PPC 2025

Official Solutions

PPC 2025 - 운영진



✓ 출제진

- 곽민성 minsung05
- 권찬우 kwoncycle
- 김경민 greydra_k
- 김재환 carotinoid1807
- 이유담 petamingks
- 임병찬 hyperbolic
- 오승혁 menborong
- 장래오 leo020630
- 정윤성 slah007

✓ 검수진

- gggkik
- jthis
- plast
- qvixnh22

✓ 테스터

- 이원기 qwerty_po
- 조정훈 justin8021
- mathtw1030, lion0625
- dohoon, gmroh06, gubshig
- paull04, jsj0412
- 6729skl, kolorvxl, m2002c

문제	문제 이름	의도한 난이도	출제자
A	어게인 포닉스	**Easy**	장래오[leo020630], 권찬우[kwoncycle], 정윤성[slah007]
B	사각형 모험	**Medium**	장래오[leo020630]
C	PPC와 CPP	**Hard**	임병찬[hyperbolic], 오승혁[menborong]
D	포닉스와 미끄럼틀	**Medium**	이유담[petamingks]
E	읽음	**Hard**	정윤성[slah007]
F	MMSQ	**Challenging**	장래오[leo020630]
G	Golden Section Search	**Easy**	장래오[leo020630]
H	Home Sweet Home	**Medium**	장래오[leo020630]
I	쿼리도	**Easy**	장래오[leo020630], 곽민성[minsung05]
J	사과 게임	**Hard**	김경민[greydra_k]
K	체스평평설	**Medium**	권찬우[kwoncycle]
L	통나무	**Hard**	권찬우[kwoncycle], 장래오[leo020630]
M	돈복사	**Easy**	권찬우[kwoncycle], 김재환[carotinoid1807]
C2	PPC와 CPP 2	**Challenging**	임병찬[hyperbolic], 오승혁[menborong]

A. 어게인 포닉스

`string`
출제진 의도 – **Easy**

- 본 대회 통계: 제출 19번, 정답 14명 (정답률 73.684%)
- 본 대회 First Solve: **트랄랄레로 트랄랄라**, 1분
- Open Contest 통계: 제출 159번, 정답 117명 (정답률 76.730%)
- Open Contest First Solve: **yyyy7089**, 0분
- 문제 아이디어: 장래오 `leo020630`, 권찬우 `kwoncycle`, 정윤성 `slah007`
- 문제 세팅: 장래오 `leo020630`



- 문자열에 등장하는 모든 PO를 PHO로 변경합시다.
- python의 `replace()` 함수 등을 이용해 쉽게 구현할 수 있습니다.
- P.S. 여러분의 응원 덕에 포스텍은 다시 한 번 ICPC World Finals에 진출하게 되었습니다.
- 감사합니다!

B. 사각형 모험

dp

출제진 의도 – **Medium**

- 본 대회 통계: 제출 20번, 정답 2명 (정답률 10.000%)
- 본 대회 First Solve: **트랄랄레로 트랄랄라**, 33분
- Open Contest 통계: 제출 64번, 정답 17명 (정답률 28.125%)
- Open Contest First Solve: **pyb1031**, 9분
- 문제 아이디어: 장래오 `leo020630`
- 문제 세팅: 장래오 `leo020630`

- ✓ 포닉스는 항상 최단 경로로만 움직입니다.
- ✓ 따라서 경로는 $(1, 1)$에서 오른쪽 또는 아래로 움직여 (N, M)에서 끝나는 꼴이며, 총 $N + M - 1$개의 칸을 거칩니다.
- ✓ (N, M)에 도착했을 때 사과의 개수가 x라 하면, 포닉스가 가진 돈은 $x \times A + (N + M - 1 - x) \times B = x \times (A - B) + (N + M - 1) \times B$입니다.
- ✓ x로 가능한 최솟값과 최댓값은 간단한 DP로 구할 수 있습니다. 이를 min_x, max_x라 합시다.

Claim: 모든 $min_x \leq x \leq max_x$ 에 대해, 사과를 정확히 x 개 포함하는 경로가 항상 존재합니다.

- ✓ 증명은 아래와 같습니다.
- ✓ 사과를 min_x 개 포함하는 경로를 P_1, max_x 개 포함하는 경로를 P_2 라 합시다.
- ✓ 경로에서 오른쪽에 인접한 칸으로 이동하는 것을 R, 아래에 인접한 칸으로 이동하는 것을 D라고 합시다.
- ✓ 경로에서 DR을 RD로 바꾸거나, RD를 DR로 바꾸면 사과의 개수가 최대 1 만큼 변화합니다.
- ✓ 위 연산만으로 P_1 을 P_2 로 변경할 수 있으므로 그 과정에서 모든 $min_x \leq x \leq max_x$ 에 대해 사과의 개수가 x 인 경로를 반드시 거치게 됩니다.

