서문

오는 11월 12일로 다가오는 교내 SW경진대회를 맞아, ALchemy 내부에서 준비대회를 열게 되었습니다. 대회에 참여하신 분들 모두 수고하셨고, 이번에 참여하지 못하신 분들 또한 시간을 잡아놓고 대회 문제를 풀어보셔도 좋겠습니다. SW경진대회에 참가하시는 분들 좋은 결과 가져오시길 기대해봅니다.
이번 준비대회에서는, 제 1회, 2회의 SW경진대회 출제 경향이 많이 반영되었습니다. 이윤수(lys9546)님과 박종현(belline0124)님의 예년 대회 참여 경험을 바탕으로, 이전 대회의 난이도와 비슷한 수준으로 구현, 수학, 문자열, 그래프 이론, DP 등 다양한 알고리즘 분류의 문제들을 출제하려 노력했습니다. 문제는 저 정영도(0do)가 이윤수님과 함께 선정했습니다.
이번에 진행된 준비대회처럼, 여러분들의 의견을 수용하여 이런 방식의 대회를 이후에도 더 여는 것을 기획하고 있습니다! 이 부분은 나중에 기획이 구체화된 후 다시 한번 말씀드릴 수 있도록 하겠습니다.

A. 세 막대

문제 링크: https://www.acmicpc.net/problem/14215
난이도: B3
분류: 수학/구현/기하학

출제자 코멘트:

- 가능한 많은 분들에게 참여상을 드리고자 이 문제를 준비하게 되었습니다.
- 실제 대회에서는, 가장 쉬운 난이도의 문제가 이 문제보다 더 어렵게 출제될 가능성도 있습니다.

문제 해설:

- 편의상 a가 큰 값을, b가 중간 값을, c가 작은 값을 가진다고 가정하겠습니다. 실제 입력은 다르게 주어질 수 있는데, [조건문] 또는 [배열의 정렬]을 이용하여 a, b, c 값을 크기 순으로 나열해주시면 됩니다.
- 삼각형의 성질을 이용하여 풀 수 있는 문제입니다. 삼각형의 성질인 가장 긴 변이 나머지 두 변의 합보다 작아야 한다는 점을 사용할 수 있습니다(a < b + c)
- a < b + c라면, 막대의 길이를 더 줄일 필요가 없으므로 a+b+c를 출력하면 됩니다.
- a >= b + c일 경우, a = b + c - 1 까지는 줄어들어야 삼각형의 성질을 만족하며 가장 큰 둘레를 갖습니다. 따라서 이 경우는 b + c + (b + c - 1) 이 답이 됩니다.

B. 이삿짐 센터

문제링크: https://www.acmicpc.net/problem/16237
난이도: S4
분류: 수학/구현/그리디 알고리즘

출제자 코멘트:

- 문제 풀이에 필요한 사고를 떠올리는건 어렵지 않을 수 있으나, 구현이 번거롭고 까다로운 문제입니다. 교내 대회에서도 이러한 문제가 등장할 수 있습니다.
- 해설과 접근방식이 일치했으나 구현에 실패했다면, 구체적인 설계와 연습이 필요할 수 있습니다.

문제 풀이:

- 위 예시 사진이 증명해주듯이 무거운 것부터 바구니에 집어넣어야 합니다.
- 아래의 과정을 따라가면, 바구니에 무거운 것부터 집어넣을 수 있습니다.
- 5kg, 4kg, 3kg 물건끼리는 서로 같은 바구니 안에 2개 이상 들어있을 수 없습니다.
- 4kg 물건이 들어간 바구니 안에는, 남는 1kg 물건을 추가로 담을 수 있습니다.
- 3kg 물건이 들어간 바구니 안에는, 남는 2kg 물건을 추가로 담을 수 있습니다.
- 3kg 물건에 담을 2kg 물건이 없다면, 대신 1kg 물건 2개를 추가로 담을 수 있습니다.
- 2kg 물건이 남아있다면, 바구니에 2kg 물건 2개를 담을 수 있습니다. 이 바구니에 남는 1kg 물건을 추가로 담을 수 있습니다.
- 남는 1kg 물건들을 바구니에 담으면 됩니다.
- 배열, 조건문, 반복문, 연산자 등을 활용하여 위 과정을 구현하시면 됩니다.

