"진화 2 (evolution2)" 문제 풀이

작성자: 구재현

부분문제 1

올바른 번호 배정은 유일하며, 최초 생명체에서 시작해서 유일한 자식을 반복해서 따라가면 이 번호 배정을 구할 수 있다.

부분문제 2

최초 생명체를 0번으로 배정하는 모든 배정이 올바르다. 즉, $N-1$ 개의 수를 정렬하는 문제로 해석할 수 있다. 일반적인 Merge Sort를 구현하면 만점을 받을 수 있다.

부분문제 3

$A(v)$를, 생명체 v 를 루트로 한 서브트리의 정점들을 탄생 시점 기준으로 정렬한 수열이라고 하자. v 가 리프 노드인 경우, $A(v) = \{v\}$ 이다. v 가 리프 노드가 아닌 경우, v 의 두 자식 $\{w_1, w_2\}$ 에 대해 $A(w_1), A(w_2)$ 를 계산하고, Merge Sort의 요령으로 두 수열을 합쳐주면 만점을 받을 수 있다.

부분문제 4

정점 v 에 대해, $P(v)$ 를 가능한 모든 $A(v)$ 의 개수의 경우의 수라고 하자. v 의 자식을 $\{w_1, w_2, \ldots, w_k\}$ 라고 했을 때, 다음과 같은 점화식이 성립한다:

$$P(v) = \begin{cases} 1 & \text{if } k = 0 \\ \binom{|A(w_1)|+\ldots+|A(w_k)|}{|A(w_1)|,\ldots,|A(w_k)|} \prod P(w_i) & \text{otherwise} \end{cases}$$

길이 L, S 의 두 수열을 $K \log_2(\binom{L+S}{L})$ 번 이하의 `compare` 연산으로 합칠 수 있는 알고리즘이 있다고 하자. $\binom{x_1+x_2+\ldots+x_k}{x_1,x_2,\ldots,x_k} = \binom{x_1+x_2}{x_2} \times \binom{x_1+x_2+x_3}{x_3} \times \ldots \binom{x_1+x_2+\ldots+x_k}{x_1,x_2,\ldots,x_k}$ 이기 때문에, 이 알고리즘을 통해서 $A(w_1), A(w_2)$ 를 합쳐준 후 그 수열에 $A(w_3), A(w_4), \ldots$ 를 순서대로 합쳐주면 전체 문제를 $K \log_2 P$ 번의 `compare` 연산으로 해결할 수 있다. 위와 같은 알고리즘이 있다면, 자식의 개수나 자식을 어떤 순서로 합쳐주는가는 전혀 중요하지 않고, 두 개의 수열을 적은 `compare` 연산으로 합쳐줄 수만 있으면 된다.

문제는 길이 L, S 의 두 수열을 최적의 연산으로 합칠 수 있는 알고리즘을 찾는 것으로 환원되었다. 일반성을 잃지 않고 $L \geq S$를 가정하자. 길이 L의 수열을 *큰 수열*, 길이 S 의 수열을 *작은 수열* 이라고 하자. 버킷을 만들듯이, 큰 수열을 S 개의 길이 $\lceil \frac{L}{S} \rceil$ 의 수열로 분할한다. 분할된 큰 수열의 맨 앞 원소를 모아서 작은 수열과 Merge해 주면, $2S-1$ 번의 연산으로 각 작은 수열의 원소가 큰 수열의 어떠한 부분에 들어가는지를 알 수 있다. 해당 부분에서 어떤 특정한 위치에 들어가는지는 이분 탐색을 통해서 계산할 수 있다. 이를 사용하면 $2S + \log_2(\frac{L}{S}+1)$ 번의 `compare` 연산으로 문제를 해결할 수 있고 80점 이상을 받을 수 있다.

