너무딱딱하지않게팀명을정해봤어어떤종류의대회인지알려주**, 67분
- 출제자: 김승현`gytjdttop`

- 매 i 번째 이동마다 상대거리 기준으로 같은 방향으로 이동한다면 변화는 0, 반대 방향이면 $\pm 2^i$ 의 변화가 일어납니다. 변화는 항상 짝수입니다.
- 따라서 $|A-B|$ 가 홀수면 두 점은 만날 수 없습니다.
- 최소 이동횟수는 상대 거리를 이진수로 표현했을 때 최상위 1비트의 위치로 알 수 있습니다.

N. Nimble Rendezvous

- 최소 이동횟수를 K 라 할 때, K 번 이동 후 두 점이 만날려면 다음과 같아야 됩니다.

$$A + \sum_{i=1}^{K} (\pm 2^{i-1}) \;=\; B + \sum_{i=1}^{K} (\pm 2^{i-1})$$

- $\sum_{i=1}^{K} (\pm 2^{i-1})$ 의 값들의 집합을 S_K 라고 하겠습니다.

$$S_K = \left\{ \sum_{i=1}^{K} \sigma_i 2^{i-1} : \sigma_i \in \{\pm 1\} \right\}.$$

- S_K 는 $-(2^K - 1)$ 부터 $(2^K - 1)$ 까지 모든 홀수가 한 번씩 존재합니다. 공차 2 인 길이 2^K 등차수열로 볼 수 있습니다.

- 이전 식에서 B를 이항하여 정리하면 두 점이 K 번 이동 후 만나는 위치 집합과 같습니다.

$$S_K \cap \left(S_K + (A - B)\right)$$

- S_K 를 $(A - B)$ 만큼 평행 이동한 교집합의 원소 개수가 되며, 이를 구하면 $2^K - \frac{|A - B|}{2}$ 가 나옵니다.
- 시간복잡도는 $O(\log |A - B|)$ 입니다. 이외에도 MITM 등의 $O(\sqrt{|A - B|})$ 풀이가 존재합니다.

I. Island Cities

`mst`, `parametric_search`, `binary_search`
출제진 의도 – **Hard**

- 제출 0번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 조서현 `tjgus1668`

- 다리 강화가 없다면, X 의 최댓값은 최대 스패닝 트리(Maximum Spanning Tree)에 사용되는 간선의 가중치 중 최솟값과 동치입니다.
- 문제를 바꿔서 간선의 가중치를 다리의 하중을 X 이상으로 만들 때 필요한 예산으로 생각해 봅시다.
- 원래 간선의 가중치가 이미 X 이상이라면 강화할 필요가 없으므로 필요한 예산은 0 이고, 미만이라면 $\lceil \frac{X - w_i}{x_i} \rceil$ 번의 추가 강화가 필요하므로 $\lceil \frac{X - w_i}{x_i} \rceil \times y_i$ 만큼의 예산이 필요합니다.
- 필요한 예산을 간선의 가중치로 하는 그래프에서 최소 스패닝 트리(Minimum Spanning Tree)를 구하고 총 예산 B 와 비교하여 X 의 **가능 여부**를 확인할 수 있습니다.

- X 의 가능 여부는 단조성을 띕니다. 즉 $X = a$ 일 때 가능하다면 $X = b\ (b \leq a)$ 일 때도 가능합니다.
- 따라서 매개변수 X 에 대해 이분 탐색을 사용하여 $X = a$ 일때는 가능하지만 $X = a + 1$ 일 때는 불가능한 a 를 찾아주면 되고 그게 바로 X 의 최댓값입니다.
- 다리 강화 횟수는 MST 의 간선으로 사용됐는데 원래 가중치가 X 미만이라면 $\lceil \frac{X - w_i}{x_i} \rceil$ 이고 나머지 경우는 0 으로 설정합니다.
- MST 를 구하는데 $O(M \log M)$, 가능한 X 의 범위는 $1 \leq X \leq 10^{6+5} + 10^5$ 이므로 총 시간 복잡도는 $O(M \log M \log 10^{6+5} + 10^5)$ 입니다.

