‘‘바보 자물쇠’’ 문제 풀이

작성자: 구재현

서술의 편의를 위해 각 알파벳 소문자를 숫자에 대응시킨다. 예를 들어, `'a'` 는 0, `'b'` 는 1.. `'z'` 는 25와 같은 식이다. 등장할 수 있는 서로 다른 알파벳의 수를 Σ 라고 표기한다. 예를 들어, 만점 서브태스크에서 $\Sigma = 26$ 이다.

부분문제 1

자물쇠가 풀리는 경우는 총 $N+1$ 가지가 있는데, 맨 앞 i 개 문자가 0 이고 뒤 $N-i$ 개 문자가 1 인 경우이다. 각 경우에 대해서, 두 문자열이 다른 위치의 개수를 센 후, 그 최솟값을 출력하면 된다. 시간 복잡도는 $O(N^2)$ 이다.

부분문제 2

부분문제 1의 풀이를 최적화한다. 모든 $0 \le i \le N$에 대해, 맨 앞 i 개 문자 중 1 의 개수, 맨 뒤 $N-i$ 개 문자 중 0 의 개수를 빠르게 셀 수 있으면 된다. 이는 부분 합을 계산하여 해결할 수 있다. 매 쿼리마다 부분합을 계산한 후 최솟값을 계산하면, $O(QN)$ 시간에 문제가 해결된다.

부분문제 3

자물쇠가 풀리는 시점에서 1 이 등장하는 위치가 구간 $i, i+1, \ldots, j-1$ 을 이룬다고 하자. j 이후 위치에서는 2 가 등장할 것이고, $i-1$ 이전 위치에서는 0 이 등장할 것이다. $F(i)$ 를 $i-1$ 이전 위치에 0 이 적히도록 하는 최소 시간, $G(j)$ 를 j 이후 위치에 2 가 적히도록 하는 최소 시간이라고 정의하자. 답은 $F(i), G(j)$, 그리고 몇가지 부분합으로 표현할 수 있으며, F, G 역시 모두 부분합으로 계산할 수 있다. 식이 다소 복잡하여 자세한 설명은 생략한다. $O(QN)$ 시간에 문제가 해결된다.

부분문제 5

어떠한 문자 x 를 새로운 문자 y 로 바꾸는 데 드는 연산 수는 $|x-y|$ 임을 관찰하자.

동적 계획법을 사용한다. $dp[i][j]$ 를, i 번 문자를 j 이하의 어떤 값으로 설정했을 때 드는 최소 연산 수라고 정의하자. 상태 전이는 두 가지이다:

- i 번 문자를 j 로 설정한다: 이 경우의 연산 수는 $dp[i-1][j] + |j - S[i]|$ 이다.
- i 번 문자를 $j-1$ 이하의 값으로 설정한다: 이 경우의 연산 수는 $dp[i][j-1]$ 이다.

이 동적 계획법은 간단하게 $O(N\Sigma)$ 시간에 계산할 수 있다. 시간 복잡도는 $O(QN\Sigma)$ 이다.

다른 동적 계획법을 사용하면 $O(QN\Sigma^2)$ 나 $O(QN\Sigma \log N)$ 의 시간 복잡도가 나올 수 있다. 이 경우에는 부분문제 4가 통과되지만, 부분문제 5는 통과되지 않을 수 있다.

부분문제 6

KOI 2014 중등부에 출제된 *금광* 문제와 유사한 방법으로 해결할 수 있다.

세그먼트 트리 자료구조를 사용하자. 트리의 각 노드에는 다음과 같은 정보를 저장한다:

- 구간에 있는 0 의 개수
- 구간에 있는 1 의 개수
- 구간의 앞을 0 으로 채우고, 뒤를 1 으로 채웠을 때 필요한 최소 연산 수

두 노드가 주어졌을 때, 구간에 있는 0 의 개수와 1 의 개수는 쉽게 합칠 수 있다. 최소 연산 수를 합칠 때는, 0 과 1 의 *분기점* 을 어디로 설정하는 지에 따라 두 가지 경우가 있다. 만약 0 과 1 의 *분기점* 이 왼쪽 구간에서 형성된다면, 오른쪽 구간은 1 로 결정되니 추가되는 연산 수는 오른쪽 구간의 0 의 개수이다. *분기점* 이 오른쪽 구간에서 형성된다면, 같은 이유로 추가되는 연산 수는 왼쪽 구간의 1 의 개수이다. 이 둘 중 최솟값을 취하면 된다.

이러한 세그먼트 트리를 구성하면 갱신 쿼리를 효율적으로 처리할 수 있다. 시간 복잡도는 $O(N+Q\log N)$ 이다.

부분문제 7

부분문제 6의 풀이를 확장하면 부분문제 7을 해결할 수 있다. 세부적인 사항이 다소 복잡하여 설명은 생략한다. 부분문제 8의 풀이가 비효율적으로 구현되었을 경우에도 부분문제 7에서 점수를 얻을 수 있다.

부분문제 8

부분문제 6의 풀이를 일반화할 경우, 행렬과 유사한 형태의 구조를 얻을 수 있다. 세그먼트 트리 자료구조의 노드에, $\Sigma \times \Sigma$ 의 크기를 가지는 2차원 배열 $Cost[][]$ 를 정의하자. 이 때, $Cost[i][j]$ 는 구간의 맨 앞 원소를 i 로 설정하고, 맨 뒤 원소를 j 이하로 설정하며 해당 구간을 증가하게 만들려 할 때 필요한 최소 연산 수이다.

