2024 고려대학교 프로그래밍 경시대회 KCPC

공식 해설

Host and Sponsors



HYUNDAI MOBIS

STARTLINK

출제자

✓	**김동우**	kidw0124	고려대학교 사이버국방학과 20
✓	**김재우**	eoaud0108	고려대학교 사이버국방학과 20
✓	**박진한**	jinhan814	고려대학교 기계공학부 20
✓	**이종우**	yijw0930	고려대학교 사이버국방학과 21
✓	**박세준**	devluyten	고려대학교 수학과 22
✓	**위재현**	finimage	고려대학교 데이터과학과 23

검수자

- amel
- edwardblue
- kevinlys00
- lighter
- rachel2003
- rootsquare
- tlsdydaud1
- tmdghks
- w8385
- wnguscjf01

번호	문제	의도한 난이도	출제자
PA	사랑은 고려대입니다	**Easy**	위재현
PB	KCPC에 등장할 알고리즘 맞히기	**Easy**	이종우
PC	차이를 M이하로	**Challenging**	김동우, 김재우

번호	문제	의도한 난이도	출제자
1A2A	$\frac{1}{2}$(MatKor+ALPS)=AlKor	**Easy**	김동우, 위재현
2B	차이를 M이상으로	**Easy**	김동우, 김재우
2C	충전하기	**Medium**	김동우
2D	4-cycle (Easy)	**Medium**	박진한
1B2E	현대모비스와 함께하는 편안한 주행	**Medium**	박세준
2F	트리 고치기	**Hard**	위재현
1C2G	과녁 맞히기	**Hard**	김동우, 위재현
1D2H	격자 이동하기	**Expert**	김동우
1E	얼음마법사 동우의 대회 검수	**Expert**	이종우
1F	차이를 M으로	**Challenging**	김동우, 이종우
1G	4-cycle (Hard)	**Challenging**	박진한
1H	히스토그램 직사각형으로 쪼개기	**Challenging**	김동우, 김재우

PA. 사랑은 고려대입니다

`implementation`
출제진 의도 – **Easy**

- ✓ 본 대회 (Div.2 + Div.1)
  - First Solve: **정성훈**, 0분 27초
  - **맞았습니다!!** 비율: **66.667%** (14/21)
- ✓ Open Contest
  - First Solve: **dabbler1**, 0분 25초
  - **맞았습니다!!** 비율: **84.507%** (231/284)
- ✓ 출제자: 위재현
- ✓ 에디토리얼 작성자: 위재현

- ✓ 단순 반복문 구현으로 해결할 수 있다.

PB. KCPC에 등장할 알고리즘 맞히기

`string`, `tree_set`, `implementation`
출제진 의도 – **Easy**

- ✓ 본 대회 (Div.2 + Div.1)
  - First Solve: **조진익**, 9분 20초
  - **맞았습니다!!** 비율: **41.176%** (7/17)
- ✓ Open Contest
  - First Solve: **dgsw1306**, 5분 14초
  - **맞았습니다!!** 비율: **59.286%** (80/140)
- ✓ 출제자: 이종우
- ✓ 에디토리얼 작성자: 이종우

PB. KCPC에 등장할 알고리즘 맞히기

- ✓ C++ STL을 사용하여 쉽게 풀 수 있는 문제이다.
- ✓ `std::string`을 이용하여 문자열을 저장한다.
- ✓ `std::map`을 이용하여 각 태그의 등장 횟수를 구한다.
- ✓ 이러한 자료구조가 기본적으로 제공되지 않는 언어를 사용할 경우 정렬을 이용하는 방법도 있다.

