2024 DGIST 알고리즘 경진대회

Official Solutions

2024 DGIST 알고리즘 경진대회 - 운영진 및 검수진

✓ 운영진

- 김도훈 `99asdfg`
- 박신욱 `bnb2011`
- 오상규 `lambda`
- 우은규 `ekwoo`
- 채척 `cocjr0208`

✓ 검수진

- dongwook7
- gs20036
- jinhan814
- jyheo98
- leo020630
- sjhi00
- tlsdydaud1
- w8385
- yyyy7089

	문제	의도한 난이도	출제자
A	장기	**Easy**	채척cocjr0208, 우은규ekwoo
B	4-LSB	**Easy**	채척cocjr0208, 우은규ekwoo
C	DPS	**Medium**	우은규ekwoo
D	반복수	**Medium**	김도훈99asdfg
E	Symmetry	**Medium**	박신욱bnb2011
F	N-가위바위보	**Hard**	김도훈99asdfg
G	트랙 정리하기	**Hard**	김도훈99asdfg
H	Starlight Express	**Hard**	박신욱bnb2011
I	팀 선발	**Challenging**	김도훈99asdfg
J	Double Up 2	**Challenging**	박신욱bnb2011

A. 장기

`math, arithmetic, implementation`

출제진 의도 – **Easy**

- 제출 201번, 정답 147번, 정답률 73.134%
- First Solve: **ychangseok**, 1분
- 출제자: 채척 `cocjr0208`, 우은규 `ekwoo`

A. 장기

- 첫 번째 줄에 입력받은 척이의 기물의 수와 각 기물에 대한 점수를 곱한 후 그 합 A를 구합니다.
- 두 번째 줄에 입력받은 은규의 기물의 수와 각 기물에 대한 점수를 곱한 후, 1.5점을 추가하여 그 합 B를 구합니다.
- A와 B의 대소를 비교하여 정답을 출력합니다.
- 총 시간복잡도는 $\mathcal{O}(1)$ 입니다.

B. 4-LSB

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

- 제출 180번, 정답 138번, 정답률 76.667%
- First Solve: **ychangseok**, 3분
- 출제자: 채척 `cocjr0208`, 우은규 `ekwoo`

B. 4-LSB

- 입력받은 세 십진수의 값을 각각 진법변환을 통해 이진수로 변환합니다(예외처리: 십진수 $0 \sim 7$의 경우, 앞단에 0을 추가적으로 채워야 합니다).
- 슬라이싱을 통해 변환된 이진수의 하위 4비트 값들을 추출한 후, 주어진 순서대로 이어붙입니다.
- 도출한 값을 다시 십진수로 변환한 후, 정답을 출력합니다(예외처리: 값이 세자리수 이하라면, 앞단에 0을 추가적으로 채워야 합니다).
- 총 시간복잡도는 $\mathcal{O}(1)$입니다.
- 이외에도 modular arithmetic을 이용한 풀이 또한 가능합니다!

C. DPS

`math`

출제진 의도 – **Medium**

- 제출 106번, 정답 41번, 정답률 38.679%
- First Solve: **hyperbolic**, 11분
- 출제자: 우은규[ekwoo]

C. DPS

- $[0, X]$ 구간에 입힌 데미지에 대해 각 스킬이 시전 시간동안 온전히 시전되는 횟수는 총 $\lfloor E/(R_i + A_i) \rfloor$ 회이며, 그동안 입힌 데미지의 총 양은 $D_i \times \lfloor E/(R_i + A_i) \rfloor$ 입니다.
- 남은 시간 중 스킬의 마지막 시전 시간은 대기 시간을 제외한 시간이며 그동안 입힌 데미지의 양은 $max(0, (\{E \mod (R_i + A_i)\} - R_i)/A_i * D_i$ 입니다.
- $[S, E]$ 구간 동안 입힌 총 데미지는 $[0, E]$ 구간에서 입힌 데미지에서 $[0, S]$ 에서 입힌 데미지를 빼서 구할 수 있습니다.
- $[S, E]$ 구간 동안 입힌 총 데미지 양을 측정 시간 $(S - E)$ 로 나누어 해답을 구할 수 있습니다.
- 총 시간복잡도는 $\mathcal{O}(N)$ 입니다.

