제 3회 전국 대학생 프로그래밍 대회 동아리 연합 여름 대회

The Problemsetters and Judges

8월 17일

A : Alchemy (출제: wook)

문제가 길고 영어라 읽는 데 괴로움이 있지만, 간단한 문제입니다. 각각의 재료에 대해, 그 재료를 합성할 수 있는 적당한 순서를 구하면 됩니다. 단순하게 모든 경우를 다 탐색하는 방법으로도 충분히 풀 수 있습니다.

Solution 1. n이 작아서 약간의 간단한 커팅을 포함한 백트래킹으로도 충분히 시간 안에 YES가 나오도록 출제했습니다. . . . 만 이렇게 푼 팀이 별로 없더군요 orz

A : Alchemy (출제 : wook)

Solution 2. 2^n DP로 풀 수 있습니다.

- $f(S)$: 현재 flask에 들어있는 재료의 집합이 S일 때 만들어지는 물질 (없으면 NONE)
- 어떤 물질을 만들기 위해서 필요한 재료물질이 S라고 하면, 재료 물질 i를 하나 정하고, $f(S \setminus \{i\}) =$ NONE 일 경우 $S \setminus \{i\}$ 를 flask에 넣은 직후 i를 넣으면 됩니다.

이런 방식으로 재귀적으로 순서를 재구성하면 됩니다.

A : Alchemy (출제 : wook)

Solution 3. 좀 더 생각해보면 쉽게 풀 수 있는 방법이 있습니다. (다항시간)

- 결국 겹치는 재료 물질은 마지막에 넣어야 하기 때문에
- 어떤 물질을 만들기 위해 필요한 재료 물질들의 재료 집합들을 다 찾고 이것들의 전체 common element를 하나 찾습니다.
- 없으면 불가능. 있으면 이 common element를 마지막에 넣으면 됩니다.

많이 했던 실수들

- 테스트 케이스 사이에 빈 줄 안 찍기 (...)
- 출력의 맨 첫 줄에 빈 줄 찍기 (P모대 D모팀)

B : Bunker (출제 : 꿀호떡)

길이 L을 하나 정하고, 다음과 같은 규칙으로 배열을 채웁니다.

$$C_i = 1 \text{ if } S_i = S_{i+L} \text{ else } 0$$

그렇게 되면, $2L$개의 연속된 1이 곧 우리가 원하는 3번 연속되는 시퀀스가 됩니다. 연속된 1이 $2L$개 이상인 지점을 모두 세면 총 시퀀스의 개수가 되고, 그 중 L이 가장 큰 것을 구해 출력하면 됩니다.

C : 용감한 오리 (출제 : Pekaz)

이동 가능성에 대한 그래프를 구성하여 두 정점 사이의 경로가 존재할 수 있는지 확인하면 됩니다.

이는 DFS나 BFS 같은 그래프 탐색 알고리즘으로 해결 가능하지만, 이 문제의 경우 제한이 충분히 작으므로 Floyd-Warshall 같은 구현이 간단한 알고리즘을 대신 사용해도 무방합니다. 혹은 Union-Find 로도 reachability를 구할 수 있습니다.

D : 어서와, 영문 이름은 처음이지? (출제 : Antagonist)

전형적인 할당 문제(Assignment Problem)입니다.

이는 Hungarian Method나 Min-cost Max-flow로 해결할 수 있습니다.

팀노트를 가진 자!

E : Annie and Tibber (출제 : Astein)

- 점 P를 기준으로 각 별을 각도 순서대로 정렬한 후, i번째 별이 몇 번째로 나타나는지를 $a[i]$에 저장합니다.
- 마찬가지로 Q를 기준으로 별을 각도 순서대로 정렬한 후, i 번째 별이 몇 번째로 나타나는지를 $b[i]$ 에 저장합니다.
- 그러면 문제는 결국 $a[i] < a[j]$ 이면서 $b[i] > b[j]$ 인 i, j 쌍을 찾는 문제가 됩니다.