PC. 차이를 M 이하로

`dp`, `sorting`, `sweeping`, `segment_tree`
출제진 의도 – **Challenging**

- ✓ 본 대회 (Div.2 + Div.1)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/3)
- ✓ Open Contest
  - First Solve: **swoon**, 190분 12초
  - **맞았습니다!!** 비율: **11.111%** (1/9)
- ✓ 출제자: 김동우, 김재우
- ✓ 에디토리얼 작성자: 김재우

PC. 차이를 M 이하로

- DP를 이용한 풀이를 생각해보자.
- $DP[i]$를 i번째 원소에 대해 시행을 하지 않을 때, i번째 원소까지 조건을 만족하도록 하는 시행횟수의 최솟값으로 정의하자.
- $j < i$인 j에 대해 j와 i 사이의 수들 모두에 대해 시행을 적용하여 조건을 만족하도록 만들 수 있다면 $DP[i] = DP[j] + (i - j + 1)$으로 생각할 수 있다.
- 만약, 위와 같은 조건을 만족하는 j가 없다면 i번째 원소 앞에 있는 모든 원소들에 대해 시행을 적용해야 하므로 $DP[i] = i - 1$이다.
- 따라서, $DP[i] = i - 1$로 초기화해두고 j를 순회하면서 $|A_i - A_j| \leq (i - j)M$을 만족할 때마다 $DP[i] = \min(DP[i], DP[j] + (i - j + 1))$로 갱신해주면 된다.
- 이를 반복문을 이용해 구현하면 $\mathcal{O}\left(N^2\right)$의 풀이를 얻을 수 있다.

PC. 차이를 M 이하로

- 이를 더 최적화하기 위해 $DP[i]$의 값이 갱신되기 위한 i, j의 조건을 살펴보면 다음과 같다.
  - $j < i$
  - $(j-i)M \leq A_i - A_j \leq (i-j)M$
- 두 번째 조건을 만족하면 $j \leq i$는 자연스럽게 만족하게 되기 때문에, $i \neq j$만 만족하도록 처리를 해주면 첫 번째 조건은 신경쓰지 않아도 된다.
- 두 번째 조건을 조금 더 정리해보면 $A_j - jM \leq A_i - iM$, $A_i + iM \leq A_j + jM$일 때 j에 대한 정보를 바탕으로 i에 대한 정보를 갱신할 수 있음을 알 수 있다.

PC. 차이를 M 이하로

- ✓ 그런데, 우리는 최대한 많은 값들을 시행하지 않고 그대로 놔두고자 하므로 증가하는 순서대로 고를 때 가장 많이 골라주어야 한다.
- ✓ 이는 $A_i + iM$ 기준으로 내림차순으로 정렬한 후 $A_i - iM$ 의 값을 기준으로 Longest Increasing Subsequence를 구하면 되므로 $\mathcal{O}(N \lg N)$ 의 시간복잡도로 해결할 수 있다.
- ✓ 혹은 Dynamic Segment Tree 또는 Persistent Segment Tree를 이용해 비슷한 방법으로 위의 DP를 최적화해도 $\mathcal{O}(N \lg N)$ 의 시간복잡도로 문제를 해결할 수 있으나, 상수가 더 크고 구현량이 조금 더 많다.

1A2A. $\frac{1}{2}$(MatKor+ALPS)=AlKor

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

- 본 대회 (Div.2 + Div.1)
  - First Solve: Div.2 - **이민재**, 4분 4초 / Div.1 - **하준서**, 4분 14초
  - **맞았습니다!!** 비율: Div.2 - **32.558%** (14/43) / Div.1 - **38.462%** (5/13)
- Open Contest
  - First Solve: **dabbler1**, 4분 22초
  - **맞았습니다!!** 비율: **48.780%** (79/164)
- 출제자: 김동우, 위재현
- 에디토리얼 작성자: 위재현

1A2A. $\frac{1}{2}$(MatKor+ALPS)=AlKor

- N 이 홀수일 때와 짝수일 때로 나누어서 문제를 해결할 수 있다.
- N 이 홀수일 때, $N^2, N, 0$ 을 출력한 후, 제한된 범위 안에 들어갈 a 와 $-a$ 를 겹치지 않도록 반복하여 출력하면 된다. $A_{N+1} = N + 1$ 이면 조건을 만족한다.
- N 이 짝수일 때, 범위 안에 들어갈 a 와 $-a$ 를 겹치지 않도록 반복하여 출력하면 된다. $A_{N+1} = 0$ 이면 조건을 만족한다.
- 그 외에도 다양한 방법들이 존재한다.