D. Designing a Tree

`trees`, `bfs`, `segment_tree`

출제진 의도 – **Hard**

- 제출 8번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 조서현 `tjgus1668`

D. Designing a Tree

- 문제를 반대로 생각해 봅시다.
- N 번 정점은 유일하게 j 를 선택할 수 없는 정점입니다. 가능한 다른 모든 정점을 N 번 정점이 속한 컴포넌트 C 에 연결되도록 해봅시다.
- 즉, 어떤 정점 i 가 $L_i \leq N \leq R_i$ 이라면 i 번 정점은 N 번 정점을 선택해야 합니다.
- C 에 속한 N 이 아닌 다른 정점 V 가 있을 때, j 를 선택하지 않은 다른 정점 i 가 마찬가지로 $L_i \leq V \leq R_i$ 라면 i 번 정점은 V 번 정점을 선택해야 C 에 연결되도록 할 수 있습니다.
- 이 과정에서 사이클은 발생하지 않기 때문에 C 는 트리 구조를 유지합니다.
- 최종적으로 $|C| = N$ 이라면 트리로 만들 수 있고, 아니라면 트리로 만들 수 없습니다.

D. Designing a Tree

- N 번 정점에서부터 시작하는 BFS를 수행한다고 생각해봅시다. u에서 이동할 수 있는 정점은 $L_v \leq u \leq R_v$ 인 v 입니다.
- 각 정점 별로 조건을 만족하는 정점을 저장하는 리스트를 배정하면 되지만, 최악의 경우 공간 복잡도와 시간 복잡도가 $O(N^2)$ 이 되므로 효율적인 방법을 생각해야 합니다.

D. Designing a Tree

- 세그먼트 트리를 사용합니다. 세그먼트 트리의 각 노드에는 해당 노드가 관리하는 범위를 $[l, r]$ 이라 할때, $L_i \leq l \leq r \leq R_i$ 인 모든 i 를 리스트에 저장하고 있습니다. 각 i 가 중복으로 저장되는 횟수는 $O(\log N)$ 번으로 제한되므로 메모리 문제가 해결됩니다.
- u 에서 이동할 수 있는 정점을 찾는다고 한다면 세그먼트 트리의 루트에서 리프로 이동하면서 만나는 모든 노드에 저장된 정보를 추가하면 됩니다.
- 방문한 모든 정점은 이후 C 에 연결되므로 다시 방문할 필요가 없을뿐더러, 시간 복잡도가 깨져버리므로, 방문 이후에는 노드를 초기화해야 합니다.
- 전체 과정에서 세그먼트 트리를 통해 얻어지는 정보의 크기는 $O(N \log N)$ 이므로 전체 $O(N \log N)$ 에 해결할 수 있습니다.

E. Euler Tour Problem

`trees`, `string`, `euler_tour_technique`, `dfs`, `bruteforcing`

출제진 의도 – **Challenging**

- 제출 8번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 조서현 `tjgus1668`

- 어떤 정점의 자식 정점 방문 순서를 재배열해서 사전순으로 가장 앞서는 문자열을 만들려면, 자식 정점을 방문할 때 만들어지는 **서브 트리 문자열**끼리 정렬하면 됩니다.
- 따라서 모든 정점에 대해 각각 정렬해보고 가장 앞서는 문자열을 출력하면 됩니다.
- 정렬할 때는, 두 **서브 트리 문자열** A, B가 주어지면 $A + B < B + A$인지 확인하면 됩니다. $A < B$가 아닌 이유는 한 **서브 트리 문자열**이 다른 **서브 트리 문자열**의 접두사가 되면 $A < B$라고 $A + B$가 $B + A$보다 사전순으로 더 앞서는 문자열이라는 보장이 없기 때문입니다.

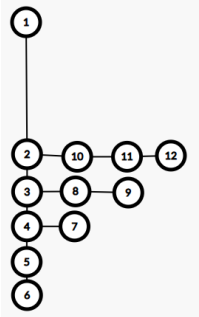
- 하지만 **서브 트리 문자열**이 만들어지는 과정을 생각해보면 절대로 접두사 관계에 있는 두 **서브 트리 문자열**은 존재할 수 없습니다.
- 모든 **서브 트리 문자열**은 `1`로 시작하고 `0`으로 끝나는데, `1`을 만나면 +1, `0`을 만나면 -1를 더한다고 생각해 봅시다.
- 한 **서브 트리 문자열**이 다른 **서브 트리 문자열**의 접두사가 되려면 끝이 아닌 어딘가에 지금까지의 합이 0이 되는 위치가 존재해야 합니다. 오일러 투어를 통해 생성되는 S의 특성상 그런 경우는 없으므로 이 문제에서는 $A+B<B+A$를 $A<B$로 조건을 완화할 수 있습니다.

E. Euler Tour Problem

- DFS를 수행하여 S를 생성합니다. 어떤 정점 u가 전체 S에서 차지하는 부분 문자열의 인덱스 in[u], out[u]를 저장합니다.
- 각 정점의 자식 정점을 정점 번호 순으로 방문했을 때(S)와 사전 순으로 정렬 후 방문했을 때(S')를 비교하면, 처음 바뀌는 위치 pos[u]를 구할 수 있습니다.
- pos가 가장 작은 정점을 정렬하는게 최적이므로 pos가 유일한 경우 해당 정점을 정렬해서 S'를 출력하면 됩니다.

