

“고등부 1번. 야구 시즌” 문제 풀이

작성자: 윤교준

각각의 지역리그 내에서 두 팀을 뽑는 경우의 수는 $\binom{M}{2}$ 이고, 지역리그가 N 개이므로, 같은 지역리그끼리의 경기 수는 $N \cdot \binom{M}{2} \cdot A$ 이다.

서로 다른 두 지역리그를 뽑는 경우의 수는 $\binom{N}{2}$ 이고, 두 지역리그가 결정되면 경기할 팀을 고르는 경우의 수는 M^2 이므로, 다른 지역리그끼리의 경기 수는 $\binom{N}{2} \cdot M^2 \cdot B$ 이다.

따라서, 총 경기 수는 $N \cdot \binom{M}{2} \cdot A + \binom{N}{2} \cdot M^2 \cdot B$ 이다.

$A = B \cdot k$ 를 대입하고, 총 경기 수가 D 이하여야 한다는 조건을 표현하면 아래 부등식을 얻을 수 있다:

$$\frac{NM((M-1)k + (N-1)M)}{2} \cdot B \leq D$$

따라서, 다음 부등식을 얻을 수 있다:

$$B \leq \frac{2D}{NM((M-1)k + (N-1)M)}$$

위의 부등식을 만족하는 자연수 B 가 존재하는지 판별하는 것과, 만족하는 최대 자연수 B_{\max} 를 찾는 작업은 $O(1)$ 에 해결할 수 있다.

“고등부 2번. 초직사각형” 문제 풀이

작성자: 김준겸

부분문제 1

N 의 크기가 10 이하로 매우 작다. 따라서 모든 경우의 수를 탐색한 뒤, 부피를 실제로 계산하여 카드를 사용하는 방법 중 부피가 최대가 되는 것을 고르면 된다. 시간복잡도는 $O(N!)$ 이다.

부분문제 2

모든 T_i 가 A 이기 때문에, B, C, D 는 변화하지 않는다. 선택한 카드의 U_i 를 차례대로 t_1, t_2, \dots, t_K 라고 하면 총 부피는 $(A_0 + t_1 + t_2 + \dots + t_K)$ 가 된다. 따라서 카드를 선택하는 순서는 답에 영향을 줄 수 없고, 부피가 큰 순서대로 K 개의 카드를 골라 사용하면 된다. 결과적으로 U_i 를 내림차순으로 정렬한 뒤 앞쪽부터 K 개의 카드를 출력하면 되고, 시간복잡도는 $O(N \log N)$ 이다.

부분문제 3

T_i 가 A, B, C, D 인 카드를 각각 K_A, K_B, K_C, K_D 개 골랐다고 하자. 부분문제 2에서와 마찬가지로 선택한 카드에 대해 부피를 식으로 나타내 보면, 카드를 선택하는 순서가 총 부피에 영향을 주지 않는다는 사실을 알 수 있다. 또한 만약 T_i 가 X 이고 U_i 가 Y_0 인 카드를 골랐을 때 T_i 가 X 이고 U_i 가 $Y > Y_0$ 인 카드를 고르지 않았다면, 전자를 후자로 대체해서 더 큰 부피를 얻을 수 있음을 알 수 있다. 따라서 각 카드 종류에 대해, 카드 종류마다 사용할 개수 K_X 를 결정했다면 가장 큰 순서대로 K_X 개를 사용하는 것이 최적이다.

따라서 $K_A + K_B + K_C + K_D = K$ 가 되도록 K_A, K_B, K_C, K_D 를 정할 수 있는 모든 경우의 수를 생각할 수 있다. $K_D = K - K_A - K_B - K_C$ 이기 때문에, K_A, K_B, K_C 를 정하면 K_D 는 자동으로 정해지게 된다. 따라서 $O(N^3)$ 가지의 경우에 대해 카드의 종류 X 마다 내림차순으로 K_X 개의 카드를 사용한 뒤 총 부피를 계산할 수 있고, 종류별로 카드를 미리 $O(N \log N)$ 시간에 전처리해두면 이 과정은 $O(N)$ 에 수행할 수 있다. 시간복잡도는 $O(N^4)$ 이다. 정렬해놓은 각 카드 종류에 대해 우리가 사용하는 값이 앞쪽부터 K_X 개 원소의 합이라는 점을 이용하여 미리 Prefix Sum을 계산해놓으면, 복잡도를 $O(N^3)$ 으로 줄일 수 있다. 두 복잡도 모두 부분문제 3을 해결하기에 충분하다.