- ✓ 따라서 C가 $min_x \leq x \leq max_x$에 대해 $x \times (A-B) + (N+M-1) \times B$ 꼴인지 확인하면 문제를 해결할 수 있습니다.
- ✓ $A = B$라면 $C = (N+M-1) \times A$인지 확인하는 것과 같으므로 $A \neq B$라고 가정합시다.
- ✓ 이는 $C - (N+M-1) \times B$가 $A-B$의 배수이며, $min_x \leq \dfrac{C-(N+M-1)\times B}{A-B} \leq max_x$ 임과 동치입니다.
- ✓ 따라서 DP로 min_x, max_x를 구한 후 각 질문을 $\mathcal{O}(1)$으로 해결할 수 있습니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(NM+K)$입니다.

C. PPC와 CPP

pigeonhole_principle, ad_hoc
출제진 의도 – **Hard**

- 본 대회 통계: 제출 38번, 정답 0명 (정답률 0.000%)
- 본 대회 First Solve: **-**, -분
- Open Contest 통계: 제출 95번, 정답 1명 (정답률 1.053%)
- Open Contest First Solve: **kyo20111**, 225분
- 문제 아이디어: 임병찬hyperbolic, 오승혁menborong
- 문제 세팅: 임병찬hyperbolic

- 먼저, 주어진 문자열에서 C의 개수 $\times 2 =$P의 개수라고 가정합니다. 만약 아니라면, 이 문제의 답은 자명하게 NO입니다.
  다음을 모두 만족하는 문자열을 **좋은 문자열**이라고 정의합니다.
  - 이 문자열에서 C의 개수 $\times 2 =$P의 개수가 성립
  - 문자열의 시작 혹은 끝중 하나 이상이 C

  예를 들어, CPPPCP의 경우 좋은 문자열입니다.

Claim 1: 모든 **좋은 문자열**은 PPC 문자열입니다. 즉, 문제에 주어진 연산을 반복하여 빈 문자열로 만들 수 있습니다.

- ✓ 길이가 $3N$인 좋은 문자열 $A[1, 2, \cdots, 3N]$을 생각해봅시다. 일반성을 잃지 않고, $A[1] = C$라고 가정합니다.
- ✓ $A[2, 3, \cdots, 3N]$는 $N-1$개의 C, $2N$개의 P로 구성되어 있습니다. 이 문자열에서, C로 인하여 나누어지는 구간은 총 N개로, (첫번째 C의 왼쪽, 첫번째 C와 두번째 C 사이, ... , 마지막 C 오른쪽) 입니다.
- ✓ 비둘기 집의 원리에 의하여, 어떠한 구간 안에는 반드시 2개 이상의 P가 존재해야 합니다. i번째 C와 $i+1$번째 C 사이의 공간에 P가 2개 이상 존재한다고 가정합시다. 그렇다면, 저희는 CPP 혹은 PPC를 무조건 찾을 수 있습니다.

- ✓ 결론적으로, $A[2, 3, \cdots, 3N]$ 는 어떻게 배열되어 있던 간에 무조건 PPC 혹은 CPP를 찾을 수 있습니다. 그러므로 연산을 반복하여 PP만 남길 수 있습니다.
- ✓ $A[1] = C$ 이기 때문에 CPP를 찾아 지워서, 최종적으로 빈 문자열로 만들 수 있습니다. 따라서 모든 좋은 문자열은 PPC 문자열입니다.

C. PPC와 CPP



Claim 2: 어떤 문자열이 PPC 문자열이라는 것과 그 문자열이 좋은 문자열들로 쪼개진다는 것과 동치입니다. 즉, 길이가 N 인 PPC 문자열에 대하여, 적당한 $i_1, i_2, \cdots, i_j$ 가 존재하여, 구간 $[1, i_1], [i_1 + 1, i_2], \cdots, [i_j + 1, N]$ 들은 전부 좋은 문자열입니다.

- ✓ 만약 어떤 문자열이 좋은 문자열들로 쪼개진다고 가정합시다. Claim 1에서 모든 좋은 문자열은 빈 문자열이 된다는 것을 보였기 때문에, 이 문자열 역시 쉽게 빈 문자열이 된다는 것을 보일 수 있습니다. 따라서, 이 문자열은 PPC 문자열입니다.
- ✓ 반대로, 어떤 문자열이 PPC 문자열이면 좋은 문자열들로 쪼개진다는 것을 증명하겠습니다. 귀류법으로, 만약 PPC 문자열인데 좋은 문자열들로 쪼개지지 않는 문자열이 존재한다고 가정하겠습니다. 그러한 문자열중 가장 길이가 작은 문자열을 A라고 하겠습니다.
- ✓ 문자열 A는 PPC 문자열이기 때문에, 이 문자열을 빈 문자열로 만드는 방법이 존재합니다. 그 한가지 방법에 대해서, 제일 마지막으로 지워진 PPC 혹은 CPP를 생각해봅시다. 일반성을 잃지 않고, 제일 마지막으로 사용한 연산을 CPP라고 가정하겠습니다.

