"팀 만들기 (teambuilding)" 문제 풀이

작성자: 김세빈

부분문제 1

남학생과 여학생을 고르는 모든 방법을 확인하고 최댓값을 구하면 된다. 시간 복잡도는 $O(NMQ)$ 이다.

부분문제 2

앞으로의 설명의 편의를 위해 여학생들의 번호를 반전하자. 0번 여학생은 $M-1$번 여학생이 되고, $M-1$번 여학생은 0번 여학생이 된다.

시나리오의 수 Q가 작으므로 하나의 시나리오를 효율적으로 해결하는 것에 집중하자. $L_1[k]=0$, $R_1[k]=N-1$, $L_2[k]=0$, $R_2[k]=M-1$인 경우를 생각하도록 한다.

i번 남학생과 j번 여학생이 팀을 이뤘을 때, 팀의 실력을 $C[i][j]$라 정의하자. 학생들의 발상 능력이 증가하고 구현 능력이 감소한다는 사실을 이용하면 아래와 같은 성질을 관찰하고 증명할 수 있다.

$0 \le i_1 < i_2 \le N-1$이고 $0 \le j_1 < j_2 \le M-1$일 때, 다음이 성립한다.

$$C[i_1][j_1] + C[i_2][j_2] \ge C[i_1][j_2] + C[i_2][j_1]$$

증명은 다음과 같이 할 수 있다.

$$\begin{aligned}
&(C[i_1][j_1] + C[i_2][j_2]) - (C[i_1][j_2] + C[i_2][j_1]) \\
&\quad = (A_1[i_1] + A_2[j_1])(B_1[i_1] + B_2[j_1]) + (A_1[i_2] + A_2[j_2])(B_1[i_2] + B_2[j_2]) \\
&\qquad - (A_1[i_1] + A_2[j_2])(B_1[i_1] + B_2[j_2]) - (A_1[i_2] + A_2[j_1])(B_1[i_2] + B_2[j_1]) \\
&\quad = (A_1[i_1]B_2[j_1] + A_2[j_1]B_1[i_1] + A_1[i_2]B_2[j_2] + A_2[j_2]B_1[i_2]) \\
&\qquad - (A_1[i_1]B_2[j_2] + A_2[j_2]B_1[i_1] + A_1[i_2]B_2[j_1] + A_2[j_1]B_1[i_2]) \\
&\quad = (A_1[i_2] - A_1[i_1])(B_2[j_2] - B_2[j_1]) + (A_2[j_1] - A_2[j_2])(B_1[i_1] - B_1[i_2]) \\
&\quad \ge 0
\end{aligned}$$

위 성질을 만족하는 배열을 Monge array라고 한다. Monge array는 최적화 문제에서 흔하게 나타나며, 다양하게 활용할 수 있는 성질들을 가지고 있어 중요하다. 일례로, Monge array인 $C[i][j]$는 다음과 같은 성질을 가지고 있다:

i가 주어졌을 때, $C[i][j]$의 값이 최대가 되도록 하는 j 중 가장 작은 것을 j_i^*라 하자. 만약 $0 \le i_1 < i_2 \le N-1$이라면, $j_{i_1}^* \le j_{i_2}^*$가 성립한다. 다시 말해, 팀의 실력을 최대화하는 여학생의 번호는 남학생의 번호가 증가함에 따라 단조증가한다.

증명은 다음과 같이 할 수 있다. 귀류법을 사용해 $j_{i_1}^* > j_{i_2}^*$라 가정하자. 정의에 의해, $C[i_1][j_{i_1}^*] > C[i_1][j_{i_2}^*]$이고, $C[i_2][j_{i_2}^*] \ge C[i_2][j_{i_1}^*]$이다 (등호에 유의하자). 이때 $C[i_1][j_{i_2}^*] + C[i_2][j_{i_1}^*] < C[i_1][j_{i_1}^*] + C[i_2][j_{i_2}^*]$가 되어 Monge array의 조건에 모순이다.

모든 $0 \le i \le N-1$에 대해 j_i^*를 구하여 부분문제 2를 해결할 수 있다. j_i^*가 단조증가하므로, 이는 분할 정복 최적화(Divide and Conquer Optimization)라는 최적화 기법을 통해 가능하다.

시간 복잡도는 $O(Q(N+M)\log N)$이다.

