



SOONGSIL  
UNIVERSITY  
1897



# 2020 SKH Algorithm Camp

주관



후원



승고한 대회 풀이집  
마라톤



# 이메일 포렌식

---

고려대 Alkor  
이동관

Marathon A

# 이메일 포렌식

---

- n: 문자열의 길이, k: @korea.ac.kr의 길이(약 10)
- 문자열을 뒤에서부터 parsing한다면  $O(k^*n)$ 으로 쉽게 찾을 수 있습니다.
- 스택 등의 자료구조로도  $O(k^*n)$ 으로 쉽게 찾을 수 있습니다.
- 더 나이브하게 시작 지점과 끝 지점을 잡아 이메일 형식이 맞는지 확인하는 방식으로도  $O(n^2)$ 으로 구할 수 있습니다.
- 구현 과정에서, kr에 대한 예외처리 정도만 신경 쓰면 되는 문제였습니다.

# 알약을 찾아라!

---

한양대 ALOHA  
김휘수

Marathon B

# 알약을 찾아라!

---

- 알고리즘 분류 : 수학
- 어떤 수 A를 나타내기 위해서 필요한 최소의 환자 수 구하기
  - ⇒ 약에 번호 0 ~ N-1을 붙이고 그 번호를 2진수로 표현하기 (ex  $100 = 1100100$ )
  - ⇒ 사람에도 번호를 붙인다 ( $2^0, 2^1, 2^2, 2^3 \dots$ )
  - ⇒ 해당 자릿수가 1이면 약을 먹이고 0이면 먹이지 않는다.
  - ⇒ 시간이 지난 뒤, 효과가 나타난 환자의 번호를 다 더한 값과 일치하는 약의 번호가 찾고자 하는 약이다.
- 따라서 N을 2진수로 나타내기 위한 최소한의 bit의 수가 답이다.

# 귀차니즘 성서

---

숭실대 SCCC  
이로건

Marathon C

# 귀차니즘 성서

---

- 의도한 풀이 : 3차원 DP
- 3차원 DP를 이용해 [날짜] [지금까지 이동한 횟수] [현재 위치한 대학]을 상태로 가지고 값은 현재 상태의 최댓값을 가져간다.
- 최악의 경우 합이 int 최대범위를 넘어감으로 DP 배열은 long long으로 잡아야 한다.
- U와 P가 입력되었을 때 U를 숫자로 바꾼다. ex) S=0, K=1, H=2

# 귀차니즘 성서

---

- DP를 구현하기 전에 DP 배열은 -1로 초기화를 해준다.
- 성서가 첫날에 한양대에서 출발함으로 DP 배열의  $[0][0][2]=0$ 을 해준다.

# 귀차니즘 성서

---

- 현재 상태  $[i][j][k]$ 는  $i-1$ 번째에서  $i$ 번째 U와 P를 입력받아 만들어진다.
  - 현재  $i$ 번에 있을 때  $i+1$ 번째 U와 P를 입력받아 U에 따라 다음과 같이  $i+1$ 을 갱신한다.

대학이 U와 다르고 이동하는 경우

$$\text{arr}[i+1][j+1][U] = \max(\text{arr}[i+1][j+1][U], \text{arr}[i][j][k]+P)$$

대학이 U와 다르고, 이동하지 않는 경우

$$\text{arr}[i+1][j][k] = \max(\text{arr}[i+1][j][k], \text{arr}[i][j][k]);$$

대학이 U의 경우

$$\text{arr}[i+1][j][U] = \max(\text{arr}[i+1][j][U], \text{arr}[i][j][k]+P)$$

- 출력값은 N 번째 날의 모든 상태 중 가장 큰 값을 출력하면 된다.

# 오해받는 숭고한 캠프

---

고려대 ALPS  
공인호

Marathon D

# 오해받는 숭고한 캠프

- a. 한 알파벳이 두 번 이상 등장하는 경우,  
절대로 새로운 숭고한 캠프의 이름이 될 수 없다. (ex. AABC)
- b. 어떤 네 자리 알파벳 조합에 대해서 아래와 같은 4x4 행렬을 만들었을 때, 역행렬이 존재하는  
조건이 곧 문제에서 주어진 조건이 된다.
- c. 모든 네 자리 알파벳 조합에 대해, 아래 행렬을 만들고 역행렬이 존재하는지 계산하여, 주어진  
조건을 만족하는지 확인한다.
- d. 각 동아리 이름에 어떤 알파벳이 몇 개 있는지는 중요하지 않기 때문에,  
전처리를 통해 각 동아리 이름마다 각 알파벳의 등장 여부를 계산해 놓는다.

	첫 번째 알파벳	두 번째 알파벳	세 번째 알파벳	네 번째 알파벳
첫 번째 동아리 이름				
두 번째 동아리 이름	각 entry의 값: 동아리 이름에 해당 알파벳이 포함되어 있는가? (1 or 0)			
세 번째 동아리 이름				
네 번째 동아리 이름				

# 오해받는 승고한 캠프

---

- 예를 들어, ALKOR, ALPS, ALOHA, SCCC에 대해서 “SLHP”라고 이름을 새로 짓는다면, 행렬은 아래와 같고 이에 대한 역행렬이 존재한다.

	S	L	H	P
ALOHA	0	1	1	0
ALKOR	0	1	0	0
ALPS	1	1	0	1
SCCC	1	0	0	0

- 만약 “ALOS”라고 이름을 새로 짓는다면 행렬은 아래와 같으며 역행렬이 존재하지 않는다.

	A	L	O	S
ALOHA	1	1	1	0
ALKOR	1	1	1	0
ALPS	1	1	0	1
SCCC	0	0	0	1

# 오해받는 숭고한 캠프

---

- 우선 각 이름에 대한 전처리는 최대 40000번의 계산이 필요하다.
- 모든 알파벳의 조합은  $26C4$ 개 존재하며,  
각 조합에 대해서 표를 만드는데 16번의 계산이,  
역행렬이 존재하는지 판단하기 위해 최대 16번의 계산이 필요하다.
- $40000 + 26C4 * 32 = 518,400$ 으로 앞에 어떤 상수가 불더라도  
1초안에 해결됨을 확신할 수 있다.