2B. 차이를 M 이상으로

`dp, greedy`
출제진 의도 – **Easy**

- ✓ 본 대회 (Div.2)
  - First Solve: **조진익**, 27분 15초
  - **맞았습니다!!** 비율: **36.842%** (6/19)
- ✓ Open Contest
  - First Solve: **dabbler1**, 8분 5초
  - **맞았습니다!!** 비율: **64.516%** (39/62)
- ✓ 출제자: 김동우, 김재우
- ✓ 에디토리얼 작성자: 김재우

2B. 차이를 M 이상으로

- ✓ 이웃한 두 수의 차이가 M 이상이 되도록 만들어야 하기 때문에, 어떤 한 원소를 골라 시행을 하게 되면 그 원소를 무한히 큰 수로 만드는 것이 이득이다.
- ✓ 따라서 $DP[i][0]$ 을 i 번째 원소에 대해 시행을 하지 않을 때 시행횟수의 최솟값, $DP[i][1]$ 을 i 번째 원소에 대해 시행을 할 때 시행횟수의 최솟값으로 정의하여 DP를 적용할 수 있다.
  - $|A_i - A_{i-1}| \geq M$ 인 경우
    - $DP[i][0] = \min(DP[i-1][0], DP[i-1][1])$
    - $DP[i][1] = \min(DP[i-1][0], DP[i-1][1]) + 1$
  - $|A_i - A_{i-1}| < M$ 인 경우
    - $DP[i][0] = DP[i-1][1]$
    - $DP[i][1] = \min(DP[i-1][0], DP[i-1][1]) + 1$

2B. 차이를 M 이상으로

- ✓ DP를 사용하지 않고도 앞에서부터 차례대로 이전 항과의 차이를 계산하여 그 값이 M 보다 작을 때마다 시행을 적용하는 Greedy 풀이도 가능하고, 시간복잡도는 두 풀이 모두 $\mathcal{O}(N)$ 이다.
- ✓ 주어지는 수의 범위가 10^{12} 까지 가능하므로 int 대신 long long 등의 자료형을 써야 한다.

2C. 충전하기

`tree_set`
출제진 의도 – **Medium**

- ✓ 본 대회 (Div.2)
  - First Solve: **정성훈**, 52분 26초
  - **맞았습니다!!** 비율: **4.110%** (3/73)
- ✓ Open Contest
  - First Solve: **menborong**, 17분 20초
  - **맞았습니다!!** 비율: **31.325%** (26/83)
- ✓ 출제자: 김동우
- ✓ 에디토리얼 작성자: 김동우

2C. 충전하기

- ✓ 1번 쿼리가 들어오게 되면 다음을 처리해야 한다.
  - 현재 남아있는 포트 중 i 이상인 최소의 포트를 찾아야 한다.
  - 그러한 포트가 존재하면 꽂고, 해당 포트가 자금 꽂은것을 저장해야 한다.
- ✓ 2번 쿼리가 들어오게 되면 다음을 처리해야 한다.
  - 해당 포트에 꽂혀있는 지 확인해야 한다.
  - 꽂혀있다면 언제 꽂았는지 알아내고, 뽑는다.

2C. 충전하기

- ✓ std::set과 배열을 통해 처리할 수 있다.
- ✓ 꽂혀있지 않은 포트를 set으로 만들고, 포트별로 언제 꽂았는지 배열에 저장한다
- ✓ 1번 쿼리가 들어오면 set.lower_bound 등을 통해 찾을 수 있다.
- ✓ 포트에 꽂고 뽑는 것은 set에서 erase, insert를 해준다.
- ✓ 꽂을 때마다 배열에 몇번째인지 업데이트 해준다.
- ✓ 각 쿼리별로 $\mathcal{O}(\lg N)$, 총 $\mathcal{O}(Q \lg N)$에 처리 가능하다.

