23Q4 s&L Lighter Cup Editorial

by

small_lighter 그룹

문제		의도한 난이도	출제자
A1	짜고 치는 가위바위보 (Small)	**Easy**	`lighter, kidw0124, devluyten`
A2	짜고 치는 가위바위보 (Large)	**Medium**	`lighter, kidw0124, devluyten`
B1	커플 파괴자 민욱이 (Small)	**Medium**	`asdf1705`
B2	커플 파괴자 민욱이 (Large)	**Hard**	`asdf1705`
C1	Mike Sees The Storm (Small)	**Hard**	`rustiebeats`
C2	Mike Sees The Storm (Large)	**Hard**	`rustiebeats`
D1	OR & XOR (Small)	**Hard**	`kongum`
D2	OR & XOR (Large)	**Challenging**	`kongum`

A. 짜고 치는 가위바위보

`bruteforce, backtracking (Small)`
`DP (Large)`
출제진 의도 – **Easy** / **Medium**

- 제출 52번, 맞은 사람 36명 (정답률 73.08%) (Small)
- 제출 15번, 맞은 사람 12명 (정답률 86.67%) (Large)
- 처음 푼 사람: **xiaowuc1**, 5분 (Small) / **xiaowuc1**, 9분 (Large)
- 출제자: `lighter, kidw0124, devluyten`

A1. 짜고 치는 가위바위보 (Small)

- Smallant가 내고자 했던 가위바위보의 문자열의 길이는 최대 20으로 제한되어 있습니다.
- 따라서 가위바위보의 정보를 일부분만 취하는 경우의 수는 최대 $2^{20}-1 = 1048575$가지 입니다.
- 조건을 만족하도록 모든 경우의 수를 탐색하며 문제를 해결하면 됩니다.
- 시간 복잡도는 총 $\mathcal{O}\left(2^N\right)$이 됩니다.

A2. 짜고 치는 가위바위보 (Large)

- 가위바위보의 문자열의 길이는 최대 $1,000,000$ 입니다.
- 때문에, Small의 접근 방법을 그대로 적용하면 시간 초과가 납니다.
- 효율적인 시간 관리를 위해 DP를 사용해봅시다.

A2. 짜고 치는 가위바위보 (Large)

- 문제에서는 조건이 성립하도록 라운드를 "없애는" 방법의 수를 따졌습니다.
- 하지만 앞에서부터 문자열을 순회하며 해당 라운드를 추가할 것인지, 추가하지 않을 것인지의 여부를 따지는 문제로 바꿔서 생각해도 무방합니다.
- 다음과 같은 상태 9개를 정의합시다.

A2. 짜고 치는 가위바위보 (Large)

	이전 문자	→	현재 문자
1	주먹	→	주먹
2	주먹	→	가위
3	주먹	→	보
4	가위	→	주먹
5	가위	→	가위
6	가위	→	보
7	보	→	주먹
8	보	→	가위
9	보	→	보

A2. 짜고 치는 가위바위보 (Large)

- 특정 상황에서 이긴 뒤에 곧바로 비기면 안된다는 조건이 있었습니다.
- 이 조건에 맞춰서 문자를 하나하나 볼때마다 9가지의 상태를 전이 시키면 됩니다.
- $\mathcal{O}(N)$의 시간복잡도로 해결할 수 있습니다.

B. 커플 파괴자 민욱이

`ad-hoc`, `greedy`, `constructive` (`Small`)
`ad-hoc`, `greedy`, `backtracking`, `constructive` (`Large`)
출제진 의도 – **Medium** / **Hard**

- 제출 30번, 맞은 사람 6명 (정답률 20.00%) (Small)
- 제출 3번, 맞은 사람 2명 (정답률 66.67%) (Large)
- 처음 푼 사람: **nick832**, 44분 (Small) / **nick832**, 65분 (Large)
- 출제자: `asdf1705`

- 기본적으로 어떠한 커플끼리도 이웃하지 않게 하려면, 이웃해 있는 커플 사이를 갈라서 묶음을 만들면 됩니다.
- 위의 규칙을 따라서 묶음을 만들면 해결됩니다.
- 하지만, 예외가 두 가지 존재합니다.