C. PPC와 CPP

- 제일 마지막으로 지워진 CPP에 대해서, 원래 문자열 A에서 C,P,P의 위치를 각각 a, b, c라고 정의하겠습니다. 즉, 적절한 연산을 진행하여, 원래 문자열에서 a번째에 위치한 C, b번째에 위치한 P, c번째에 위치한 P로 이루어진 "CPP"를 만들 수 있습니다. 그리고 이 문자열을 지움으로써 최종적으로 원래 문자열은 빈 문자열로 만들 수 있습니다.
- 그렇다면, 구간 $[1, a-1]$은 빈 문자열로 바꿀 수 있어야 합니다. 즉,$A[1, a-1]$는 PPC 문자열입니다. 원본 문자열 A가 좋은 문자열로 쪼개지지 않는 문자열중에 가장 작은 길이를 가진다고 했으므로, $A[1, a-1]$는 좋은 문자열들로 쪼개집니다.
- 또한 $A[a, N]$은 그 자체로 좋은 문자열입니다. 왜냐하면, 구간 $[a, N]$에서 C의 개수 $\times 2 =$ P의 개수가 성립하며, 구간의 시작인 a위치가 C이기 때문입니다.
- 그러므로, 문자열 A는 좋은 문자열들로 쪼개집니다. 이는 가정에 모순입니다. 그러므로, 모든 PPC 문자열은 좋은 문자열들로 쪼개집니다.

Claim 3: 어떤 문자열 $A[1, \cdots, N]$ 가 좋은 문자열들로 쪼개진다는 것과 다음 3가지중 1가지 이상을 만족한다는 것과 동치입니다.

- ✓ $A[1] = \mathrm{C}$
- ✓ $A[N] = \mathrm{C}$
- ✓ 다음을 만족하는 $1 \leq i < N$ 이 존재한다.
  - 구간 $[1, i], [i+1, N]$ 각각에 대해서, C의 개수 $\times 2 = $ P의 개수
  - $A[i] = A[i+1] = \mathrm{C}$

A 가 C의 개수 $\times 2 = $ P의 개수를 가정하고 있다는 것에 유의해주세요. 만약 C의 개수 $\times 2 \neq$ P의 개수라면 위 Claim은 성립하지 않고 무조건 PPC 문자열이 아닙니다.
Claim 3과 P,C의 누적합을 이용하여 어떤 문자열이 PPC 문자열인지 $\mathcal{O}(N)$ 에 확인이 가능합니다.



이 문제의 상위버전으로 다음과 같은 문제가 존재합니다:
https://atcoder.jp/contests/agc066/tasks/agc066_c

C2. PPC와 CPP 2

`pigeonhole_principle`, `ad_hoc`, `inclusive_exclusive_principle`, `generating_function`
출제진 의도 – **Challenging**

- ✓ Open Contest 통계: 제출 0번, 정답 0명 (정답률 0.000%)
- ✓ Open Contest First Solve: -, -분
- ✓ 문제 아이디어: 임병찬[hyperbolic], 오승혁[menborong]
- ✓ 문제 세팅: 임병찬[hyperbolic]



PPC와 CPP에서 증명한 Claim과 비슷하게, 다음 Claim을 보일 수 있습니다.

Claim: 어떤 문자열 $A[1, \cdots, (K+1)N]$가 좋은 문자열들로 쪼개진다는 것과 다음 3가지중 1가지 이상을 만족한다는 것과 동치입니다.

- ✓ $A[1] = \mathsf{C}$, 전체 구간에서 C의 개수 $\times K = \mathsf{P}$의 개수
- ✓ $A[(K+1)N] = \mathsf{C}$, 전체 구간에서 C의 개수 $\times K = \mathsf{P}$의 개수
- ✓ 다음을 만족하는 $1 \leq i < N$이 존재한다.
  - 구간 $[1, (K+1)i], [(K+1)i+1, (K+1)N]$ 각각에 대해서, C의 개수 $\times K = \mathsf{P}$의 개수
  - $A[(K+1)i] = A[(K+1)i+1] = \mathsf{C}$

- 먼저, $A[1] = C$ 혹은 $A[(K+1)N] = C$ 인 PPC 문자열의 개수를 세봅시다. 이는 $\binom{(K+1)N-2}{N-2} + \binom{(K+1)N-2}{N-1} \times 2$ 입니다. 이제 일반적으로 $A[1] = A[(K+1)N] = P$ 라고 가정합니다.
- 집합 S_i 를 **길이가** $(K+1)N$ **인 문자열 중에서,** $A[1] = A[(K+1)N] = P, A[(K+1)i] = A[(K+1)i+1] = C$ **고, 구간** $[1,(K+1)i], [(K+1)i+1,(K+1)N]$ **에서 C의개수** $\times K =$ **P의 개수를 만족하는 문자열의 집합**이라고 정의하겠습니다.
- 그렇다면, $A[1] = A[(K+1)N] = P$ 를 만족하는 PPC 문자열의 개수는, 집합 $S_1 \cup S_2 \cup \cdots \cup S_{N-1}$ 의 크기와 같습니다.

