목차

문제		의도한 난이도	출제자
A	모르고리즘 회장 정하기	Beginner	김태윤 ystaeyoon113
B	버터 녹이기	Easy	김태윤 ystaeyoon113
C	과제 제출하기	Easy	김태윤 ystaeyoon113
D	미로 보수	Medium	김형진 plast
E	카르텔 님 게임	Medium	김형진 plast
F	피자 쌓기	Medium	김건우 coconut99
G	슈퍼 트리 뽀개기	Hard	김형진 plast
H	선로 조립	Hard	전우제 kiwiyou

출제진 소개

- ✓ 김건우 (coconut99)
  - 2022 google codejam round3 진출
  - 2022 연세대학교 프로그래밍 대회 출제 및 검수
  - 2023 연세대 x 고려대 프로그래밍 대회 출제 및 검수
  - 2023 신촌지역 대학생 프로그래밍 대회 (SUAPC) 1위
  - 2023 삼성전자 대학생 프로그래밍 대회 (SCPC) 5등상
- ✓ 김태윤 (ystaeyoon113)
  - 2020 국제 대학생 프로그래밍 대회 (ICPC), 한국본선 6위
  - 2023 신촌지역 대학생 프로그래밍 대회 (SUAPC) 1위
  - 2023 전국 대학생 프로그래밍 대회 (UCPC) 19위
  - 2023 삼성전자 대학생 프로그래밍 대회 (SCPC) 4등상

출제진 소개

- ✓ 김형진(plast)
  - 2018 한국정보올림피아드(KOI) 은상
  - 2020 연세대학교 신입생 프로그래밍 대회 2위
  - 2023 신촌지역 대학생 프로그래밍 대회(SUAPC) 2위
  - 2023 현대모비스 프로그래밍 대회 14위
  - 2023 삼성전자 대학생 프로그래밍 대회(SCPC) 4등상
- ✓ 전우제(kiwiyou)
  - 2022 전국 대학생 프로그래밍 대회(UCPC) 28위
  - 제1회 청소년IT경시대회 출제진
  - 제1회 미적확통컵 출제진
  - 제1회 가지컵 주최

NAVER D2 START LINK

검수진 소개

- 곽재혁(dreami63) - 연세대학교
- 문정후(mjhmjh1104) - 경기과학고등학교
- 이국렬(lky7674) - 연세대학교
- 이성현(hibye1217) - 한양대학교
- 장래오(leo020630) - 포항공과대학교
- 정용진(playsworld16) - 연세대학교

A. 모르고리즘 회장 정하기

`string`
출제진 의도 – **Beginner**

- ✓ 제출 58번, 정답 30명 (정답률 55.172%)
- ✓ 처음 푼 사람: **주4일제 도입 시급**, 0분
- ✓ 출제자: **김태윤(ystaeyoon113)**

A. 모르고리즘 회장 정하기

- ✓ 길이가 3인 string들에만 관심이 있습니다.
- ✓ 많은 언어의 minimum 함수는 사전 순으로 앞선 string을 구할 수 있습니다.
- ✓ 따라서 길이가 3인 string들 중 최소값을 찾으면 됩니다.
- ✓ string의 길이가 5를 넘어가지 않으므로 $O(N)$으로 분석할 수 있습니다.

B. 버터 녹이기

`sort`, `greedy`
출제진 의도 – **Easy**

- ✓ 제출 65번, 정답 20명 (정답률 30.769%)
- ✓ 처음 푼 사람: **대회랑 일정 겹치는데 ㅠㅠ**, 13분
- ✓ 출제자: **김태윤(ystaeyoon113)**

B. 버터 녹이기

- 먼저 각 버터를 x 값이 증가하는 순서로 정렬했다고 생각해 봅시다.
- 이때, 정렬된 상태에서 i 번째 버터와 j 번째 버터가 섞였다고 합시다. ($i < j$)
- 그렇다면 당연히 i 번째부터 j 번째에 존재하는 모든 버터들은 섞였음을 관찰할 수 있습니다.