# 점 매칭

---

숭실대 SCCC  
김도현

Marathon E

# 점 매칭

---

- 항상  $\text{floor}(N/2)$ 개의 직선을 만들 수 있음을 발견하자.
- 이제 그러한 직선의 집합을 효율적으로 만들어야 하는데, 출제자가 찾은 제일 간단한 풀이는 다음과 같다.
- 모든 점을  $\text{pair}<\text{int},\text{int}>$ 로 정렬한 후, 인접한 두 원소를 연결하면 된다.
- 제한을 일부러 애매하게 줘서, 정렬하면 바로 풀리는 걸 숨겼다.
- $N \log N$  체커 짜는게 귀찮은게 제일 큰 이유지만 ㅋㅋ!

# 바이토닉 트리

---

한양대 ALOHA  
박정호

Marathon F

# 바이토닉 트리

---

- 바이토닉 수열은 기본적으로 3가지 유형이 있다.

1. 강한 증가 수열      ex ) [1, 2, 3, 4, 5]
2. 강한 감소 수열      ex ) [5, 4, 3, 2, 1]
3. 증가 후 감소 수열    ex ) [1, 3, 5, 4, 2]

- 이 3가지 유형의 수열의 공통된 특징은 다음과 같다.

- 같은 수가 연속하게 나타나지 않는다.  
ex ) [1, 1, 1, 2, 1]은 바이토닉 수열이 아니다. 1이 연속하게 나타나기 때문이다.
- 한번 감소 구간에 들어가면 다시 증가하지 않는다.  
ex ) [1, 5, 4, 3, 4]는 바이토닉 수열이 아니다. 5 이후로 감소 구간에 들어갔는데  
다시 “3, 4”와 같이 증가 구간이 생겼기 때문이다.
- 따라서 우리는 이 두가지 특징을 이용해서 이 문제를 해결해야 한다.

# 바이토닉 트리

---

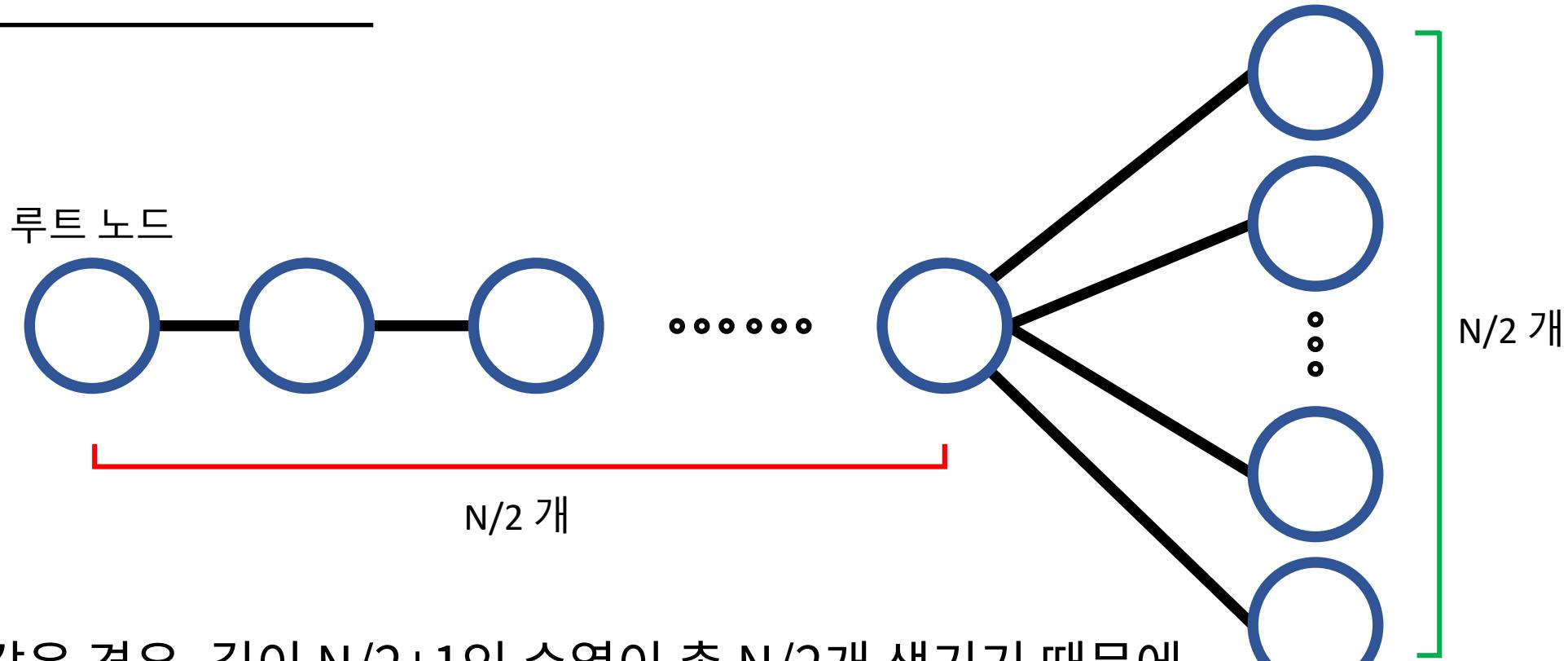
- 어떤 수열이 바이토닉 수열인지 확인하는 방법은 다음과 같다.
- 수열의 처음부터 끝까지 다음 과정을 반복한다.  
이때 수열은 처음에 증가 구간에 있다고 본다.
- 수열의 현재 수 ( $A[i]$ ) 와 바로 이전 수 ( $A[i-1]$ ) 를 비교한다.
- 두 수가 같다면, 이 수열은 바이토닉 수열이 아니다.  
 $A[i-1]$ 이 더 크다면, 이 수열은 이후 감소 구간이라고 본다.  
 $A[i]$ 가 더 크다면, 이 수열이 현재 증가 구간일 때는 문제가 없다.  
이 수열이 현재 감소 구간이었다면 이 수열은 바이토닉 수열이 아니다.
- 위의 반복이 끝나고, 한번도 “바이토닉 수열이 아닐” 조건에 있지 않았다면,  
이 수열은 바이토닉 수열이다.
- 이 과정의 시간복잡도는  $O(N)$ , 여기서  $N$ 은 수열의 길이이다.