- ✓ 이 집합의 크기는 포함배제의 원리를 이용하여 계산할 수 있습니다. 즉, 모든 $1 \le i_1 < i_2 < \cdots < i_k \le N-1$을 만족하는 $(i_1, \cdots, i_k)$에 대하여, $(-1)^{k+1}|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}|$의 합이 집합 $S_1 \cup S_2 \cup \cdots \cup S_{N-1}$의 크기입니다.
- ✓ $|S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}|$의 값은 몇일까요? 집합 S_i의 정의에 의하여, $S_{i_1} \cap S_{i_2} \cap \cdots \cap S_{i_k}$에 속하는 문자열은 $A[1] = A[(K+1)N] = P, A[(K+1)i_1] = A[(K+1)i_1+1] = A[(K+1)i_2] = A[(K+1)i_2+1] = \cdots = A[(K+1)i_k+1] = C$를 만족해야 합니다. 그리고, 각 구간 $[(K+1)i_j+1, (K+1)i_{j+1}]$에 대해서, C의 개수 $\times K =$ P의 개수를 만족해야 합니다.

- 다항식 F_1을 **한쪽 끝이 P, 다른 한쪽 끝이 C이면서 C의 개수** $\times K =$ **P의 개수를 만족하는 길이가** $(K+1)i$**인 문자열의 개수**의 생성함수라고 정의합시다. 마찬가지로, 다항식 F_2을 **양쪽 끝이 C이면서 C의 개수** $\times K =$ **P의 개수를 만족하는 길이가** $(K+1)i$**인 문자열의 개수**의 생성함수라고 정의합시다.
- 그러면, $F_1 = \sum_{i=1}^{N} \binom{(k+1)i-2}{i-1} x^i, F_2 = \sum_{i=2}^{N} \binom{(K+1)i-2}{i-2} x^i$를 만족한다는 것을 쉽게 계산할 수 있습니다.
- 이제 포함배제 식을 정리하면, $A[1] = A[(K+1)N] = P$인 PPC 문자열의 개수는 $[x^N]F_1(1 - F_2 + F_2^2 - F_2^3 \cdots)F_1$이라는 것을 알 수 있습니다.



- $(1 - F_2 + F_2^2 - F_2^3 \cdots)$를 계산하는 방법은 크게 2가지가 존재합니다. 첫번째는 $[x^0]F_2 = 0$을 만족하므로 $(1 - F_2 + F_2^2 - F_2^3 + (-1)^N F_2^N)$만 계산해도 충분하고, 이것을 분할정복 + FFT로 $\mathcal{O}(N \log^2 N)$에 계산하는 방법입니다. 두번째로, 역시 $[x^0]F_2 = 0$를 만족하기에 $(1 - F_2 + F_2^2 - F_2^3 \cdots) = \dfrac{1}{1 + F_2}$가 성립하며, 다항식 나눗셈을 사용하여 $\mathcal{O}(N \log N)$에 계산하는 방법입니다.
- 최종적으로, $[x^N]F_1(1 - F_2 + F_2^2 - F_2^3 \cdots)F_1$는 $\mathcal{O}(N \log^2 N)$ 혹은 $\mathcal{O}(N \log N)$ 시간에 계산이 가능하며, 기존에 구해둔 $\binom{(K+1)N-2}{N-2} + \binom{(K+1)N-2}{N-1} \times 2$과 더해주면 그것이 전체 PPC 문자열의 개수가 됩니다.

D. 포닉스와 미끄럼틀

`math`

출제진 의도 – **Medium**

- 본 대회 통계: 제출 16번, 정답 6명 (정답률 37.500%)
- 본 대회 First Solve: **녜힁**, 60분
- Open Contest 통계: 제출 58번, 정답 17명 (정답률 29.310%)
- Open Contest First Solve: **yyyy7089**, 5분
- 문제 아이디어: 이유담`petamingks`
- 문제 세팅: 이유담`petamingks`



Claim: 경사가 최대인 두 점을 x 좌표가 증가하는 순으로 $P_i = (i, A_i)$, $P_j = (j, A_j)$ 라 하면, $i + 1 = j$ 입니다. 즉, 경사가 최대인 두 점은 항상 인접한 x 좌표를 가집니다.

- ✓ 증명은 아래와 같습니다.
- ✓ 일반성을 잃지 않고 $A_i < A_j$ 라 합시다.
- ✓ 만약 $i < k < j$ 인 점 $P_k = (k, A_k)$ 가 존재한다면 아래의 세 경우 중 하나로 분류할 수 있습니다.
  - 1. 점 P_k 가 선분 $\overline{P_iP_j}$ 보다 위에 있다면, 선분 $\overline{P_iP_k}$ 의 기울기가 선분 $\overline{P_iP_j}$ 의 기울기보다 큽니다.
  - 2. 점 P_k 가 선분 $\overline{P_iP_j}$ 보다 아래에 있다면, 선분 $\overline{P_kP_j}$ 의 기울기가 선분 $\overline{P_iP_j}$ 의 기울기보다 큽니다.
  - 3. 점 P_k 가 선분 $\overline{P_iP_j}$ 위에 있다면, 선분 $\overline{P_iP_k}$ 와 선분 $\overline{P_kP_j}$ 의 기울기는 $\overline{P_iP_j}$ 의 기울기와 같습니다.
- ✓ 따라서, $\max(|\frac{A_k - A_i}{k - i}|, |\frac{A_j - A_k}{j - k}|) \geq |\frac{A_j - A_i}{j - i}|$ 입니다.

