1. 감소하는 성장률의 비극 시리즈

https://www.acmicpc.net/problem/28684 (감소하는 성장률의 비극 1)

https://www.acmicpc.net/problem/28685 (감소하는 성장률의 비극 2)

https://www.acmicpc.net/problem/28686 (감소하는 성장률의 비극 3)

C. 감소하는 성장률의 비극 1

`Math, Greedy, Stack, Geometry, [optional : Line Segment Intersection, Convex Hull]`
출제진 의도 – **Hard**

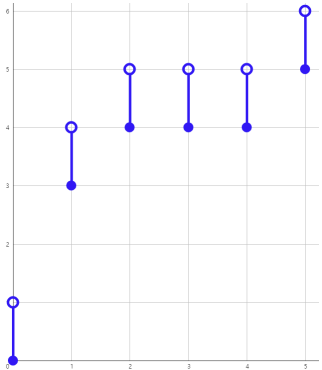
- ✓ Solve 정보
  - First Solve: **dontsaymyid**, 45분 6초
  - **맞았습니다!!** 비율: **8.772%** (10/114)
- ✓ 출제자: 주하늘, 김동우, 이종우
- ✓ 에디토리얼 작성자: 김동우

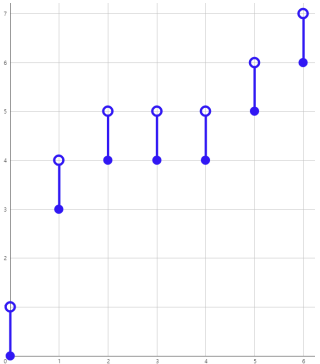
C. 감소하는 성장률의 비극 1

- 날짜가 지날수록 실제 전일 대비 성장 값이 단조 감소해야 한다.
- 이를 다르게 생각하면 x 축을 날짜 축, y 축을 실제 키 축으로 두어 점을 찍어 사이를 선분으로 그릴 때, x 좌표가 커지면 기울기가 작아지거나 유지되어야 한다.
- i 번째 날에 유림이가 공책에 쓴 값이 a_i 라고 하면, 실제 가능한 구간은 $[a_i - 0.5, a_i + 0.5)$ 이다.
- 여기서 그래프를 평행 이동해도 답에는 변화가 없으므로 처음 구간을 $x = 0, y \in [0, 1)$ 로 하자.
- 그러면 입력되는 값이 $a_1, a_2, \cdots a_n$ 이라 하고, $S_i = \sum_{j=1}^{i} a_i$ 라 하면, $x = i$ 일때 구간은 $y \in [S_i, S_i + 1)$ 이다.

C. 감소하는 성장률의 비극 1

✓ 즉, 입력으로 “3 1 0 0 1”이 들어오면 왼쪽, “3 1 0 0 1 1”이 들어오면 오른쪽이다.



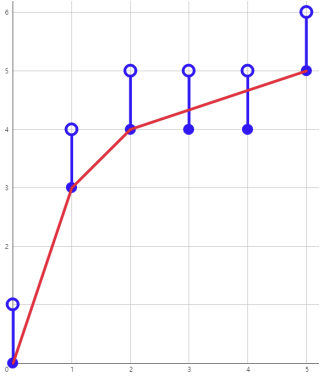


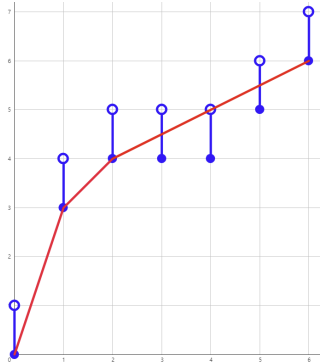
C. 감소하는 성장률의 비극 1

- ✓ 결론적으로 여기서 모든 선분을 통과하는 오목함수를 찾는 것이고, 이는 선분의 아래 점들로 이루어진 Monotone Chain(Convex Hull의 절반)을 찾는 것과 같다.
- ✓ 물론 Convex Hull 알고리즘을 바로 적용해도 되나, x 좌표가 증가하는 방향으로 되어 있고, x 가 증가함에 따라 y 도 단조 증가하므로, 답을 다양한 방식을 통해 구할 수 있다.
- ✓ x 를 증가 시키며 스택에 점을 삽입하며 단조 감소를 유지시키는 방식(LIS 혹은 히스토그램 문제와 유사하지만 점의 기울기의 업데이트가 필요함)으로 구할 수 있고,
- ✓ 혹은 아래의 점에 Convex Hull 알고리즘을 적용시켜 구할 수 있다
- ✓ 출제자는 Monotone Chain에서 한쪽 껍질만 구하는 방식으로 Main AC 코드를 작성하였다
- ✓ 즉, Convex Hull보다 쉬운, 그에 포함되는 방식으로 구할 수 있다.

C. 감소하는 성장률의 비극 1

✓ 즉, “310001”은 왼쪽, “310011”은 오른쪽이 그려진다..





C. 감소하는 성장률의 비극 1

- 이렇게 Monotone Chain을 만들었다면, 그 Monotone Chain이 기존 $N+1$개의 선분 구간을 지나는지 확인해준다.
- 직선의 방정식 혹은 선분교차 등으로 확인할 수 있다.
- 이때, $N+1$개의 선분 구간의 아래점은 포함되지만, 위의 점은 아니라는 것을 주의한다.
- 만약 모든 선분 구간과의 교점이 생기면, 해당 Monotone Chain이 예시 이므로 “Not Provable”, 교점이 안 생기는 선분 구간이 존재하면 “Provable”을 출력한다.
- 직전 슬라이드의 왼쪽은 모든 구간 교점이 생기지만, 오른쪽은 $x=4$일때 상한은 지나지만 교점이 생기지 않는다.
- Monotone Chain 등의 방식을 통해 구하는 과정에서 $\mathcal{O}(N)$이 소요되며, 교차 판정도 $\mathcal{O}(N)$이 소요된다.

F. 감소하는 성장률의 비극 2

Solution 1 : (Tags of "감소하는 성장률의 비극 1"), Two Pointer, Brute force
Solution 2 : Math, Number Theory, Continued Fractions, Geometry, Ad-hoc, Case work
출제진 의도 – **Expert**

- ✓ Solve 정보
  - First Solve: **cki86201**, 136분 48초
  - **맞았습니다!!** 비율: **22.222%** (4/18)
- ✓ 출제자: 주하늘, 김동우, 이종우
- ✓ 에디토리얼 작성자: 김동우(solution 1), 주하늘(solution 2)

F. 감소하는 성장률의 비극 2 - solution 1

Math, Greedy, Stack, Geometry, Two Pointer, Brute force, [optional : Line Segment Intersection, Convex Hull]

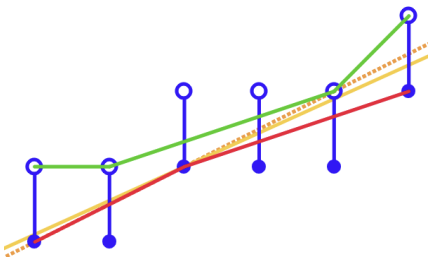
- **Solution 1은 “감소하는 성장률의 비극 1”의 풀이는 모두 했다고 생각하고 진행합니다.**

F. 감소하는 성장률의 비극 2 - solution 1

- “감소하는 성장률의 비극 1”는 단조 감소하는 직선 중 어느 것을 찾아도 되지만, 이번에는 그 중 직선, 그 중에서도 기울기가 가장 큰 것을 찾아야 한다.
- 기존에 아래 점들에 대한 Monotone Chain을 찾았다면, 위쪽 점들에 대한 볼록함수인 Monotone Chain을 찾으면 된다.
- “0 1 0 0 1”의 입력을 보자.

F. 감소하는 성장률의 비극 2 - solution 1

- ✓ 아래의 빨간색, 초록색과 같이 Monotone Chain을 그렸다면, 그 사이를 지나는 최대 기울기의 직선을 찾아야 한다.
- ✓ 아래에서는 노란색실선을 최대한 돌려 주황색점선을 만드는 것이 상한일 것이다.
- ✓ 결국 상한은 빨간색, 초록색 들의 점 중 하나씩을 접한다는 것을 알 수 있다.



F. 감소하는 성장률의 비극 2 - solution 1

- ✓ 먼저 브루트포스를 하게 되면 Monotone Chain의 최대 점의 개수는 Number of points on Convex hull with lattice points에 따라 $N^{2/3}$ 임을 알 수 있다. 즉, 각각의 점의 개수가 $N^{2/3}$ 을 넘지 않고, 가능한 모든 쌍을 조합해 검사하면 $\mathcal{O}\left(N^{4/3}\right)$ 에 해결가능하다. 즉, 점을 $\mathcal{O}(N)$ 에 찾고, $\mathcal{O}\left(N^{4/3}\right)$ 에 문제를 풀 수 있다.
- ✓ 두 번째로 투포인터 방법은 위쪽 Monotone Chain의 가장 오른쪽 점(a)과 아래쪽 Monotone Chain의 가장 왼쪽 점(b)을 시작점으로 하여 선분을 그렸을 때, 위쪽 Monotone Chain과 만난다면 점(a)를 위쪽 Monotone Chain 바로 직전의 점으로 업데이트 하고, 반대로 아래쪽 Monotone Chain 바로 다음 점으로 점(b)를 업데이트한다, 이렇게 하다가 성공하는 시점을 출력하면 된다. 이렇게 풀게되면 Monotone Chain을 $\mathcal{O}(N)$ 에 찾고, $\mathcal{O}(N)$ 에 해결 가능하므로 총 $\mathcal{O}(N)$ 에 해결 가능하다.

F. 감소하는 성장률의 비극 2 - solution 2

Math, Number Theory, Continued Fractions, Geometry, Ad-hoc, Case work

- **Solution 2는 “감소하는 성장률의 비극 3” 문제 풀이의 강한 스포일러가 될 수 있습니다. 열람 전에 유의하시기 바랍니다.**
- 해당 솔루션은 추후 공개하도록 하겠습니다.