# 바이토닉 트리

---

- Naïve한 풀이는 다음과 같다.
  1. 주어진 루트 노드에서 시작해서 트리를 순회한다.
  2. 트리 순회 중 리프 노드에 도달하면, 루트에서부터의 경로를 모두 저장한다.
  3. 모든 순회가 끝난 후 저장된 모든 경로들에 대해서 바이토닉 수열인지 체크한다.
- 이 풀이의 경우 리프 노드의 개수  $L$ 만큼 수열이 생기고, 각 수열의 길이가 최대 트리의 높이  $H$ 가 된다.
- 따라서  $O(H)$ 의 바이토닉 수열 확인 과정을  $L$ 회 반복해야 한다.  
즉, 이 풀이의 시간복잡도는  $O(L^*H)$ 가 된다.  
대체로  $L$ 이 증가하면  $H$ 가 감소하는 양상을 띄기에  
대부분의 경우에는 이 풀이도 1초에 통과한다.  
**하지만 반례가 존재한다.**

# 바이토닉 트리



- 이와 같은 경우, 길이  $N/2+1$ 인 수열이 총  $N/2$ 개 생기기 때문에,  
최악의 경우인  $N = 100,000$ 의 경우,  $50,001 * 50,000 = 2,500,050,000$ 회의  
반복이 필요하게 되고,  
TLE를 받게 된다.

# 바이토닉 트리

---

- 앞의 풀이가 TLE가 발생한 이유는,  
여러 수열에 공통으로 존재하는 구간을 한번만 체크하지 않고, 여러 번 체크했기  
때문이다.  
(앞의 예시에서 빨간색으로 표시된 부분)
- 하지만, 모든 경로마다 수열을 따로 저장하는 방식은 공통 구간을 한번만 체크하기  
어렵다.
- 즉, 모든 수열을 따로 저장하지 않고 진행하는 방식이 필요하다.

# 트리 가짓수 세기

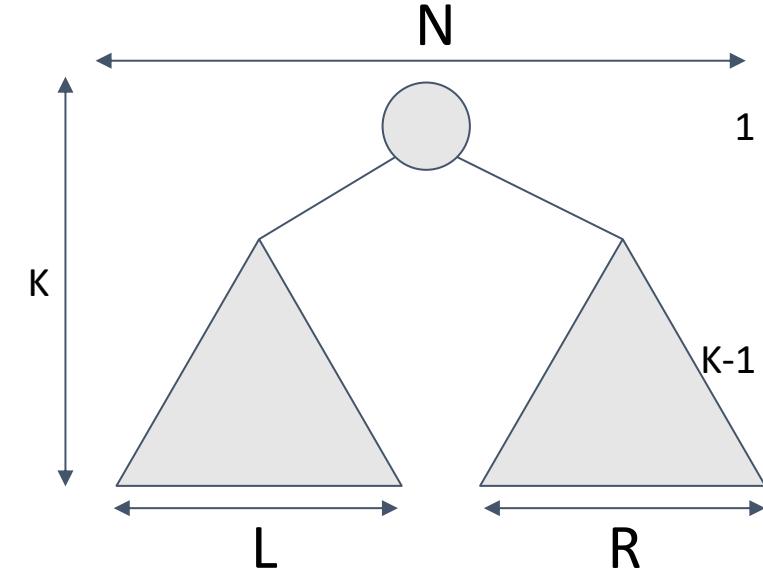
---

충실대 SCCC  
김민상

Marathon G

# 트리 가짓수 세기 - 분석

- N : 트리에 들어가는 노드의 개수
- K : BST 트리의 높이
- L : 왼쪽 서브트리에 있는 노드의 개수
- R : 오른쪽 서브트리에 있는 노드의 개수
- 왼쪽 서브트리의 노드의 개수가 L개라고 하면 오른쪽 서브트리의 노드의 개수는  $N - L - 1$ 개이다.
- 노드의 개수가 N개이고 높이가 K인 BST가 만들어지는 경우의 수 = 노드의 개수가 L개이고 높이가 K-1인 BST가 만들어지는 경우의 수 \* 노드의 개수가 R개이고 높이가 K-1인 BST가 만들어지는 경우의 수
- L이 가능한 범위를 계산해보면  $0 \leq L \leq N - 1$  이다.
- 이를 가지고 점화식을 세워보자.



# 트리 가짓수 세기

---

- $DP[K][N]$  : 높이  $K$ 에서 노드  $N$ 개를 놓을 수 있는 경우의 수

$$dp[K][N] = \sum_{i=0}^{N-1} dp[K-1][i] * dp[K-1][N-i-1]$$

- 
- 시간 복잡도 :  $O(N^2 K)$
  - 위 점화식을 잘 소스코드로 옮기면 됩니다.

# 제가 수학 문제를 하나 가져왔어요

---

숭실대 SCCC  
권욱제

Marathon H

# 제가 수학 문제를 하나 가져왔어요

---

1.  $x, y$ 가 증가함에 따라  $f(x, y)$ 가 엄청 빠른 속도로 증가한다.
2. 유독  $f(i, i-1)$ 의 변화량만 굉장히 작다.
3.  $f(1, 1) = 1$ 일 때,  $f(3000, 2999) = 4,498,500$ 이다.
4. 초항을  $n$ 배 증가시켜 점화식을 계산하면, 모든 항의 값이 정확히  $k$ 배 증가한다.

4번이 제일 중요한데,  $f(1, 1) = 2$ 이고  $f'(1, 1) = 6$ 일 때,  
 $3f(x, y) = f'(x, y)$ 가 성립한다는 말입니다.

성질을 잘 찾으셨나요?

$f(1, 1) = 1$ 의 모든  $f(x, y) \leq 4,498,500$ 를 set에 넣어두고 답을 찾으면 됩니다. ^-^v

# 제가 수학 문제를 하나 가져왔어요

---

문제에 스텔링 수가 등장하지만 풀이랑은 별로 관계가 없습니다.

ㅎㅎ;; 저.. 쿰!!

# 이진수 탐색

---

고려대 Alkor  
이경렬

Marathon I

# 이진수 탐색

---

- 케이스 분류를 통해 패턴을 찾자!
- 111…1 을 1로 줄이고, 000…0을 0으로 줄였을 때 다음과 같은 케이스들이 있을 것이다.