- ✓ 따라서 경사의 최댓값은 $\max_{1 \le i < N} (|A_i - A_{i+1}|)$ 입니다.
- ✓ 경사의 최솟값은 경사의 역수의 최댓값과 같습니다.
- ✓ 따라서 점 $(A_1, 1), \cdots, (A_N, N)$ 에 대해 경사의 최댓값을 구한 후 역수를 취해 계산해줄 수 있습니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(N)$ 또는 $\mathcal{O}(N \log N)$ 입니다.

E. 읽음

`string`, `case_work`

출제진 의도 – **Hard**

- 본 대회 통계: 제출 10번, 정답 0명 (정답률 0.000%)
- 본 대회 First Solve: -, -분
- Open Contest 통계: 제출 13번, 정답 3명 (정답률 23.077%)
- Open Contest First Solve: **cohenpf**, 90분
- 문제 아이디어: 정윤성[slah007]
- 문제 세팅: 정윤성[slah007]

E. 읽음



- 우선, 문제에서 주어진 대로 읽지 않은 사람의 수를 세는 대신 읽은 사람의 수를 세겠습니다. j 번째 메시지를 읽은 사람의 수를 k_j 명이라 합시다. N 명에서 읽지 않은 사람의 수를 빼면 됩니다.
- i 번째 사람이 맨 마지막으로 읽은 메시지가 A_i 번째 메시지라고 합시다. 즉, i 번째 사람은 $1, 2, \ldots, A_i$ 번째 메시지를 읽었습니다. 만약 i 번째 사람이 메시지를 하나도 읽지 않았다면 A_i = 0입니다.
- i 번째 사람이 맨 마지막으로 보낸 메시지가 B_i 번째 메시지라고 합시다. B_i 값은 주어진 메시지 목록을 통해 구할 수 있고, $B_i \leq A_i \leq M$ 을 만족해야 합니다. 만약 i 번째 사람이 메시지를 한 번도 보내지 않았다면 $B_i = 0$ 입니다.
- 우리는 A_i 값을 아직 정확히 모르지만, A_i 값을 최소 B_i 에서 시작해서 각각의 k_j 를 만족하도록 조금씩 늘린다고 생각해 봅시다.
- k_j 가 내림차순이 아니거나, 모든 A_i 값을 B_i 로 배정해도 어떤 j 번째 메시지의 읽은 사람의 수가 k_j 명을 초과한다면 정답은 `IMPOSSIBLE`입니다.

E. 읽음



- 이외의 경우 모든 k_j 를 만족하도록 A_i 를 적당히 늘리는 것이 항상 가능합니다. 이때 A_i 값을 할당하는 방법이 유일하다면 정답은 UNIQUE, 그렇지 않다면 정답은 POSSIBLE 입니다.
- 모든 A_i 값을 B_i 로 배정할 때 j 번째 메시지를 읽은 사람이 정확히 k_j 명이 되는 경우 조건을 만족하는 방법은 유일합니다.
- 그렇지 않더라도 $A_i \geq j$ 까지 늘릴 수 있는 i 를 전부 다 추가했을 때 j 번째 메시지를 읽은 사람이 정확히 k_j 명이 되는 경우에도 조건을 만족하는 방법은 유일합니다.
- 정리하면 $j-1$ 번째 메시지를 읽은 사람들의 집합을 순차적으로 갱신하면서, j 번째 메시지를 읽은 사람 수가 정확히 k_j 이거나, $j-1$ 번째 메시지를 읽었지만 j 번째 메시지를 읽지 않은 모든 사람을 추가해야 k_j 가 되는 경우 정답은 UNIQUE 입니다.
- 단 하나의 j 에 대해서라도 위의 조건을 만족하지 않는다면 정답은 POSSIBLE 입니다.

F. MMSQ

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

- 본 대회 통계: 제출 3번, 정답 0명 (정답률 0.000%)
- 본 대회 First Solve: -, -분
- Open Contest 통계: 제출 6번, 정답 1명 (정답률 16.667%)
- Open Contest First Solve: **pyb1031**, 148분
- 문제 아이디어: 장래오 `leo020630`
- 문제 세팅: 장래오 `leo020630`



- 1번 쿼리가 주어지지 않고, $l = 1, r = N$ 인 2번 쿼리가 한 번만 주어지는 문제를 먼저 해결해 봅시다.
- $i \leq j$ 가 고정되었다고 가정합시다.
  $sum(i,j) + max(i,j) - min(i,j) = \max_{i \leq a, b \leq j} (sum(i,j) + A_a - A_b)$ 로 표현할 수 있습니다.
- 따라서 구간 $1 \leq i \leq j \leq N$ 을 고르고, 구간 안에서 원소 하나를 골라 더하고 하나를 골라 빼서 얻은 최댓값을 구하는 문제로 변형할 수 있습니다.
- 변형된 문제의 답은 원래 문제의 답과 같습니다.