.

C. 비슷한 단어

문제링크: https://www.acmicpc.net/problem/1411
난이도: S3
분류: 문자열/브루트포스 알고리즘

출제자 코멘트:

- 문제 설명이 매우 직관적인 것은 아니었던 것 같습니다. 또한, 문제를 이해는 했는데 어떻게 구현해야할지 떠올리기 어려울 수도 있습니다. SW경진대회에서도, 이러한 문제들이 등장할 수 있습니다.
- 문제 출제 및 대회 검수를 맡아주신 윤수님은 C번이 B번보다 쉽다는 평가를 내려주셨습니다.

문제 풀이:

- 맵[딕셔너리]를 이용하여 한 단어에서 첫 번째로 나오는 알파벳에 0, 두 번째로 나오는 알파벳에 1, … 이런 식으로 번호를 부여해줍시다. 예) {“a”: 0, “b”: 1}
- 예를 들어, ‘aa’, ‘bb’, ‘aba’, ‘cdc’ 이렇게 네 개의 단어가 있다면, ‘aa’ = [0, 0], ‘bb’ = [0, 0], ‘aba’ = [0, 1, 0], ‘cdc’ = [0, 1, 0] 이런 모양이 됩니다.
- 모양이 같은 두개의 단어가, 서로 비슷한 단어가 됩니다. 모든 단어의 모양을 배열 형태로 바꿔준 후, 모든 가능한 쌍을 구해주면 됩니다.

- 정답자 코드를 참고해보시는 것이 문제 풀이 이해에 도움이 될 듯 합니다.
- [Python] http://boj.kr/0d306c6653e7461e902b05884cac20f8