I. 감소하는 성장률의 비극 3

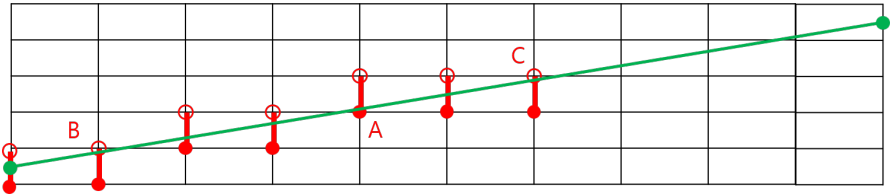
`Math`, `Number Theory`, `Stern-Brocot Tree`, `DP`, `Geometry`, `Ad-hoc`
출제진 의도 – **Challenging**

- ✓ Solve 정보
  - First Solve: **해당 없음**
  - **맞았습니다!!** 비율: **해당 없음** (0/0)
- ✓ 출제자: 주하늘
- ✓ 에디토리얼 작성자: 주하늘

I. 감소하는 성장률의 비극 3

- ✓ 문제를 간략화하기 위하여, 원래는 키를 반올림하여 측정해야 하지만, 정수 m에 대해 $m \times 10^{-3}\text{cm}$ 꼴로 표현되는 가장 가까운 키로 버림하여 측정하는 것으로 바꾸어도 정답은 변하지 않음이 설명된다.
- ✓ 성장률이 기약분수 $\frac{q}{p}(\times 10^{-3}\text{cm/일})$로 표현되는 상수 성장률 성장이 설명하는 전일 대비 성장 데이터는 주기 p를 가지게 된다. 성장률이 $\frac{q}{p}$로 일정하더라도 키의 초깃값이 달라짐에 따라 데이터도 변화하는데, 성장률이 일정하다면 한 순환마디를 '돌려' 다른 순환마디를 만들 수 있음을 보일 수 있다.
- ✓ 다음 예를 관찰하자. $y = 0.4x + 0.1$과 $y = 0.4x + 0.8$의 $x = 0,\ 1,\ 2,\ 3,\ 4,\ 5$에서의 함숫값을 정수로 버림하면 각각 $0,\ 0,\ 0,\ 1,\ 1,\ 2$, $0,\ 1,\ 1,\ 2,\ 2,\ 2$이다. 이 두 수열의 계차를 구하면 $\frac{q}{p} = \frac{2}{5}$에 대한 서로 다른 두 순환마디 $[0,\ 0,\ 1,\ 0,\ 1]$, $[1,\ 0,\ 1,\ 0,\ 0]$을 얻는다. 첫 순환마디를 왼쪽으로 두 칸 '돌리면' 두 번째 순환마디가 될 것이다.

- 한편 어떤 길이 N의 데이터가 한 상수 성장률 성장으로 설명된다고 할 때, 이는 $\frac{q}{p}(p \leq N)$ 꼴 성장률의 상수 성장률 성장으로도 설명될 수 있다. 이를 아래와 같이 증명하자.



- 위의 도식을 참조하라. 이는 길이 6의 데이터 $0,\ 1,\ 0,\ 1,\ 0,\ 0$을 설명하는 기울기 $\frac{4}{9}$의 상수 성장률 성장을 표현한 것이다.

I. 감소하는 성장률의 비극 3

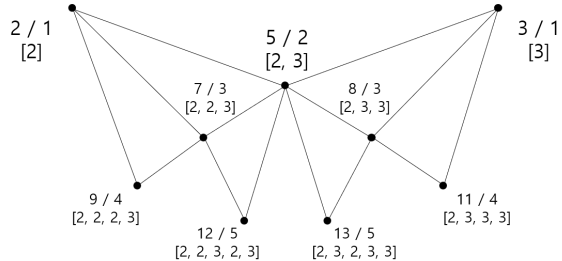
- ✓ 같은 데이터를 설명하는 다른 상수 성장률 성장을 찾기 위해, 기울기를 유지한 채 직선이 어느 구간의 아래끝 A에 닿을 때까지 평행하게 내린다. 이 직선은 여전히 데이터를 설명한다.
- ✓ 직선을 A를 중심으로 시계방향으로 회전시킨다. A보다 x좌표가 작은 어떤 구간의 위끝에 닿거나, A보다 x좌표가 큰 어떤 구간의 아래끝에 닿을 때까지 진행한다. 만약 다른 구간의 아래끝에 닿았을 경우, 이 직선은 여전히 데이터를 설명하며, 또한 x좌표가 최대 N만큼 차이 나는 두 정수점을 지나고 있기에 $\frac{q}{p}(p \leq N)$ 꼴 기울기를 가진다. 그렇지 않은 경우, 찾은 점을 B라고 하자.
- ✓ 반시계방향으로 같은 작업을 반복한다. 만일 A보다 x좌표가 큰 어떤 구간의 위끝에 처음으로 닿았을 경우, 그 점을 C라고 하자(한 아래끝과 다른 위끝에 동시에 닿은 경우에는 위끝을 생각한다).
- ✓ 이제 $\overleftrightarrow{BC}$와 평행하고 A를 지나는 직선은 기울기가 $\overleftrightarrow{BA}$의 기울기 초과 $\overleftrightarrow{AC}$의 기울기 미만이 되므로, B와 C의 construction에 의해 이 직선은 데이터를 설명한다.

I. 감소하는 성장률의 비극 3

- ✓ 이외에도 성장률이 정수 n과 $n+1$ 사이가 되는 상수 성장률 성장이 설명하는 데이터에는 오직 n과 $n+1$만이 등장할 수 있음을 관찰할 수 있다.
- ✓ 이제 문제는 다음과 같이 요약된다: 0 이상 9 이하이며 기약분수로 표현했을 때 분모가 N 이하인 모든 유리수 성장률에 대하여, 해당 성장률의 순환마디를 적당히 '돌리고' 무한 번 반복한 후 주어진 데이터와 비교하여 LCS를 구할 때, $N - LCS$ 길이 의 최솟값을 구하여 출력하고, 해당 최솟값의 argmin이 되는 성장에 대해 데이터에서 LCS에 포함되지 않는 값들을 전부 출력하면 된다.
- ✓ Naive하게 생각하면, 따져야 할 성장률은 $\mathcal{O}\left(N^2\right)$ 개이며, 길이 p의 순환마디를 돌리는 방법이 $\mathcal{O}\left(p\right)$ 가지, 다시 데이터와 LCS를 구하기 위해 각 경우마다 $\mathcal{O}\left(N\right)$의 연산이 소요된다. 분모가 N 이하인 기약분수 하나를 뽑을 때 그 분모의 기댓값이 $\frac{N}{2}$ 정도 또는 그 이상이라고 어림잡으면, 결국 이는 $\mathcal{O}\left(N^4\right)$에 동작할 것이라고 볼 수 있다.
- ✓ $N \leq 500$임을 감안하면 이는 부적절하다.

I. 감소하는 성장률의 비극 3

✓ 성장률의 순환마디가 갖는 규칙성을 부가적으로 파악해야 해결할 수 있다. 이는 다음과 같이 표현된다.



✓ 위는 2와 3 사이의 유리수들을 포함하는 Stern-Brocot tree를 다른 형태로 그린 것이다(물론 일반적으로 정수 n과 $n+1$ 사이의 유리수들을 포함하도록 그릴 수 있다). 그래프의 간선마다 위쪽 끝점이 아래쪽 끝점의 부모 노드가 된다고 하면, 제일 위의 두 노드 $2/1, 3/1$을 제외한 노드들은 각각 왼쪽 부모와 오른쪽 부모 노드를 갖는다.

I. 감소하는 성장률의 비극 3

- ✓ 먼저 정수 q에 대해 성장률 $q/1$에 대응하는 노드의 순환마디를 $[q]$로 저장해 둘 수 있다.
- ✓ 한 노드 q/p의 왼쪽 부모 노드 ql/pl 및 오른쪽 부모 노드 qr/pr에 대하여, $p = pl + pr,\ q = ql + qr$이 성립한다. 이와 함께 두 부모 노드의 순환마디가 저장되어 있을 때, 다음이 성립한다:
- ✓ $[$**순환마디 규칙**$]$ ql/pl의 저장된 순환마디를 왼쪽, qr/pr의 저장된 순환마디를 오른쪽에 두고 이어붙여 만든 길이 $pl + pr = p$의 열은 q/p의 순환마디가 된다.
- ✓ 이에 따르면, 깊이 우선 탐색과 유사한 과정을 통해 재귀적으로 모든 유리수 성장률에 대한 순환마디를 구할 수 있게 된다. (모든 기약분수 성장률이 정확히 한 번씩 탐색되게 된다.)
- ✓ 순환마디 규칙은 Farey sequence 내지는 Stern-Brocot tree의 특징적인 성질로부터 얻을 수 있다. 위의 그래프에 맞추어 이 성질을 서술하면 다음과 같다 : 한 노드 q/p와 그 부모 노드 s/r에 대하여, $|ps - qr| = 1$이 성립한다.

I. 감소하는 성장률의 비극 3

- ✓ 노드 q/p와 그 왼쪽, 오른쪽 부모 노드들 ql/pl, qr/pr이 주어진다고 하자. 좌표평면 위에 세 점 $O(0,\,0),\ A(pl,\,ql),\ B(p,\,q)$를 그리고 세 점을 꼭짓점으로 하는 삼각형을 그려 보자. $\overleftrightarrow{OA}$의 기울기는 ql/pl이고, $\overleftrightarrow{AB}$의 기울기는 이보다 큰 qr/pr임에 유의하라.
- ✓ 위 Stern-Brocot tree의 성질에 의하여 $\triangle OAB$의 넓이는 $\frac{1}{2}$이다. 여기서 픽의 정리를 적용해 보자. $q/p,\ ql/pl,\ qr/pr$이 전부 기약분수이므로 삼각형의 변에 놓인 점 개수는 3개이고, 따라서 삼각형의 넓이는 삼각형 내부의 점 개수 $+\,\frac{3}{2}\,-\,1$로도 표현된다.
- ✓ 이 둘을 동일시하면, $\triangle OAB$ 내부에는 격자점이 놓이지 않음을 보일 수 있다. 이제 두 곧은 경로 $O \to B$, $O \to A \to B$를 실제 성장을 나타내는 시간–키 함수의 그래프로 보고 이 두 성장이 설명하는 데이터를 추출하면, 두 경로 사이에 격자점이 놓이지 않는 상황에서 각 정수 x 좌표마다의 두 함숫값을 가까운 정수로 버림하고 있으므로 두 데이터는 온전히 일치한다.