- 변형된 문제는 다음과 같은 DP로 해결할 수 있습니다.
- 원소 하나를 골라 더하는 연산을 연산 1, 원소 하나를 골라 빼는 연산을 연산 2라 합시다.
- $DP(i, 0)$ = 연산 1과 연산 2를 모두 하지 않은 상태에서의 i번째 원소를 끝으로 하는 구간 중 최댓값
- $DP(i, 1)$ = 연산 1만 한 상태에서의 i번째 원소를 끝으로 하는 구간 중 최댓값
- $DP(i, 2)$ = 연산 2만 한 상태에서의 i번째 원소를 끝으로 하는 구간 중 최댓값
- $DP(i, 3)$ = 연산 1과 연산 2를 모두 한 상태에서의 i번째 원소를 끝으로 하는 구간 중 최댓값
- 답은 $\max_{1 \le i \le N}(DP(i, 3))$ 입니다.



- 답을 더 간단하게 정리해 봅시다. 상태를 6가지로 정의합니다.
- $(x, 0)$ = 아직 구간을 선택하지 않은 상태 $(x < i)$
- $(x, 1)$ = 구간이 진행 중이지만, 연산 1과 연산 2를 모두 하지 않은 상태 $(i \le x \le j)$
- $(x, 2)$ = 구간이 진행 중이고, 연산 1만 한 상태 $(i \le x \le j)$
- $(x, 3)$ = 구간이 진행 중이고, 연산 2만 한 상태 $(i \le x \le j)$
- $(x, 4)$ = 구간이 진행 중이고, 연산 1과 연산 2를 모두 한 상태 $(i \le x \le j)$
- $(x, 5)$ = 구간을 벗어난 상태 $(j < x)$
- $0 \le i \le N, 0 \le a, b \le 5$에 대해, 상태 (i, a)에서 상태 $(i + 1, b)$로 향하는 간선의 비용을 정의에 따라 정해줄 수 있습니다.
- $(0, 0)$에서 $(N + 1, 5)$로 가는 최장경로의 비용이 문제의 답이 됩니다.

- 전체 문제를 해결하기 위해서는 세그먼트 트리를 사용합니다.
- 1번 쿼리의 경우, 세그먼트 트리를 사용하면 각 간선의 가중치를 빠르게 업데이트할 수 있습니다.
- 2번 쿼리의 경우, $(l-1, 0)$ 에서 $(r+1, 5)$ 로 가는 최장경로의 비용이 문제의 답이 됩니다.
- $dist(l, r, a, b)$ 를 $(l-1, a)$ 에서 (r, b) 로 가는 최장경로의 비용으로 정의합시다.
- 이는 $m = \lfloor \frac{l+r}{2} \rfloor$ 에 대해 $dist(l, r, a, b) = \max_{0 \le c \le 5} (dist(l, m, a, c) + dist(m+1, r, c, b))$ 로 계산할 수 있습니다.
- 따라서, 세그먼트 트리의 각 노드에 이 정보를 저장하면 $O(6^3 \times \log N)$ 에 각 쿼리를 처리할 수 있습니다.
- 총 시간복잡도는 $O((N+Q) \times 6^3 \times \log N)$ 입니다.



- 또는 KOI 2014 중등부 4번 금광 문제로 잘 알려진 최대 연속합을 구하는 세그먼트 트리의 구조를 응용해 해결할 수도 있습니다.
- 이 경우 원래 구조에서 필요로 하는 4개의 정보를 각각 연산1/2 여부에 따라 저장해야 하기 때문에 각 노드에 16개의 정보가 필요합니다.
- merge 연산 구현이 다소 복잡함에 유의합시다.
- 총 시간복잡도는 $O((N+Q) \times 16 \times \log N)$ 입니다.

G. Golden Section Search

`math`
출제진 의도 – **Easy**

- 본 대회 통계: 제출 30번, 정답 13명 (정답률 43.333%)
- 본 대회 First Solve: **트랄랄레로 트랄랄라**, 14분
- Open Contest 통계: 제출 18번, 정답 15명 (정답률 88.889%)
- Open Contest First Solve: **yaksha**, 54분
- 문제 아이디어: 장래오[leo020630]
- 문제 세팅: 장래오[leo020630]



- ✓ 모든 $1 \leq i < N$ 에 대해 i 번 문제에 대해 황금 분할 하는 것이 가능한지 판단합시다.
- ✓ $\sum_{1 \leq k \leq i} D_k$ 으로 가능한 값은 $[\sum_{1 \leq k \leq i} L_k, \sum_{1 \leq k \leq i} R_k]$ 범위의 모든 정수입니다.
- ✓ $\sum_{i < k \leq N} D_k$ 으로 가능한 값은 $[\sum_{i < k \leq N} L_k, \sum_{i < k \leq N} R_k]$ 범위의 모든 정수입니다.
- ✓ 따라서 두 구간 $[\sum_{1 \leq k \leq i} L_k, \sum_{1 \leq k \leq i} R_k]$ 와 $[\sum_{i < k \leq N} L_k, \sum_{i < k \leq N} R_k]$ 간에 교집합이 존재하면 i 번 문제에 대해 황금 분할 하는 것이 가능합니다.
- ✓ 각 값은 누적합 배열을 이용해 구하거나 또는 반복문 과정에서 함께 구해줄 수 있습니다.
- ✓ 총 시간복잡도는 $\mathcal{O}(N)$ 입니다.