부분문제 3

부분문제 2에서와 같이 모든 $0 \le i \le N-1$에 대해 j_i^*를 구하면, i가 주어졌을 때 $C[i][j]$의 최댓값을 구할 수 있다. 이 값을 $D[i]$라 하자. 부분문제 3에서 각 시나리오의 답은 D에서의 구간 최대 쿼리 (Range Maximum Query)를 통해 구할 수 있다. 시간 복잡도는 $O((N+M+Q)\log N)$이다.

부분문제 4

먼저, 모든 시나리오에서 0번 남학생을 고를 수 있는 경우를 생각해 보자. 즉, 모든 $0 \le k \le Q-1$에 대해 $L_1[k]=0$인 경우를 생각해 보자. 또한, $R_1[k]=n$인 시나리오만 고려하자. 우리의 목표는 n을 0부터 $N-1$까지 증가시키며 모든 시나리오의 답을 구하는 것이다.

이번에는 j가 주어졌을 때, $C[i][j]$의 값이 최대가 되도록 하는 $0 \le i \le n$ 중 가장 작은 것을 i_j^*라 하자. 이때, 부분문제 2에서 관찰한 성질에 의해 i_j^*는 j에 대해 단조증가한다. i_j^*의 값이 동일한 j들을 하나의 구간으로 묶어 생각하면, i_j^*의 정보를 순서쌍들의 배열 $(a_1,b_1),\cdots,(a_t,b_t)$로 나타낼 수 있다. 이것의 의미는 구간 $(b_{l-1},b_l]$에 속하는 j들에 대해 i_j^*의 값이 a_l이라는 것이다 ($1 \le l \le t$). 이때 $b_0=0$, $b_t=M-1$로 정의된다. $a_1<\cdots<a_t$, $b_1<\cdots<b_t$가 성립한다는 사실을 알 수 있다.

어떤 j가 주어졌을 때 i_j^*의 값을 구하기 위해서는 이 순서쌍들의 배열에서 이분 탐색을 하면 된다. 따라서 하나의 시나리오에 대한 답을 $O(\log M)$에 구할 수 있다.

이제 n이 증가함에 따라 이 순서쌍들의 배열이 어떻게 변화하는지 관찰하면, 배열의 오른쪽 끝부분에서만 변화가 일어난다는 사실을 알 수 있다. 따라서 순서쌍들을 스택으로 관리하여 효율적으로 처리할 수 있다. i_j^*의 값이 동일한 구간을 계산하는 데는 이분 탐색이 필요하다. 따라서 n을 0부터 $N-1$까지 증가시키는 데 필요한 시간 복잡도는 $O(N\log M)$이다.

이제 $L_1[k]=0$인 경우, 즉 시나리오가 남학생들의 prefix에 대해 주어지는 경우 문제를 $O((N+Q)\log M)$에 해결할 수 있다. Prefix에 대해 문제를 해결할 수 있고 이를 뒤집으면 suffix에 대해서도 마찬가지이므로, 분할 정복 (Divide and Conquer) 기법을 활용해 부분문제 4를 $O(N\log N\log M+Q\log M)$에 해결할 수 있다. 이는 잘 알려진 방법이며, 간단히 설명하면 다음과 같다.

- $0 \le m \le N-1$인 m을 고르고, m을 포함하는 모든 시나리오에 대한 답을 구하자. 즉, $L_1[k] \le m \le R_1[k]$를 만족하는 시나리오에 대한 답을 구하자.
- 구간 $[L_1[k],R_1[k]]$에 대한 답을 얻기 위해서는 두 구간 $[L_1[k],m]$과 $[m,R_1[k]]$에 대해 문제를 해결한 뒤 최댓값을 취하면 된다. 이는 각각 m에서 시작하는 prefix와 suffix이므로, 위에서 설명한 방법과 같이 해결할 수 있다.
- 이제 남은 시나리오들은 모두 m을 포함하지 않는 시나리오들이다. 즉, 구간 $[0,m-1]$ 또는 $[m+1,N-1]$에 완전히 포함되는 시나리오들이다. 따라서 두 구간 $[0,m-1]$과 $[m+1,N-1]$에 대해 같은 과정을 반복하면 된다.
- 각 과정에 대해, m을 구간의 중점으로 잡으면 전체 시간 복잡도에 $O(\log N)$이 곱해진다.