I. 감소하는 성장률의 비극 3

- ✓ 따라서 키의 초깃값이 정수이고 성장률이 q/p인 상수 성장률 성장이 설명하는 데이터의 앞 p개의 항은, 키의 초깃값이 정수이고 성장률이 각각 $ql/pl,\ qr/pr$인 상수 성장률 성장이 설명하는 데이터들의 앞 $pl,\ pr$개의 항들을 concatenate한 것과 일치한다.
- ✓ 이를 다르게 표현하면 곧 순환마디 규칙이 된다.

- ✓ 하지만, 우리는 각 q/p의 순환마디를 알아내야 할 뿐만 아니라, 그 순환마디를 '돌린' 것의 무한반복과 입력받은 데이터 사이의 LCS 길이를 구하고, 그렇게 얻은 모든 $N - LCS$ 길이 중 최소를 찾아야 한다.
- ✓ DP를 활용하여 해결한다. 두 부모 노드들에 대해 일정한 정보들이 저장되어 있으면, 그 자식 노드에 대해 같은 정보를 $\mathcal{O}(N)$에 저장할 수 있도록 할 것이다. 그 정보로부터 원하는 값을 $\mathcal{O}(N)$에 찾아낼 것이다.

I. 감소하는 성장률의 비극 3

- ✓ 성장률 q/p에 대하여, 다음의 2가지 정보가 배열로 저장될 것이다.
  1. 이 정보는 크기 p의 정보이다. 각 $i = 0, 1, \ldots, p-1$에 대하여, 순환마디의 오른쪽 i개 항만 취하여 부분열을 만들고, 이 열을 데이터의 초항부터와 비교하여 LCS를 구해 나간다. 만일 그러다가 데이터의 끝에 도달한다면, 그렇다는 사실과 함께(Boolean: true) 해당 부분열의 몇 개의 원소까지 데이터 안에 '집어넣을' 수 있는지가(integer) 저장될 것이다. 만일 해당 열 전부를 데이터 안에 '집어넣었지만' 데이터의 오른쪽 끝에 도달하지 못했다면, 그러지 못했다는 사실과 함께(Boolean: false) 부분열을 모두 '집어넣고' 나서 데이터의 몇 번째 항에 도달했는지가(integer) 저장될 것이다.

  2. 이 정보는 크기 n의 정보이다. 각 $i = 0, 1, \ldots, n-1$에 대하여, 이번엔 순환마디 전체를 데이터의 i번째 항부터와 비교하여 LCS를 구해 나간다. 저장되는 정보는 1번째 정보와 동일하다(Boolean, integer 하나씩).
- ✓ 부모 $ql/pl,\ qr/pr$의 정보들로부터 q/p에 대한 1번째 정보를 다음 페이지에서와 같이 $\mathcal{O}(N)$에 저장할 수 있다. 2번째 정보를 저장하는 방식도 유사하므로 이는 생략한다.

I. 감소하는 성장률의 비극 3

- ✓ q/p의 순환마디는 길이가 pl인 ql/pl의 순환마디와 길이가 pr인 qr/pr의 순환마디를 이어붙인 것과 같다. 따라서 $0 \le i < pr$에 대하여, q/p의 순환마디의 오른쪽 i개 항을 취하면 그것은 qr/pr의 순환마디의 오른쪽 i개 항과 같다. 즉, qr/pr의 1번째 정보의 i번째 값을 그대로 가져와 저장하면 된다.
- ✓ $i \ge pr$이라면, q/p의 순환마디의 오른쪽 i개 항은 ql/pl의 순환마디의 오른쪽 $(i-pr)$개 항과 qr/pr의 순환마디를 이어붙인 것과 같다. 즉, ql/pl의 1번째 정보의 $(i-pr)$번째 값을 가져온다. 그 Boolean 값이 false라면, 그 integer 값은 데이터 상의 항 번호가 될 것이다. qr/pr의 2번째 정보의 해당 '항 번호'번째 값을 다시 가져온다. 그 Boolean 값이 false라면, 그 integer 값을 저장하면 된다.
- ✓ i를 pr부터 하나씩 늘리다 보면 두 번째로 찾은 Boolean 값이 true가 될 수 있다. 그렇다면 그때부터는 데이터에 '집어넣은' 부분열의 원소 개수를 $(i-pr)$과 두 번째 integer 값의 합으로 구할 수 있다.
- ✓ i를 더 늘리다 보면 첫 번째로 찾은 Boolean 값조차 true가 될 수 있다. 그렇다면 그때부터는 데이터에 '집어넣은' 부분열의 원소 개수는 그냥 첫 번째 integer 값으로서 구할 수 있다.

I. 감소하는 성장률의 비극 3

- ✓ 이제 이렇게 저장되어 있는 q/p의 정보로부터, q/p 성장률의 순환마디를 '돌리고' 무한 번 반복한 후 주어진 데이터와 비교한 LCS 길이를 순환마디를 몇 번 '돌렸는지'에 따라 전부 구해 낼 것이다.
- ✓ 각 $i = n-1, \ldots, 0$에 대하여, '돌리지' 않은 성장률의 순환마디를 무한 번 반복한 후 주어진 데이터의 i번째 항부터와 비교한 LCS 길이를 순차적으로 구할 것이다.
- ✓ 이를 위하여, 주어진 i에 대해 2번째 정보의 i번째 값을 가져온다. 만약 그 Boolean 값이 true라면, 그 integer 값을 사용하면 된다. 만약 false라면, 순환마디를 데이터에 '집어넣은' 후의 데이터 상에서의 위치 j가 있으니, 그 j에 대해 구해 놓은 값에 순환마디 하나의 길이 p를 더한 값을 사용하면 된다.
- ✓ 이 값들을 모두 구했다면, 이로부터 유사한 방식으로 각 $i = 0, 1, \ldots, p-1$에 대하여, 순환마디를 i회 왼쪽으로 '돌리고' 무한 번 반복한 후 주어진 데이터의 초항부터와 비교한 LCS 길이를 얻을 수 있다(1번째 정보를 적절히 활용한다).
- ✓ 각 q/p에 대하여 위의 정보 저장, LCS 길이 도출 과정 전부가 $\mathcal{O}(N)$에 동작한다.

- ✓ 초기에 데이터를 입력받으면, 각 $n = 0, 1, \ldots, 8$에 대하여 n과 $n+1$만 남긴 sub-data를 형성한 뒤, n과 $n+1$ 사이의 분모가 length(sub-data) 이하인 성장률들에 대해 위의 DP 과정을 거친다.
- ✓ 이로부터 고려해야 할 모든 성장에 대한 $N - LCS$ 길이의 최솟값을 찾고, 그 argmin이 되는 성장을 같이 찾는다. 이 과정은 결국 $\mathcal{O}\left(N^2\right)$ 개의 성장률마다 $\mathcal{O}\left(N\right)$회, 총 $\mathcal{O}\left(N^3\right)$ 회의 연산 내에 완수된다. ($0, 1, \ldots, 9$의 정수 성장률들도 처리해야 함에 유의하라.)
- ✓ 최솟값을 출력하고, 해당 성장으로부터 데이터를 복원하여 주어진 데이터와 비교한 LCS를 구한다. 주어진 데이터에서 LCS에 포함되지 않은 항들을 제거(즉, 출력)하고 나면, 데이터의 남은 부분은 상수 성장률 성장으로 설명될 것이다.

2. 다포체수

https://www.acmicpc.net/problem/28689

G. 다포체수

`Math`, `Number Theory`, `Polynomial Basis`, `Interpolation`, `Combinatorics`, `Ad-hoc`

출제진 의도 – **Expert**

- ✓ Solve 정보
  - First Solve: **flappybird**, 129분 58초
  - **맞았습니다!!** 비율: **37.500%** (3/8)
- ✓ 출제자: 주하늘
- ✓ 에디토리얼 작성자: 주하늘

G. 다포체수

- 먼저 문제를 요약하면, 다항식을 다른 다항식으로 보내는 일종의 연산자 S를 $[S(q)](n) := \sum_{k=1}^{n} q(k)$로 정의할 때, $[S^K(q)](n)$의 계수들을 P로 나눈 나머지를 구하는 문제로 생각할 수 있다.

  풀이 1.

- 다음의 식을 관찰하자: $\sum_{k=1}^{n} k(k-1)\cdots(k-s+1) = \frac{1}{s+1}(n+1)n(n-1)\cdots(n-s+1)$ ($s \geq 1$). 이에 대한 증명은 우변에 n을 대입한 것에서 $(n-1)$을 대입한 것을 빼 보면 된다.
- 이를 활용하기 위하여, polynomial basis $\{1,\ n,\ n(n-1),\ \ldots,\ n(n-1)\cdots(n-d+1)\}$를 구성한 뒤 주어진 다항식을 이들의 선형결합으로 우선 분해한다. 그러면 $q(n) = c_0 + c_1 n + c_2 n(n-1) + \cdots + c_d n(n-1)\cdots(n-d+1)$의 형태가 만들어진다.

G. 다포체수

- ✓ 위의 공식을 활용하면, 상수항을 제외한 각 항에 대해 다음이 성립함을 확인할 수 있다:

$$[S^K(c_i n(n-1)\cdots(n-i+1))](n)\,(i \geq 1) = c_i \cdot i! \binom{n+K}{i+K} =$$
$$c_i \binom{i+K}{i}^{-1} \cdot \left[\frac{(n+1)(n+2)\cdots(n+K)}{K!}\right].$$

- ✓ 여기서 두 번째 변의 이항계수 $\binom{n+K}{i+K}$는 symbolic expression이다.
- ✓ 참고 : Binomial coefficients as polynomials
- ✓ 상수항에 대해서는 공식이 약간 바뀌어서($\sum_{k=1}^{n} 1$은 $(n+1)$이 아니라 n이 되기 때문), 위의 공식을 $i = 0$에 대해서까지 적용하여 전부 더한 뒤 $c_0 \binom{n+K-1}{K-1}$을 빼 주어야 정확한 답이 된다.

G. 다포체수

- ✓ 두 번째 변까지만을 활용하여 $[S^K(q)](n) = \left[\sum_{i=0}^{d} c_i \cdot i! \binom{n+K}{i+K}\right] - c_0 \binom{n+K-1}{K-1}$ 을 계산해도 되고(이때 각 i에 대하여 $\binom{n+K}{i+K}$를 순차적으로 구할 수 있다), 또는 세 번째 변까지 구하여 c_i들을 $c_i \binom{i+K}{i}^{-1}$로 대체한 다항식을 새로 구한 뒤, 거기에 $\left[\frac{(n+1)(n+2)\cdots(n+K)}{K!}\right]$라는 다항식을 곱해 주어도 된다(여전히 상수항 처리는 해야 한다).
- ✓ 시간복잡도는 c_i들을 구하기 위해 $\mathcal{O}\left(d^2\right)$, K 차 다항식을 만들어내기 위해 $\mathcal{O}\left(K^2\right)$, 다항식을 곱하기 위해 $\mathcal{O}\left(dK\right)$ 정도가 각각 소요된다.