# 이진수 탐색 - 케이스 관찰

---

- 1) 01, 0101, 010101, ...
- 2) 10, 1010, 101010, ...
- 3) 010, 01010, 0101010, ...
- 4) 101, 10101, 1010101, ...
- 5) 00000...
- 6) 11111...
- 7) X

# 이진수 탐색 - 1

---

- 1) 01, 0101, 010101, … 의 경우에서  
‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때

01 →             $a=1, b=1, c=1, d=0$

0101 →         $a=2, b=2, c=2, d=1$

010101 →      $a=3, b=3, c=3, d=2$

→  $c-d=1$  이면서,  $a \geq c, b \geq c$ 일 때 패턴 1)이 성립

# 이진수 탐색 - 2

---

- 2) 10, 1010, 101010, … 의 경우에서  
‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때

10 → a=1, b=1, c=0, d=1

1010 → a=2, b=2, c=1, d=2

101010 → a=3, b=3, c=2, d=3

→ d-c=1이면서, a>=d, b>=d일 때 패턴 2)가 성립

# 이진수 탐색 - 3

---

- 3) 010, 01010, 0101010, … 의 경우에서  
‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때

010 → a=2, b=1, c=1, d=1

01010 → a=3, b=2, c=2, d=2

0101010 → a=4, b=3, c=3, d=3

→ c=d이면서,  $a \geq c+1$ ,  $b \geq c$ 일 때 패턴 3)이 성립

# 이진수 탐색 - 4

---

- 4) 101, 10101, 1010101, … 의 경우에서  
‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때

101 → a=1, b=2, c=1, d=1

10101 → a=2, b=3, c=2, d=2

1010101 → a=3, b=4, c=3, d=3

→ c=d이면서, a>=d, b>=d+1일 때 패턴 4)가 성립

# 이진수 탐색 - 5

---

- 5) 0000… 의 경우에서  
‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때

$a > 0$ 이면서  $b=c=d=0$

# 이진수 탐색 - 6

---

- 6)  $1111\cdots$  의 경우에서 '0'의 개수를 a, '1'의 개수를 b, '01'의 개수를 c, '10'의 개수를 d라고 할 때

$b > 0$ 이면서  $a=c=d=0$

# 이진수 탐색 - 7 (x)

---

- 7) X 의 경우에서 '0'의 개수를 a, '1'의 개수를 b, '01'의 개수를 c, '10'의 개수를 d라고 할 때

$a=b=c=d=0$

이럴 땐 개행 문자 '\n'을 출력함에 유의.

# 이진수 탐색 - 5, 6, 7

---

- ‘0’의 개수를 a, ‘1’의 개수를 b, ’01’의 개수를 c, ’10’의 개수를 d라고 할 때, 1~7)의 경우 모두  $|c-d| \leq 1$  이어야 한다. 아닌 경우 ‘-1’을 출력.
- A=0인 경우엔 패턴 6 혹은 패턴 7이고, b=0인 경우에는 패턴 5 혹은 패턴 7이기 때문에, a나 b 둘 중 하나가 0인데 c나 d가 1 이상인 경우엔 ‘-1’ 출력. (따로 예외처리를 하면 편하다!)
- 예외처리를 할 때,  $a,b \geq 1$ 인데,  $c=d=0$ 인 경우도 ‘-1’ 출력

# 이진수 탐색 - 1

---

- $c-d=1$ 인 경우는 패턴 1)인데,  $a>=c$ ,  $b>=c$ 가 아닌 경우 '-1' 출력

패턴 1)은 01, 0101, 010101… 이므로 '01'의 개수에 맞춰서 패턴 1) 중 몇 번째 이진수를 선택해야 하는지 골라야 한다.

가장 큰 이진수를 출력해야 하기 때문에 '01'의 개수를 맞추기 위해 필수적으로 사용해야 하는 '0'과 '1'의 개수를 제외한 나머지 '0'과 '1'를 알맞게 배치해야 하는데, '1'은 가장 왼쪽의 '1', '0'은 가장 오른쪽의 '0'에 몰빵.

## 이진수 탐색 - 2

---

- $d-c=1$ 인 경우는 패턴 2)인데,  $a>=d$ ,  $b>=d$ 가 아닌 경우 '-1' 출력

패턴 2)는 10, 1010, 101010… 이므로 '10'의 개수에 맞춰서 패턴 2) 중 몇 번째 이진수를 선택해야 하는지 골라야 한다.

가장 큰 이진수를 출력해야 하기 때문에 '10'의 개수를 맞추기 위해 필수적으로 사용해야 하는 '0'과 '1'의 개수를 제외한 나머지 '0'과 '1'를 알맞게 배치해야 하는데, '1'은 가장 왼쪽의 '1', '0'은 가장 오른쪽의 '0'에 몰빵.

# 이진수 탐색 - 3, 4

---

- $c=d$ 인 경우는 패턴 3,4)인데,  $a>=c+1$ ,  $b>=c$  이거나  $c=d$ 이면서,  $a>=d$ ,  $b>=d+1$  가 아닌 경우 '-1' 출력

패턴 3, 4에서 '10'의 개수에 따라서 몇 번째 이진수를 선택해야 하는지 골라야 한다.

가장 큰 이진수를 출력해야 하기 때문에 '10'의 개수를 맞추기 위해 필수적으로 사용해야 하는 '0'과 '1'의 개수를 제외한 나머지 '0'과 '1'를 알맞게 배치해야 하는데, '1'은 가장 왼쪽의 '1', '0'은 가장 오른쪽의 '0'에 몰빵.

# 3N 투어링

---

고려대 ALPS  
서태수

Marathon J

# 3N 투어링

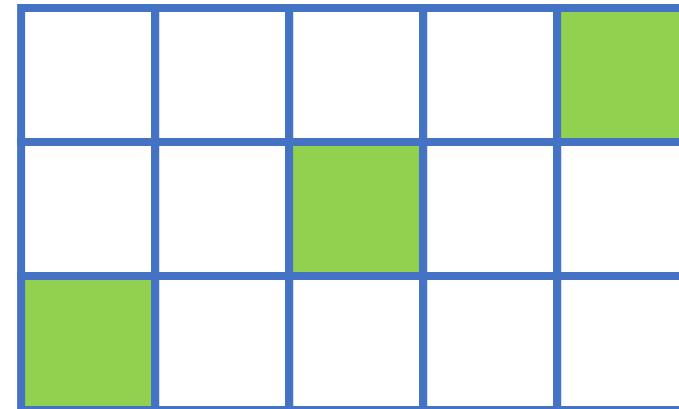
---

- 스포부터 하고 가자면 태수는 N에 상관없이 이 놀이를 성공합니다.
- 푸는 사람마다 매우 다양한 풀이가 나올 수 있는 문제입니다.
- 제가 작성한 풀이와 다른 풀이를 찾더라도 전혀 슬퍼하지 않으셔도 됩니다.  
오히려 여러분들이 찾은 풀이가 더 좋을 수 있어요.

# 3N 투어링

---

- 저는 기본적으로 이러한 형태를 이용했습니다.
- 그렇기 때문에  $N \% 5$ 를 기준으로 케이스 분석을 해주었습니다.



