완전 기초 스터디!

여섯 번째 모임

키-파

June 22, 2018

목차

질의응답 (w5-h1)

- **80점**, 80점, 60점, 60점
- 시간 $O(n + q)$, 메모리 $\Theta(\max a_i)$ 풀이:
  1. 크기가 10^8 인 거대한 참/거짓 배열 `c`를 선언합니다
  2. 수열을 입력받았을 때 `c[`a_i`]`의 값을 참으로 만듭니다
  3. 쿼리를 처리할 때 `c[`q_i`]`의 값이 참이면 `Yes`, 아니면 `No`를 출력합니다
- 각 `c[i]`에 대해 필요한 bit 수는 1 bit이고, 10^8 bits $\approx$ 11.92 MB이므로 메모리 제한(16MB)을 아슬아슬하게 넘지 않는 것을 확인할 수 있습니다

질의응답 (w5-h1)

- 시간 $O(n \log n + q \log n)$, 메모리 $\Theta(n)$ 풀이:
  1. 수열을 입력받은 뒤 정렬합니다 ($O(n \log n)$)
  2. 각 쿼리를 $O(\log n)$ 에 해결하여, $O(q \log n)$ 에 나머지 문제를 해결합니다:
     1. 현재 수열에 원소가 없으면 `No`를 출력합니다
     2. 현재 수열의 중앙을 봅니다
     3. 중앙의 수가 q_i 와 같으면, `Yes`를 출력합니다
     4. 중앙의 수가 q_i 보다 크면, 왼쪽 반만을 대상으로 1로 돌아가 다시 탐색합니다
     5. 중앙의 수가 q_i 보다 작으면, 오른쪽 반만을 대상으로 1로 돌아가 다시 탐색합니다
- 정렬을 어떻게 $O(n \log n)$ 만에 할 수 있을까요?
- 쿼리를 $O(\log n)$ 만에 해결하는 알고리즘은 왜 옳을까요?

일억의 법칙 (w5-h2)

- **80점**, 40점
- 시간 $O(n^2)$ (10점) 풀이: 나와 있는 코드를 긁어서...
- 시간 $O(n)$ ($\geq$30점) 풀이: 다음 식을 이용합니다

$$\sum_{j=1}^{i} ij = \frac{i^2(i+1)}{2}$$

- 원하는 만큼 점수를 받지 못했다면, 모든 부분에서 overflow를 확인합시다
  - 자료형이 `int`인 변수 두 개를 곱할 수 **없습니다**
  - 자료형이 `long long int`인 변수 세 개를 연이어 곱할 수 **없습니다**
  - 코드는 대강 이런 꼴이 될 것입니다:

```
(long long int)a * b % mod * c % mod *···
```

일억의 법칙 (w5-h2)

- 시간 $O(1)$ (100점) 풀이: 다음 식을 이용합니다

$$\sum_{i=1}^{n}\sum_{j=1}^{i} ij = \frac{1}{2}\left(\left(\frac{n(n+1)}{2}\right)^2 + \frac{n(n+1)(2n+1)}{6}\right)$$
$$= \frac{1}{24}n(n+1)(n+2)(3n+1)$$

  - 공대생의 친구 wolframalpha!
- 역시나 원하는 점수를 받지 못하였다면, overflow를 확인합시다

검증 vs 계산 (w5-h3)

- **100점**, 80점
- 의도는 counting sort입니다: 수열의 각 원소가 10^6 이하의 양수로 매우 작음
- 저보다 wikipedia의 Sorting Algorithm 문서가 설명을 더 잘합니다
- side note:
  - quick sort는 이름과 다르게 최악의 경우 아주 느립니다 ($O(n^2)$)
    - 상수를 줄이는 대신 정렬 시간이 n^2 이 되는 확률을 $\frac{1}{n!}$ 으로 낮추는 방법
    - PS에서는 이런 정렬을 저격하는 test case가 들어올 수 있기에 보통 어떤 경우에도 $O(n \log n)$ 이 보장된, 랜덤 수열에서는 상수 배 느리지만 모든 수열에 대해 $O(n \log n)$ 을 보장하는 merge sort나 heap sort를 사용합니다
  - `#include <algorithm>`을 쓰면 `sort`라는 함수를 사용할 수 있습니다
    - 사용법은 `sort(a, a+n)`: `a`는 배열 이름, `n`은 배열 크기
    - $O(n \log n)$ 시간을 **보장합니다**