D. 반복수

`bruteforcing`
출제진 의도 – **Medium**

- ✓ 제출 86번, 정답 24번, 정답률 27.907%
- ✓ First Solve: **dbgusdn012**, 16분
- ✓ 출제자: 김도훈[99asdfg]

D. 반복수

- 임의의 K 자리수 x 로 만들 수 있는 K-반복수의 개수를 생각해 봅시다.
- 가령, 3 자리수 123 으로 만들 수 있는 10^{12} 이하의 K-반복수는 다음과 같습니다.
  - $123, 1231, \cdots, 12312312, 123123123, 1231231231, 12312312312, 123123123123$
- 위와 같이, K 자리수 x 로 만들 수 있는 K-반복수의 개수는 12 개 이하입니다.
- 즉, 1 이상 10^{12} 이하의 반복수의 개수는 아무리 많아도 $10^6 \times 12$ 개 정도입니다.
- 따라서, 10^6 이하이며 K 자리수인 모든 수에 대해, 모든 반복수를 순회하며 각 반복수들이 A 이상 B 이하이며 M 으로 나누어떨어지는지 확인해 주면 됩니다.
- 총 시간복잡도는 $\mathcal{O}\left(10^K \log_{10} B\right)$ 입니다.

E. Symmetry

`combinatorics`
출제진 의도 – **Medium**

- 제출 36번, 정답 18번, 정답률 50.000%
- First Solve: **hyperbolic**, 32분
- 출제자: 박신욱`bnb2011`

E. Symmetry

- $A_{x,y}$: 대칭 격자의 가운데 세로축이 x 열과 $x+1$ 열 사이에 있을 때, y 행에서 세로축을 중심으로 하는 최대 팰린드롬 길이라고 정의합니다.
- 모든 A의 값을 계산하는 것은 나이브하게 $\mathcal{O}\left(NM^2\right)$ 시간에 가능합니다.
- 세로축을 고정했을 때 대칭 격자의 개수는 $\sum_{i=1}^{N}\sum_{j=i}^{N}\min\left(A_{x,i}, A_{x,i+1}, \cdots, A_{x,j}\right)$ 입니다.
- 위 수식의 값은 $\mathcal{O}\left(N^2\right)$ 시간에 구할 수 있고, 이 작업은 $\mathcal{O}\left(M\right)$ 번 반복해야 합니다.
- 따라서 총 시간복잡도는 $\mathcal{O}\left(NM(N+M)\right)$ 입니다. Manacher 알고리즘과 Monotone Stack을 이용하면 $\mathcal{O}\left(NM\right)$ 시간에도 해결할 수 있습니다.

F. N-가위바위보

`case_work, greedy`
출제진 의도 – **Hard**

- 제출 27번, 정답 3번, 정답률 11.111%
- First Solve: **golazcc83**, 54분
- 출제자: 김도훈[99asdfg]

F. N-가위바위보

- ✓ 어떤 카드 x에 대해, $x-1, x+1$을 제외한 나머지 모든 $N-2$장의 카드는 모두 무승부 처리가 됩니다.
- ✓ 이제부터 각 카드에 대해 무승부를 낼 수 있는 카드를 "무승부 카드", 그렇지 못한 두 장의 카드를 "승패 카드"라고 합시다.
- ✓ 직관적으로, 상당히 많은 카드들에 대해 무승부 처리를 낼 수 있을 것으로 짐작할 수 있으며 이는 다음 관찰으로 연결될 수 있습니다.
  - 관찰. N번의 대결 중 최소 $N-2$회는 무승부를 낼 수 있다.
  - 증명: 상대가 낸 순서대로 1번째 카드부터 $N-2$번째 카드까지 무승부 처리가 되도록 아무 카드나 내더라도, $N-2$번째 대결에서 반드시 세 장의 카드가 손에 남아있으며, 그중 적어도 하나의 카드는 반드시 무승부 카드이므로 $N-2$번째 대결까지는 그리디하게 카드를 내도 항상 무승부가 가능하다.