풀이 2.

- ✓ $\sum_{k=1}^{n} q(k)$ 를 풀어서 $q(1)+q(2)+\cdots+q(k)$ 로 쓰고, $\sum_{k=1}^{n}\sum_{j=1}^{k} q(j)$ 를 풀어서 $(q(1))+(q(1)+q(2))+\cdots+(q(1)+q(2)+\cdots+q(n))$ 으로 쓰는 식으로 다중 합의 모든 항을 열거했을 때, 각 $q(k),\ k=1,\ 2,\ \ldots,\ n$ 이 몇 번씩 등장하는지를 셀 수 있다.
- ✓ 특히 n 에서 시작하여 k 로 끝나는 길이 K 의 단조감소수열 하나마다 $q(k)$ 항 하나가 대응되는데, 이를테면 $K=3,\ n=3$ 일 때 수열 $(3,\ 2,\ 2)$ 에 대응하는 $q(2)$ 항은 $[\{q(1)\}]+[\{q(1)\}+\{q(1)+q(2)\}]+[\{q(1)\}+\{q(1)+\underline{q(2)}\}+\{q(1)+q(2)+q(3)\}]$, 즉 3번째 대괄호 안의 2번째 중괄호 안의 2번째 항이다.

G. 다포체수

- 따라서 $[S^K(q)](n) = \sum_{k=1}^{n} c_k q(k)$ (c_k 는 주어진 $(n-k+1)$ 개의 수로 이루어지는 길이 $K-1$ 의 단조감소수열의 개수)임을 얻는다. c_k 는 다시 중복조합 ${}_{n-k+1}H_{K-1} = \binom{n+K-k+1}{K-1}$ 으로 얻어진다.
- 결국 다음과 같은 풀이가 성립한다. $q(n)$ 에 $1 \sim (d+K)$ 를 대입하여 함숫값들을 얻는다. 이 값들로부터 위 공식을 써서 $n = 0, 1, \ldots, (d+K)$ 에 대한 $[S^K(q)](n)$ 을 구할 수 있다(이항계수를 빠르게 구하기 위해 계승의 모듈러 값들을 전처리해 둔다). 여기까지의 시간복잡도는 $\mathcal{O}\left((d+K)^2\right)$ 이고, 이제 이 $(d+K+1)$ 개의 함숫값으로부터 interpolation을 통해 다항식 $S^K(q)$ 를 복원하면 된다.
- 막바지의 interpolation은 구해 둔 함숫값들이 특수하기에($x = 0, 1, \ldots, d+K$ 에 대한 함숫값들이므로) Newton's interpolation을 적용하기 좋고, $\mathcal{O}\left((d+K)^2\right)$ 에 완수할 수 있다.
- 참고 : Newton Polynomial Application

3. 1차원 돌 게임 시리즈

https://www.acmicpc.net/problem/31541 (1)

https://www.acmicpc.net/problem/31542 (2)

H1. 1차원 돌 게임 1

Game Theory
출제진 의도 – **Hard**

- ✓ 본 대회
  - First Solve: **주마늘**, π 분 e 초
  - **맞았습니다!!** 비율: **NaN%** (1/0)
- ✓ Open Contest
  - First Solve: **주하늘**, 299 분 59 초
  - **맞았습니다!!** 비율: **100%** (1/1)
- ✓ 출제자: 주하늘
- ✓ 에디토리얼 작성자: 주하늘

H1. 1차원 돌 게임 1

- ✓ Subtask 1에서의 게임은 아무리 길어도 6턴 안에는 종료되므로 편하게 구현할 수 있다.
- ✓ $N = 4, 5, 6, 7$일 때에만 선공이 패배하게 된다.

- ✓ $b(n, k)$ $(n \geq 0, k \geq 1)$를 첫 턴에 선공이 k개를 가져가는 동일한 게임에서 선공의 승리 여부로 두고 $b(n, 1)$을 결정하면 충분하다. 게임 내에서 k가 계속 변화하므로, 다른 k들에 대해서도 이 승리 여부 값을 조사해 주어야 $b(n, 1)$도 구할 수 있는 것이다.
- ✓ Subtask 2에서는 이 $b(n, k)$를 $n \leq 1000, k \leq 45$까지 구하면 충분하다. $1 + 2 + \cdots + 45 > 1000$이기 때문에, $n \leq 1000$의 범위에서는 $k \geq 46$인 상태에서는 반드시 선공이 패배하는 것으로 둔다.
- ✓ 연속한 n개의 돌에서 k개를 가져감으로써 도달할 수 있는 모든 남은 (연속한) 돌 개수 n'에 대해, $b(n', k + 1)$ 중 하나라도 거짓(후공 승)이 있으면, $b(n, k)$는 참(선공 승)으로 결정되며, 아닌 경우 $b(n, k)$는 거짓(후공 승)으로 결정된다.

H1. 1차원 돌 게임 1

- ✓ n과 k에 따른 n'의 값으로 가능한 것들을 조사하면, 한 구석의 k개를 가져갈 때 $(n-k)$개가 남는 것이 최대이며, 가장 가운데의 k개를 가져가 두 부분의 돌의 개수를 최대한 비슷하게 맞추고 적은 쪽을 버리는 것이 최소이다. 이때는 $\lceil \frac{n-k}{2} \rceil$개가 남는다. 이를 식으로 정리하면:

$$b(n,\, k) = \sim \left[b\left(\left\lceil \frac{n-k}{2} \right\rceil ,\, k+1 \right) \wedge \cdots \wedge b(n-k,\, k+1) \right]. \quad (n \geq k) \quad \cdots (1)$$

  $n < k$인 경우에는 선공이 행동할 수 없으므로 선공의 패배이다.

- ✓ 이를 구현한 Subtask 2의 시간복잡도는 $\mathcal{O}\left(N^{\frac{5}{2}}\right)$이다. $N^{\frac{1}{2}}$는 k의 범위에 해당한다.

H1. 1차원 돌 게임 1

- ✓ 대부분의 경우에 $b(n,\ k) = b(n-1,\ k)$ 인 것을 관찰할 수 있다. 식 (1) 의 우변에서, n 을 1 증가시키면 $\lceil \frac{n-k}{2} \rceil$ 와 $(n-k)$ 는 최대 1 증가하므로, AND 를 취하는 대부분의 항은 그대로 남고 한두 개의 항이 추가되거나 제외될 뿐이기 때문이다.
- ✓ k 를 고정시키고 n 을 증가시키면서 관찰할 때, 새롭게 값이 거짓인 항 $b(n-k,\ k+1)$ 이 추가될 경우 한동안은 $b(n,\ k)$ 가 계속 참이 된다. 반대로 $b(\lceil \frac{n-k}{2} \rceil,\ k+1)$ 이 제외되면서 AND 를 취하는 항들이 모두 참이 될 경우 한동안은 $b(n,\ k)$ 가 계속 거짓이 되는 것이다.
- ✓ 따라서 $k \geq 1$ 과 각 $m \in \mathbb{N}$ 에 대해 $f_m(k),\ g_m(k)$ 를 다음과 같이 귀납적으로 정의한다:
  (i) $f_1(k) \equiv 0$ 이다.
  (ii) $m \geq 1$ 에 대해 $g_m(k) \equiv \min\{n \mid n > f_m(k),\ b(n,\ k) = TRUE\}$ (cf. $g_1(k) = k$.)
  (iii) $m \geq 2$ 에 대해 $f_m(k) \equiv \min\{n \mid n > g_{m-1}(k),\ b(n,\ k) = FALSE\}$

- ✓ 위 정의를 풀이하면, 고정된 k와 증가하는 n에 대해 $b(n,\ k)$가 거짓이 되는 m번째 구간의 시작이 $f_m(k)$이고, $b(n,\ k)$가 참이 되는 m번째 구간의 시작이 $g_m(k)$이다.
- ✓ 즉, $f_m(k)$부터 $g_m(k)-1$까지의 n에 대해서는 $b(n,\ k)=FALSE$가 되고, $g_m(k)$부터 $f_{m+1}(k)-1$까지의 n에 대해서는 $b(n,\ k)=TRUE$가 되도록 $f_m(k),\ g_m(k)$를 정의한다.

Thm. $f_2(k)=3k+1$.

- ✓ pf. k를 고정하자. $f_1(k)=0,\ g_1(k)=k$이므로, k보다 크면서 $b(n,\ k)=FALSE$가 되는 최소의 n이 $f_2(k)$일 것이다. 그런데 $b(0,\ k+1)=\cdots=b(k,\ k+1)=FALSE$이므로, Prop. 3에 의하여 $n\geq k$이고 $\lceil\frac{n-k}{2}\rceil\leq k$인 모든 n에 대하여 $b(n,\ k)=TRUE$임을 얻는다. 즉 임의의 k와 $k\leq n\leq 3k$인 모든 n에 대하여 $b(n,\ k)=TRUE$이다. 이는 모든 k에 대하여 성립한다.

- 다시 k를 고정하고, 위 결론을 $(k+1)$에 대해 적용할 수 있다. 이로부터 다시 $b(3k+1,\ k)$를 구하면 $= \sim [b\,(k+1,\ k+1) \wedge \cdots \wedge b(2k+1,\ k+1)] = FALSE$이다.
- 위 두 결론에 의하여 $f_2(k) = 3k+1$이다. ■

- 이와 같이 모든 $f_m(k)$를 귀납적으로 결정할 수 있으나, 위 증명에서와 같이 이전의 여러 결론들을 엮어 다음 결론을 만들어야 하는 형태이기 때문에 증명에 주의를 요한다.
- 예로, 한 m에 대해 다음을 순차적으로 증명하도록 하면 m에 대한 수학적 귀납법 증명이 완성된다.
  (i) 모든 k에 대해 $g_m(k)$는 잘 정의되고, $g_m(k) = f_m(k+1) + k$이다.
  (ii) 모든 k에 대해 $g_m(k) \le g_m(k+1)$이다.
  (iii) 모든 k에 대해 $f_{m+1}(k)$는 잘 정의되고, $f_{m+1}(k) = 2g_m(k+1) + k - 1$이다.