부분문제 4

모든 U_i 가 1이므로, T_i 가 A, B, C, D 인 카드를 각각 K_A, K_B, K_C, K_D 개 골랐다고 하면 총 부피는 $(A_0 + K_A)(B_0 + K_B)(C_0 + K_C)(D_0 + K_D)$ 이다. 별도의 제약이 없는 실수 범위에서 $x + y + z + w = C$ 일 때 $xyzw$ 를 최대화하는 값은 $x = y = z = w$ 일 때이기 때문에, 이 경우에도 가능한 $(X_0 + K_X)$ 들을 비슷하게 맞추어야겠다는 직관을 가질 수 있다. 따라서 남은 카드가 존재하여 증가시킬 수 있는 종류의 카드 중 가장 $(X_0 + K_X)$ 가 작은 종류의 카드를 하나 사용하는 것을 반복하는 욕심쟁이 전략을 세울 수 있다. 이는 실제로 최적이다.

부피를 최대화했을 때 $A < B + 1 \leq C \leq D$ 이고, 이 때 사용한 카드 중 A 종류가 아닌 카드가 있으면서 A 종류 카드가 남아있다고 가정하자. 그러면 카드의 적용 순서는 최종 부피와 관련이 없으므로 A 종류가 아닌 카드를 마지막에 사용했다고 생각해도 무방하다. 그러면 A 종류 카드를 사용했을 때 증가하는 부피는 BCD 이고 다른 종류 카드를 사용했을 때 증가하는 부피는 이보다 반드시 작으므로 A 종류 카드로 기존에 사용한 카드를 대체할 수 있다. 따라서 최종 결과에서 A 가 최솟값(마지막 카드를 제외하였을 때 $A = B$ 였던 경우는

고려하지 않음)인 경우에 A 종류인 카드는 반드시 모두 사용했거나, 또는 A 종류가 아닌 카드를 사용한 적이 없어야 한다.

정리해서, 카드의 개수가 충분하다면 $A = B$ 이거나 $A + 1 = B$ 여야 하고, 그렇지 않다면 모든 A 종류 카드를 사용했어야 한다. A 를 고정한 뒤 같은 방법으로 B 와 C 에도 같은 논리를 적용할 수 있다. 결과적으로 부피를 최대화 했을 때 최댓값과 2 이상 차이 나는 원소는 해당하는 종류의 카드를 모두 소모한 뒤여야 하고, 따라서 값을 증가시킬 수 있는 원소 중 최솟값을 가지는 원소를 반복적으로 증가시키는 것이 최적이다. 시간복잡도는 $O(N)$ 이다.

부분문제 5

먼저 K_C 는 최대 $O(N)$ 개의 값만을 가지므로 K_C 를 고정할 수 있다. 그러면 $K_A + K_B = K - K_C$ 로 계산할 수 있다. K_C 가 고정된 상황에서 K_A 와 K_B 가 얼마의 값을 가져야 하는지는 빠르게 계산할 수 있다.

A 종류의 카드와 B 종류의 카드의 U_i 를 내림차순으로 정렬해놓은 수열을 각각 $A_1, A_2, \dots, A_a, B_1, B_2, \dots, B_b$ 라고 하자. $f(x)$ 와 $g(x)$ 를 가능한 정수 범위의 x 에 대해서 $f(x) = (A_0 + A_1 + \dots + A_x)$, $g(x) = (B_0 + B_1 + \dots + B_{P-x})$ 라고 하자. 그러면 $f(x)g(x)$ 가 위로 볼록하다. 그 이유는 $h(x) = f(x)g(x)$ 라 할 때 $h(x) - h(x-1) \geq h(x+1) - h(x)$ 이기 때문이다. 이는 간단한 식 전개를 통해 증명할 수 있다.

따라서 $h(x)$ 에 대한 삼분 탐색을 수행하거나, $h(x+1) - h(x)$ 에 대한 이분 탐색을 수행하는 방법으로 주어진 $K_A + K_B$ 에 대한 최적을 구할 수 있다. 총 $O(N)$ 개의 K_C 에 대해 각 $O(\log N)$ 시간에 $K_A + K_B$ 에 대한 최적이 찾아지므로, 총 시간복잡도는 $O(N \log N)$ 이다.