E. Euler Tour Problem

- pos가 같으면 어떻게 해야할까요? 복잡한 문자열 자료구조가 필요할 것 같지만, 해답은 pos가 같은 경우가 최대 몇 번일까에서 찾을 수 있습니다.



- 한 정점 u와 그 정점의 부모 정점 p의 pos가 같다는 것은, p의 u가 아닌 다른 자식 중 u와 pos가 같은 정점 v가 존재한다는 뜻인데 루트로 향할 수록 v 서브트리의 깊이는 점점 길어져야 합니다. 따라서 pos가 같은 경우는 **많지 않습니다.**

E. Euler Tour Problem

- 체인의 길이 K 가 주어질 때 정점 개수는 $O(K^2)$ 으로 표현됩니다. 즉, 정점 개수가 N 일 때, 체인의 최대 길이는 $O(\sqrt{N})$ 입니다.
- pos가 같은 경우들을 따로 모아주고 최대 $O(\sqrt{N})$ 개의 정점에 대해 문자열을 나이브하게 비교해주는 것만으로도 정답을 시간 내에 구할 수 있습니다.
- 따라서 $O(N\sqrt{N})$ 에 해결할 수 있습니다. 부분 문자열 비교에 사용되는 자료구조는 Rolling Hash 정도면 충분합니다. 사실 문자열 비교 함수를 잘 구현하면 별도의 자료구조 없이도 구현할 수 있습니다.
- 아까 언급했던 것처럼, Suffix Array 등의 문자열 자료구조를 빡세게 잘 구현하면 $O(N \log N)$ 까지 줄일 수는 있습니다. 다만, 상수가 크고, $O(N\sqrt{N})$ 풀이의 상수가 꽤 작기 때문에 보통은 더 느리게 동작합니다.

K. Konpaku Youmu

`trees`, `centroid_decomposition`, `merge_sort_tree`

출제진 의도 – **Challenging**

- 제출 0번, 정답 0팀 (정답률: 0%)
- 처음 푼 팀: **?**, ?분
- 출제자: 최준원 `kaorin`

- 주어진 수식의 시간 복잡도는 $O(N^2)$ 이므로 다른 방법으로 계산합니다. 어떤 간선 $e = (u, v, w)$ 가 주어진 수식에 기여하는 횟수를 안다면 문제의 정답을 구할 수 있습니다.
- 트리에서 e 를 제거했을 때 u 쪽 컴포넌트를 A, v 쪽 컴포넌트를 B 라고 하겠습니다.
- A 에 있는 어떤 정점 a 와 B 에 있는 어떤 정점 b 에 대해서, $dist_K(a, b)$ 가 e 를 사용한다는 것은 $dist(a, u) + w \leq K$ 임을 의미합니다. 반대로 $dist_K(b, a)$ 가 e 를 사용한다는 것은 $dist(b, v) + w \leq K$ 임을 의미합니다.

- 따라서 각 간선에 대해, ($w \times (|A| \times$ B의 정점 중 v로부터의 거리가 K-w 이하인 정점의 개수) + ($|B| \times$ A의 정점 중 u로부터의 거리가 K-w 이하인 정점의 개수))를 누적하면 됩니다.
- 센트로이드 분할과 정렬/이분 탐색을 사용하면, 특정 정점에서 거리가 일정 이하인 정점의 개수를 $O(\log^2 N)$에 구할 수 있습니다.
- 트리에서 루트를 잡고 오일러 투어 방문 순서를 이용하여 머지 소트 트리를 구성하면, 특정 정점을 루트로 하는 서브 트리에서 거리가 일정 이하인 정점의 개수를 $O(\log^2 N)$에 구할 수 있습니다.
- 이 두 자료구조를 적절히 이용하여 위 수식을 계산하면 됩니다.

- 루트를 포함하는 컴포넌트를 A, 포함하지 않는 컴포넌트를 B라고 할 때, 센트로이드 분할을 이용해서 u에서 거리가 $K-w$ 이하인 정점의 개수를 구하고, 거기에 머지 소트 트리를 이용해서 v를 루트로 하는 서브 트리에서 거리가 $K-2w$ 이하인 정점의 개수를 빼면 A 컴포넌트의 정점 중 u로부터의 거리가 $K-w$ 이하인 정점의 개수를 구할 수 있습니다.
- B의 경우는 머지 소트 트리만 사용하면 구할 수 있습니다.
- 따라서 총 $O(N \log^2 N)$에 해결할 수 있습니다.