H1. 1차원 돌 게임 1

- ✓ 이에 따르면, 모든 $m \geq 1$에 대하여 $f_m(k),\ g_m(k)$는 k에 대한 일차함수임을 얻는다.
- ✓ 이제 $f_m(k) = a_m k + b_m,\ g_m(k) = c_m k + d_m$이라 두면 $a_1 = 0,\ b_1 = 0,\ a_{m+1} = 2c_m + 1,$ $b_{m+1} = 2c_m + 2d_m - 1,\ c_m = a_m + 1,\ d_m = a_m + b_m$이라고 하는 점화식을 얻는다.
- ✓ 이를 풀면 다음의 결과를 얻는다:
- ✓ $f_m(k) = (3 \cdot 2^{m-1} - 3)k + (6m - 17)2^{m-1} + 11,\ g_m(k) = (3 \cdot 2^{m-1} - 2)k + (3m - 7)2^m + 8.$
- ✓ $f_m(1) = (3m - 7)2^m + 8,\ g_m(1) = (6m - 11)2^{m-1} + 6.$

- ✓ 예상한 대로 n이 증가함에 따라 $b(n,\ 1)$이 바뀌는 지점은 드물게 존재한다. N의 제한에 맞춰 이 지점들을 배열에 저장하고, 입력에 주어지는 n마다 이 n이 어떤 두 지점 사이에 놓이는지를 찾아 답을 반환하면 된다. 구현의 효율성에 따라 Subtask 3 또는 Subtask 4를 맞힐 수 있다.
- ✓ 이분탐색을 활용하면 쿼리당 시간복잡도는 $\mathcal{O}(\log N)$이다.

H2. 1차원 돌 게임 2

Game Theory
출제진 의도 – **Hard**

- ✓ 본 대회
  - First Solve: **주마늘**, π 분 e 초
  - **맞았습니다!!** 비율: **NaN%** (1/0)
- ✓ Open Contest
  - First Solve: **주하늘**, 299 분 59 초
  - **맞았습니다!!** 비율: **100%** (1/1)
- ✓ 출제자: 주하늘
- ✓ 에디토리얼 작성자: 주하늘

H2. 1차원 돌 게임 2

- $b(n,\ k)\ (n \geq 0,\ k \geq 1)$를 첫 턴에 선공이 k개 이하를 가져갈 수 있는 동일한 게임에서 선공의 승리 여부로 둔다. 게임 내에서 k가 계속 변화하므로, 다른 k들에 대해서도 이 승리 여부 값을 조사해 주어야 $b(n,\ 1)$도 구할 수 있는 것이다.
- H1번 문제와 구분되는 점은, k가 클 때 선공이 할 수 있는 행동 집합이 k가 작을 때의 그것을 온전히 포함한다는 점이다. 즉, $b(n,\ k)$가 참이라면, 모든 $l \geq k$에 대해 $b(n,\ l)$ 역시 참이다.
- 따라서 $b(n,\ k)$를 전부 구하는 대신 $b(n,\ k)$가 참이 되는 최소의 k만을 저장하면 충분하다.
- 이를 $c(n)$이라 하자. $c(n) = 1$일 때 해당 n에서 선공이 승리한다고 결정하면 된다.

- Subtask 1의 $N \leq 20$은 직접 손으로 계산해 볼 수 있을 정도의 범위이기도 하다.
- $N = 2,\ 3,\ 8,\ 18,\ 19$일 때에만 선공이 패배한다.

H2. 1차원 돌 게임 2

- ✓ Subtask 2에서는 고정된 n에 대해, $k \leq 45$의 범위에서 ($1 + 2 + \cdots + 45 > 1000$이므로 이 위의 k 값에는 도달할 수 없다) $b(n,\ k)$를 조사하여 최초로 참이 되는 k의 값인 $c(n)$을 구한다.
- ✓ 연속한 n개의 돌에서 k개를 가져감으로써 도달할 수 있는 모든 남은 (연속한) 돌 개수 n'에 대해, $b(n',\ k+1)$ 중 하나라도 거짓(후공 승)이 있으면 모든 $r \geq k$에 대해 $b(n,\ r)$은 참(선공 승)이고 $c(n) = k$이다. 그렇지 않다면 $b(n,\ k)$는 거짓(후공 승)이다.

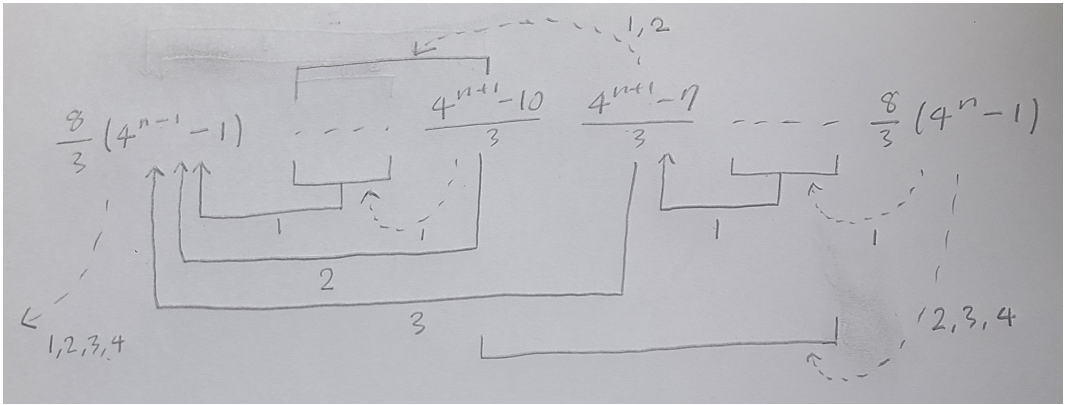
- ✓ 시간복잡도는 $\mathcal{O}\left(N^{\frac{5}{2}}\right)$이다. $N^{\frac{1}{2}}$는 k의 범위에 해당한다.
- ✓ 각 $b(n,\ k)$를 구할 때마다 $1 \leq l \leq k$개를 가져가는 모든 경우를 고려하여 상한 시간복잡도 $\mathcal{O}\left(N^{\frac{7}{2}}\right)$로 구현하더라도, $c(n) = 1$인 n이 대부분이기에 풀이가 충분히 통과할 수 있다. 통과하지 않더라도 런타임 전의 전처리를 통해 통과시킬 수 있을 것이다.

H2. 1차원 돌 게임 2

- ✓ Subtask 3의 풀이는 작은 n에 대한 결과의 규칙성을 찾아 전략을 explicit하게 표현하는 것이다.
- ✓ 연속한 n개의 돌에서 k개를 가져감으로써 n'개의 연속한 돌을 남길 수 있다면 이를 n에서 n'으로 향하는 화살표로 표현하고, k를 화살표 옆에 적는 표기법을 사용한다. 또 그것이 '이기는' 행동이라면 실선 화살표로, '지는' 행동이라면 점선 화살표로 표현한다. 선공과 후공 중 유리한 쪽은 항상 실선 화살표를 따라가는 것이 전략이고, 불리한 쪽에게는 가능한 모든 행동이 점선 화살표로 표시된다.
- ✓ $c(n) = t$일 때, n에서 나가는 화살표는 $1 \leq k < t$에 대응하는 점선 화살표들과 $k = t$에 대응하는 실선 화살표 하나이다. 그러나 유리한 쪽이 실선 화살표만 따라간다면 어떤 n에 대해서는 적당한 $s < t$에 대해 $k > s$인 경우는 발생하지 않을 수 있다. 이 경우는 $k > s$에 대응하는 화살표들은 생략한다.
- ✓ 구간 $[a, b]$의 모든 수에서 공통된 k에 대해 한 지점 n으로 향하는 실선 화살표가 있는 경우, 이를 $[a, b]$에서 n으로 향하는 실선 화살표 하나로 축약한다. 또 한 지점 n과 한 k에 대응하는 모든 점선 화살표의 도착점이 항상 구간 $[a, b]$ 안에 놓이는 경우 이를 n에서 $[a, b]$로 향하는 점선 화살표 하나로 축약한다.

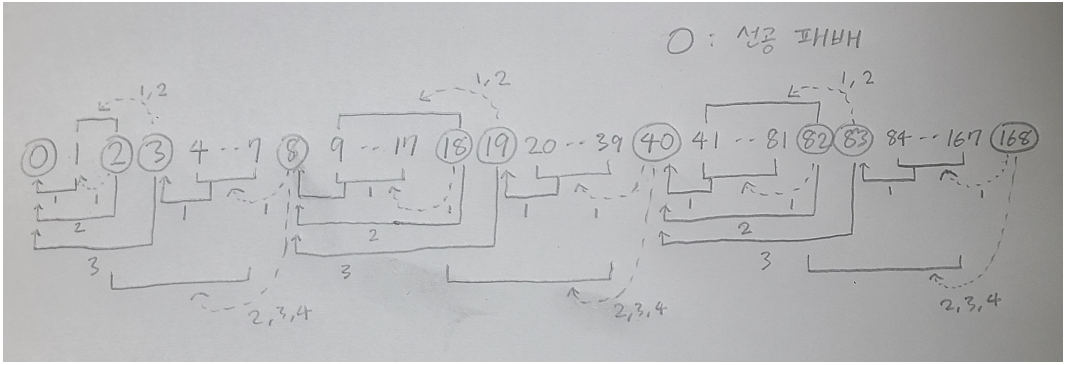
H2. 1차원 돌 게임 2

- 전략은 다음 그림과 같이 표현된다. ($n = 1, 2, 3, \ldots$)



H2. 1차원 돌 게임 2

- 이를 모든 n에 대해 겹쳐서 펼친 그림은 다음과 같다.



H2. 1차원 돌 게임 2

- 즉 오직 $\frac{8}{3}(4^{n-1}-1)$, $\frac{4^{n+1}-10}{3}$, $\frac{4^{n+1}-7}{3}$ 꼴의 지점에서 선공이 패배한다.
- 이러한 지점은 $\mathcal{O}(\log N)$ 개 발생하므로, 이를 계산하여 Subtask 3, 4를 해결할 수 있다.