별해. 부분문제 6에서 부피의 증가율을 계산하는 대신, 직접 부피를 계산해 보고 가장 많이 증가하는 카드를 선택하는 욕심쟁이 전략을 수립할 수 있다. 이 때 가능한 부피의 범위는 64비트 정수를 넘어가므로, Python/Java에서 지원하는 큰 수 자료형을 사용하지 않았거나 C/C++에서 64비트 초과 정수 자료형을 직접 구현하지 않았다면 항상 최대 부피가 $10^{36} = 10^{18}$ 보다 작은 부분문제 5에서만 정답을 받을 수 있다.

부분문제 6

부분문제 4에서 욕심쟁이 전략이 성립하는 이유에 대해 다른 관점에서 분석해볼 수 있다. K_A 를 1 증가시키면, 총 부피는 $(A_0 + K_A)(B_0 + K_B)(C_0 + K_C)(D_0 + K_D)$ 에서 $(A_0 + K_A + 1)(B_0 + K_B)(C_0 + K_C)(D_0 + K_D)$ 이 되므로 부피가 $\frac{A_0 + K_A + 1}{A_0 + K_A}$ 배 증가한다. 따라서 아무 카드도 사용하지 않은 상태에서 종류 X 의 카드를 하나 사용하면 $\frac{X_0 + 1}{X_0}$ 배 부피가 늘어나고, 두 개 사용하면 $\frac{X_0 + 2}{X_0 + 1}$ 배 부피가 늘어나고... 와 같이 증가한다. 따라서 부피의 증가분이 아닌 증가비는 다른 종류의 카드를 몇 장 사용했느냐와 완전히 독립적이다. 이런 성질들을 고려하였을 때 $\frac{X_0 + K_X + 1}{X_0 + K_X}$ 를 최대로 하는 X 는 현재 $X_0 + K_X$ 가 가장 작은 X 이고, 따라서 $X_0 + K_X$, 즉 현재의 최솟값을 반복적으로 증가시키는 것이 최적이다.

부분문제 6에 대해서도 동일한 원리를 적용할 수 있다. T 종류이고 U 의 값을 가지며, 같은 종류의 카드를 내림차순으로 정렬하였을 때 i 번째에 오는 카드를 하나 사용하면 총 부피는

$$\frac{T_0 + T_1 + \dots + T_{i-1} + T_i}{T_0 + T_1 + \dots + T_{i-1}} = 1 + \frac{T_i}{T_0 + T_1 + \dots + T_{i-1}}$$

배만큼 증가한다. 이는 역시 다른 종류의 카드를 얼마나 사용했느냐에 완전히 독립적이고 T_i 를 미리 내림차순으로 정렬했기 때문에 부피의 증가율도 갈수록 감소한다는 사실을 알 수 있다. 따라서 이 값을 모든 카드에 대해서 계산해둔 뒤 정렬하여 값이 가장 큰 K 개의 카드를 선택할 수 있다. 분모를 미리 전처리해두거나 계산 과정에서 유지할 수 있으므로, 시간복잡도는 $O(N \log N)$ 이다.

정렬할 때 두 분수를 비교해야 하는데, 양 변에 공통적으로 존재하는 1을 소거하면 분자는 10^5 이하이고, 분모는 $(N+1) \times 10^6 \leq 2 \times 10^{11} + 10^6$ 이하이므로 64비트 정수만을 이용해 $\frac{p_1}{q_1} < \frac{p_2}{q_2}$ 를 $p_1q_2 < p_2q_1$ 과 같이 바꾸어 비교할 수 있다.

“고등부 3번. 공통 부분 수열 확장” 문제 풀이

작성자: 강태규

부분문제 1

W 의 확장으로 가능한 수열은 W 의 앞에 문자를 붙이는 경우, 뒤에 문자를 붙이는 경우로 총 52가지이다. 52가지의 모든 확장된 W 에 대해 W 가 X 와 Y 의 부분수열인지 확인하여 부분문제를 해결할 수 있다. 확장된 W 는 길이가 2이므로, X 에서 처음에 나오는 $W[1]$ 의 위치보다 마지막에 나오는 $W[2]$ 의 위치가 더 뒤일 경우 W 는 X 의 부분수열이다.

