"고등부 3번. 오름차순" 문제 풀이

작성자: 이종영

부분문제 1

D_i를 $A_i \le A_{i+1}2^{D_i}$를 만족하는 최대의 정수라고 정의하자. 이 배열은 log 함수를 이용해 $O(N)$에 구하거나, 직접 반복문을 이용해 $O(N \log A)$에 구할 수 있다. $i = l$부터 $r-1$까지 순회하며, $D'_j = D_j$로 초기화된 배열에 대해, $D'_i > 0$이라면 답에 D'_i를 더하고, D'_{i+1}에 D'_i를 더하는 방법으로 시간 복잡도 $O(NQ)$에 문제를 해결할 수 있다. 자세한 내용은 "두 배" 문제의 풀이를 참고하면 좋다.

부분문제 2

$[l, r]$에서의 답과 $[l, r+1]$에서의 답을 구하는 과정은 마지막 D'_r에 대한 처리를 제외하면 동일하다. 각 l에 대해 r을 증가시키면서 모든 답을 전처리하면 시간 복잡도 $O(N^2 + Q)$에 문제를 해결할 수 있다.

부분문제 4

부분문제 1의 풀이에서 $D'_i > 0$ 대신 $D'_i \ge 0$을 기준으로 해도 변함이 없음을 알 수 있다. 또, $A_i \ge A_{i+1}$이라는 조건 안에서는 $D_i \ge 0$을 만족하므로 $D'_i \ge 0$을 만족한다. 즉, 답은 $(r-l)D_l + \cdots + D_{r-1}$이다. $(r-l)D_l + \cdots + D_{r-1} = r(D_l + \cdots + D_{r-1}) - (lD_l + \cdots + (r-1)D_{r-1})$을 만족하므로 D_i의 부분합 및 iD_i의 부분합 배열을 구해둔 후 $O(1)$에 쿼리를 답할 수 있다.

부분문제 5

구간 내의 첫 2를 기준으로 1과 2로 최종 값이 정해진다. 구간 내의 첫 2는 동적 프로그래밍을 통해 각 i에 대해 인덱스가 i 이상인 가장 앞의 2의 위치를 구해둔다면 $O(1)$에 구할 수 있다. 실제 비용은 1의 개수를 저장하는 부분합 배열을 이용하면 $O(1)$에 구할 수 있다.

부분문제 6

부분문제 5의 풀이를 응용해, 최종 값이 2^k가 되는 구간을 각 k에 대해 구하는 방식으로 해결할 수 있다.

부분문제 7

nxt_i를 $i+1$부터 순회했을 때, $D'_j < 0$이 되는 최소 j라고 정의하자. D_i의 부분합 배열 S에 대해, nxt_i는 $S_j < S_i$이며 $i < j$를 만족하는 최소 j이다. 이러한 경우 nxt 배열을 스택을 통해 $O(N)$에 구할 수 있음이 잘 알려져 있다. 구체적으로는, i를 N부터 1까지 순회한다. 현재 스택의 크기가 1 이상이며 스택의 top j에 대해 $S_j \ge S_i$를 만족한다면 그렇지 않을 때까지 스택에서 pop 연산을 해 준다. 이후 스택의 크기가 1 이상이라면 스택의 top j가 nxt_i가 된다. (스택의 크기가 0이라면 nxt_i는 존재하지 않으며, 편의상 이 경우 $nxt_i = N$으로 정의하자.) 마지막으로 스택에 i를 push하면 된다.

이제 $[l, r]$쿼리에 대해 D' 값이 0 이상으로 유지되는 구간들은 $[l, nxt_{l-1}-1], [nxt_{l-1}+1, nxt_{nxt_{l-1}}-1], \cdots$이다. $cost_i$를 $[i+1, nxt_i - 1]$ 구간에 대한 비용으로 정의하자. 부분문제 4의 풀이를 활용하면 $cost$ 배열을 구할 수 있다. 이제 각 쿼리에 대해 $i = l-1$로 시작해, $nxt_i \le r-1$인 동안 답에 $cost_i$를 더한 후, $i = nxt_i$로 이동한다. 마지막으로 $[i+1, r-1]$ 구간에 대한 비용을 더해주면 쿼리에 대한 답을 구할 수 있다.

이 방식은 얼핏 보기에는 nxt 배열을 타고 가는 것이 최악의 경우 $O(N)$이 소요된다고 생각할 수 있지만, $2A_{i+1} \le A_{nxt_i+1}$을 만족하기 때문에 사실은 $O(\log A)$의 시간 복잡도를 가진다. 따라서 문제를 $O(N +$

$Q \log A)$ 시간 복잡도에 해결할 수 있다. 모범 코드의 경우 D 배열을 $O(N \log A)$에 계산해 $O((N+Q) \log A)$의 시간 복잡도를 가지며, 이 방식 또한 문제를 해결하는 데에 충분하다.

nxt배열을 타고 가는 것이 $O(\log A)$가 소요된다는 관찰을 하지 않거나, nxt_i를 $S_j \leq S_i$이며 $i < j$를 만족하는 최소 j로 정의한 경우에도 문제를 해결할 수 있다. 이 경우 sparse table을 활용한 $O((N+Q) \log N)$ 방식, jump table을 활용한 $O(N + Q \log N)$ 방식 등이 가능하다.

두 방식을 합치면 거의 $O(N+Q)$와 유사한 $O((N+Q) \log \log A)$, $O(N + Q \log \log A)$의 시간 복잡도로도 문제를 해결할 수 있다.