4. 하늘아 군대 잘 가고(前 이등병의 편지)

https://www.acmicpc.net/problem/31548

M. 이등병의 편지

`Multivariable Polynomials`, `Generating Functions`, `Sum over Subsets DP`, `FWHT`
출제진 의도 – **Challenging**

- ✓ 본 대회
  - First Solve: **주마늘**, π 분 e 초
  - **맞았습니다!!** 비율: **NaN%** (1/0)
- ✓ Open Contest
  - First Solve: **주하늘**, 299 분 59 초
  - **맞았습니다!!** 비율: **100%** (1/1)
- ✓ 출제자: 주하늘
- ✓ 에디토리얼 작성자: 주하늘

M. 이등병의 편지

- ✓ Subtask 1에서는 가능한 모든 공연 순서를 검사하면 충분하다.
- ✓ 전등의 상태를 꺼짐 -0, 켜짐 -1, 점멸 -2의 수로 표현하고, 한 부원이 전등을 조작하는 방식 역시 조작하지 않음 -0, 'L' -1, 'R' -2의 수로 표현할 수 있다.
- ✓ 그러고 나면, 현재 전등의 상태를 나타내는 수에서 조작 방식을 나타내는 수를 더한 뒤 3으로 나눈 나머지를 취하면 조작 이후 바뀐 전등의 상태를 나타내는 수가 된다.
- ✓ M명의 부원들 중에서 중복을 허용하여 총 K명을 뽑아 공연 순서를 형성하는 방법의 수는 총 M^K 가지이다. 각 공연 순서에 대해, 모든 전등에 대해 이 합의 나머지를 계산하고, 그 값들이 전부 0이 되는 공연 순서의 가짓수를 세어 주면 된다. 백트래킹 알고리즘을 활용하여 공연 순서를 사전순으로 탐색하고, 합의 나머지를 계산할 때 직전 공연 순서와 달라진 부분만 갱신해 줌으로써 실행 시간을 개선할 수 있다.
- ✓ 시간복잡도는 $\mathcal{O}\left(NM^K\right)$이다.

M. 이등병의 편지

- ✓ Subtask 2는 공연 길이 K에 대한 DP를 통해 해결한다.
- ✓ 각 전등이 3가지 상태를 가질 수 있고, 그러한 전등이 N개가 있으므로 전등들의 상태 조합은 총 3^N 가지이다. 3진법의 비트마스크를 통해 이 상태 조합들을 $0 \sim (3^N - 1)$의 정수에 대응시킬 수 있다.
- ✓ 한편 부원 한 명이 전등들을 조작하는 방식들 역시 3진법의 비트마스크를 통해 $0 \sim (3^N - 1)$에 대응시킬 수 있다. 그렇게 하면 전등들의 초기 상태 조합 하나와 부원의 조작 방식 하나를 입력받아, 해당 부원이 한 번 조작한 이후 전등들의 바뀐 상태 조합을 출력하는 함수를 고려할 수 있다.
- ✓ 3진법으로 표현된 두 수를 비트마다 더하고 3으로 나눈 나머지를 취하면 되는 것이다.

M. 이등병의 편지

- ✓ 초기에 M 명의 부원들의 조작 방식을 입력받으면, 각 $0 \sim (3^N - 1)$ 번의 전등들의 초기 상태 조합 번호에 대해, 모든 부원들의 조작 방식에 대해 위 함수를 적용하여 한 명의 부원의 조작을 통해 도달할 수 있는 M 개의 상태 조합 번호들을 저장해 둔다. 이 과정은 $\mathcal{O}\left(3^N NM\right)$ 에 완수할 수 있다.
- ✓ 이제 t 초 길이의 공연 이후 상태 k($k = 0,\ 1,\ \ldots,\ 3^N - 1$)에 도달하는 공연 순서의 가짓수를 k 번째 성분으로 갖는 벡터 v_t($t = 0,\ 1,\ \ldots$)를 고려할 수 있다.
- ✓ 초기에 모든 전구는 꺼져 있으므로, v_0 은 0 번 성분만이 1 이고 나머지는 0 인 벡터로 둔다.
- ✓ v_{t-1} 로부터 위에 저장해 둔 '경우의 수 전이'를 통해 v_t 를 구할 수 있다. v_t 를 영벡터로 초기화하고, 한 명의 부원의 조작을 통해 i 번째 상태를 j 번째 상태로 만들 수 있을 때마다 v_t 의 j 번째 성분에 v_{t-1} 의 i 번째 성분을 더하는 것이다. 이를 K 번 반복하여 v_K 를 구하고, 0 번 성분을 추출하여 답을 얻을 수 있다.
- ✓ 이 과정의 시간복잡도는 $\mathcal{O}\left(3^N MK\right)$ 이고, 따라서 총 시간복잡도는 $\mathcal{O}\left(3^N M(N+K)\right)$ 가 된다.

M. 이등병의 편지

- ✓ Subtask 2의 풀이에서, M 명의 부원의 조작 방식을 입력받아 상태별 경우의 수를 $(t-1)$ 초에서 t 초로 전이시키는 정보를 저장하였다. 이것을 단순히 배열의 묶음으로 저장하는 대신, $3^N \times 3^N$ 의 행렬을 만들고 행렬곱을 활용하면 Subtask 3을 해결할 수 있다.
- ✓ $[A]_{ji} (0 \le i,\ j \le 3^N - 1)$ 를 i 번 상태에서 한 명의 부원의 조작을 통해 j 번 상태를 만드는 경우의 수로 두면 $v_t = Av_{t-1}$ 이 성립하게 된다. 따라서 $v_K = A^K v_0$ 을 계산하고 0 번 성분을 추출하면 된다. 이는 Exponentiation by Squaring을 통해 $\mathcal{O}\left(3^{3N} \lg K\right)$ 에 완수할 수 있다.
- ✓ 총 시간복잡도는 $\mathcal{O}\left(3^N NM + 3^{3N} \lg K\right)$ 이다. 행렬곱의 큰 시간복잡도를 희생하여 K 에 대한 로그 시간을 확보하는 풀이이다.

M. 이등병의 편지

- ✓ 구하는 경우의 수를 K에 대한 수열로 생각하면, 해당 수열은 일정한 형태의 선형 점화식을 따를 것으로 예상할 수 있다. 특히 전등들의 상태 조합이 총 3^N 가지이므로, K초 길이의 공연 이후 각 상태에 도달하는 공연 순서의 가짓수를 K에 대한 수열로 표현하면, 3^N 개의 수열이 생기며, 이들을 연립하여 일차 선형 점화식 계(system)를 얻을 수 있다.
- ✓ 따라서 구하는 경우의 수는 3^N 차의 선형 점화식을 따름을 알 수 있고, 이보다 더 작은 차수의 점화식의 존재성을 기대할 수 있다. 결국 Subtask 2의 방식으로 앞 몇 개의 항만을 계산하고, Berlekamp-Massey 알고리즘을 통해 최소 선형 점화식을 얻어 K째 항을 구한다는 풀이법을 얻는다.
- ✓ 최소 선형 점화식의 차수의 상한을 어떤 D로 잡으면, 적어도 $2D$개의 초기 항을 계산하여 투입해야 한다. 이는 Subtask 2의 방법으로 $\mathcal{O}\left(3^N M(N+2D)\right)$의 시간에 완수된다. 이후는 최소 선형 점화식과 K째 항을 계산하는 과정으로, 특히 키타마사 알고리즘을 쓰면 $\mathcal{O}\left(D^2 \lg K\right)$에는 수행할 수 있다.

M. 이등병의 편지

- D를 3^N 정도로 잡으면 Subtask 3을 맞히기엔 충분하지만, Subtask 4까지 통과되는 시간복잡도는 아니다. 결국 Subtask 4에서는는 D를 어디까지 줄일 수 있을지가 관건이 된다.
- 하지만 알고리즘의 특성상, 실전에서는 제한 시간을 초과하지 않으면서도 충분히 큰 D를 잡은 뒤 위의 풀이를 시도하는 것이 효율적인 접근법이 될 수 있다.
- 결론은, D를 $\mathcal{O}\left(M^2\right)$ 규모로 줄여 잡아도 충분하므로 이 풀이가 Subtask 4를 해결한다는 것이다.
- 한 증명은 다음과 같다.

M. 이등병의 편지

- Subtask 3에서 잡은 행렬 A를 고려해 보자. i가 3진법 수 $\overline{i_N i_{N-1} \cdots i_2 i_1}_{(3)}$으로 표현되고, a번 부원의 조작 방식이 $k^{(a)} = \overline{k_N k_{N-1} \cdots k_2 k_1}_{(3)}$으로 표현된다고 할 때, i번 상태에서 해당 부원의 조작을 통해 도달하는 상태 $j = \overline{j_N j_{N-1} \cdots j_2 j_1}_{(3)}$에 대해 $j_r \equiv i_r + k_r \mod 3$이다.
- $v_t = Av_{t-1}$의 관계가 성립하므로, v_t의 0번 성분을 모아 만든 수열은 A의 minimal polynomial을 characteristic equation으로 갖는 선형 점화식을 만족시키게 된다.
- A의 minimal polynomial의 차수의 상계를 탐색해 보자.

$$\begin{cases} N = 2,\ M = 3 \\ k^{(1)} = \overline{00}_{(3)} \\ k^{(2)} = \overline{21}_{(3)} \\ k^{(3)} = \overline{10}_{(3)} \end{cases} \implies A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

M. 이등병의 편지

- ✓ 두 3진법 수 $i,\ k$를 비트별로 더하고 3으로 나눈 나머지를 취하는 연산을 $i \oplus k$와 같이 쓰자.
- ✓ $3^N \times 3^N$ 행렬 C_k는 $M = 1,\ k^{(1)} = k$인 경우에 대한 A로 정의한다. 즉, $[C_k]_{ji} = \delta_{i(i\oplus k)}$ 이다.
- ✓ $A = \sum_{a=1}^{M} C_{k^{(a)}}$ 가 성립한다. 이제 이 연산 $\oplus$와 행렬 C_k 들에 대해 분석하자.
  - $\oplus$에 대한 교환법칙과 결합법칙이 성립한다.
  - $\forall k:\ \{i \oplus k:\ 0 \le i \le 3^N - 1\} = \{0,\ 1,\ \ldots,\ 3^N - 1\}$. 즉 C_k는 permutation matrix이다.
  - $1,\ 3,\ \ldots,\ 3^{N-1}$과 $\oplus$를 사용해 $0 \sim 3^N - 1$의 모든 수를 표현할 수 있다.
  - $[C_{k_1} C_{k_2}]_{ji} = \sum_{p=0}^{3^N} I[p = i \oplus k_1 \wedge j = p \oplus k_2] = I[j = i \oplus (k_1 \oplus k_2)] = [C_{k_1 \oplus k_2}]_{ji}$.

