"초등부 4번. 상자 보관" 문제 풀이

작성자: 반딧불

부분문제 1

이 부분문제에서는 $N \leq 6$으로 매우 작다. 각 상자에 대해, 해당 상자가 포함하는 다른 상자를 지정(또는 아무 상자도 포함하지 않는다고 지정)하는 모든 경우를 브루트 포스를 통해 모두 시도해볼 수 있다. 구현에 따라 $O(N^N)$ 또는 $O(N!)$ 등의 시간 복잡도가 나올 수 있다.

부분문제 2

이 부분문제에서는 $s_i = c_i + 1$로, 모든 상자의 보관 용량이 그 크기보다 1만큼 작다. 따라서 두 상자의 크기가 같으면 두 상자의 보관 용량도 같게 된다.

어떤 상자에도 포함되지 않은 상자 a_1이 다른 상자 a_2를 직접 포함하고, 상자 a_2는 다른 상자 a_3을 직접 포함하고, ..., 상자 a_{k-1}는 상자 a_k를 직접 포함하고, 상자 a_k는 다른 상자를 포함하지 않는 관계가 존재할 때, 상자 $a_1, a_2, \cdots, a_k$를 **상자 체인**이라고 정의하자. 자명하게도, 한 상자 체인에 존재하는 상자들은 s_i 값이 모두 달라야 함을 알 수 있다. 따라서, 문제의 답, 즉 모든 상자를 포함하는 데 필요한 최소 상자 체인의 수는 임의의 v에 대해, $v = s_i$인 i의 개수보다는 커야 한다. 이러한 i의 개수를 cnt_v라고 정의하자.

앞에서 보았듯이 답은 최소

$$\max_{1 \leq v} cnt_v$$

이상이어야 한다. 따라서, 만약 정확히 위 개수만큼의 상자 체인으로 모든 상자를 포함할 수 있는 해가 존재한다면, 해당 해가 최적해이다. 부분문제 2의 경우, 이러한 최적해가 존재함을 간단히 알 수 있다.

부분문제 2의 조건에 의해, 상자 k개가 임의로 주어졌을 때, 모든 상자가 $s_i = c_i + 1$을 만족하고 모든 상자의 s_i 값이 다르다면, 이 상자들로 상자 체인을 만드는 것이 항상 가능하다. 만약 s_i 값이 같은 상자들이 존재한다면, s_i 값이 같은 cnt_{s_i}개의 상자들에 임의의 순서로 1부터 cnt_{s_i} 사이의 *라벨*을 하나씩 부여하자. 이제 라벨이 같은 상자들끼리 모으면 해당 상자들의 s_i 값은 모두 다르므로, 각각의 묶음으로 상자 체인을 하나씩 만들 수 있다. 라벨의 최댓값은 cnt_v의 최댓값과 같으므로, 이렇게 만든 해가 최적해이다.

이제 필요한 일은 상자가 하나씩 추가될 때 cnt_v의 최댓값을 관리하는 것뿐이다. s_i 값이 하나 추가될 때마다 `std::map`과 같은 자료구조에 개수를 늘려서 관리하는 방식으로 해결할 수 있다. 시간 복잡도는 $O(N \log N)$이다.

부분문제 3

어떤 상자 a가 다른 상자 b를 직접 포함할 때, 쌍 (a, b)의 **포함 관계**가 있다고 정의하자. 만약 N개의 상자를 p개의 상자 체인으로 넣었다면, 총 $N - p$개의 포함 관계가 형성되었다는 뜻과 같다. 문제의 조건에 의해, 쌍 (a, b)의 포함 관계가 형성될 수 있는 조건은 $s_b \leq c_a$이다.

문제는 상자 체인의 개수를 최소화할 것을 요구하고 있지만, 이것을 반대로 생각하면, 포함 관계의 수를 최대화해야 하는 문제로도 해석하는 것이 가능하다. 따라서, $s_b \leq c_a$ 조건을 만족하게 최대 몇 개의 포함 관계를 만들 수 있는지 찾으면 된다.

답에 대한 이분 탐색(parametric search)을 이용하자. N개의 가방이 있을 때, p개의 포함 관계를 만들 수 있는지 분석하자. 이를 위해 어떤 s_b 값들과 c_a 값들을 고를지 생각해야 한다. 그리디하게 생각하면, 가장

작은 p개의 s_b 값과 가장 큰 p개의 c_a 값을 사용하는 것이 최적이다. 저 값들을 쌍을 짓는 최적의 방법은, p개의 s_b 값들을 오름차순으로 정렬하고, p개의 c_a 값들도 오름차순으로 정렬한 뒤, 크기 순으로 매칭하는 것이다. 모든 매칭에서 $s_b \le c_a$가 성립한다면 p개의 포함 관계를 만들 수 있다는 뜻이다.