부분문제 2

X 의 i 번째 문자부터 j 번째 문자까지의 부분 문자열을 $X[i:j]$ 라고 쓰고, $Xp[i]$ 를 $W[1:i]$ 가 $X[1:p]$ 의 부분수열이 되는 최소의 p 라고 쓰자. $Xp[i]$ 는 $X[Xp[i-1] + 1:|X|]$ 에서 처음으로 나오는 $W[i]$ 의 위치와 같으므로 Xp 배열을 $O(|X|)$ 에 구할 수 있다.

$Xp[|W|]$ 가 존재할 경우 W 는 X 의 부분수열이다. $26(|W|+1)$ 가지의 모든 확장된 W 에 대해 W 가 X 와 Y 의 부분수열인지 확인하여 부분문제를 해결할 수 있다. 시간복잡도는 $O(26|W|(|X| + |Y|))$ 이다.

부분문제 3

Xp 와 비슷하게 $Xs[i]$ 를 $W[i:|W|]$ 가 $X[p:|X|]$ 의 부분수열이 되는 최대의 p 라고 쓰자. Xs 배열도 $O(|X|)$ 에 구할 수 있다.

0 이상 $|W|$ 이하의 어떤 i 가 존재하여 $X[Xp[i]+1:Xs[i+1]-1]$ 와 $Y[Yp[i]+1:Ys[i+1]-1]$ 사이에 공통된 문자 c 가 존재한다면, W 는 $W[1:i] + c + W[i+1:|W|]$ 로 확장 가능하다. 가능한 모든 i 에 대해 $(Xs[i+1]-Xp[i]-1)(Ys[i+1]-Yp[i]-1)$ 번의 비교로 공통된 문자가 있는지 확인할 경우 시간복잡도는 $O(|W||X||Y|)$ 이다.

단, 공통된 문자 c 를 찾는 즉시 중단하면 비교 횟수가 감소한다. $Xp[i+1] < Xs[i+1]$ 와 $Yp[i+1] < Ys[i+1]$ 가 동시에 성립한다면, $X[Xp[i]+1:Xs[i+1]-1]$ 와 $Y[Yp[i]+1:Ys[i+1]-1]$ 에 모두 $W[i+1]$ 가 존재하여 중단할 수 있게 된다. 그 전까지는 $Xp[i+1] = Xs[i+1]$ 와 $Yp[i+1] = Ys[i+1]$ 중 하나가 반드시 성립한다. 비교 횟수를 자세히 쓰면 다음과 같다.

$$\begin{aligned}
 & \sum_i (Xs[i+1] - Xp[i])(Ys[i+1] - Yp[i]) \\
 & \leq \sum_i (|X| + |Y|)(Xp[i+1] = Xs[i+1] ? Xs[i+1] - Xp[i] : Ys[i+1] - Yp[i]) \\
 & = \sum_i (|X| + |Y|)(Xp[i+1] = Xs[i+1] ? Xp[i+1] - Xp[i] : Yp[i+1] - Yp[i]) \\
 & \leq (|X| + |Y|)^2
 \end{aligned}$$

따라서 시간복잡도는 $O((|X| + |Y|)^2)$ 이다.

부분문제 4

X_p, X_s, Y_p, Y_s 배열은 모두 증가한다. $X[X_p[i]+1:X_s[i+1]-1]$ 에서 각 문자가 등장하는 횟수를 알고 있다면, $X[X_p[i+1]+1:X_s[i+2]-1]$ 에서 각 문자가 등장하는 횟수는 $X[X_s[i+1]:X_s[i+2]-1]$ 에 등장하는 문자의 횟수를 더하고, $X[X_p[i]+1:X_p[i+1]]$ 에 등장하는 문자의 횟수를 빼서 구할 수 있다. $X[X_s[i+1]:X_s[i+2]-1]$ 와 $X[X_p[i]+1:X_p[i+1]]$ 의 길이를 모두 더하면 $O(|X|)$ 이므로 모든 i 에 대해 $X[X_p[i]+1:X_s[i+1]-1]$ 와 $Y[Y_p[i]+1:Y_s[i+1]-1]$ 에서 각 문자가 등장하는 횟수를 시간복잡도 $O(|X| + |Y|)$ 에 관리하여 답을 구할 수 있다.