    즉, $C_{k_1} C_{k_2} = C_{k_1 \oplus k_2} = C_{k_2} C_{k_1}$ 이 성립한다.

- ✓ C_k 들끼리 commute하므로, C_k 들 중 대각화 가능한 것끼리는 simultaneously diagonalizable 하다. 즉, 공통된 eigenvector들로 이루어진 한 basis를 찾을 수 있다.
- ✓ C_k 들의 eigenvalue들을 관찰하자. $C_1,\ C_3,\ \ldots,\ C_{3^{N-1}}$ 의 eigenvalue들과 eigenvector들은 관찰을 통해 비교적 쉽게 얻을 수 있다. 일례로 아래는 $N = 2$인 경우의 $C_1,\ C_3$ 을 표현한 것이다.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

- ✓ 각 eigenvalue $1,\ \omega,\ \omega^2$ 에 3^{N-1} 차원의 eigenspace가 대응하며, 따라서 $C_1,\ C_3,\ \ldots,\ C_{3^{N-1}}$ 은 모두 diagonalizable하다. (ω 는 $x^3 = 1$의 실수가 아닌 한 근이다.)

- ✓ $C_1, C_3, \ldots, C_{3^N-1}$ 의 공통된 eigenvector들 $v_1, \ldots, v_{3^N}$ 을 잡자. 그 중 임의의 하나를 v 라 하자.
- ✓ 임의의 $0 \le k \le 3^N - 1$ 에 대해, C_k 를 여러 $C_1, C_3, \ldots, C_{3^N-1}$ 의 곱으로 표현할 수 있다. 따라서 다음 곱 $C_k v$ 를 고려하면, v 는 C_k 의 eigenvector가 되며, 대응하는 eigenvalue는 ω 의 거듭제곱으로서 다시 $1, \omega, \omega^2$ 중 하나가 된다는 것을 알 수 있다.
- ✓ 즉, 모든 $C_0, \ldots, C_{3^N-1}$ 들 전부가 simultaneously diagonalizable하고, 발생하는 모든 eigenvalue는 $1, \omega, \omega^2$ 중 하나이다.
- ✓ 따라서 $A = \sum_{a=1}^{M} C_{k^{(a)}}$ 는 diagonalizable하고, A 의 모든 eigenvalue는 $a+b+c=M$ 인 음이 아닌 정수 a, b, c 에 대해 $a + b\omega + c\omega^2$ 의 형태로 표현된다.
- ✓ $1 + \omega + \omega^2 = 0$ 임을 고려하여 이 형태의 서로 다른 복소수의 개수를 세면 $\frac{1}{2}(M^2 + 3M + 2)$ 개이다.

M. 이등병의 편지

- ✓ 따라서 A의 서로 다른 eigenvalue의 개수는 $\frac{1}{2}(M^2+3M+2)$개 이하임을 알 수 있다. 이들을 $\lambda_1 \sim \lambda_s$로 두면, A는 diagonalizable이므로 minimal polynomial $\prod_{i=1}^{s}(x-\lambda_i)$를 갖는다.
- ✓ A의 minimal polynomial의 차수의 한 상계를 $\frac{1}{2}(M^2+3M+2)$로 구할 수 있다. 이는 v_K의 0번 성분들을 모아 만든 수열이 만족하는 최소 선형 점화식의 characteristic equation의 차수의 상계이기도 하다. ■

- ✓ 데이터 중에는 $M=10$이고, $D=\frac{1}{2}(M^2+3M)=65$로 두게 되면 $2D$개의 초기 항을 투입하였을 때 잘못된 점화식이 반환되어 답이 틀리는 것이 있다.
- ✓ Subtask 4에 적용되는 이 풀이의 최종 시간복잡도는 $\mathcal{O}\left(3^N M(N+M^2)+M^4 \lg K\right)$이다.

- ✓ Subtask 5를 해결하기 위해서는, 구하는 경우의 수를 적절한 다항식의 계수의 형태로 표현한 뒤 그 다항식을 대신 분석하는 생성함수의 아이디어를 적용한다.
- ✓ N 개의 전등 각각에 변수 $x_1 \sim x_N$ 을 대응시키고, $j = 1, \ldots, M$ 번 부원 각각에 대해 단항식 $p_j(x_1, \ldots, x_N) = \prod_{i=1}^{N} x_i^{r_{ji}}$ 를 잡는다. 여기서 r_{ji} 는 j 번 부원이 i 번 전등을 조작하는 방식을 나타내는 $0, 1, 2$ 중 하나의 수이다.
- ✓ 이제 다항식 $g(x_1, \ldots, x_N) = (p_1 + p_2 + \cdots + p_M)^K$ 를 고려하자. 이 다항식의 거듭제곱을 전부 전개하여 단항식의 합으로 쪼개면, M^K 개의 항 각각이 공연 순서 한 가지에 대응한다. 이 중에서 공연이 끝난 후 모든 전등이 다시 꺼지는 공연 순서에 대응하는 것들을 선별하여 그 개수를 세야 한다.
- ✓ 그런데 위의 조건은 정확히, 해당 항 $\prod_{i=1}^{N} x_i^{s_i}$ 의 모든 지수 s_i 들이 3 의 배수가 되는 것과 동치이다.

- ✓ 한 단항식 x^s 에 $1,\ \omega,\ \omega^2$ 을 각각 대입하여 더한 식을 고려하면, s 가 3 의 배수일 경우 3 이고, 아닐 경우 0 의 값을 가짐을 확인할 수 있다.
- ✓ 마찬가지로, $\prod_{i=1}^{N} x_i^{s_i}$ 에 $\{(a_1,\ a_2,\ \ldots,\ a_N) :\ a_1,\ \ldots,\ a_N = 1,\ \omega,\ \omega^2\}$ 의 3^N 가지 값의 조합을 대입하여 전부 더한 값은, 오직 모든 s_i 들이 3 의 배수일 때만 3^N 이고, 다른 경우 0 이다.
- ✓ 따라서, 다음 식 $\frac{1}{3^N} \sum_{x_1,\ldots,x_N \in \{1,\omega,\omega^2\}} g(x_1,\ \ldots,\ x_N)$ 의 값이 정확히 구하는 경우의 수와 일치한다.
- ✓ Subtask 5에서는 이 3^N 가지 대입값을 직접 대입하여 다음과 같이 식의 값을 계산하면 충분하다.

M. 이등병의 편지

- ✓ 각 전등에 대해, 모든 부원이 해당 전등을 조작하는 방식을 입력받아 저장해 둔다.
- ✓ 최초에 모든 x_i 에 1 의 값을 대입하면 $p_1 = p_2 = \cdots = p_M = 1$ 이고, $g(x_1, \ldots, x_N) = M^K$ 이다.
- ✓ 대입하는 값을 조금씩 바꾸어 가면서 모든 값을 탐색한다. 그러는 동안 각 단항식 p_j 의 값($1, \omega$, 또는 ω^2)을 저장하며 관리하면, 대입 당 평균 $\mathcal{O}(M)$ 번의 계산으로 모든 $g(x_1, \ldots, x_N)$ 의 값을 구할 수 있다.
- ✓ 이를 위해 대입값의 순서를 잘 조정해서, 모든 대입값을 순회하면서도 각 대입마다 대입값이 바뀌는 변수 x_i 의 개수를 줄일 필요가 있다. Subtask 1에서처럼 백트래킹을 하여도 무방하고, I 번째 대입값에서 $(I+1)$ 번째 대입값으로 넘어갈 때 $x_{(I\text{의 }3\text{진법 표현의 끝에 연속한 숫자 }0\text{의 개수})}$ 에만 ω 를 곱하는 방식도 있다.
- ✓ 위 방식은 그레이 코드의 원리를 3 진법에 적용하는 것으로, 이를 통해 각 대입마다 단일 변수 x_i 의 대입값을 조정하고, 저장되어 있는 p_j 들의 값을 활용해 바뀐 $g(x_1, \ldots, x_N)$ 의 값을 계산한 뒤, 다시 p_j 들의 값을 갱신하는 식의 구현법을 구성할 수 있다.

- ✓ 얻은 $g(x_1, \ldots, x_N)$의 값을 전부 합한 뒤 3^N으로 나누어 주면 정답이 된다. 구현 과정에서 복소수 ω를 포함한 덧셈, 곱셈, 거듭제곱 등의 연산을 구조체로 구현하면 편리하다. $10^9 + 7 = P$로 나눈 나머지를 관리하는 방식은 실수에서와 다르지 않아, 계산 중 실수부 혹은 허수부가 $0 \sim (P-1)$의 범위를 초과할 때마다 P로 나눈 나머지로 대체해 주는 것으로 충분하다.
- ✓ 시간복잡도는 다음과 같다. 먼저 입력을 위해 $\mathcal{O}(NM)$이 소요된다. 3^N번의 대입을 하는 과정에서, 각 대입마다 $\mathcal{O}(M)$의 계산 시간과 함께 K제곱 연산 하나를 위해 $\mathcal{O}(\lg K)$가 추가로 소요된다.
- ✓ 따라서 총 시간복잡도는 $\mathcal{O}\left(3^N(M + \lg K)\right)$이다.