어떠한 부모 노드의 $Cost[i][j]$ 를 계산할 때는, 오른쪽 자식 노드의 맨 앞 원소 k 를 고정하면, 왼쪽 자식 노드의 $Cost[i][k]$ 와 오른쪽 자식 노드의 $Cost[k][j]$ 의 합 만큼의 연산 수가 필요함을 알 수 있다. 이를 모든 $i \le k \le j$ 에 대해 순회하며 최솟값을 구하면 된다. 시간 복잡도는 $O((N+Q\log N)\Sigma^3)$ 이다.

부분문제 9

만점을 받기 위해서는 전체 문제를 $\Sigma - 1$ 개의 부분문제 6 인스턴스로 변환시키는 아이디어가 필요하다. 모든 $1 \le i \le \Sigma - 1$ 에 대해, $bin(S,i)$ 를 $S_j \ge i$ 일 때 1 이고 아닐 때 0 인 길이 N 의 이진 문자열이라고 정의하자. 또한, $cost(S)$ 를 어떠한 수열 S 를 단조증가하게 만드는 최소 연산 수라고 하자. 다음 사실이 성립한다.

Theorem. $cost(S) = \sum_i cost(bin(S,i))$

Proof. $cost(S) \ge \sum_i cost(bin(S,i))$: $cost(S)$ 의 답을 주는 어떠한 최적해 $opt(S)$ 를 생각해 보자. 이제 각각의 i 에 대해 $bin(S,i)$ 를 $bin(opt(S),i)$ 로 변환할 경우, 이들 중 j 번 위치에서 다른 값을 가지는 i 의 개수는 정확히 $|S_j - opt(S)_j|$ 개임을 관찰할 수 있다. 고로 우변은 $cost(S)$ 초과일 수 없다.

$cost(S) \leq \sum_i cost(bin(S,i))$: 각 i 에 대한 최적해 $opt(bin(S,i))$ 를 생각해 보자. 또한, $zero_i$ 를 $opt(bin(S,i))$ 에 있는 0 의 개수라고 하자.

만약 $zero_i$ 가 단조증가 수열이라면, i 번 문자를 S 의 $[zero_i, zero_{i+1})$ 구간에 배정함으로서 $cost(bin(S,i))$ 의 최적해를 구성할 수 있다 ($zero_0 = 0, zero_\Sigma = N$). 이 수열은 증가하고, $|S_j - opt(S)_j|$ 는 이 수열 중 j 번 문자가 달라진 수열의 개수와 일치한다. 고로, 이 경우 $cost(bin(S,i))$ 의 합의 연산 수로 S 를 단조증가하게 변환했다고 할 수 있다.

$zero_i$ 가 단조증가 수열이 아니라고 하자. 즉, $zero_i > zero_{i+1}$ 인 위치 i 가 존재한다고 하자. 이 경우 $opt(bin(S,i))$ 와 $opt(bin(S,i+1))$ 의 최적해를 서로 바꿔도 답이 늘어나지 않음을 증명할 수 있다. $bin(S,i+1)$ 이 $bin(S,i)$ 보다 구간 $[zero_{i+1}, zero_i)$ 에서 더 많은 0 을 포함하고 있기 때문이다. 고로 이 경우 $zero_i$ 와 $zero_{i+1}$ 을 바꿔줄 수 있고, 버블 정렬의 요령으로 이를 반복해주면 연산 수에서 손해를 보지 않고 $zero_i$ 가 단조증가 수열이라고 가정할 수 있다. ■

Theorem에 의해, 부분문제 6의 풀이를 그대로 적용해서 문제를 해결할 수 있다. 시간 복잡도는 $O((N + Q \log N)\Sigma)$ 이다.

여담

사이클이 없는 방향성 그래프 $G = (V, E)$ 의 각 정점에 $N = |V|$ 개의 수 $A_1, A_2, \ldots, A_N$ 이 적혀 있을 때, G 의 모든 간선 $u \to v$ 에 대해 $A_u \leq A_v$ 를 만족하게끔 하는 문제를 **Isotonic Regression** 이라고 한다. 이 문제는, 그래프를 $i \to i+1$ 형태의 단순한 선 그래프로 생각하고, 숫자를 바꾸는 비용을 $|x - y|$ (L_1 metric) 으로 두었을 때의 Isotonic Regression을 푸는 문제라고 생각할 수 있다.

선 그래프의 L_1 Isotonic Regression은 잘 알려진 문제이고 (BOI 2004 Sequence), 경시대회에서 *Slope trick* 이라고 부르는 방법으로 해결된다. Leftist heap을 사용한 풀이도 있지만, 다소 복잡한 풀이이다. 이 문제에서 제안된 접근을 사용하면, *Slope trick* 과 완전히 다른 방향에서 $O(N \log N)$ 풀이를 얻을 수 있다.

이러한 접근은 기존 *Slope trick* 보다 더 직관적이고 (특히 최적해의 복원), 제한된 형태의 업데이트를 지원할 수 있으며, 더 다양한 문제를 해결할 수 있다는 장점이 있다. 만약에 Isotonic Regression을 $A_i \in \{0, 1\}$ 이라는 제약 조건 하에서 해결한다면, 트리에서는 DP를 사용하여 $O(N)$ 시간에, DAG에서는 Network flow 를 사용하여 $O(N^{1+o(1)})$ 시간에 해결할 수 있다. 이러한 풀이는 N 번 반복하여 원래 문제의 풀이로 복원되거나, A_i 가 작을 때 갱신을 지원하는 dynamic algorithm을 사용하여 (예를 들어, 트리의 경우에는 Top tree 를 사용하여 여러 갱신이 지원된다) 다양한 방향의 효율적인 알고리즘을 얻을 때 사용할 수 있다.

이 문제는 IOI 2023에 Call for tasks를 통하여 제안되었던 문제이다.