# 3N 투어링

---

- N이 5 미만이면 하드코딩 해줍니다.
- 정말 마음대로 하면 됩니다.

3
2
1

3	2
6	5
1	4

7	2	5
4	9	8
1	6	3

3	10	5	8
6	9	2	11
1	4	7	12

# 3N 투어링

---

- $N \% 5 == 0$  일 때
- (모든 케이스의 핵심은 최대한 오른쪽으로 갔다가 한 칸 이동하고 최대한 왼쪽으로 갔다가… 의 반복입니다.)

13	12	23	26	3	16	9	20	29	6
24	25	2	15	10	21	28	5	18	7
1	14	11	22	27	4	17	8	19	30

# 3N 투어링

---

- $N \% 5 == 1$  일 때

27	26	13	16	3	30	23	10	19	6	33
14	15	2	29	24	11	18	5	32	21	8
1	28	25	12	17	4	31	22	9	20	7

# 3N 투어링

---

- $N \% 5 == 2$  일 때

15	16	31	28	3	12	19	34	25	6	9	22
30	29	2	13	18	33	26	5	10	21	36	23
1	14	17	32	27	4	11	20	35	24	7	8

# 3N 투어링

---

- $N \% 5 == 3$  일 때

17	16	33	30	3	20	13	36	27	6	23	10	39
32	31	2	19	14	35	28	5	22	11	38	25	8
1	18	15	34	29	4	21	12	37	26	7	24	9

# 3N 투어링

---

- $N \% 5 == 4$  일 때

17	18	33	36	3	14	21	30	39	6	11	24	27	42
34	35	2	15	20	31	38	5	12	23	28	41	8	9
1	16	19	32	37	4	13	22	29	40	7	10	25	26

# 3N 투어링

---

- 일반성을 잃지 않고 N이 더 커졌을 때도 똑같은 방식으로 진행하면 됩니다.

# 태양처럼

---

고려대 Alkor  
이동관

Marathon K

# 태양처럼

---

- $n \log n$  복잡도의 알고리즘이라면 어떤 풀이도 가능합니다!
- 세그먼트 트리, sqrt decomposition 알고리즘 아무거나 알아보시면 좋습니다.

# 숭고한 초콜릿

---

고려대 ALPS  
공인호

Marathon L

# 승고한 초콜릿 - 아이디어

---

- 길이가 N인 초콜릿을 크기가 같은 M 개의 조각으로 나누어 줄 수 있는,
  - M-1 개의 금의 조합 개수를 찾아야 합니다.
- 
- 우선 첫 번째 아이디어는, 초콜릿의 넓이가 4N으로 고정되어 있으며,
  - 이에 초콜릿 조각 하나의 넓이가  $4N/M$ 라는 것입니다.
- 
- 이 사실과 사다리꼴 및 삼각형의 넓이 공식을 이용하면,
  - 선택되는 금들은 반드시  $(a_i+b_i) \%(2N/M) == 0$ 을 만족함을 알 수 있습니다.
  - 또한 선택된 M-1 개의 금들의  $(a_i+b_i)/(2N/M)$  값이 1~M-1 범위에서 모두 다른 것을 알 수 있습니다.

# 승고한 초콜릿

---

- a)  $(a_i + b_i) \% (2N/M)! = 0$ 인 모든 금을 제거합니다.
- b)  $(a_i + b_i) / (2N/M)$  값을 바탕으로 모든 금을 1~M-1 중 하나로 매핑시킵니다.  
 $(a_i + b_i) / (2N/M) = x$ 인 금의 집합을  $S_x$ 라고 부릅시다.
- c) 모든  $S_x$ 를  $(a_i, b_i)$ 순으로 정렬합니다.
- d)  $dp[x][t] = S_x$ 중 t번째 직선을 골랐을 때,  $S_{\{x+1\}}, \dots, S_{\{M-1\}}$  각 집합에서 금을 하나씩 고를 때 서로 겹치지 않게 고를 수 있는 경우의 수라고 정의합시다.
- e)  $dp[x][t] = \text{sum}(\text{dp}[x+1][t1], \dots, \text{dp}[x+1][t2])$  를 만족합니다.  
t1:  $a[S_x[t]] \leq a[S_{\{x+1\}}[l]]$ 를 만족하는 l 중 최소값  
t2:  $b[S_x[t]] \leq b[S_{\{x+1\}}[r]]$ 를 만족하는 r 중 최대값  
t2는  $a[S_{\{x+1\}}[r]] \leq a[S_x[t]] + 2N/M$ 를 만족하는 r 중 최대값이기도 합니다.

# 승고한 초콜릿

---

- f) k에 대해 M-1부터 시작해 1 까지 dp 테이블을 채워주면 됩니다.
- g) 특정 x에 대해서 dp 테이블을 모두 채운 후 prefix-sum을 이용하여,  
 $\text{prefix}[t] = \text{dp}[x][1] + \dots + \text{dp}[x][t]$ 를 구해줍니다.
- h) 이분탐색 또는 투 포인터로 l과 r을 찾아  $\text{prefix}[r]-\text{prefix}[l-1]$ 을 이용하여, x-1에 대한 dp 테이블을 채웁니다.
- i) 단, dp 테이블을 2차원 그대로 잡아버리면 메모리 초과가 나기 때문에,  
 $\text{dp}[t]$ ,  $\text{prefix}[t]$  두 개의 1차원 배열만 잡고 dp 갱신, prefix 갱신, dp 갱신 … 을 반복해줍니다.
- j) 마지막 iteration이 끝나고,  $\text{prefix}[\text{s\_1.size}()]$ 를 출력하면 됩니다.

# 승고한 초콜릿

---

- 우선 정렬을 위해 총  $O(K \log K)$ 의 연산이 필요합니다.
- 최악의 상황을 가정하기 위하여,  
모든 금이  $(a_i + b_i) \% (2N/M) == 0$ 을 만족한다고 가정합시다.
- $(a_i + b_i) / (2N/M)$ 의 값이  $x$ 인 금의 개수  $p_x$ 라고 할 때,  
 $p_1 + \dots + p_{M-1} = K$ 를 만족합니다.
- 따라서 투 포인터를 사용하여  $l, r$ 을 구하는 경우,  
DP값을 모두 구하기 위해  $O(K)$ 의 연산이 필요하고,
- 이분탐색을 사용하더라도,  $O(K \log K)$ 의 연산이 필요합니다.
- 따라서, 총 시간 복잡도는  $O(K \log K)$ 가 됩니다.