B. 버터 녹이기

- 조금 쉬운 예시로 1번 버터와 3번 버터가 섞인 상태를 가정해 보겠습니다.
- 그렇다면 당연히 1번 버터와 3번 버터 사이에 있는 2번 버터는 섞인 상태입니다.
- 그리고 당연히 1번 버터와 3번 버터가 섞이기 이전에, 1번 버터와 2번 버터가 섞였거나 혹은 2번 버터와 3번 버터가 섞였을 것입니다.
- 따라서, 어떤 버터들이 섞이게 된다면 가장 먼저 섞이게 되는 버터들은 어떤 인접한 버터들이라는 사실을 관찰할 수 있습니다.

B. 버터 녹이기

- 그러므로 모든 버터들을 x 값에 따라 정렬한 뒤, 인접한 버터들만 살펴보면 됩니다.
- 이때 i 번과 $i+1$ 번째 버터가 만나는 시간을 구하는 방법은 다양한 방법이 있습니다.
- 그중 하나는 t 초 후의 i 번째 버터의 오른쪽 끝과, $i+1$ 번째 버터의 왼쪽 끝을 비교하는 방법입니다.
- $x_i + min(t, h_i/2) < x_{i+1} - min(t, h_{i+1}/2)$ 인 t 의 최대값을 구하면 됩니다.
- 이는 이분탐색이나 간단한 조건분기로 구할 수 있습니다.
- 정렬하는 과정에서 $O(NlogN)$ 이 소요되고 답 계산은 $O(N)$ 또는 $O(NlogN)$ 이내로 구할 수 있습니다.

C. 과제 제출하기

`brute force, greedy`
출제진 의도 – **Easy**

- ✓ 제출 46번, 정답 12명 (정답률 26.087%)
- ✓ 처음 푼 사람: **주4일제 도입 시급**, 11분
- ✓ 출제자: **김태윤(ystaeyoon113)**

C. 과제 제출하기

- ✓ 성현이가 문제를 푸는 순서를 이미 정했다면 성현이의 최적 전략은 쉽게 생각할 수 있습니다.
- ✓ 그때의 최적 전략은 해당 지식을 꼭 필요로 하는 때에만 공부하는 것입니다.
- ✓ 즉, i 번째 문제를 풀어야 하는데 i 번째 문제를 풀기 위해 모르는 지식이 있다면 그때 공부하는 전략입니다.
- ✓ 이는 각 지식 j 에 대해 성현이가 언제까지 기억하고 있는지 (t_j) 를 배열 등으로 관리해서 $O(N+M)$ 에 쉽게 해결할 수 있습니다.

C. 과제 제출하기

- ✓ 일반적인 경우에, 성현이가 어떤 순서로 문제를 해결해야 하는지는 알 수 없습니다.
- ✓ 따라서 성현이가 문제를 해결하는 모든 순서들에 대해서 답을 구하고 그중 최소값을 취해야 합니다.
- ✓ 풀어야 하는 문제수가 7 이하이므로 문제를 해결하는 모든 순서는 7! = 5040 개를 넘지 않습니다.
- ✓ 따라서 모든 $M!$ 개의 순서에 대해서 성현이의 최적 전략으로 문제를 풀면 됩니다.
- ✓ 시간복잡도는 $O(M! \times (N+M))$ 입니다.

D. 미로 보수

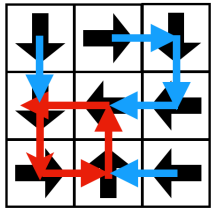
`graph`
출제진 의도 – **Medium**

- ✓ 제출 43번, 정답 7명 (정답률 16.279%)
- ✓ 처음 푼 사람: **대회랑 일정 겹치는데 ㅠㅠ**, 41분
- ✓ 출제자: **김형진(plast)**

D. 미로 보수

- 미로의 어느 부분에 점프대를 설치해야 사람들이 미로 밖으로 빠져나갈 수 있는지 생각해 봅시다.
- 미로 내부에서 특정한 사이클이 형성되어 영영 벗어나지 못하게 될 때, 반드시 이 사이클 안에 점프대를 설치해야만 해당 사이클에 있는 사람들이 미로를 벗어납니다.
- 이를 통해 고민해 보면, 미로 내부에 있는 사이클마다 사이클 내부에서 가장 싼 지역에 점프대를 하나 설치하는 것이 최선의 전략입니다.