F. N-가위바위보

- 위 관찰을 토대로, 상대가 낸 카드의 종류의 개수에 따라 경우의 수를 나눠서 문제를 해결할 수 있습니다.
- 상대가 낸 카드의 종류의 개수가 4장 이상이라고 가정합시다.
- 위 관찰에 따라, 반드시 $N-4$회는 무승부 처리가 가능하며, 카드를 내는 순서를 적절히 바꿔서 서로 다른 4가지 종류의 카드를 남겨둘 수 있습니다.
- 그 이후 남은 4장의 카드 중 2장의 카드를 처리한 뒤, 상대의 3번째 카드와 4번째 카드에서 각각 승패 카드만이 남았더라도, 반드시 처음 처리한 두 장의 카드 중 한 장의 카드와 맞바꿔 각각의 상대 카드에 대해 무승부 처리가 가능하도록 카드를 바꿔낼 수 있습니다.
- 따라서, 상대가 낸 카드의 종류의 개수가 4장 이상이면 정답은 반드시 N입니다.
- 이제, 상대가 낸 카드의 종류가 3개 이하인 경우들을 하나씩 확인해 봅시다.

F. N-가위바위보

1. 상대가 낸 카드의 종류가 모두 같은 경우
   - 무승부 카드의 종류는 $N-2$개이므로, $N-2$회 무승부를 낼 수 있음은 자명합니다.
2. 상대가 낸 카드의 종류가 총 2개인 경우

2.1 두 종류의 카드의 승패 카드 종류가 겹치지 않는 경우

- 이 경우는 두 종류의 카드 중 등장 횟수가 많은 카드는 최소 2회는 등장했으므로, 등장 횟수가 적은 카드의 승패 카드를 등장 횟수가 많은 카드에 사용하면 등장 횟수가 적은 카드는 모두 무승부 처리가 가능합니다.
- 이때, 등장 횟수가 적은 카드의 등장 횟수가 2회 이상이라면 위의 경우와 마찬가지로 두 종류의 카드 모두 무승부 처리를 낼 수 있으며, 등장 횟수가 1회라면 등장 횟수가 많은 카드는 반드시 1회는 무승부 처리가 되지 않는 경우가 발생하며, 최대 무승부 횟수는 $N-1$회가 됩니다.

2.2 두 종류의 카드의 승패 카드 종류가 1 개 겹치는 경우

- 두 종류의 카드의 겹치는 무승부 카드를 낼 때는 항상 무승부를 낼 수 없으며, 그 외에 각 카드의 승패 카드를 서로 다른 카드의 등장에 사용하면 반드시 $N-1$ 번 무승부 처리가 됨을 확인할 수 있습니다.

2.3 두 종류의 카드의 승패 카드 종류가 2 개 겹치는 경우

- 해당 경우는 $N=4$ 인 경우에만 등장하며, $\{1,1,3,3\}$ 과 같은 경우가 그 예시입니다. 이 경우, 단순히 모든 경우를 순회해 보거나 직접 넣어보며 (혹은 예제 3의 출력을 보며) 정답이 2 임을 파악할 수 있습니다.

3. 상대가 낸 카드의 종류가 총 3개인 경우
   - 등장하는 카드들 중, 가장 많이 등장한 세 장의 카드를 각각 A, B, C라 합시다.
   - 세 종류의 카드에 대해, A의 승패 카드 두 장을 각각 B, C에 넣어줍시다.
   - 이때, 등장한 총 카드의 종류가 3개이고 A와 B의 무승부 카드가 완전히 겹치며, C의 등장 횟수가 1회인 경우에는 적어도 한 번은 무승부 처리가 될 수 없습니다. 이 상황은 위에서 언급한 바와 같이 $N = 4$인 경우에만 발생합니다.
   - 그 외의 경우, A의 두 장의 승패 카드들을 B 혹은 C에 무승부로 적절히 나눠줄 수 있습니다.
   - 그 후, B의 승패 카드들을 각각 A와 C에, C의 승패 카드들을 각각 A와 B에 무승부 처리가 되도록 나눠주면 항상 N번의 무승부 처리가 가능합니다.

F. N-가위바위보

- ✓ 이상을 정리하면, 다음을 구현하면 최대 무승부 횟수를 구할 수 있습니다.
  - 상대가 낸 카드가 모두 같은 종류의 카드인 경우: $N-2$
  - 상대가 낸 카드의 종류가 총 2개인 경우
    - 한 종류의 카드가 $N-1$회 등장하거나, 승패 카드 종류가 1개 겹치는 경우: $N-1$
    - 두 종류의 승패 카드 종류가 2개 겹치는 경우: $N-2$
    - 그 외의 경우: N
  - 상대가 낸 카드의 종류가 총 3개 이상인 경우
    - 가장 많이 등장한 두 종류의 승패 카드가 완전히 겹치는 경우: $N-1$
    - 그 외의 경우: N