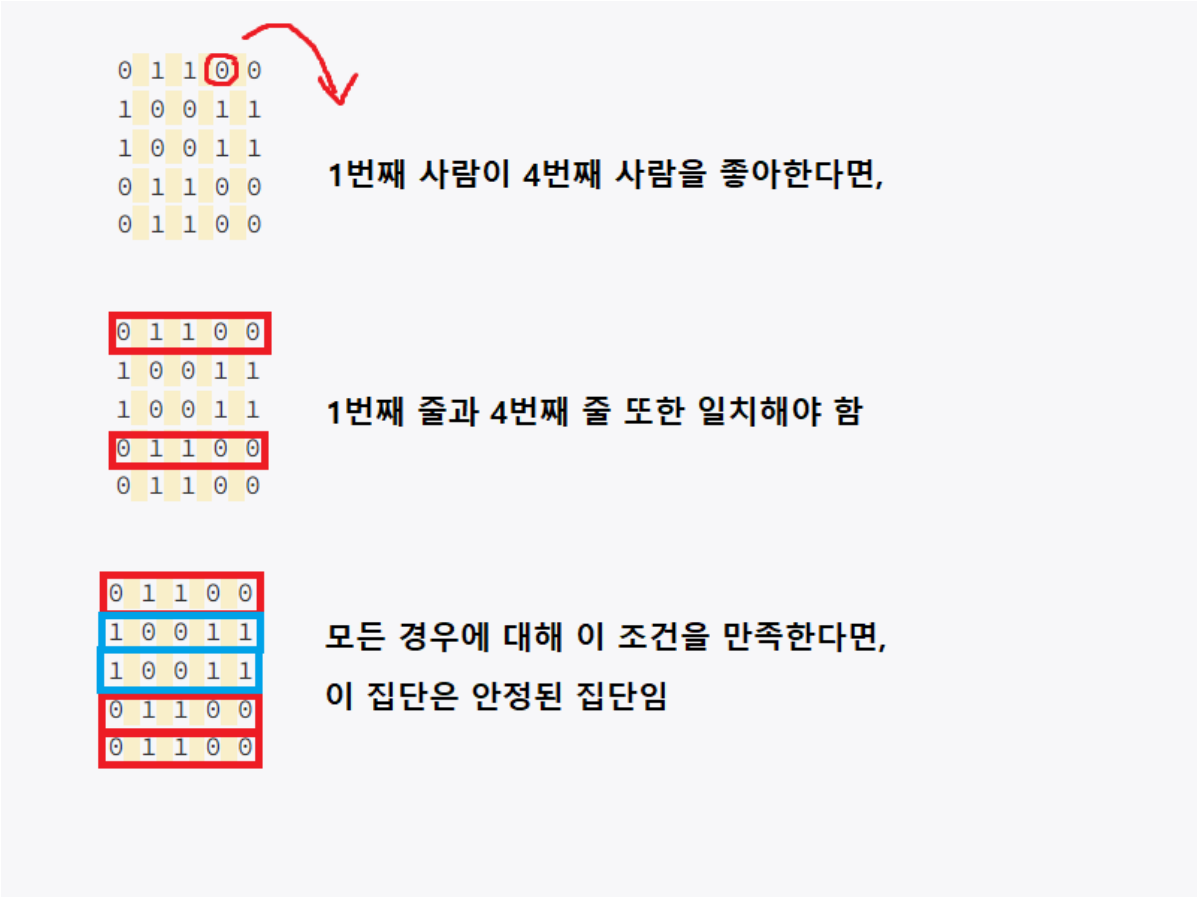
D. 안정된 집단

문제링크: https://www.acmicpc.net/problem/2653
난이도: G4
분류: 그래프 이론/구현

최초 해결: 권순찬(sketom)님, 64분

출제자 코멘트:
- 문제 난이도가 상승하는 부분입니다.
- ALchemy 내에서 잘 푸는 사람들의 경우 3~4문제 정도 해결할거라고 예상했고, 이 문제 해결여부에 따라서 이번 대회의 승자가 갈리겠다 생각했습니다. 실제로는 2명이나 이 문제를 해결하게 되었네요.

문제 해설:
- i 번째 사람이 j 번째 사람을 좋아한다면, i 번째 사람과 j 번째 사람은 같은 소집단에 있어야 합니다.
- 같은 소집단에 있는 사람들끼리는, 좋아하는 사람과 싫어하는 사람의 정보가 일치해야 합니다.
- 즉 안정된 집단이 되기 위해서는, i가 j를 좋아한다면, n * n의 인접행렬에서 i번째 줄과 j번째 줄 또한 일치해야 합니다. 인접행렬 안에서 하나라도 이 규칙에 위배되는 경우가 있을 경우, 이 집단은 안정된 집단이 아닙니다.



- 그림으로 다시 정리할 경우 다음과 같습니다. 이 방법을 통해, 집단이 안정된 집단인지 여부를 판별할 수 있습니다.

```
for i in range(n):
    if not visited[i]:
        for j in range(n):
            if arr[i][j] == 0:
                visited[j] = True
                print(...)
```

- 이후 위 코드처럼 방문(visited) 배열을 사용하거나, Union-Find 알고리즘을 구현하는 등 적절한 후처리를 통해 안정된 집단에 대한 번호를 출력해주시면 됩니다.

E. 낮잠 시간

문제링크: https://www.acmicpc.net/problem/1988
난이도: G4
분류: DP

최초 해결: -

출제자 코멘트:

- 알고리즘을 공부하시는 분들 중 학습에 어려움을 느끼는 대표적인 알고리즘 중 하나가 DP일 것이라고 생각됩니다. 이번 대회에서도 E번 문제에 대한 해결자가 없었습니다.