D. 미로 보수



- 다음과 같은 미로가 있다고 할 때, 주의해야 할 점은 9개의 칸 중 어느 칸에서 출발해도 사람들은 미로 밖으로 영영 벗어날 수 없지만, 9개의 칸 모두가 사이클에 포함되는 것이 아닌 빨간색 경로에 해당하는 부분만이 사이클이라는 점입니다.

D. 미로 보수

- ✓ 그래프를 dfs로 탐색해 나가면서, 현재 탐색하고 있던 위치로 되돌아왔다면 사이클이 있음을 알 수 있습니다.
- ✓ 그래프의 방문 여부를 저장할 때 여러 상태로 나눠 저장합니다. 완전히 방문이 끝난 점과, 방문하기는 했지만, 해당 위치로 되돌아올 경우 사이클이라고 생각할 수 있는 점으로 분류해 상황에 맞게 관리하는 방식으로 사이클을 찾을 수 있습니다.

E. 카르텔 님 게임

`game theory`
출제진 의도 – **Medium**

- ✓ 제출 20번, 정답 4명 (정답률 20.000%)
- ✓ 처음 푼 사람: **종이비행기**, 80분
- ✓ 출제자: **김형진 (plast)**

E. 카르텔 님 게임

- ✓ 플레이어 A, B 두 명은 같은 팀이기 때문에 팀이 이기기 위해서 같은 전략을 취합니다.
- ✓ 따라서 플레이어 A, B가 선택하는 행동들은 사실 플레이어 한 명만 플레이하고, 그 플레이어가 2 이상 K 이하의 정수를 선택해 돌을 가져가는 것과 완전히 같은 상황입니다.

E. 카르텔 님 게임

- ✓ 이제 문제가 조금 더 간단해졌습니다. 3명이 플레이하는 게 아닌 2명이 플레이하는 전형적인 님 게임으로 문제가 바뀝니다.
- ✓ 플레이어 AB, C가 있고, AB는 2 이상 K 이하의 정수를 선택해 그만큼의 돌을 가져가고, 플레이어 C는 1 이상 K 이하의 정수를 선택해 그만큼의 돌을 가져갔을 때 어느 플레이어가 이기는지의 문제를 풀어봅시다.

E. 카르텔 님 게임

- ✓ 각 플레이어의 필승전략은 플레이어 C를 기준으로 생각하면 비교적 쉽게 아이디어를 찾을 수 있습니다.
- ✓ 플레이어 C가 돌을 받았을 때 남은 돌이 1 이상 K 이하인 경우, 플레이어 C가 바로 모든 돌을 가져갈 수 있으므로 C가 승리합니다.
- ✓ 플레이어 C가 돌을 받았을 때 남은 돌이 $K+1$ 인 경우, 플레이어 C가 어떻게 돌을 가져가더라도 플레이어 AB는 1 이상 K 이하의 돌을 받게 됩니다. 즉 플레이어 AB가 바로 모든 돌을 가져갈 수 있으므로 C가 패배합니다.
- ✓ 플레이어 C가 돌을 받았을 때 남은 돌이 $K+2$ 이상인 경우, 다음과 같은 전략을 취할 수 있습니다.

E. 카르텔 님 게임

- 플레이어 C가 돌을 받았을 때 남은 돌이 $K+2$ 이상인 경우, 다음과 같은 전략을 취할 수 있습니다.
- 1) 플레이어 C가 받은 돌의 개수가 $K+1$의 배수가 아닌 경우, 항상 플레이어 C는 남은 돌을 $K+1$의 배수가 되도록 플레이어 AB에게 넘길 수 있습니다. 이 경우 일반적인 님 게임과 같은 전략을 취하면 플레이어 C가 승리합니다.
- 2) 플레이어 C가 받은 돌의 개수가 $K+1$의 배수인 경우, 플레이어 C는 K 개의 돌을 가져갑니다. 이러면 플레이어 AB는 $K+1$로 나눈 나머지가 1인 만큼 돌을 받게 되는데, 어떻게 하더라도 이다음에 플레이어 C는 받는 돌의 개수가 $K+1$의 배수가 아니게 됩니다. 이후로는 1)의 전략을 취하면 플레이어 C가 승리합니다.

