

“중등부 3번 / 고등부 2번. XOR 최대” 문제 풀이

작성자: 나정휘

부분문제 1

$N \leq 30$ 으로 입력이 매우 작으므로 S 에서 s_1 과 s_2 를 선택하는 $O(N^4)$ 가지 경우를 모두 확인하면 문제를 해결할 수 있다.

부분문제 2

두 가지 풀이를 설명한다.

풀이 1

음이 아닌 정수 N 개로 구성된 배열 A 에서 두 수를 골라 XOR한 값을 최대화하는 문제는 트라이(Trie)라는 자료구조를 이용해 $O(N \log(\max A_i))$ 에 해결할 수 있음이 잘 알려져 있다.

이 문제는 S 의 부분문자열 $O(N^2)$ 개 중 2개를 골라 XOR한 값을 최대화하는 문제이며, 각 부분문자열의 길이는 N 이하이므로 트라이(Trie)를 이용해 $O(N^3)$ 시간에 해결할 수 있다.

풀이 2

일반성을 잃지 않고 $|s_1| \geq |s_2|$ 라고 하자. 또한, S 에서 가장 먼저 등장하는 1의 위치를 x 라고 하자. 이때 인덱스는 1부터 시작한다.

관찰 1. s_1 이 S 의 접두어(prefix)인 경우만 생각해도 된다.

증명: s_1 이 S 의 접두어가 아닌 답이 있다고 가정하자. s_1 의 바로 앞 문자가 1이면 s_1 을 앞으로 한 칸 확장해서 값을 증가시킬 수 있으므로 모순이고, s_1 의 바로 앞 글자가 0이면 앞으로 한 칸 확장해도 답이 변하지 않는다. 따라서 s_1 이 S 가 접두어가 아니라면 접두어가 될 때까지 s_1 을 확장할 수 있다.

관찰 2. 두 부분문자열을 XOR해서 뒤에서부터 $N - x + 1$ 번째 비트보다 더 높은 비트를 켤 수 없다.

증명: $t \geq N - x + 2$ 번째 비트를 켜기 위해서는 $S[1 \cdots x - 1]$ 사이에 1이 존재해야 하지만, x 의 정의에 의해 $S[x]$ 보다 앞에 1이 존재할 수 없다.

관찰 3. s_1 은 $S[x \cdots N]$ 을 포함하는 형태만 고려해도 된다.

증명: s_1 이 $S[x \cdots N]$ 을 포함하지 않으면 $N - x + 1$ 번째 비트를 켜지 못함을 보이면 되고, 관찰 2와 비슷하게 증명할 수 있다.

세 가지 관찰을 종합하면 다음과 같은 결론을 얻을 수 있다.

정리 1. $s_1 = S$ 인 정답이 존재한다.

증명: 관찰 3에 의해 s_1 이 S 의 접미어(suffix)인 정답이 존재한다. 또한, 관찰 1에 의해 s_1 을 앞으로 확장한 더라도 답이 감소하지 않는다. 따라서 $s_1 = S$ 인 정답이 존재한다.

S 의 모든 부분문자열 s_2 를 보면서, S 와 s_2 를 XOR한 값 중 최댓값을 구하면 된다. 이 경우 s_2 로 가능한 부분문자열이 $O(N^2)$ 가지이고 XOR 연산을 수행하는 데 $O(N)$ 시간이 걸린다. 따라서 $O(N^3)$ 시간에 문제를 해결할 수 있다.

부분문제 3

두 가지 풀이를 설명한다.

풀이 1

부분문제 2의 두 번째 풀이를 트라이(Trie)를 이용해 최적화하면 $O(N^2)$ 시간에 해결할 수 있다.

풀이 2

편의상 S 에 0과 1이 모두 1개 이상 등장하는 경우만 생각하자. S 에 1이 등장하지 않으면 정답은 0이고, 0이 등장하지 않으면 정답은 S 의 마지막 비트를 0으로 바꾼 것이다.

S 에서 가장 처음 등장하는 1과 연속한 1의 개수를 C , 그 바로 뒤에 등장하는 0의 위치를 P 라고 하자.

관찰 4. s_2 의 첫 글자를 $S = s_1$ 의 P 번째 글자와 XOR 연산하는 정답이 존재한다.

증명: $S[P]$ 는 0이고 그 앞($S[P-1]$)에 1이 있으므로 XOR한 결과의 P 번째 비트를 1로 만드는 방법이 존재하며, 그 위치를 1로 만들지 않고 더 좋은 답을 만들 수 없다. 또한, s_2 의 첫 글자가 $S[P]$ 가 아닌 그보다 더 앞에 있는 문자와 매칭되더라도 답이 증가하지 않음은 쉽게 알 수 있다.

이 관찰을 이용하면 s_2 의 후보를 C 개로 줄일 수 있다. 예를 들어 $S = 011110ABCDE\dots$ 형태라면, S 와 s_2 를 매칭하는 방법은 아래 네 가지 뿐이다.

```
S = 011110ABCDE...
s2 =      11110A...
s2 =      1110AB...
s2 =      110ABC...
s2 =      10ABCD...
```

이와 같이 s_2 의 후보는 $C(\leq N)$ 개밖에 존재하지 않는다. 실제로 고려해야 하는 s_2 가 $O(N)$ 가지이므로 $O(N^2)$ 시간에 정답을 구할 수 있다.

부분문제 4

부분문제 3의 두 번째 풀이처럼 $S[P\dots N]$ 과 XOR한 값이 최대가 되는 s_2 를 찾는 방식으로 해결할 것이다.

A 가 주어졌을 때 $A \oplus B$ 를 최대화하는 B 를 찾는 것은 $\text{not}A \oplus B$ 를 최소화하는 B 를 찾는 것과 같다. 이러한 B 를 찾는 문제는 가능한 B 의 후보들 중 $\text{not}A$ 보다 작거나 같으면서 가장 큰 수, 또는 $\text{not}A$ 보다 크거나 같으면서 가장 작은 수 중 하나가 정답이다. 따라서 부분문자열의 사전 순 비교를 $O(T(N))$ 시간에 할 수 있으면 전체 문제를 전처리 시간을 제외하고 $O(N \times T(N))$ 시간에 해결할 수 있다.

문자열 $A \# \text{not}A$ 의 접미사 배열(Suffix Array)를 사용하면 전처리를 $O(N \log N)$, 부분문자열의 사전 순 비교를 $O(1)$ 에 할 수 있고, 문자열 해시와 이분 탐색을 사용하면 전처리를 $O(N)$, 부분문자열의 사전 순 비교를 $O(\log N)$ 에 할 수 있다. 두 방법 모두 전체 시간 복잡도는 $O(N \log N)$ 이다.

부분문제 5

부분문제 3의 두 번째 풀이를 최적화할 것이다. $S = 011110ABCDE\dots$ 형태라고 가정하자.

$S = 011110ABCDE\dots$

$s_2 = 11110A\dots$

$s_2 = 1110AB\dots$

$s_2 = 110ABC\dots$

$s_2 = 10ABCD\dots$

만약 $A = 1$ 이면 $s_2 = 10ABCD$ 로 확정할 수 있고, $A \neq 1, B = 1$ 이면 $s_2 = 110ABC$ 로 확정할 수 있다. 이와 같이 P 이후로 처음 나오는 1의 위치를 $P + i$ 라고 하면, s_2 는 $P - \min(C, i)$ 에서 시작하는 문자열로 확정할 수 있다. 따라서 정답이 되는 s_2 를 $O(N)$ 에 구할 수 있어 전체 문제를 $O(N)$ 시간에 해결할 수 있다.