# 리그 오브 숭고한

---

고려대 ALPS  
서태수

Marathon M

# 리그 오브 숭고한

---

- 유저 또는 몬스터가 죽고 나서 부활하기 전에 행동을 할 수 없습니다.
- 유저 또는 몬스터가 죽고 나서 부활한 뒤에도 죽어 있다고 판단하면 안됩니다.
- 같은 팀을 죽일 수는 없습니다.
- 아이템을 살 때 아이템의 가격보다 현재 갖고 있는 골드가 더 많아야 합니다.
- 골드는 정확히 1000ms당 1골드가 들어옵니다. 500ms 지났다고 0.5골드가 들어오는 형식이 아닙니다.

# 리그 오브 숭고한

---

- 같은 아이템을 여러 개 살 수 있습니다. 다만 갖고있는 총 아이템의 개수가 4개를 초과하면 안됩니다.
- 현재 갖고 있지 않은 아이템을 팔면 안됩니다.
- 아이템을 팔 때, 샀을 때의 절반의 가격만 돌려받을 수 있습니다. 가격이 홀수라면 소수점은 버립니다.
- 레벨업을 할 때 한 번에 2레벨 이상 올릴 수 있지만, 15보다 커지지 않도록 잘 코딩해야 합니다.
- 포탑을 파괴하면 파괴한 사람은 300골드, 그와 같은 팀 사람들은 150골드를 받습니다.

# 리그 오브 숭고한

---

- 포탑을 6개 이상 파괴할 수 없습니다.
- 넥서스를 파괴하기 전에 2개 이상의 포탑을 파괴해야 합니다.
- 당연하겠지만 상대팀의 포탑은 세면 안됩니다.
- 넥서스가 파괴되고나서 그 어떠한 정보도 타임스탬프에 저장 돼있으면 안됩니다.

# 리그 오브 숭고한

---

- 앞에서 언급한 내용들을 하나도 빠짐없이, 꼼꼼하게 잘 코딩하면 정답을 받을 수 있습니다!

# 숭고한 마라톤 대회

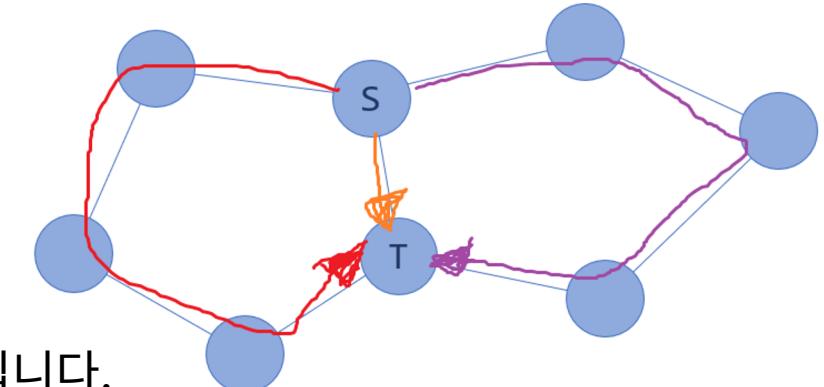
---

숭실대 SCCC  
이성서

Marathon N

# 승고한 마라톤 대회 - 관찰 1

- 시작점과 끝점을 잘 정했을 때 겹치지 않는 3개의 경로를 만들 수 있는 그래프가 어떤 성질을 갖는지 관찰합시다.



- 두 단순 사이클이 공유하는 간선이 있다면?
  - 공유 정점 두 개를 각각 시작점과 끝점으로 정할 시 항상 3개의 경로가 생깁니다.
- 두 단순 사이클이 공유하는 간선이 없다면?
  - 항상 3개의 경로를 만드는 방법이 존재하지 않습니다.

(몰라도 문제풀이에 지장은 없지만, 이러한 성질을 갖는 그래프는 선인장 그래프와 동치입니다.)



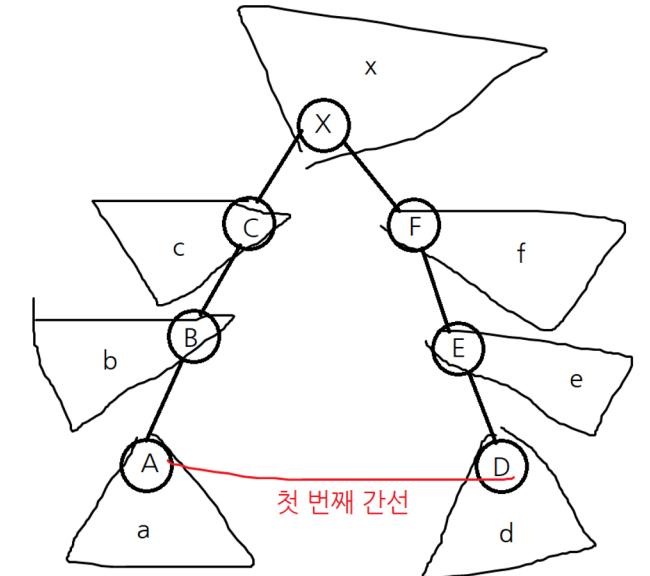
공유 간선이 없는 쪽이 생각하기 편하므로 전체 경우의 수에서 3개의 경로를 못 만드는 경우의 수를 빼서 구합시다.

# 승고한 마라톤 대회 - 관찰 2

- 간선을 2개나 놓아야 하면 생각하기 복잡하므로 간선 하나를 먼저 놓고 생각합시다.
- 첫 번째 간선을 (A, D)에 놓았다면?
  - (A, B), (B, C), (C, X), (X, F), (F, E), (E, D) 6개의 간선을 끊었을 때 생기는 7개의 서브트리 중 하나에 두 번째 간선의 양 끝점이 모두 들어가야 합니다.

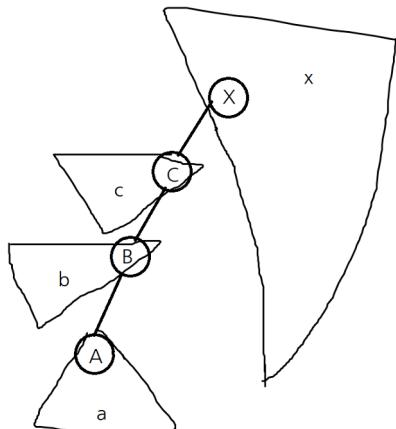
$$S(n) = n(n-1)/2 - (n-1)$$

라 하면 두 번째 간선을 놓는 경우의 수는  
 $S(a) + S(b) + S(c) + S(d) + S(e) + S(f) + S(x)$



# 승고한 마라톤 대회 - 루트-서브트리

- 루트를 고정하고 DFS를 돌리는데, 모든 정점 X에 대해  $X = \text{LCA}(A, B)$ 인  $(A, B)$  간선을 첫 번째 간선으로 놓고
- 두 번째 간선까지 잘 놓아서 목표를 달성하는 경우의 수를 정답에 더합니다.
- 두 간선을 놓는 순서가 바뀌는 경우도 같은 답이므로, 중복 제거를 위해 최종 답을 2로 나눠야 합니다.
- $X$ 가 첫 번째 간선의 끝점인 경우를 루트-서브트리 경로, 나머지 경우를 서브트리-서브트리 경로라고 부르겠습니다.