- DP문제는 프로그래머의 사고를 요구하는 부분이 많아, 주로 코딩테스트보다는 대회에 더 많이 출제되는 경향이 있습니다. 물론 코딩테스트에 출제되지 않는 것은 아닙니다. 교내대회를 대비하여 DP 알고리즘을 연습해보시는 것을 추천드립니다.

문제 해설:
- 이 문제는 DP(다이나믹 프로그래밍)에 대한 개념적인 이해가 없으면 풀이가 불가능한 문제입니다. DP에 대한 이해가 있다는 것을 전제로 해설하겠습니다.

- 문제 입력으로 주어지는 시간별 회복량을 energy 배열에 저장한다고 가정하겠습니다.
- '현재 시간이 언제인지', 그리고 '잠을 몇 시간 잤는지' 이렇게 두개의 정보가 저장된 2차원 배열을 2개 만들어야 합니다. 잠을 자지 않고 있는 상태를 나타내는 idle 배열, 그리고 잠을 자며 회복중인 상태를 나타내는 sleep 배열을 만든다고 가정하겠습니다.
- 다음과 같은 점화식으로 이 문제를 해결할 수 있습니다.

$$idle[i][j] = max(idle[i-1][j], sleep[i-1][j])$$
$$sleep[i][j] = max(idle[i-1][j-1], sleep[i-1][j-1]+energy[i])$$

- 이때, i(0<i<N)은 현재 시간, j(0<j<=B)는 잠을 잔 횟수를 의미합니다.
- idle[i][j]에서는 이전 시간까지 잠을 자지 않은 케이스와 잠을 잔 케이스 중, 어떤 케이스의 피로 회복량이 더 많았는지 비교합니다.
- sleep[i][j]에서는, 방금 막 잠에 든 케이스(피로가 회복되지 않음)와 잠을 자고 있던 케이스 중, 어떤 케이스의 피로 회복량이 더 많았는지 비교합니다.
- 이외에는 약간의 디테일한 처리만 해주면 문제를 해결하실 수 있습니다.

- 설명드린 해설 외에도 풀이방법이 다양할 것이라 생각되는 문제입니다. DFS등 탐색을 이용한 풀이도 가능할 것 같습니다.

F. 밤편지

문제링크: https://www.acmicpc.net/problem/23258
난이도: G1
분류: DP/그래프/플로이드-워셜

최초 해결: -

출제자 코멘트:
- 풀 수 있는 사람이 있을 거라고 예상하진 않았으나, '교내대회에 이정도 난이도의 문제까지 나올 수 있다' 라는 것을 알려드리기 위해 출제한 문제입니다.
- 그럼에도 정말 진심으로 괜찮은 문제이니 나중에라도 풀어보시는걸 추천드립니다!

문제 해설:
- 이 문제는 플로이드-워셜에 대한 개념적인 이해가 없으면 풀이가 불가능한 문제입니다. 플로이드-워셜에 대한 이해가 있다는 것을 전제로 해설하겠습니다.

- 이슬을 2^C방울 마실 수 없다는 의미는 번호가 C 이상인 집은 거쳐갈 수 없고, C 미만인 집은 거쳐가도 상관없다는 뜻입니다. 2^1+2^2+...+2^(C+1) < 2^C이기 때문입니다.
- 플로이드-워셜 알고리즘의 가장 바깥쪽 반복문은 거쳐갈 수 있는 정점을 나타냅니다. 즉, 플로이드 -워셜 알고리즘에서, 가장 바깥쪽 반복문 변수의 값이 k일때, dist 배열에는 1부터 k번 정점까지 거쳐갈 수 있을때의 최단거리가 저장이 됩니다.
- 따라서 이 문제에서는, k에 대하여 3차원 dist 배열을 만들어두면, 배열을 참조하는 것으로 모든 질문에 답할 수 있습니다.