전위 표현식 (w5-h4)

- **75점**, 5점
- 문제를 보고 검색을 해서 자료구조를 찾을 생각을 하셨으면 안 됩니다!
- 수 하나도 전위 표현식이라 하면, 전위 표현식은 다음 형태 중 하나입니다:
  - n
  - + e_1 e_2
  - - e_1 e_2
  - * e_1 e_2

전위 표현식 (w5-h4)

- 풀이: 표현식 e를 입력받아서 계산하는 함수 `eval`을 만든다고 하면,
  1. 수를 하나 입력받습니다
  2. 음이 아닌 정수 n이라면, 그것이 표현식의 결과가 됩니다
  3. 아니라면 뒤이어 나오는 두 개의 표현식을 평가합니다:
     $$v_1 = \texttt{eval}(), v_2 = \texttt{eval}()$$
  4. 각각의 연산자에 맞게 $v_1 + v_2, v_1 - v_2, v_1 v_2$ 중 하나를 돌려줍니다
- 틀렸을 때 집중적으로 봐야 할 부분:
  - **overflow**: 두 `int`는 곱할 수 없습니다!
  - **negative division**: `w2-h8`을 기억하시나요?
    - 뺄셈이 들어간 나머지는 코드가 대강 이런 꼴이 됩니다:

```
(a + mod - b) % mod
```

더하자! (w5-h5)

- **제출 기록 없음**
- 풀지 말라고 만든 문제가 맞습니다
  - 조교분이 못 풀었기 때문에...
- $a_1, a_2, \cdots, a_{i-1}, a_i, a_{i+1}, \cdots, a_n$ 으로 만들 수 있는 수들의 집합은 $a_1, a_2, \cdots, a_{i-1}, (a_1 \oplus a_i), a_{i+1}, \cdots, a_n$ 으로 만들 수 있는 수들의 집합과 **같습니다**
- $\geq$30점 풀이: 모든 조합을 다 해 보면 됩니다...
  - 시간은 $O(n \cdot 2^n)$ 이며, $n = 20$ 일 때 이 값은 $\approx$2000만이 되므로 일억의 법칙을 위배하지 않습니다
  - 모든 조합을 다 해 보기 위해 비트 연산자를 활용하면 좋습니다

더하자! (w5-h5)

- 100점 풀이: $s := 0$으로 두고, 다음 작업을 30번 시행한 다음 s가 정답이 됩니다
  1. 가장 큰 수 m을 찾습니다
  2. $a_i := \min(a_i, a_i \oplus m) \quad \forall i = 1, 2, \cdots, n$
  3. $s := \max(s, s \oplus m)$
- 왜 될까요?

더하자! (w5-h5)

대수학적 interpretation:

- 각각의 수는 $\mathrm{F} = \mathbb{Z}_2$ 의 vector F^{30} 에 대응합니다
- 목표는 이 vector들의 spanning set 중 가장 **큰** 원소를 찾는 것입니다
- 모든 vector space는 basis를 가지고, 이 spanning set은 F^{30} 의 subspace입니다
  - 이 vector space의 **basis** vector는 많아 봐야 30개
- 여기서 **큰** 순서는 vector의 앞에 1이 있을수록 큼
- vector들로 matrix를 만든 뒤 그 matrix의 row reduced echelon form을 생각합니다
  - 이 row reduced echelon form의 row space와 원래 matrix의 row space가 같습니다
  - [최초의 1]을 보고 이 vector를 고를지 말지 결정할 수 있음