분할 정복 대신 세그먼트 트리를 활용하여도 유사한 방법으로 해결할 수 있다. 이 경우 시간 복잡도는 $O(N\log M+Q\log N\log M)$이고, fractional cascading이라는 최적화 기법을 통해 $O(N\log M+Q(\log N+\log M))$로 줄일 수 있다.

부분문제 5

부분문제 4에서는 모든 시나리오에 대해 가능한 여학생의 번호가 유일하므로, 즉 $L_2[k] = R_2[k]$이므로, 순서쌍들의 배열 $(a_1, b_1), \cdots, (a_t, b_t)$에서 단 한 번의 이분 탐색을 통해 시나리오의 답을 구할 수 있었다. 전체 문제에서는 가능한 여학생의 번호 또한 구간으로 주어지기 때문에, 이를 해결하기 위해서는 몇 가지 과정이 더 필요하다.

순서쌍들의 배열 $(a_1, b_1), \cdots, (a_t, b_t)$에서 이분 탐색을 통해 $L_2[k] \le b_l$이 성립하는 최소의 $1 \le l \le t$과 $b_r < R_2[k]$가 성립하는 최대의 $1 \le r \le t$을 구할 수 있다. 이때, 시나리오를 통해 주어진 구간 $[L_2[k], R_2[k]]$를 다음과 같이 여러 개의 구간으로 분할할 수 있다. 각 구간에 대한 답 중 최댓값이 시나리오의 답이 된다.

$$[L_2[k], b_l], \ \ (b_l, b_{l+1}], \ \ \cdots, \ \ (b_{r-1}, b_r], \ \ (b_r, R_2[k]]$$

각 구간에 대한 답은 다음과 같이 구할 수 있다.

- 구간 $[L_2[k], b_l]$에서는 팀의 실력을 최대화하는 남학생의 번호가 a_l로 고정된다. 이는 부분문제 4에 주어진 상황에서 남녀를 바꾼 것과 같다. 따라서 부분문제 4와 동일한 방법으로 해결할 수 있다.
- 구간 $(b_r, R_2[k]]$에서는 팀의 실력을 최대화하는 남학생의 번호가 a_{r+1}로 고정된다. 마찬가지로 부분문제 4와 동일한 방법으로 해결할 수 있다.
- $l < s \le r$에 대해, 구간 $(b_{s-1}, b_s]$에서 팀의 실력을 최대화하는 남학생의 번호는 a_s이다. a_s번 남학생과 구간 $(b_{s-1}, b_s]$에서 고른 여학생이 구성할 수 있는 팀의 실력의 최댓값을 $D[s]$로 정의하자. $D[s]$의 값은 위에서와 마찬가지로 부분문제 4와 동일한 방법으로 해결할 수 있다. 한편 $D[s]$는 시나리오, 즉 $L_2[k]$와 $R_2[k]$의 영향을 받지 않는 값이다. 따라서 순서쌍들의 배열 $(a_1, b_1), \cdots, (a_t, b_t)$을 구성하는 과정에서 D 값을 미리 전처리할 수 있다. 스택에서 push 또는 pop이 일어날 때마다 해당하는 위치의 값을 다시 계산해 주면 된다. 이제 시나리오에 대한 답을 구할 때는 D에서 구간 $[l+1, r]$에 대한 구간 최대 쿼리(Range Maximum Query)를 통해 구간 $(b_l, b_{l+1}], \ \cdots, \ (b_{r-1}, b_r]$을 한번에 처리할 수 있다.

위에서는 다루지 않았지만 $l = r$이거나 $l = r + 1$인 경우도 존재할 수 있어 따로 처리해야 한다. 자세한 설명은 생략한다.

부분문제 4에서는 하나의 시나리오에 대한 답을 $O(\log M)$에 구할 수 있었다. 부분문제 4의 풀이를 사용하는 횟수를 세어 보면, 전체 문제를 해결하는 시간복잡도가 $O((N+M)\log N \log M + Q \log N)$라는 것을 알 수 있다.

분할 정복 대신 세그먼트 트리를 활용하여도 유사한 방식으로 해결할 수 있다. 이 경우 시간 복잡도는 $O((N+M)\log N \log M + Q \log N \log M \log(N+M))$ 또는 $O((N+M)\log N \log M + Q \log^2(N+M))$이다.