위 풀이에서 적용할 수 있는 한 가지 최적화는, 각 분할된 수열의 크기가 정확히 $\frac{L}{S}+1$ 일 필요가 없다는 것이다. 분할된 수열에서의 위치를 찾기 위해서 어차피 이분 탐색을 해야 하기 때문에, 분할된 수열의 크기를 2 의 지수승으로 맞춰준 후 그에 따라서 분할의 개수를 줄이면 이분 탐색의 쿼리 횟수를 그대로 유지하면서 Merge Sort의 쿼리 횟수를 줄일 수 있다. 분할된 수열의 크기를 2^i 라고 두었을 때, 정확히 몇 번의 `compare` 연산이 필요한지를 간단한 수식으로 계산할 수 있다. 이 수식을 최소화하는 i 를 구한 후, 이 i에 따라서 큰 수열을 분할하고 위 알고리즘을 적용하면 된다. 이 전략을 사용하면 100점을 받을 수 있다. 프로그램을 통해서 이 전략을 모든 $1 \leq S \leq L, S+L \leq 10000$ 에 대해서 검증해 보면, $K \leq 1.37$ 이 계산된다.

큰 K 가 나오는 경우는 대다수 S, L 이 굉장히 작기 때문에, 이 경우에 대해서 최적 전략을 하드 코딩하는 식으로 더 작은 K 를 얻을 수도 있다. 하지만 이는 만점을 받기 위해서 필요한 최적화는 아니다.

부분 점수를 받을 수 있는 다른 풀이 몇 가지를 소개한다: 작은 수열의 각 원소에서 이분 탐색을 하는데 특정 원소의 위치가 결정되면 다른 원소의 이분 탐색 범위를 결정된 위치로 한정시켜서 탐색 범위를 줄이는 풀이가 있다. 각 원소를 탐색하는 순서를 분할 정복처럼 중간 / 그 다음 중간과 같이 결정하면, 분할 정복 최적화와 비슷한 코드를 얻을 수 있으며, 이 방법도 $O(S \log_2 \frac{L}{S})$ 번의 쿼리를 사용함을 증명할 수 있다. 이 방법은 잘 최적화할 경우 만점을 얻을 수 있다. Treap을 사용하여 수열을 관리한 후, 이 글과 같은 방법으로 Treap을 합쳐주어도 $O(S \log_2 \frac{L}{S})$ 번의 쿼리를 사용한다. 이 방법은 60점 이상을 받는다.

Remark 1. Stirling's inequality에 의해:

$$(L+S)\log_2(L+S) - L\log_2(L) - S\log_2(S) = \log_2(\binom{L+S}{S}) + O(\log_2 L + \log_2 S - \log_2(L+S))$$

우변에 붙은 Error term인 $O(\log_2 L + \log_2 S - \log_2(L+S))$ 는 충분히 큰 L, S 에 대해 작다. (실제 증명에서 무시할 수 없지만) 이를 무시하고, $(L+S)\log_2(L+S) - L\log_2(L) - S\log_2(S) = \log_2(\binom{L+S}{S})$ 라고 가정하자. 이 가정하에 $S + S\log(1+\frac{L}{S})$ 번의 쿼리를 사용하는 알고리즘이 최적임을 증명한다. 모든 $x \in (0, 1]$ 에 대해 $x \le \log_2(1+x)$이다. 고로:

$$\begin{aligned}
\frac{S}{L} &\le \log_2(1+\frac{S}{L}) \\
S &\le L\log_2(S+L) - L\log_2(L) \\
S + S\log_2(S+L) &\le (S+L)\log_2(S+L) - L\log_2(L) \\
S + S\log_2(S+L) - S\log_2(S) &\le (S+L)\log_2(S+L) - L\log_2(L) - S\log_2(S) \\
S + S\log_2(1+\frac{L}{S}) &\le (S+L)\log_2(S+L) - L\log_2(L) - S\log_2(S)
\end{aligned}$$

이와 같은 방식으로 $O(S + S\log(1+\frac{L}{S}))$ 번의 쿼리를 사용하는 알고리즘이 asymptotically optimal함을 증명할 수 있다.

Remark 2. 입력이 Rooted tree가 아니라 DAG인 경우에도, $O(\log P)$ 번의 쿼리를 사용하여 숨겨진 순서를 복원하는 알고리즘이 존재한다. (Simpler Optimal Sorting from a Directed Acyclic Graph) 이 경우, P 는 DAG의 가능한 위상 정렬의 경우의 수에 대응된다. 트리에서 이 알고리즘을 적용할 경우, 최소 $K \ge 2$ 가 되기 때문에 만점을 받기는 어렵다.