정수

2-bit integer:

00	01	10	11
⋮	⋮	⋮	⋮
-8	-7	-6	-5
-4	-3	-2	-1
0	1	2	3
4	5	6	7
⋮	⋮	⋮	⋮

정수

32-bit integer:

$00\cdots00$	$00\cdots01$	$\cdots$	$01\cdots11$	$10\cdots00$	$\cdots$	$11\cdots11$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
-2^{32}	$-2^{32}+1$	$\cdots$	$-2^{31}-1$	-2^{31}	$\cdots$	-1
0	1	$\cdots$	$2^{31}-1$	2^{31}	$\cdots$	$2^{32}-1$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

덧셈/뺄셈/곱셈

- 덧셈/뺄셈의 경우 중간 계산 과정이 2^{31} 을 넘어도 최종 계산 결과만 `int` 범위 안에 들어오면 아무 문제 없음!
  - w2-h4(세 수 정렬하기)를 기억하시나요?
    2. 중간 수는 다음과 같이 구함

```
middle = (a + b + c) - (biggest + smallest)
```

  - a, b, c 각각이 10^9 까지 될 수 있으므로 `a + b + c`는 $3 \cdot 10^9 \geq 2^{31}$ $\rightarrow$ overflow!
  - 왜 잘 작동했을까?
- 곱셈의 경우도 마찬가지이지만 수의 절댓값이 커지므로 쓸모는 없음
  - 곱셈에서 overflow가 나면 일단 결과를 신뢰할 수 없음

연습

- 문제는 w6-p1입니다
  - hint: 수 a 뒤에 한 자리 수 b 붙이기: $10a + b$

나눗셈?

- 어려운 문제
  - 나누어 떨어지는 경우만 생각합니다
  - 왜?
- 2-bit integer에서 3으로 나누기:

00	01	10	11
⋮	⋮	⋮	⋮
-8	-7	-6	-5
-4	-3	-2	-1
0	1	2	3
4	5	6	7
⋮	⋮	⋮	⋮

- 표현이 같은 수는 나눈 후에도 결과가 같다!
- 예쁜 형태로 정리할 수는 없을까요?

나눗셈?

- 2-bit integer에서 2로 나누기:

00	01	10	11
⋮	⋮	⋮	⋮
-8	-7	-6	-5
-4	-3	-2	-1
0	1	2	3
4	5	6	7
⋮	⋮	⋮	⋮

  - 01은 2로 나눈 결과가 없음!
  - 어떤 표현은 나눈 결과가 없을 수도 있음

- 어떨 때 이 표현이 잘 정의될까?

- p로 나눈 나머지가 같으면 모두 같은 수로 정의합니다
- 덧셈/뺄셈/곱셈은 모두 연산 후 p로 나눈 나머지로 정의합니다
  - 익히 알고 있는 결합, 교환, 분배 법칙 등이 성립합니다

Theorem (Fermat's Little Theorem)

소수 p에 대해, a가 p의 배수가 아니면,

$$a^{p-1} \equiv 1 \pmod{p}$$

가 성립한다.

- note: a와 a^{p-2}를 곱하면 1이 된다!

$\mathbb{Z}_p$ 에서의 나눗셈

- a/b 를 $a \cdot b^{p-2}$ 로 **정의**합니다
  - 정의에 의해 $(a/b) \cdot b = a$ 가 됨
- 외워 두면 좋은 두 소수

$$\begin{aligned} p &= 1,000,000,007 &&= 10^9 + 7 \\ p &= 998,244,353 &&= 7 \cdot 17 \cdot 2^{23} + 1 \end{aligned}$$

- p 가 아주 큰 경우 a^{p-2} 을 어떻게 계산?

연습

- 문제는 w6-p2입니다