2D. 4-cycle (Easy)

graph
출제진 의도 – **Medium**

- ✓ 본 대회 (Div.2)
  - First Solve: **조하성**, 137분 26초
  - **맞았습니다!!** 비율: **100.000%** (2/2)
- ✓ Open Contest
  - First Solve: **menborong**, 22분 5초
  - **맞았습니다!!** 비율: **85.714%** (13/21)
- ✓ 출제자: 박진한
- ✓ 에디토리얼 작성자: 박진한

2D. 4-cycle (Easy)

- ✓ $i = 1, \cdots, N$ 을 순회하며 다음의 과정을 반복한다.
  - c_j 를 i 에서 j 로 가는 길이가 2 인 path 의 개수로 정의하자.
  - i 를 포함하는 길이가 4 인 사이클의 개수는 $j \neq i$ 인 j 에 대한 $\frac{c_j(c_j-1)}{2}$ 의 합이다.
  - 이를 $\mathcal{O}(N+M)$ 에 계산해 정답에 더해준다.
- ✓ 위의 과정에서 길이가 4 인 사이클은 각각 4 번씩 counting 되니, 구한 정답을 4 로 나눠서 출력해주면 된다.
- ✓ 풀이의 시간복잡도는 $\mathcal{O}(N(N+M))$ 이다.

1B2E. 현대모비스와 함께하는 편안한 주행

`geometry`, `dijkstra`
출제진 의도 – **Medium**

- ✓ 본 대회 (Div.2 + Div.1)
  - First Solve: Div.2 - **정성훈**, 140분 00초 / Div.1 - **하준서**, 43분 9초
  - **맞았습니다!!** 비율: Div.2 - **16.667%** (1/6) / Div.1 - **14.286%** (3/21)
- ✓ Open Contest
  - First Solve: **menborong**, 29분 31초
  - **맞았습니다!!** 비율: **61.111%** (11/55)
- ✓ 출제자: 박세준
- ✓ 에디토리얼 작성자: 박세준

1B2E. 현대모비스와 함께하는 편안한 주행

- ✓ 현대모비스 간판을 중심으로 하는 반경 1의 원 안에서는 스트레스 없이 움직일 수 있다.
- ✓ 아래의 세 경우만을 고려하면 된다.
  - 한 구역에서 다른 구역으로 이동할 때, $\max(0, distance - 2)$ 비용 이하로 이동할 수 있다.
  - 출발점에서 구역/구역에서 도착점으로 이동할 때, $\max(0, distance - 1)$ 비용 이하로 이동할 수 있다.
  - 출발점에서 도착점으로 이동할 때, $distance$ 비용 이하로 이동할 수 있다.
- ✓ 각 비용을 간선, 출발/도착점과 간판의 위치를 정점으로 하는 그래프를 만들어 최단 거리 경로를 찾자.
- ✓ 다익스트라 알고리즘으로 $\mathcal{O}\left((V+E)\log V\right) \leq \mathcal{O}\left(N^2 \log N\right)$에 해결 가능하다.
- ✓ 실수 오차를 고려하여 구현해야 한다. double 자료형으로 해결할 수 있다.

2F. 트리 고치기

`dp, tree`
출제진 의도 – **Hard**

- ✓ 본 대회 (Div.2)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/4)
- ✓ Open Contest
  - First Solve: **menborong**, 39분 21초
  - **맞았습니다!!** 비율: **50.000%** (5/10)
- ✓ 출제자: 위재현
- ✓ 에디토리얼 작성자: 위재현

2F. 트리 고치기

- ✓ 어떤 노드를 루트로 하는 서브 트리에서 0에서 K개의 노드를 고쳤을 때의 작동하는 최대 노드 수를 각각 저장하는 트리 DP로 해결할 수 있다.
- ✓ 어떤 노드 u의 자식 노드를 v_i라고 하면, v_i를 루트로 하는 서브 트리에서 $t(0 \leq t \leq K)$개의 노드를 고쳤을 때의 작동하는 최대 노드 수를 $DP[v_i][t]$라고 하자.
- ✓ u의 DP값은 v_i들의 DP값들을 이용해 구할 수 있다.