- 총 두 사람 밖에 없고 묶음이 두 묶음인 경우를 생각해봅시다.
- 즉, 한 쌍의 커플이 서 있는 경우입니다.
- 이 경우는 어떻게 묶음을 배열하든 조건을 만족시키지 못합니다.
- 따라서, -1을 출력하고 종료하면 됩니다.

- 첫번째 예외를 확장시켜 생각해봅시다.
- 전체 인원이 3명 이상이고, 두 묶음이 만들어 졌는데 1번 묶음의 맨 앞 사람과 2번 묶음의 맨 뒷 사람이 커플인 경우를 생각해봅시다.
- 사실, 이 경우라면 어떻게 묶음을 배열하든지 조건을 만족시키지 못합니다.
- 따라서 새로운 묶음을 하나 더 만들어서 조건을 만족시키도록 해야합니다.
- 1번 묶음과 2번 묶음 중 하나를 골라, 고른 묶음의 원하는 위치에서 묶음을 둘로 나누어 3개의 묶음을 만들면 됩니다.

- 이렇게 만들어진 $1, 2, \cdots, M$ 개의 묶음들을 역순으로 배열하면 커플들이 이웃하지 않도록 할 수 있습니다.

B2. 커플 파괴자 민욱이 (Large)

- 기본적인 접근 방법과 예외 처리는 Small과 동일합니다.
- 이후 조건에 맞도록 M 개의 묶음을 배열하는 것은 백트래킹을 통해 해결할 수 있습니다.
- 시간 내에 백트래킹이 동작함을 보장할 수 있습니다.
- 어떤 묶음은 자기 앞 혹은 뒤에 오지 못하는 묶음을 최대 2개 까지 가질 수 있고, Dirac's Theorem을 사용하면 6 이상에서는 항상 답이 존재함을 보일 수 있습니다.
- 따라서 백트래킹의 단계를 최대 8 단계에서 끝나도록 할 수 있습니다.

C. Mike Sees The Storm

DP (Small) `combinatorics`, `exponentiation_by_squaring`, `modular_multiplicative_inverse`, `flt` (Large)
출제진 의도 – **Medium** / **Hard**

- 제출 15번, 맞은 사람 12명 (정답률 86.67%) (Small)
- 제출 13번, 맞은 사람 8명 (정답률 61.54%) (Large)
- 처음 푼 사람: **ncy09**, 13분 (Small) / **ncy09**, 13분 (Large)
- 출제자: `rustiebeats`

- 제한이 $1,000$ 까지임을 주목해봅시다.
- 점화식을 잘 세워서 $\mathcal{O}\left(N^3\right)$ DP로 해결할 수 있습니다.

- Small의 접근 방법을 그대로 사용하면 시간 초과가 나옵니다.
- 때문에 문제에서 요구하는 값을 하나의 식으로 정리해야 합니다.
- 우리는 수열의 최댓값이 항상 0부터 N 범위 사이에 있다는 것을 주목할 필요가 있습니다.

- 먼저, 수열의 최댓값이 N 인 경우를 생각해 봅시다.
- 이 경우는 오직 한 가지 밖에 존재하지 않습니다.
- 수열의 최댓값이 i 인 경우가 몇 가지가 나올지 생각해봅시다.
- 위의 수열이 카탈란 수와 관련이 있음을 생각하면, 이 경우는 총 $\binom{2n}{N-i} - \binom{2n}{N-(i+1)}$ 가지가 나옴을 확인할 수 있습니다.

- 따라서 식을 정리하면 아래와 같습니다.
- $N^K + \sum_{i=0}^{N-1} i^K \left(\binom{2n}{N-i} - \binom{2n}{N-(i+1)} \right)$
- 해당 조합값은 분할정복을 이용한 거듭제곱으로 빠르게 계산할 수 있습니다.