루트-서브트리 경로는 간단하게 구할 수 있습니다.  $X$ 의 첫 자식이  $C$ 인 경우,

$$+ S(x-c) + S(c-b) + S(b-a) + S(a)$$
$$+ S(x-c) + S(c-b) + S(b)$$
$$+ S(x-c) + S(c)$$
$$- (S(x-c) + S(c))$$

<-  $(X, C)$ 는 이미 존재하는 간선이므로 예외 처리

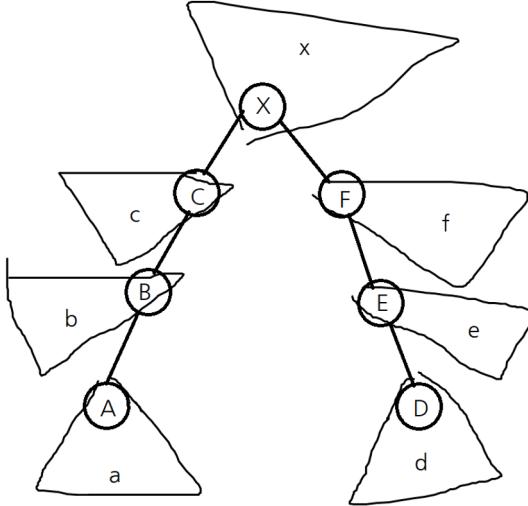
$C$ 의 서브트리 정점 개수를  $\text{sz}[C]$ 라 정의하고 굵은 글씨 합을  $\text{sum}[C]$ 라 정의하면,

$$\text{sum}[X] = \text{sum}[C] + S(\text{sz}[X] - \text{sz}[C]) * \text{sz}[C] + S(\text{sz}[X])$$

처럼  $\text{sum}[ ]$ 을 관리할 수 있습니다. 자식이 여러 개인 경우도 비슷하게 확장하면 됩니다.

# 승고한 마라톤 대회 - 서브트리-서브트리

---



루트가 답에 기여하는 정도가 계산하기 까다롭습니다.  
 $O(N^2)$ 을 피하기 위해  $S()$ 를 전개해야 합니다.

$x$ 의 자식들 서브트리 크기를 각각  $x_1, x_2, \dots, x_k$ 라 하면  
를  $O(k)$ 에 계산해야 합니다.

$$\sum_{1 \leq i < j \leq k} S(N - x_i - x_j) x_i x_j$$

# 승고한 마라톤 대회 - 서브트리-서브트리 (2 - 루트 처리)

---

- 빠르게 계산하기 위해  $V = x_i$ 들의 합,  $W = (x_i)^2$ 들의 합으로 전처리합니다.

$$\begin{aligned} & \sum_{1 \leq i < j \leq k} S(N - x_i - x_j) x_i x_j \\ &= \left[ \sum_{1 \leq i < j \leq k} (N^2 - 3N + 2)x_i x_j + (x_i^3 - 2Nx_i^2 + 3x_i^2)x_j + (x_j^3 - 2Nx_j^2 + 3x_j^2)x_i + 2x_i^2 x_j^2 \right] / 2 \\ &= [V^2(N^2 - 3N + 2) + 2V \sum_{1 \leq i \leq k} (x_i^3 - 2Nx_i^2 + 3x_i^2) + 2W^2 - 2 \sum_{1 \leq i \leq k} S(N - 2x_i)x_i^2] / 4 \end{aligned}$$

위처럼 열심히 수식 전개를 하면 풀립니다.

최종 시간복잡도는  $O(N)$ 이 됩니다.

# 어려운 이항계수 문제

---

고려대 Alkor  
오제노

Marathon O

# 어려운 이항계수 문제

---

- \_\_int128로 문제를 푼다면, 보다 수월하게 문제를 풀 수 있었습니다.
- 마라톤이라서 공식 풀이와 유사한 아이디어로 풀어도 통과할 수 있게 시간 제한과 메모리 제한을 넉넉하게 드렸습니다.
- 수학적으로 값을 구하는 거 이외의 풀이는 아마 없을 겁니다....

# 어려운 이항계수 문제

$f(x) : 1 \sim x$  중에서  $p$ 의 배수가 아닌 것의 곱

$$n! = \prod_{i=0}^{\infty} p^{\left(\left[\frac{n}{p^{i+1}}\right]\right)} \times f\left(\left[\frac{n}{p^i}\right]\right) = \prod_{i=0}^{\infty} p^{\left(\left[\frac{n}{p^{i+1}}\right]\right)} \times \prod_{i=0}^{\infty} \frac{\left(\left[\frac{n}{p^i}\right]\right)!}{p^{\left(\left[\frac{n}{p^{i+1}}\right]\right)} \left(\left[\frac{n}{p^i}\right]\right)!} = p^a \times b$$

$$a = \prod_{i=0}^{\infty} p^{\left(\left[\frac{n}{p^{i+1}}\right]\right)}, b \equiv \prod_{i=0}^{\infty} f\left(\left[\frac{n}{p^i}\right]\right) \pmod{p^4}$$

$x = px_1 + x_2$  ( $0 \leq x_2 < p$ ) 이라고 둔다면

$$f(x) = \prod_{i=0}^{x_1-1} ((pi + 1) \times (pi + 2) \cdots \times (pi + p - 1)) \times \prod_{i=px_1+1}^{px_1+x_2} i$$

# 어려운 이항계수 문제

---

$\prod_{i=px_1+1}^{px_1+x_2} i$  값은  $O(p)$  안에 구할 수 있습니다.