H. Home Sweet Home

`dijkstra`, `combinatorics`, `prefix_sum`, `binary_search`
출제진 의도 – **Medium**

- 본 대회 통계: 제출 6번, 정답 2명 (정답률 33.333%)
- 본 대회 First Solve: **장난을잘치는홍성현군**, 87분
- Open Contest 통계: 제출 14번, 정답 6명 (정답률 42.857%)
- Open Contest First Solve: **yaksha**, 71분
- 문제 아이디어: 장래오[leo020630]
- 문제 세팅: 장래오[leo020630]

- 간선이 추가되었기 때문에, 새로 추가한 간선 $E = (u, v, w)$ 가 쓸모 없는 간선이기 위해서는 모든 i 에 대해 $dist_G(i, 1) = dist_{G \cup E}(i, 1)$ 여야 합니다.
- 즉, 간선을 추가해도 1 번 정점에서 다른 정점으로의 최단거리가 변하지 않아야 합니다.
- u 번 정점과 v 번 정점을 잇는 간선을 추가했기 때문에, $dist(1, u)$ 와 $dist(1, v)$ 가 변하지 않는지만 확인하면 됩니다.
- $dist_G(1, u) \leq dist_G(1, v)$ 라 합시다. 최단 거리의 정의에 의해, $dist_G(1, u) + w < dist_G(1, v)$ 라면 $dist(1, v)$ 가 감소합니다.
- 따라서, $E = (u, v, w)$ 가 쓸모 없는 간선임은 $|dist_G(1, u) - dist_G(1, v)| \leq w$ 임과 동치입니다.

- ✓ 따라서, $1 \le u < v \le N, |dist_G(1,u) - dist_G(1,v)| \le K$ 인 u, v 에 대해 $K + 1 - |dist_G(1,u) - dist_G(1,v)|$ 을 모두 더하면 답을 구할 수 있습니다.
- ✓ 이는 $1 \le u, v \le N, u \ne v, |dist_G(1,u) - dist_G(1,v)| \le K$ 인 u, v 에 대해 $K + 1 - |dist_G(1,u) - dist_G(1,v)|$ 을 모두 더한 후 2 로 나눈 것과 같습니다.
- ✓ Dijkstra 알고리즘 등의 최단 경로 알고리즘으로 $dist_G(1,v)$ 를 모두 구한 후 오름차순으로 정렬합시다.
- ✓ 각 정점 u 에 대해 $|dist_G(1,u) - dist_G(1,v)| \le K$ 인 정점 v 는 이 배열에서 구간으로 표현되므로, 누적합과 이분 탐색을 이용하면 $K + 1 - |dist_G(1,u) - dist_G(1,v)|$ 을 모두 더한 값을 구할 수 있습니다.
- ✓ 이후 원래 존재했던 간선들에 대한 정보를 제거해야 함에 유의합시다.
- ✓ 총 시간 복잡도는 $\mathcal{O}(M \log N)$ 입니다.

I. 쿼리도

`implementation`
출제진 의도 – **Easy**

- 본 대회 통계: 제출 25번, 정답 7명 (정답률 28.000%)
- 본 대회 First Solve: **장난을잘치는홍성현군**, 20분
- Open Contest 통계: 제출 24번, 정답 10명 (정답률 41.667%)
- Open Contest First Solve: **yyyy7089**, 28분
- 문제 아이디어: 장래오 `leo020630`
- 문제 세팅: 곽민성 `minsung05`

- 벽을 반으로 자른 것을 장애물이라 합시다.
- i번째 행과 $i+1$번째 행 사이에 있는 장애물만 남긴 상태에서, 남은 장애물은 모두 짝수개여야 합니다.
- 또한, 인접한 장애물끼리 묶었을 때 각 장애물 묶음의 길이 역시 모두 짝수여야 합니다.
- 매 묶음에 대해 인접한 2개의 장애물들을 등장 순서대로 하나의 벽으로 짝지어줄 수 있습니다.

- 열과 열 사이의 장애물에 대해서도 같은 작업을 반복합시다.
- 모든 작업을 마친 후 십자 모양으로 교차하는 벽이 없는지 판별하면 됩니다.
- 총 시간 복잡도는 $\mathcal{O}\left(N^2\right)$ 입니다.

J. 사과게임

`dp`, `prefix_sum`
출제진 의도 – **Hard**

- 본 대회 통계: 제출 9번, 정답 0명 (정답률 0.000%)
- 본 대회 First Solve: **-**, -분
- Open Contest 통계: 제출 31번, 정답 6명 (정답률 19.355%)
- Open Contest First Solve: **yaksha**, 12분
- 출제자: 김경민`greydra_k`

- ✓ $DP_{i,j}$ 를 구간 $[i, j]$ 에서 얻을 수 있는 최대 점수로 정의합시다. $(1 \leq i \leq j \leq N)$
- ✓ 이웃한 두 구간 $[a, k]$, $[k+1, b]$ 에 대한 해 $DP_{a,k}$, $DP_{k+1,b}$ 를 알 때, 두 구간을 이어붙여 $DP_{a,b}$ 를 구할 수 있을까요?
- ✓ 크게 세 가지 경우로 나눠서 생각할 수 있습니다.
  - 1. 두 구간 모두 사과를 전부 없앤 경우, $DP_{a,b} = DP_{a,k} + DP_{k+1,b}$
  - 2. 두 구간에서 일부 사과가 남았는데 숫자의 합이 10이어서 없앨 수 있는 경우, $DP_{a,b} = DP_{a,k} + DP_{k+1,b} + 1$
  - 3. 이외의 경우, $DP_{a,b} = DP_{a,k} + DP_{k+1,b}$