D. OR & XOR

`bruteforce`, `greedy`, `bitmasking` (Small)
`math`, `DP`, `bitmasking`, `dp_bitfield`, `dp_sum_over_subsets` (Large)
출제진 의도 – **Hard** / **Challenging**

- 제출 36번, 맞은 사람 12명 (정답률 36.11%) (Small)
- 제출 10번, 맞은 사람 4명 (정답률 40.00%) (Large)
- 처음 푼 사람: **riroan**, 8분 (Small) / **hjroh0315**, 37분 (Large)
- 출제자: `kongum`

- OR 연산을 $\vee$, AND 연산을 $\wedge$, XOR 연산을 $\veebar$라고 합시다.
- 이때, $(x \vee y) = x + y - (x \wedge y)$, $(x \veebar y) = x + y - 2(x \wedge y)$가 성립합니다.
- 이를 이용하면, OR과 XOR 연산을 합과 AND 연산으로 바꿀 수 있습니다.
- $\sum_{i=1}^{N}\sum_{j=1}^{N}(A_i + B_j)$는 쉽게 구할 수 있으므로, AND 연산 했을 때 가장 큰 항 p개를 찾는 문제로 바꿔서 생각할 수 있습니다.

- 수열 A에 값 x가 몇 개 있는지를 $\mathrm{cnt}A_x$, 수열 B에 값 y가 몇 개 있는지를 $\mathrm{cnt}B_y$라고 합시다.
- $0 \le A_i, B_j < 2^u$ 이므로, $0 \le (A_i \wedge B_j) < 2^u$ 입니다.
- 따라서 $0 \le x, y < 2^u$를 만족하는 모든 순서쌍 (x, y)에 대해, $(x \wedge y)$과 그 값을 가지는 항이 몇 개인지를 $\mathrm{cnt}A_x \times \mathrm{cnt}B_y$를 통해 $\mathcal{O}(4^u)$의 시간 복잡도로 구할 수 있습니다.

- $(x \wedge y)$를 구한 값을 계수 정렬을 통해 가장 값이 큰 항 p개의 합을 구합니다.
- 구한 값과 $\sum_{i=1}^{N}\sum_{j=1}^{N}(A_i + B_j)$를 더한 값이 정답이 됩니다.
- 따라서, 총 시간 복잡도는 $\mathcal{O}(4^u)$입니다.
- small버전의 경우, $u = 10$의 제한을 가지므로 시간 제한 내에 문제를 해결할 수 있습니다.
- $(x \vee y) = x + y - (x \wedge y)$, $(x \veebar y) = x + y - 2(x \wedge y)$을 생각하지 못했더라도 $0 \leq (x \vee y) - (x \veebar y) < 2^u$임을 관찰했다면 비슷한 방법으로 풀 수 있습니다.

- $x \subseteq i$를 $(x \wedge i) = x$를 만족하는 i라고 합시다.
- $x \subset i$를 $(x \wedge i) = x$와 $i \neq x$를 만족하는 i라고 합시다.
- AND 연산 했을 때의 값이 x인 항의 개수를 $\mathrm{cnt}V_x$라고 합시다.
- 정의에 따라 아래의 등식이 성립합니다.
- $$\sum_{mask \subseteq k} \mathrm{cnt}V_k = \sum_{mask \subseteq i} \mathrm{cnt}A_i \times \sum_{mask \subseteq j} \mathrm{cnt}B_j, (0 \leq mask < 2^u)$$
- $$\Rightarrow \mathrm{cnt}V_{mask} = \sum_{mask \subseteq i} \mathrm{cnt}A_i \times \sum_{mask \subseteq j} \mathrm{cnt}B_j - \sum_{mask \subseteq k} \mathrm{cnt}V_k$$

- $x = 0$부터 시작하여 $x < 2^u$까지 순회하며 위의 값을 구합니다.
- 이때, $mask \subset x$라면 $x \vee mask$를 x로, $mask \subseteq x$라면 $x + 1$을 x로 하여 순회합니다.
- $mask \subseteq x$를 만족하는 각 x마다 위의 값을 $\mathcal{O}(1)$의 시간 복잡도로 구할 수 있습니다.
- $\sum_{mask=0}^{2^u-1} n(\{x|mask \subseteq x\}) = \sum_{i=0}^{u} 2^i \binom{u}{i} = 3^u$이므로, $\mathrm{cnt}V_{mask}(0 \leq mask < 2^u)$를 구하는 데 필요한 시간 복잡도는 $\mathcal{O}(3^u)$입니다.
- Large 버전은 u의 제한이 17이므로 시간 제한 안에 문제를 해결할 수 있습니다.
- Sum Over Subset DP라고 알려진 방법으로도 $\mathrm{cnt}V_{mask}(0 \leq mask < 2^u)$를 빠르게 구할 수 있습니다.