M. 이등병의 편지

- ✓ Subtask 5의 풀이는 3^N 회의 대입 각각에 대해 $\mathcal{O}(M)$ 회의 계산이 요구되어, 총 $\mathcal{O}(3^N M)$ 의 시간복잡도를 피할 수 없다. 3^N 개의 대입값을 하나씩 투입하지 않고 통합적으로 처리함으로써 이를 개선하면 Subtask 6을 해결할 수 있다.
- ✓ 부원이 모든 전구를 조작하는 방식은 총 3^N 가지이다. 부원들의 정보를 입력받을 때, 각 방식을 채택하고 있는 부원 수를 세어 3^N 크기의 벡터 v에 저장해 둔다. 이는 다항식 $p_1 + p_2 + \cdots + p_M$ 에서 각 단항식 $x_1^0 x_2^0 \cdots x_N^0 \sim x_1^2 x_2^2 \cdots x_N^2$ 의 등장 횟수들을 저장해 둔 것과 같다.
- ✓ 이때 비트마스크를 활용하여, $x_1^{s_1} x_2^{s_2} \cdots x_N^{s_N}$ 의 등장 횟수를 $k = \overline{s_N s_{N-1} \cdots s_1}_{(3)}$ 번째 성분에 놓자.
- ✓ $x_1, \ldots, x_N$ 의 대입값이 $\omega^{r_1}, \omega^{r_2}, \ldots, \omega^{r_N}$ 으로 주어질 때, 다항식 $p_1 + \cdots + p_M$ 의 값을 $v_0, v_1, \ldots, v_{3^N-1}$ 을 사용하여 표현하면 다음과 같다:

$$(p_1 + \cdots + p_M)|_{x_i = \omega^{r_i}\ \forall i} = \sum_{0 \le k \le 3^N - 1,\ k = \overline{s_N s_{N-1} \cdots s_1}_{(3)}} \omega^{\left(\sum_{i=1}^{N} r_i s_i\right)} v_k.$$

- 각 대입값의 조합 $(r_1, r_2, \ldots, r_N)$ 에 대하여, $\overline{r_N r_{N-1} \cdots r_1}_{(3)}$ 번 성분이 해당 대입값을 투입한 $p_1 + \cdots + p_M$ 의 값으로 정의되는 3^N 차원 벡터 w 를 고려하자.

$$(N=1)\ \ w = \begin{bmatrix} v_0 + v_1 + v_2 \\ v_0 + \omega v_1 + \omega^2 v_2 \\ v_0 + \omega^2 v_1 + \omega v_2 \end{bmatrix},\ \ (N=2)\ \ w = \begin{bmatrix} v_0 + v_1 + v_2 + v_3 + v_4 + v_5 + v_6 + v_7 + v_8 \\ v_0 + \omega v_1 + \omega^2 v_2 + v_3 + \omega v_4 + \omega^2 v_5 + v_6 + \omega a^7 + \omega^2 v_8 \\ v_0 + \omega^2 v_1 + \omega v_2 + v_3 + \omega^2 v_4 + \omega v_5 + v_6 + \omega^2 a^7 + \omega v_8 \\ v_0 + v_1 + v_2 + \omega v_3 + \omega v_4 + \omega v_5 + \omega^2 v_6 + \omega^2 v_7 + \omega^2 v_8 \\ v_0 + \omega v_1 + \omega^2 v_2 + \omega v_3 + \omega^2 v_4 + v_5 + \omega^2 v_6 + v_7 + \omega v_8 \\ v_0 + \omega^2 v_1 + \omega v_2 + \omega v_3 + v_4 + \omega^2 v_5 + \omega^2 v_6 + \omega v_7 + v_8 \\ v_0 + v_1 + v_2 + \omega^2 v_3 + \omega^2 v_4 + \omega^2 v_5 + \omega v_6 + \omega v_7 + \omega v_8 \\ v_0 + \omega v_1 + \omega^2 v_2 + \omega^2 v_3 + v_4 + \omega v_5 + \omega v_6 + \omega^2 v_7 + v_8 \\ v_0 + \omega^2 v_1 + \omega v_2 + \omega^2 v_3 + \omega v_4 + v_5 + \omega v_6 + v_7 + \omega^2 v_8 \end{bmatrix}$$

- w 를 구할 수 있다면, 각 성분을 K 제곱하여 모두 더한 뒤 3^N 으로 나누어 답을 얻는다. 하지만 앞선 Subtask 5의 풀이는 w 를 $\mathcal{O}\left(M3^N\right)$ 에 구하므로 부적격이다.

- ✓ 위의 v와 w 간의 관계는 수학적으로 다음과 같이 서술된다. 다음 행렬:

$$X = \begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega & \omega^2 \\ 1 & \omega^2 & \omega \end{bmatrix}$$

  에 대하여, $w = (\underbrace{X \otimes X \otimes \cdots \otimes X}_{N})v$. 여기서 $\otimes$는 텐서곱을 의미한다.

- ✓ '대칭성'을 표명하는 이 성질을 활용하면 v로부터 w를 보다 빠른 시간에 계산할 수 있다. 이후 w의 각 성분을 K제곱하여 전부 더한 뒤 3^N으로 나누어 답을 얻는다.
- ✓ Sum over Subsets DP를 3진법으로 확장 적용하는 방식으로 이를 달성한다.

- 다음의 의사코드를 통해 이 과정을 정리할 수 있다. 이는 $\mathcal{O}\left(N3^N\right)$ 에 동작한다.

initialize v, a vector of length 3^N
repeat for each step $= 1, 3, \ldots, 3^{N-1}$ {
    divide v into blocks of consecutive elements with the common size $[3 \times \text{step}]$
    repeat for each block, for each $j = 0, 1, \ldots, (\text{step} - 1)$ {
        let $a, b, c = \text{block}[j], \text{block}[j + \text{step}], \text{block}[j + 2 \cdot \text{step}]$
        replace $\text{block}[j], \text{block}[j + \text{step}], \text{block}[j + 2 \cdot \text{step}] =$
            $a + b + c,\ a + b\omega + c\omega^2,\ a + b\omega^2 + c\omega$
    }
}

- 귀납적으로, 이것이 실제로 $(\underbrace{X \otimes X \otimes \cdots \otimes X}_{N})v$ 를 계산하게 됨을 확인할 수 있다.

- ✓ 사실, 이 관찰은 간접적으로 3 진법에서의 Fast Walsh-Hadamard Transform의 형태를 제시한다.
- ✓ Subtask 4의 풀이에서 정의한 연산 $\oplus$에 대하여, 두 3^N 차 벡터 간의 convolution $v * w$를 $[v * w]_k = \sum_{i=0}^{3^N-1} \sum_{j=0}^{3^N-1} \delta_{k(i\oplus j)} v_i w_j$ 로 정의하고, 또 $\mathcal{F}(v) := (\underbrace{X \otimes X \otimes \cdots \otimes X}_{N})v$라 하자.
- ✓ $N = 1$인 경우, $v = \begin{bmatrix} a \\ b \\ c \end{bmatrix}, w = \begin{bmatrix} d \\ e \\ f \end{bmatrix}$ 에 대해 다음이 성립한다: ($\cdot$는 coordinatewise multiplication)

$$\mathcal{F}(v * w) = \mathcal{F}(v) \cdot \mathcal{F}(w) = \begin{bmatrix} ad + ae + af + bd + be + bf + cd + ce + cf \\ ad + \omega ae + \omega^2 af + \omega bd + \omega^2 be + bf + \omega^2 cd + ce + \omega cf \\ ad + \omega^2 ae + \omega af + \omega^2 bd + \omega be + bf + \omega cd + ce + \omega^2 cf \end{bmatrix}.$$

- ✓ 이는 모든 N에 대해 성립한다. 좌변과 우변의 k번 성분의 $v_i w_j$의 계수를 서로 비교함으로써 이를 검증할 수 있다($0 \le i,\ j,\ k \le 3^N - 1$).
- ✓ 한편 $X^{-1} = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & \omega^2 & \omega \\ 1 & \omega & \omega^2 \end{bmatrix}$ 이다. 상수를 제외하면 X에서 ω의 자리에 ω^2 혹은 $\overline{\omega}$를 대입한 형태이다. 그리고 텐서곱의 성질 $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$에 따르면, 역변환 $\mathcal{F}^{-1}$이 $\mathcal{F}^{-1}(w) = (\underbrace{X^{-1} \otimes X^{-1} \otimes \cdots \otimes X^{-1}}_{N})w$로 주어짐을 알아낼 수 있다.
- ✓ 이것을 통해 즉시 다른 풀이를 얻을 수 있다. Subtask 2에서의 v_t를 가져오고, Subtask 6의 앞선 풀이에서의 v를 가져올 때, 전이 식을 $v_t = v_{t-1} * v$로 표현할 수 있기 때문이다.

- 결국 다음 풀이가 성립한다.

$$v_K = \mathcal{F}^{-1}\left[\mathcal{F}(v_K)\right] = \mathcal{F}^{-1}\left[\mathcal{F}(v_0 * \underbrace{v * v \cdots * v}_{K})\right] = \mathcal{F}^{-1}\left[\mathcal{F}(v_0) \cdot \underbrace{\mathcal{F}(v) \cdot \mathcal{F}(v) \cdots \mathcal{F}(v)}_{K}\right].$$

- $\cdot$은 coordinatewise multiplication임에 유의하자. $\mathcal{F}(v)$를 K번 곱하는 과정은 $\mathcal{O}\left(3^N \lg K\right)$에 완수된다. $\mathcal{F}$ 및 그 역변환은 앞선 풀이와 같은 알고리즘으로 각각 $\mathcal{O}\left(N3^N\right)$에 완수된다.
- 그런데, v_0은 0번 성분만이 1이고 나머지는 0인 벡터이므로, $\mathcal{F}(v_0)$은 모든 성분이 1인 벡터이다. 이는 $\cdot$에 대한 항등원에 해당한다. 한편, 구해야 하는 값은 v_K의 0번 성분이므로, 최종적으로 $\mathcal{F}^{-1}$을 취한 뒤 0번 성분을 택하는 것은 모든 성분의 합을 구하여 3^N으로 나누는 것과 동치이기도 하다.
- 즉, 이 풀이는 v를 입력받아 앞선 풀이에서의 알고리즘으로 FWHT를 계산한 뒤, 모든 성분의 K제곱의 합을 구하고 3^N으로 나누어 주는 과정으로 reduce된다. 이는 사실 앞선 Sum of Squares DP를 사용한 풀이와 완전히 같은 계산 과정임을 관찰할 수 있다.

- ✓ 위 풀이에 따르면, 생성함수를 활용하지 않고도 이와 같은 3 진법에서의 FWHT를 통해 경우의 수 전이 식을 표현하여 문제를 해결할 수 있다.
- ✓ 하지만 이 서로 다른 두 접근 방식은 결국 공통된 알고리즘과 같은 계산 과정을 갖는 하나의 풀이로 귀결된다. FWHT의 형태를 파악하는 데 있어 생성함수를 통한 접근이 nudge로 작용할 수 있다.

- ✓ 시간복잡도는 다음과 같다. v를 입력받는 과정에서 $\mathcal{O}(NM)$, Sum over Subsets DP 혹은 FWHT를 계산하는 과정에서 $\mathcal{O}\left(N3^N\right)$, 각 성분을 K 제곱하고 전부 합치는 과정에서 $\mathcal{O}\left(3^N \lg K\right)$ 가 소요된다.
- ✓ 따라서 총 시간복잡도는 $\mathcal{O}\left(NM + 3^N(N + \lg K)\right)$ 이다.