$$((pi + 1) \times (pi + 2) \cdots \times (pi + p - 1))$$

$$\begin{aligned} &\equiv \left( (p-1)! + pi(p-1)! \left( \sum_{j=1}^{p-1} j^{-1} \right) + p^2 i^2 (p-1)! \left( \sum_{j=1}^{p-1} \sum_{z=j+1}^{p-1} (jz)^{-1} \right) \right. \\ &\quad \left. + p^3 i^3 (p-1)! \left( \sum_{j=1}^{p-1} \sum_{z=j+1}^{p-1} \sum_{w=z+1}^{p-1} (jzw)^{-1} \right) \right) \pmod{p^4} \end{aligned}$$

$$\left( \sum_{j=1}^{p-1} j^{-1} \right) = \varepsilon_1', \left( \sum_{j=1}^{p-1} \sum_{z=j+1}^{p-1} (jz)^{-1} \right) = \varepsilon_2', \left( \sum_{j=1}^{p-1} \sum_{z=j+1}^{p-1} \sum_{w=z+1}^{p-1} (jzw)^{-1} \right) = \varepsilon_3', \left( \sum_{j=1}^{p-1} j^{-2} \right) = E_2, \left( \sum_{j=1}^{p-1} j^{-3} \right) = E_3$$

# 어려운 이항계수 문제

---

$$\varepsilon_0' \equiv (p-1)!, \quad \varepsilon_2' \equiv \frac{(\varepsilon_1')^2 - E_2}{2}, \quad \varepsilon_3' \equiv \frac{(\varepsilon_1')^3 - 3\varepsilon_1'E_2 + 2E_3}{6}$$

$$((pi+1) \times (pi+2) \cdots \times (pi+p-1)) \equiv (\varepsilon_0' + pi\varepsilon_0'\varepsilon_1' + p^2i^2\varepsilon_0'\varepsilon_2' + p^3i^3\varepsilon_0'\varepsilon_3')(mod\ p^4)$$

Let  $\varepsilon_0 \equiv \varepsilon_0'(mod\ p^4)$ ,  $\varepsilon_1 \equiv \varepsilon_0'\varepsilon_1'(mod\ p^4)$ ,  $\varepsilon_2 \equiv \varepsilon_0'\varepsilon_2'(mod\ p^4)$ ,  $\varepsilon_3 \equiv \varepsilon_0'\varepsilon_3'(mod\ p^4)$

$$f(x) \equiv \prod_{i=0}^{x_1-1} (\varepsilon_0 + pi\varepsilon_1 + p^2i^2\varepsilon_2 + p^3i^3\varepsilon_3) \times \prod_{i=px_1+1}^{px_1+x_2} i \quad (mod\ p^4)$$

$$\prod_{i=0}^{x_1-1} (\varepsilon_0 + pi\varepsilon_1 + p^2i^2\varepsilon_2 + p^3i^3\varepsilon_3) \equiv \varepsilon_0^{x_1} \times \prod_{i=0}^{x_1-1} (1 + pi\varepsilon_1' + p^2i^2\varepsilon_2' + p^3i^3\varepsilon_3')$$

# 어려운 이항계수 문제

---

$$\prod_{i=0}^{x_1-1} (1 + pi\varepsilon_1' + p^2i^2\varepsilon_2' + p^3i^3\varepsilon_3') \equiv$$

$$\left( 1 + p\varepsilon_1' \sum_{j=0}^{x_1-1} j + p^2 \left[ \varepsilon_2' \sum_{j=0}^{x_1-1} j^2 + (\varepsilon_1')^2 \sum_{j=0}^{x_1-1} \sum_{z=j+1}^{x_1-1} jz \right] + \right. \\ \left. p^3 \left[ \varepsilon_3' \sum_{j=0}^{x_1-1} j^3 + \varepsilon_1' \varepsilon_2' \sum_{j=0}^{x_1-1} j \left( \sum_{z=0}^{x_1-1} z^2 - j^2 \right) + (\varepsilon_1')^3 \sum_{j=0}^{x_1-1} \sum_{z=j+1}^{x_1-1} \sum_{w=z+1}^{x_1-1} jzw \right] \right)$$

# 어려운 이항계수 문제

---

$$\text{Let } \sum_{j=0}^{x_1-1} j = \frac{(x_1 - 1)x_1}{2} \equiv S_1, \sum_{j=0}^{x_1-1} j^2 = \frac{(x_1 - 1)x_1(2x_1 - 1)}{6} \equiv S_2, \sum_{j=0}^{x_1-1} j^3 = \left(\frac{(x_1 - 1)x_1}{2}\right)^2 \equiv S_3$$

$$\prod_{i=0}^{x_1-1} (1 + pi\varepsilon_1' + p^2i^2\varepsilon_2' + p^3i^3\varepsilon_3') \equiv \\ \left( 1 + p\varepsilon_1'S_1 + p^2 \left( \varepsilon_2'S_2 + (\varepsilon_1')^2 \left[ \frac{S_1^2 - S_2}{2} \right] \right) + p^3 \left( \varepsilon_3'S_3 + \varepsilon_1'\varepsilon_2'[S_1S_2 - S_3] + (\varepsilon_1')^3 \left[ \frac{S_1^3 - 3S_1S_2 + S_3}{6} \right] \right) \right)$$

- 위와 같이 계산될 수 있으며, 시간 복잡도는  $O(p \log_p n + p \log_2 p)$ 가 됩니다.

# 어려운 이항계수 문제

---

- \_\_int64로 문제를 해결하시고 싶다면, 아래와 같은 사실들을 사용하시면 됩니다.
  - $a \equiv a_1 + a_2 p^2 \pmod{p^4}$ ,  $b \equiv b_1 + b_2 p^2 \pmod{p^4}$ ,  $0 \leq a_1, a_2, b_1, b_2 < p^2$
  - $ab \equiv a_1 b_1 + p^2(a_1 b_2 + a_2 b_1) \pmod{p^4}$
  - Let  $c \equiv (a_1 b_2 + a_2 b_1) \pmod{p^2} \rightarrow ab \equiv a_1 b_1 + p^2 c \pmod{p^4}$
- $\varphi(p^4) = p^3(p - 1)$ ,  $a^{\varphi(p^4)-1} \equiv a^{-1} \pmod{p^4}$ ,  $(ab)^{-1} \equiv (a^{-1}) \times (b^{-1}) \pmod{p^4}$
- $(ab)^{-1} \equiv (a^{-1}) \times (b^{-1}) \pmod{p^4}$  이 사실을 통해서
- $O(p \log_p n + p \log_2 p)$ 였던 시간 복잡도를 대략  $O\left(\frac{p}{10} \log_p n + p \log_2 p\right)$  정도로 만들 수 있습니다.