2F. 트리 고치기

- u를 v_k의 DP값을 고려해 갱신한다고 하자.
- 이때, 아직 갱신이 완전히 된 u가 아니므로 이는 갱신 중인 u_{tmp}라고 하자.
- $DP[u_{tmp}][t] = \max_{i+j=t}(DP[v_k][i] + DP[u_{tmp}][j])$가 성립한다.
- 이때 값을 잘못 갱신하게 되면 v_i에서 t_1개를 고친 것과 t_2개를 고친 것을 중복해서 더하는 실수가 발생할 수 있다.
- 모든 자식 노드에 대해서 u_{tmp}의 DP값을 갱신한 후 다음을 생각해보자.
- 만약 u가 고장 난 노드라면 이 노드는 무조건 고쳐야 하므로 갱신 후,
  $DP[u][i] = DP[u_{tmp}][i-1](1 \le i \le K)$처럼 한 칸씩 밀고 $DP[u][0] = 0$으로 구할 수 있다.
- 만약 u가 고장 나지 않은 노드라면 모든 $DP[u][i] = DP[u_{tmp}][i] + 1(0 \le i \le K)$씩 더해주면 된다.
- 이는 $\mathcal{O}\left(NK^2\right)$에 해결할 수 있다.

1C2G. 과녁 맞히기

`math`, `combinatorics`, `modular_multiplicative_inverse`
출제진 의도 – **Hard**

- 본 대회 (Div.2 + Div.1)
  - First Solve: Div.2 - **이민재**, 144분 32초 / Div.1 - **임동현**, 32분 47초
  - **맞았습니다!!** 비율: Div.2 - **7.143%** (2/28) / Div.1 - **25.000%** (3/12)
- Open Contest
  - First Solve: **menborong**, 44분 11초
  - **맞았습니다!!** 비율: **77.778%** (7/9)
- 출제자: 김동우, 위재현
- 에디토리얼 작성자: 위재현

1C2G. 과녁 맞히기

- 과녁을 쏠 때, 줄을 고르는 경우의 수는 $i(1 \le i \le M)$ 가 A_i 개씩 총 N 개를 나열하는 서로 다른 경우의 수, $\binom{N}{A_1, \cdots, A_M}$ 와 같다.
- 과녁이 한 개 이상 존재하는 i 번 줄을 골랐을 때, 위 혹은 아래의 과녁을 고르는 경우의 수는 $2^{A_i - 1}$ 이다.
- 이를 모두 곱하면 2^{N-M} 이 된다.
- 따라서 $\binom{N}{A_1, \cdots, A_M} 2^{N-M}$ 를 $10^9 + 7$ 로 나눈 나머지를 구하면 된다.
- 이는 팩토리얼의 모듈로 역원을 미리 구해놓고 하나씩 곱하는 방법으로 해결할 수 있다.

1D2H. 격자 이동하기

`dag`, `dp`

출제진 의도 – **Expert**

- 본 대회 (Div.2 + Div.1)
  - First Solve: Div.2 - **없음** / Div.1 - **임동현**, 136분 41초
  - **맞았습니다!!** 비율: Div.2 - **0.000%** (0/1) / Div.1 - **25.000%** (2/8)
- Open Contest
  - First Solve: **menborong**, 64분 20초
  - **맞았습니다!!** 비율: **26.316%** (5/19)
- 출제자: 김동우
- 에디토리얼 작성자: 김동우