F. N-가위바위보

- 단순히 위의 과정을 구현하면 되며, 어떤 방식으로 구현하든 $\mathcal{O}(N \log N)$ 이하 혹은 그에 준하는 정도의 시간복잡도로만 구현하면 됩니다.
- 출제자 Tip: 풀이의 깔끔한 흐름을 위해, 풀이는 "최소 $N-2$ 회는 무승부가 가능하다" -> "카드가 4 종류 이상이라면 반드시 N 번 무승부가 가능하다" -> "그 외의 경우를 각각 나눠서 해결하자"의 순서로 서술하였으나, 실제 문제를 풀 때는 그 역순으로, 즉 작은 케이스부터 확인해 보는 것이 문제를 해결하는 데 더 도움이 된다고 생각합니다.
- $N=4$ 인 경우를 미처 체크하지 못하고 경우의 수를 나눠놓았다고 해도, 논리를 잘 따라가면 예제 3 에서 $N=4$ 인 경우에만 예외 케이스가 발생함을 확신할 수 있으며, 단순히 $N=4$ 인 경우에만 완전 탐색을 하는 방식으로도 해당 문제를 해결할 수도 있습니다.

G. 트랙 정리하기

`two_pointer, deque`
출제진 의도 – **Hard**

- ✓ 제출 26번, 정답 7번, 정답률 26.923%
- ✓ First Solve: **wbcho0504**, 65분
- ✓ 출제자: 김도훈[99asdfg]

G. 트랙 정리하기

- 시작 지점에서부터, 왼쪽으로 가장 가까운 쓰레기가 있는 구역과 오른쪽으로 가장 가까운 쓰레기가 있는 구역을 각각 l, r 이라 합시다.
- A_l 이나 A_r 둘 중 하나가 0 이 되기 전까지는 항상 $[l, r]$ 구간을 왔다갔다하며 움직이게 됩니다.
- 따라서, 두 지점을 왕복하는 횟수는 $min(A_r, A_l)$ 회이며, 해당 순회가 끝나면 l 또는 r 값이 바뀌게 됩니다.
- 왕복하는 횟수와 왕복 거리를 단순히 곱함으로써 각 l, r 값에 대해 $\mathcal{O}(1)$ 만에 왕복 이동 거리를 계산해줄 수 있으며, $l = r$ 인 경우까지만 순회하면 되므로 총 시간복잡도 $\mathcal{O}(N)$ 으로 문제를 해결할 수 있습니다.

G. 트랙 정리하기

- 말로만 하면 참 쉬운 문제이지만, 실제 구현 시 다소 애를 먹을 수 있습니다.
- 이는 왕복이 l에서 끝나는지, r에서 끝나는지에 따라 이동 방향이 달라질 수 있기 때문입니다.
- 이를 해결하기 위해, 왕복 시작 구역과 종료 구역을 모두 r으로 고정하고, 왕복이 끝난 이후에는 항상 오른쪽으로 가는 방향으로 가도록 고정해 둡시다.
- 그러면, l, r 값이 바뀌는 경우 단순히 현재 위치를 r로 옮긴 뒤, 둘 사이의 거리와 왕복 횟수를 곱해줌으로써 이동 거리를 쉽게 계산할 수 있습니다.
- 단, 이 경우 $l = r$이 된 직후 $A_r = 0$인 상황과 $A_r \neq 0$인 상황을 잘 나눠서 처리해 주어야 합니다.
- 투 포인터 말고도 deque 자료구조를 사용하여 문제를 해결할 수도 있습니다.

H. Starlight Express

`graph_traversal`, `union_find`, `sorting`
출제진 의도 – **Hard**

- 제출 11번, 정답 5번, 정답률 45.455%
- First Solve: **wbcho0504**, 85분
- 출제자: 박신욱[bnb2011]

H. Starlight Express

- 문제를 그래프의 관점에서 생각해봅시다.
- 각 역을 정점으로 두고, 이동할 수 있는 두 역 사이를 간선으로 이어줍니다.
- 그러면 어떤 두 역이 연결되어 있다는 것을 **두 정점이 그래프의 같은 연결 컴포넌트에 속해 있다**와 같이 해석할 수 있습니다.