- ✓ 따라서, 플레이어 C가 받는 돌의 개수가 정확히 $K+1$ 일 때에만 플레이어 C가 패배합니다.
- ✓ N 이 K 이하인 경우, 플레이어 AB가 바로 모든 돌을 가져갈 수 있으므로 AB가 승리합니다.
- ✓ N 이 $K+3$ 이상 $2K+1$ 이하인 경우, 플레이어 AB는 플레이어 C에게 정확히 $K+1$ 개의 돌을 넘길 수 있습니다. 따라서 AB가 승리합니다.
- ✓ 그 외의 모든 경우에 대해, 플레이어 AB는 플레이어 C에게 $K+1$ 개의 돌을 넘길 수 없기 때문에 C가 승리합니다.

F. 피자 쌓기

`combinatorics`
출제진 의도 – **Medium**

- ✓ 제출 9번, 정답 3명 (정답률 33.333%)
- ✓ 처음 푼 사람: **T stands for TLE**, 48분
- ✓ 출제자: **김건우(coconut99)**

F. 피자 쌓기

- ✓ 가능한 모든 순서에 대해 보이는 피자 수를 계산하는 것으로 서브태스크 1을 해결할 수 있습니다.
- ✓ 1 부터 N 크기의 피자가 1개씩 있을 때의 정답을 a_N 이라고 할 때, a_N 에 대한 점화식을 세워 서브태스크 2를 해결할 수 있습니다.
- ✓ a_i 을 전처리해두고 $N+1$ 크기의 피자가 처음 등장하는 위치에 대한 답을 a_N 을 이용해 구한 뒤 모두 합하면 서브태스크 3을 해결할 수 있습니다.

F. 피자 쌓기

- 서브태스크 2를 해결하는 점화식 a_i을 구해봅시다.
- N개의 피자 사이에 $N+1$번째 피자를 위에서부터 i번째 위치에 끼워넣을 때, 가능한 $N!$가지 순서에 대해 보이는 피자 수의 합은 ${}_NP_{N-i} \times a_i + N!$입니다.
- 0부터 N까지 모든 i에 대해 합하면 $a_{N+1} = (N+1)! + \sum_{i=0}^{N} {}_NP_{N-i} \times a_i$입니다.
- 정리하면 $a_{N+1} = N! + N \times (N! + \sum_{i=0}^{N-1} {}_{N-1}P_{N-1-i} \times a_i) + a_N = N! + (N+1) \times a_N$ 입니다.

F. 피자 쌓기

- 이제 서브태스크 4를 해결해 봅시다.
- 어떤 피자가 보이기 위해서는, 그 피자보다 크기가 크거나 같은 피자들 중 가장 위에 있어야 합니다.
- 즉 크기 i인 피자가 보일 확률은 $\frac{a_i}{a_i + a_{i+1} + \cdots + a_N}$ 입니다.
- 가능한 모든 순서의 수를 s라 하면, 크기 i인 피자가 보이는 순서의 수는 $\frac{s \times a_i}{a_i + a_{i+1} + \cdots + a_N}$ 이므로 1부터 N까지의 i에 대해 이 값을 모두 합하면 됩니다.