1D2H. 격자 이동하기

- ✓ 대각선 길을 가장 많이 지나는 경로의 개수를 세야 한다.
- ✓ 대각선 길은 DAG이므로, $\mathcal{O}\left(N^2\right)$에 그래프를 그린다.
- ✓ DAG의 간선의 가중치는 combination을 통해 구할 수 있다.
- ✓ 팩토리얼을 $\mathcal{O}\left(a+b\right)$에 전처리 해 구해두면, 각각 $\mathcal{O}\left(1\right)$에 구할 수 있다.
- ✓ DAG에서 가장 많은 정점을 지나는 경로의 개수를 구한다.
- ✓ DAG DP를 통해 지나는 정점의 개수와 경로의 수를 저장해 $\mathcal{O}\left(N\right)$에 처리가능하다.
- ✓ $N/2$개씩 정점을 나누어 서로 연결시켜 놓으면 모든 간선이 유의미한 $\mathcal{O}\left(N^2\right)$개의 간선이 나오므로, 이보다 작게 하기는 쉽지 않다.

1E. 얼음마법사 동우의 대회 검수

`sqrt_decomposition`, `tree_set`
출제진 의도 – **Expert**

- ✓ 본 대회 (Div.1)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/9)
- ✓ Open Contest
  - First Solve: **menborong**, 150분 16초
  - **맞았습니다!!** 비율: **33.333%** (2/6)
- ✓ 출제자: 이종우
- ✓ 에디토리얼 작성자: 이종우

1E. 얼음마법사 동우의 대회 검수

- ✓ 쿼리는 최대 $Q = 100\,000$ 번만 입력된다.
- ✓ $\sqrt{Q}$ 번 초과의 쿼리와 관련된 대회의 수는 $\sqrt{Q}$ 미만이다.
- ✓ $\sqrt{Q}$ 번 이하의 쿼리와 관련된 대회는 $\sqrt{Q}$ 이하의 검수자를 가진다.
- ✓ 따라서 $\sqrt{Q}$ 번 초과의 쿼리와 관련된 대회의 검수를 진행할 때에는 받은 고통을 대회에 저장하며 $\sqrt{Q}$ 번 이하의 쿼리와 관련된 대회의 검수를 진행할 때에는 받은 고통을 검수자 개개인에 저장할 수 있다.
- ✓ 검수자 한 명이 받은 고통은 검수자 자신에게 저장된 고통과 검수자가 포함된 $\sqrt{Q}$ 번 초과의 쿼리와 관련된 대회에 저장된 고통의 합이다.
- ✓ $\mathcal{O}\left(Q\sqrt{Q}\log Q\right)$ 의 시간복잡도로 해결 가능하다.

1F. 차이를 M 으로

`number_theory`, `segment_tree`, `sweeping`
출제진 의도 – **challenging**

- ✓ 본 대회 (Div.1)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/1)
- ✓ Open Contest
  - First Solve: **swoon**, 171분 45초
  - **맞았습니다!!** 비율: **10.000%** (1/10)
- ✓ 출제자: 김동우, 이종우
- ✓ 에디토리얼 작성자: 이종우

1F. 차이를 M 으로

- ✓ 최대한 많은 수의 원소를 골라, 해당 원소들을 변경하지 않은 상태로 조건을 만족하게 만들고자 한다.
- ✓ $A_i \not\equiv A_j \mod M$ 이라면 A_i 와 A_j 는 동시에 고를 수 없다.
- ✓ $\left\lfloor \frac{A_i}{M} \right\rfloor + i \not\equiv \left\lfloor \frac{A_j}{M} \right\rfloor + j \mod 2$ 이라면 A_i 와 A_j 는 동시에 고를 수 없다.
- ✓ 따라서 동시에 선택될 가능성이 있는 최대 $\min(N, 2M)$ 개의 부분수열로 분할된다.
- ✓ 이제 **PC**. 차이를 M 이하로와 동일한 풀이가 가능하다.