H. Starlight Express

- 하지만 가능한 모든 간선을 이어주는 것은 불가능합니다. 모든 역이 같은 x 좌표 또는 y 좌표에 있는 경우를 생각해보면, 모든 역들은 서로 이동할 수 있기 때문에 이어야 하는 간선이 $\mathcal{O}\left(N^2\right)$ 개가 되기 때문입니다.
- 사실 문제에서 중요한 것은 모든 간선을 잇는 것이 아니라, 정점 간의 연결 관계를 잘 표현하는 것입니다.
- 기차 역들을 x 좌표, y 좌표에 따라 각각 오름차순으로 정렬한 뒤, 정렬 결과에서 서로 인접해 있는 두 역이 서로 이동 가능한 경우 두 역을 간선으로 이어줍니다. 이러면 $\mathcal{O}\left(N\right)$ 개의 간선만을 이용해 역들의 연결 관계를 나타낼 수 있습니다.

H. Starlight Express

- 이후 그래프에 있는 연결 컴포넌트 수에 따라서 케이스가 나뉩니다.

1. 연결 컴포넌트 개수가 두 개 이상일 때
   - 새로운 역을 추가하면 원래 연결되지 않았던 두 역을 이어줄 수 있습니다. 그리고 이동할 수 있는 역 쌍을 최대화하기 위해선 자명하게 가장 많은 정점이 포함된 두 개의 연결 컴포넌트를 잇는 것이 최적입니다.
   - (x_a, y_a)에 있는 역과 (x_b, y_b)에 있는 역을 이어주기 위해서는 (x_a, y_b) 또는 (x_b, y_a)에 새로운 역을 설치하면 됩니다. 두 역이 처음에 연결되지 않은 상태였다면, 새로 설치하는 역의 좌표가 기존에 존재하지 않았음이 보장되므로 문제의 조건을 만족합니다.

H. Starlight Express

2. 연결 컴포넌트가 하나일 때
   - 연결 컴포넌트가 하나일 때는 새로 설치할 역이 기존에 있는 역들 중 하나로 이동할 수 있기만 하면 됩니다.
   - 설치할 역의 x 좌표를 1 번 역의 x 좌표로 두고, 좌표 범위를 벗어나지 않는 선에서 y 좌표를 랜덤하게 설정하면 쉽게 구현할 수 있습니다.
   - 이때 설치할 역의 좌표가 기존 역의 좌표와 겹치지 않도록 주의해야 합니다.

✓ 연결 컴포넌트의 크기와 같은 정보를 얻는 것은 단순 그래프 탐색 또는 분리 집합을 이용해서 구현할 수 있습니다.

✓ 정렬에 $\mathcal{O}(N \log N)$, 그래프 탐색에 $\mathcal{O}(N)$ 시간이 필요하므로 총 시간복잡도는 $\mathcal{O}(N \log N)$ 입니다.

I. 팀 선발

`dp, greedy`
출제진 의도 – **Challenging**

- 제출 8번, 정답 5번, 정답률 62.500%
- First Solve: **wbcho0504**, 93분
- 출제자: 김도훈[99asdfg]

I. 팀 선발

- 각 학생의 목표 점수를 오름차순으로 정렬하여 순서대로 각각 1 번 학생부터 N 번 학생이라 합시다.
- 이때, dp 식 및 계산식을 다음과 같이 세워 봅시다.
  - $dp[i]$: i 번 이하의 학생들로 팀들을 구성한 경우 팀 실력의 합의 최댓값
  - $dp[i] = max(dp[i], dp[j] + f(j+1, i)); B_j \geq B_i - K; j \leq i - 3;$
  - $f(l, r)$: 구간 $[l, r]$ 에서 실력이 가장 높은 세 학생의 실력의 합
- 위 계산식 대로 계산하면 시간복잡도가 $\mathcal{O}\left(N^2\right)$ 으로, 시간 초과를 받게 됩니다.
- 추가 관찰을 통해 시간복잡도를 줄여봅시다.