G. 슈퍼 트리 뽀개기

`tree`, `sparse table`, `tree DP`, `small to large`, `segment tree`, `euler tour`, `merge sort`
출제진 의도 – **Hard**

- ✓ 제출 4번, 정답 2명 (정답률 50.000%)
- ✓ 처음 푼 사람: **대회랑 일정 겹치는데 ㅠㅠ**, 129분
- ✓ 출제자: **김형진 (plast)**

G. 슈퍼 트리 뽀개기

- ✓ 서브테스크 1은 노드별로 노드의 K 깊이 이하 자식의 개수를 세는 문제입니다.
- ✓ K 깊이 이하 자식의 개수는 곧, (모든 자식 노드의 개수 - K 초과 자식 노드의 개수)와도 같습니다.

G. 슈퍼 트리 뽀개기

- ✓ K 초과 자식 노드의 개수를 관리해 봅시다.
- ✓ 배열 a를 정의합니다. a[v] = K + 1번째 조상이 v 인 노드의 개수라고 정의를 하면, 어떤 노드 v 의 K 초과 자식 노드의 개수는 곧 노드 v 를 조상으로 하는 노드 집합 S에 대해 $\Sigma a[S]$ 가 됩니다.

G. 슈퍼 트리 뽀개기

- ✓ a배열은 sparse table을 이용해 각 노드에 대해 K + 1번째 조상 노드를 구해주면 $O(N \log N)$ 에 계산할 수 있습니다.
- ✓ 그 뒤 트리 DP를 통해 자신의 서브트리의 a배열의 총합을 구하고, 서브트리의 노드 개수 역시 비슷한 방식으로 구하면 각 노드에서의 K 이하 자식의 개수를 $O(N \log N)$ 에 관리해 줄 수 있습니다.

G. 슈퍼 트리 뽀개기

- ✓ 서브테스크 2는 a배열을 구하는 방식을 변화시켜야 합니다.
- ✓ sparse table을 똑같이 유지하되, sparse table에 총 거리까지도 같이 계산해서 저장해 주면, 처음으로 거리를 K 를 넘어가는 조상 노드를 이분 탐색을 통해 구할 수 있습니다. 해당 방법으로 a배열을 계산하여 문제를 해결할 수 있습니다.

G. 슈퍼 트리 뽀개기

- ✓ 해당 풀이 외에도, 스몰 투 라지 + 세그먼트 트리, 오일러 투어 + 머지소트 등의 다양한 풀이가 존재합니다.

H. 선로 조립

`dp_tree`
출제진 의도 – **Hard**

- ✓ 제출 6번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: —, –분
- ✓ 출제자: **전우제(kiwiyou)**

H. 선로 조립

- 선로는 트리이고, 선로를 옮긴 후에도 트리를 유지해야 합니다.
- 즉 트리에서 간선을 하나 옮긴 뒤 지름이 최대가 되도록 트리를 구성해야 합니다.
- 제거한 간선의 양 끝 노드 중 루트에 가까운 노드를 p, 다른 노드를 s라고 합시다.
- s를 루트로 하는 서브트리 S의 지름과 남은 트리 P의 지름을 합하고 1을 더하면 됩니다.

H. 선로 조립

- ✓ 쿼리마다 S와 P의 지름을 구한다면 $O(NQ)$로 시간 초과를 받습니다.
- ✓ 기존 트리의 지름을 DP로 구한다면, S의 지름은 쿼리당 $O(1)$에 답할 수 있습니다.
- ✓ P의 지름은 기존 트리 T의 지름 위치에 따라 경우를 나눠 구할 수 있습니다.
  1. T의 지름이 온전히 P에 있는 경우
  2. T의 지름이 간선 (p, s)를 포함하는 경우
  3. T의 지름이 온전히 S에 있는 경우

H. 선로 조립

- T의 지름을 구할 때, 각 자식 서브트리에서 첫째, 둘째, 셋째로 큰 지름을 구해둡니다.
- 2번, 3번 경우의 최대 지름을 구하여 제외한 뒤 남은 첫째 지름을 사용하면 됩니다.
- 이때 2번 경우를 제외하고 나면 루트에서 p까지의 경로를 포함하는 지름이 새로 생길 수 있습니다.
- 이는 전체 트리에서 다시 한 번 DP를 통해 구할 수 있으며, 전체 시간복잡도 $O(N+Q)$에 문제를 해결할 수 있습니다.