가방의 개수가 N개일 때 위 방법을 이용하면 $O(N \log N)$ 시간에 p의 최댓값을 구할 수 있고, 답을 알아낼 수 있다. 따라서 위 방법을 이용하면 전체 문제를 $O(N^2 \log N)$ 시간에 해결할 수 있다.

부분문제 4

부분문제 3과는 다른 방식으로 접근해 보자. 각 상자 i를 수직선 위에 (c_i, s_i)에 해당하는 구간으로 표현할 수 있다. 이때 하나의 상자 체인은 서로 분리된 (쌍마다 교집합이 공집합인) 구간의 집합으로 생각할 수 있다.

다양한 예제를 시도하며 규칙을 찾아 보면 부분문제 2와 비슷하게 문제의 답이 수직선 위의 한 지점이 구간에 포함되는 최대 횟수와 같음을 알 수 있다. 즉, 임의의 v에 대해, $v \in (c_i, s_i)$인 i의 개수의 최댓값과 같음을 알 수 있다. 일단 이 사실$(*)$을 참이라고 가정하고, 문제를 해결해 보자.

길이 100의 배열 A를 관리한다. 어떤 상자 (c_i, s_i)가 새로 들어올 경우, $A_{c_i}, A_{c_i+1}, \cdots, A_{s_i-1}$에 1씩 더한다. 해당 연산을 처리한 후, 답은 $\max_i A_i$의 값과 같다. 따라서 전체 문제를 $O(N \max s_i)$ 시간에 해결할 수 있다.

이제 풀이의 핵심이 되는 $(*)$를 증명해 보자. 답이 $\max_i A_i$ 이상이어야 함은 자명하다. 한 위치에서 겹치는 구간이 적어도 $\max_i A_i$개 있다는 뜻인데, 이 구간들이 나타내는 상자는 모두 다른 상자 체인에 포함되어야 하기 때문이다. 이제 해당 개수만큼의 상자 체인으로 모든 상자를 포함할 수 있음을 보인다.

다음과 같은 그리디한 접근을 생각한다. 모든 구간을 시작점의 오름차순으로, 시작점이 같다면 끝점의 오름차순으로 정렬한다. 구간을 왼쪽부터 하나씩 보며, 적당한 상자 체인에 넣는 과정을 반복할 것이다. 선택 방법은 다음과 같다.

- 만약 현재 존재하는 상자 체인 중 현재 구간을 넣을 수 있는 것이 있다면, 그러한 상자 체인을 아무거나 선택해 해당 구간을 넣는다.
- 만약 현재 존재하는 상자 체인 중 현재 구간을 넣을 수 있는 것이 없다면 (즉, 현재 구간이 (c_i, s_i)일 때, 모든 상자 체인에 대해 해당 상자 체인에서 가장 큰 상자의 크기 $s_j > c_i$가 성립할 때), 현재 보고 있는 구간만을 포함하는 새로운 상자 체인을 하나 만든다.

위 과정을 통해 상자들을 상자 체인으로 분배할 때 상자 체인의 총 개수가 $\max_i A_i$를 넘지 않음을 보인다. 귀류법을 사용하자. 어떤 상자 (c_i, s_i)를 보는 순간, 새로운 상자 체인을 만들게 되어 상자 체인의 총 개수가 $\max_i A_i + 1$개가 되었다고 하자. 이것은 이미 존재하는 $\max_i A_i$개의 상자 체인이 모두 현재 상자의 c_i 값보다 s_i 값이 크다는 것을 의미한다. 그러나 상자의 정렬 순서를 고려하였을 때, 이것은 현재 새로 보고 있는 상자의 c_i 값에 대해, $c_i - \epsilon$이 포함된 구간의 개수가 $\max_i A_i + 1$개임을 의미한다. 이는 A_i의 정의에 모순이다.

따라서, 위와 같은 그리디한 접근으로 상자 체인이 총 $\max_i A_i$개가 되도록 구성할 수 있음을 보였고, 풀이의 정당성이 증명되었다.

부분문제 5

부분문제 4의 풀이는 세그먼트 트리를 이용해 쉽게 최적화할 수 있다. 세그먼트 트리를 이용해 구간 $[c_i, s_i - 1]$에 1을 더하고, 전체 최댓값을 찾는 문제로 바꾸는 것이 가능하다. 이렇게 하면 전체 문제를 $O(N \log N)$에 해결할 수 있다.