1G. 4-cycle (Hard)

`graph, sqrt_decomposition`
출제진 의도 – **Challenging**

- ✓ 본 대회 (Div.1)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/0)
- ✓ Open Contest
  - First Solve: **chmpro**, 82분 39초
  - **맞았습니다!!** 비율: **5.556%** (1/18)
- ✓ 출제자: 박진한
- ✓ 에디토리얼 작성자: 박진한

1G. 4-cycle (Hard)

- ✓ degree가 B 이상인 정점을 heavy 정점, 미만인 정점을 light 정점이라 한다.
- ✓ 편의상 (a, b, c, d, a) 사이클의 각 정점의 heavy, light 여부를 HLHL 과 같이 나타내겠다.
- ✓ 길이가 4인 사이클은 사이클을 구성하는 정점의 heavy, light 여부에 따라 LLLL, HLLL, HLHL, HHLL, HHHL, HHHH과 같이 6가지 case로 나눌 수 있다.

1G. 4-cycle (Hard)

- LLLL 형태의 사이클의 개수는 light 정점 i 마다 light 정점으로 이루어진 (i, j, k) path를 순회하며 k 를 방문한 횟수 c_k 에 대해 $\frac{c_k(c_k-1)}{2}$ 를 더해 계산할 수 있다.
- (i, j) 간선의 개수는 $\mathcal{O}(M)$ 개이고, 각 j 마다 k 의 개수는 $\mathcal{O}(B)$ 개이니 시간복잡도는 $\mathcal{O}(MB)$ 이다.
- 나머지 형태의 사이클은 적어도 하나의 heavy 정점을 가지니, heavy 정점 i 마다 i 에 인접한 정점에 체크를 해둔 뒤, 그래프의 정점 k 를 순회하며 인접한 체크된 heavy, light 정점의 개수 a, b 를 구하며 각 형태 별 사이클의 개수를 구할 수 있다.
- 예를 들면 HLLL 형태의 사이클의 개수는 light 정점 k 에 대해 $\frac{b(b-1)}{2}$ 를 더해 계산할 수 있다.
- heavy 정점의 개수는 $\mathcal{O}(M/B)$ 개이고, 각 heavy 정점마다 $\mathcal{O}(N+M)$ 만큼의 연산을 수행하니 시간복잡도는 $\mathcal{O}((N+M)M/B)$ 이다.

1G. 4-cycle (Hard)

- ✓ 정리하면, 정점의 degree에 대해 sqrt decomposition을 이용하면 B를 $\mathcal{O}((N+M)^{1/2})$ 정도로 잡아줄 때 $\mathcal{O}\left(MB + (N+M)M/B\right)$ = $\mathcal{O}\left(M(N+M)^{1/2}\right)$에 문제를 해결할 수 있다.
- ✓ 정답을 구할 때 각 형태 별로 사이클이 중복으로 counting될 수 있으니, 구한 값을 중복 횟수로 나눠줘야 함에 주의하자.

1H. 히스토그램 직사각형으로 쪼개기

`dp`, `prefix_sum`, `ad_hoc`
출제진 의도 – **Challenging**

- ✓ 본 대회 (Div.1)
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/0)
- ✓ Open Contest
  - First Solve: **없음**
  - **맞았습니다!!** 비율: **0.000%** (0/0)
- ✓ 출제자: 김동우, 김재우
- ✓ 에디토리얼 작성자: 김재우

1H. 히스토그램 직사각형으로 쪼개기

- ✓ 주어진 히스토그램을 직사각형 모양으로 자르기 위해서는 높이가 바뀌는 지점마다 반드시 가로 혹은 세로로 선을 그어 나눠주어야 한다.
- ✓ i번째 직사각형부터 j번째 직사각형까지 이어진 히스토그램을 직사각형 모양으로 자른다고 할 때, 세로로 바닥 끝까지 이어진 분할이 있는지 여부를 통해 경우를 나눠볼 수 있다.
  - 세로로 바닥 끝까지 이어진 분할이 있는 경우
    - ▶ $i \leq k < j$인 k에 대해 k번째와 $k+1$번째 사이에 세로로 분할이 있다고 생각해보자.
    - ▶ 이 때의 비용은 i번째부터 k번째까지 분할 비용, k번째와 $k+1$번째 사이의 세로 분할 비용, $k+1$번째부터 j번째까지 분할 비용의 합으로 구할 수 있다.
    - ▶ 가능한 k들에 대해 모두 탐색해본 후 최솟값을 취하면 된다.