I. 팀 선발

- 직관적으로 생각하면, 팀을 선발할 때 팀을 $\{1, 3, 5\}, \{2, 4, 6\}$ 과 같이 선발하는 것은 다소 무의미한 행동입니다. 이는 팀을 $\{1, 2, 3\}, \{4, 5, 6\}$ 과 같이 선발한 것과 팀 실력의 합이 동일하기 때문입니다.
- 이를 토대로, 다음과 같은 관찰을 할 수 있습니다.
  - 관찰. $dp[i]$ 를 계산할 때, 최댓값은 항상 $j \geq i - 5$ 인 경우에 등장한다.
  - 증명. $j = i - 6$ 인 경우, $dp[i] = max(dp[i], dp[i - 6] + f(i - 5, i))$ 로 계산된다. 이때, $[i - 5, i]$ 번 학생 중 세명을 골라 하나의 팀으로 묶는 것보다 항상 $\{i - 5, i - 4, i - 3\}, \{i - 2, i - 1, i\}$ 와 같이 두 팀으로 묶는 것이 더 팀 실력의 합이 커지므로, 항상 $dp[i - 6] + f(i - 5, i) \leq dp[i - 3] + f(i - 2, i)$ 이 성립한다. $j \leq i - 6$ 인 경우도 이와 비슷하게 증명 가능하다.

- ✓ 따라서, dp 계산 식을 다음과 같이 바꿔줄 수 있습니다.
  - $dp[i] : i$ 번 이하의 학생들로 팀들을 구성한 경우 팀 실력의 합의 최댓값
  - $dp[i] = max(dp[i], dp[j] + f(j+1, i)); B_j \geq B_i - K; i-5 \leq j \leq i-3;$
  - $f(l, r)$: 구간 $[l, r]$ 에서 실력이 가장 높은 세 학생의 실력의 합
- ✓ $f(l, r)$ 의 각 값은 $\mathcal{O}(1)$ 로 구할 수 있으므로, 총 시간복잡도 $\mathcal{O}(N)$ 으로 문제를 해결할 수 있습니다.

J. Double Up 2

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

- 제출 25번, 정답 2번, 정답률 8.000%
- First Solve: **algoishard**, 125분
- 출제자: 박신욱`bnb2011`

J. Double Up 2

- 0 이상 M 미만의 모든 정수 n에 대해, n에서 $2n \bmod M$으로 가는 유향 간선을 그어줍니다.
- 이 그래프처럼 모든 정점에서 하나의 나가는 간선을 가진 그래프를 **함수형 그래프**라고 합니다.
- 함수형 그래프에 존재하는 사이클들을 각각 하나의 거대한 정점처럼 생각하면, 이 문제는 트리 DP와 유사한 방식으로 해결할 수 있음을 짐작 가능합니다.

J. Double Up 2

✓ 가장 많이 등장하는 수가 사이클에 속해있는지 아닌지에 따라 문제를 나눠서 해결합니다.

1. 가장 많이 등장하는 수가 사이클에 속해있지 않음
   - dp_n : (가능한 모든 수들을 n로 만들었을 때의 개수, 그때의 연산 시행 횟수)라고 합시다.
   - 초기의 dp 값은 $dp_n = (A$에서 n의 등장 횟수, 0)입니다.
   - 그래프의 리프 정점부터 시작해, 위상 정렬을 하듯이 dp값을 갱신해 나갑니다.
   - 전이는 $dp_n = (a, b), dp_{2n \bmod M} = (c, d)$ 라고 했을 때 $dp_{2n \bmod M} = (a + c, b + d + a)$과 같이 진행합니다.
   - 사이클에 속해있지 않은 모든 정점에 대해 모든 dp 값을 계산했을 때, dp의 첫 번째 값이 최대이면서 두 번째 값이 최소인 경우가 답의 후보가 될 수 있습니다.

J. Double Up 2

2. 가장 많이 등장하는 수가 사이클에 속해있음
   - 사이클에 속해있지 않은 수들을 최소한의 연산 횟수를 사용해 모두 사이클 안으로 집어넣고 시작합니다.
   - 각 사이클마다 사이클 위의 한 점을 고정하고, 그 점으로 사이클의 모든 수를 옮기기 위해서 몇 번의 연산이 필요한지를 선형 시간에 구할 수 있습니다.
   - 고정한 점 이외에 나머지 값으로 수를 모으는 횟수는 다음과 같이 구할 수 있습니다. n 으로 가능한 모든 수를 모으기 위해 필요한 연산의 수를 k 라고 하면, $2n \bmod M$ 으로 가능한 모든 수들을 모으기 위해 필요한 연산의 수는 $k+$ (해당 사이클에 속한 수의 개수) $-$ (원래 $2n \bmod M$ 에 있었던 수의 개수) $\times$ (사이클의 길이)와 같이 계산 가능합니다.

✓ 총 시간복잡도는 $\mathcal{O}(N+M)$ 입니다.