- 그렇다면 구간에서 사과를 모두 없앴는지, 얼마나 없앴는지는 어떻게 알아낼 수 있을까요?
- 사과를 한 번 없앨 때마다 점수는 1점이 늘어나고, 구간 내 숫자들의 합은 10이 줄어듭니다.
- 따라서 $(\sum_{a \le k \le b} A_k) - 10 \cdot DP_{a,b}$ 의 값이 구간 $[a, b]$ 에 남은 숫자들의 합을 나타내게 됩니다. 해당 값이 0일 경우 구간 내의 모든 사과를 없앴다는 의미입니다.
- 각 구간의 누적합은 $\mathcal{O}(N)$ 에 전처리 후 $\mathcal{O}(1)$ 에 알아낼 수 있고, $\mathcal{O}(N^2)$ 개의 모든 구간 $[a, b]$ 에 대해 구간을 둘로 나눌 수 있는 경우를 고려해야 하므로 총 시간복잡도는 $\mathcal{O}(N^3)$ 입니다.

K. 체스평평설

`constructive`

출제진 의도 – **Medium**

- 본 대회 통계: 제출 23번, 정답 11명 (정답률 47.826%)
- 본 대회 First Solve: **Localhost**, 30분
- Open Contest 통계: 제출 31번, 정답 6명 (정답률 19.355%)
- Open Contest First Solve: **yyyy7089**, 15분
- 문제 아이디어: 권찬우`kwoncycle`
- 문제 세팅: 권찬우`kwoncycle`

- ✓ $M \leq 2$인 경우 불가능함을 쉽게 알 수 있습니다.
- ✓ 이제 $M \geq 3$인 경우를 보면 됩니다. 결론적으로, 이들은 모두 해를 구성할 수 있습니다.
- ✓ $M = 3, 4, 5$인 경우를 손으로 직접 구성한 후, $M = 3k + t$ 꼴로 나눈 후 각각의 $t = 0, 1, 2$를 $M = 3, 4, 5$인 경우에서 확장시켜 재귀적으로 나타낼 수 있습니다.

L. 통나무

`geometry`
출제진 의도 – **Hard**

- 본 대회 통계: 제출 12번, 정답 1명 (정답률 8.333%)
- 본 대회 First Solve: **장난을잘치는홍성현군**, 219분
- Open Contest 통계: 제출 20번, 정답 1명 (정답률 5.000%)
- Open Contest First Solve: **aeren**, 147분
- 문제 아이디어: 권찬우[kwoncycle], 장래오[leo020630]
- 문제 세팅: 권찬우[kwoncycle]

L. 통나무



- 만약 통나무가 없다면 무조건 건물들을 연결할 수 있습니다. 구체적으로, 임의의 한 점을 고른 후 그 점과 다른 점을 모두 연결해버리면 됩니다. 일직선 위에 세 개 이상의 점이 있는 경우 가까운 순으로 이으면 됩니다.
- 통나무를 직선으로 확장한 후, 통나무 왼쪽의 건물들의 집합을 C_1, 오른쪽의 건물들의 집합을 C_2 라고 합시다. 또, 전체 건물들의 집합을 C 라고 합시다. 만약 $C = C_1$ or $C = C_2$ 면 위 방법대로 그냥 해를 구성하면 되니, 그렇지 않은 경우만 고려해봅시다.
- 만약 통나무가 C 의 convex hull를 가로지른다면, C_1 과 C_2 의 원소들은 절대 연결될 수 없고, No를 출력하면 됩니다.
- 만약 가로지르지 않는다면, C 의 convex hull이 통나무로 인해 끊어지지 않았으므로 C_1 과 C_2 를 이을 수 있는 점의 쌍 P_1, P_2 가 반드시 존재합니다. 이는 convex hull을 시계 방향으로 따라가면 쉽게 찾을 수 있습니다. 각 P_1, P_2 에 대해 각각의 영역에서 건물 연결을 시키면 최종적으로 모든 건물을 연결할 수 있습니다.

M. 돈복사

`implementation`
출제진 의도 – **Easy**

- 본 대회 통계: 제출 16번, 정답 14명 (정답률 87.500%)
- 본 대회 First Solve: **Localhost**, 7분
- Open Contest 통계: 제출 71번, 정답 57명 (정답률 85.915%)
- Open Contest First Solve: **aeren**, 1분
- 문제 아이디어: 권찬우`kwoncycle`
- 문제 세팅: 김재환`carotinoid1807`



- ✓ 포닉스가 돈복사를 최대 3번밖에 할 수 없음을 쉽게 관찰할 수 있습니다.
- ✓ 브루트포스로 답을 구해줍시다.