- 세로로 바닥 끝까지 이어진 분할이 없는 경우
  - 이 경우를 잘 생각해보면 반드시 높이가 최소인 지점에서의 가로 분할이 양쪽 끝까지 이어져야 한다.
  - 따라서, 최솟값을 기준으로 가로 분할이 된 상태에서의 최소 비용이 필요하다.
  - 이를 계산하기 위해서는 $DP[i][j][k]$를 i번째 직사각형부터 j번째 직사각형까지 이어진 히스토그램에서 k라는 높이로 양쪽 끝까지 가로 분할이 되어있는 상황에서의 최소 비용으로 정의하면 된다.
  - 이 때, 높이의 범위가 매우 크므로 적절히 좌표 압축을 해주면 된다.

✓ 위와 같이 DP를 정의하면 $\mathcal{O}\left(N^4 \lg N\right)$ 또는 $\mathcal{O}\left(N^4\right)$의 시간복잡도로 답을 구할 수 있다.

1H. 히스토그램 직사각형으로 쪼개기

- ✓ 이를 더 최적화하기 위해 주어진 구간 내에서 바닥 끝까지 오는 세로 분할이 없다면 높이가 최소가 되는 지점에서 가로 분할이 존재해야 하고, 반드시 양쪽 끝까지 가야 한다는 점을 이용해보자.
- ✓ $DP[i][j]$를 i번째 직사각형부터 j번째 직사각형까지 이어진 히스토그램을 직사각형 모양으로 자를 때, 구간 내 높이의 최솟값을 갖는 지점에서 가로 분할이 일어나고, 그 가로 분할이 구간의 양 끝까지 가도록 자르는 비용의 최솟값으로 정의하자.
- ✓ $DP[i][j]$의 값을 구하는 과정에서는 위에서 세로로 바닥 끝까지 이어진 분할이 있는 경우처럼 비슷하게 $i \leq k < j$인 k에 대해 k번째와 $k+1$번째 사이에서 세로 분할이 최솟값이 있는 지점까지 있다고 생각해보자. 이 때의 비용은 $DP[i][k]$, $DP[k+1][j]$, k번째와 $k+1$번째 사이의 세로 분할 비용, 가로 분할 비용의 합이다.
- ✓ 가로 분할 비용은 각 구간의 최솟값과 전체 구간의 최솟값이 일치하는지에 따라 이미 포함되었는지 여부가 달라지기 때문에 일치하지 않을 때마다 구간의 가로 길이를 더해주면 되고, 이 과정을 가능한 k에 대해 반복하여 최솟값을 취해주면 $DP[i][j]$를 구할 수 있다.

1H. 히스토그램 직사각형으로 쪼개기

- ✓ 그렇다면, 바닥까지의 세로 분할이 없는 경우는 어떻게 고려할까?
- ✓ 사실 위의 알고리즘에서 이미 세로 분할이 없는 경우도 고려되어 있다.
- ✓ i와 j 사이에는 구간의 최솟값을 가지는 지점이 당연하게도 존재할 것이기 때문에, 그 지점을 k로 잡는 순간이 바닥까지 닿는 세로 분할이 없는 경우이다.
- ✓ 따라서 위의 알고리즘을 통해 모든 DP 값을 채울 수 있다.
- ✓ 우리가 구하고자 하는 값을 얻기 위해서는 직사각형들의 처음 또는 마지막에 가로의 길이와 세로의 길이가 모두 0인 직사각형이 하나 더 있다고 생각하고 최종적으로 얻어지는 값에 가로 길이의 총합을 빼주면 된다.
- ✓ Prefix Sum을 통해 적당히 전처리를 미리 해두면 $\mathcal{O}\left(N^3\right)$에 문제의 답을 구할 수 있다.