E : Annie and Tibber (출제 : Astein)

그러면 문제는 결국 $a[i] < a[j]$ 이면서 $b[i] > b[j]$ 인 i, j 쌍을 찾는 문제가 됩니다.

이 문제는 여러 가지 방식으로 풀 수 있는데, binary indexed tree 등의 자료구조를 활용해서 풀 수 있습니다.

- 먼저 각 별에 대해 $(a[i], b[i])$ 의 쌍을 구한 후, 이들을 $a[i]$를 기준으로 정렬합니다.
- 이후에 정렬된 순서대로 별들을 고려하면서, 자료구조에 데이터 $b[i]$를 추가하거나 $b[i]$ 이상 N 이하인 데이터의 개수를 구해오는 식으로 진행하면 됩니다.

F : 강시대소동 (출제 : Corea)

SCC(Strongly Connected Component)를 구성해 해결할 수 있습니다. 이 때, SCC의 크기가 2 이상인 것의 개수가 답이 됩니다.

- 대신 재귀 함수를 이용해 DFS를 구현하는 경우 스택이 터집니다. 비재귀 짜세요. 두번 짜세요. (SCC 라이브러리 팀노트에 있나요?)

F : 강시대소동 (출제 : Corea)

- 그래프를 구성하는 것이 까다롭거나 귀찮을 수 있는데, 입력으로 나온 발자국의 좌표들을 모두 map에 넣고, 각 발자국에서 맨하탄 거리가 L인 것의 좌표가 map에 있는지 확인만 하면 됩니다.
- 기하스러운 부분은 알아서 하시면 되는데, L이 작아 중간 계산 과정에서 오차가 커도 무관합니다.

윤하 노래 좋아요. 꼭 한번씩 들어보세요. 풍선색은 윤하가 좋아하는 윤하얀색입니다.

G : Colorful Necklaces (출제 : Astein)

답이 x라고 가정해 봅시다.

- 그렇다면, 색상별로 x개를 넘는 구슬은 의미가 없습니다.
- 이 구슬들을 버리고 나서 구슬들의 총 합이 Cx 개를 넘는다면 언제나 올바른 목걸이들을 만들 수 있습니다.

따라서 답인 x에 대해 이분검색하여 해결할 수 있습니다.

G : Colorful Necklaces (출제 : Astein)

이분검색 대신에 유사한 성질을 갖는 다른 탐색법들을 활용할 수도 있습니다 (전체 구슬들의 평균 개수를 기준으로 계산한다거나)

참고: 이러한 아이디어를 얻기 위해서는 좀 더 제한된 경우의 문제부터 생각해 보면 도움이 됩니다.

H : Inventory (출제 : Being)

DP로 해결할 수 있습니다. 이를테면 $D[r][c][n][m]$ 과 같이 잡아

- 몇 개의 아이템씩을 사용했는지,
- 그리고 1x1 아이템이 들어갈 다음 칸은 어디인지를 상태로 가지면
- 해당 칸까지 빈 칸들을 셈으로써 2x1 아이템이 어디어디에 들어갔는지 알아낼 수 있으므로 상태를 표현하기에 충분합니다.

H : Inventory (출제 : Being)

이 경우 상태를 표현하는 변수들로부터 가방의 모양을 복원하는 것이 어려울 수 있으나, 반복문 기반의 DP가 아닌 재귀 형태의 메모이제이션을 선택하는 경우 백트래킹에서 판을 채워 나가는 것과 완전히 같으므로 쉽게 구현할 수 있습니다.

혹은 현재 인벤토리가 채워진 모양, 그리고 앞으로 더 집어넣어야 할 각 아이템의 개수를 하나의 상태로 정의한 후, 상태공간을 탐색하는 방식으로도 풀 수 있습니다. 단 이 경우에는 효율적인 구현이 필요합니다.

I : Mismatched Parenthesis (출제 : yuki)

단순한 구현 문제입니다. 각 괄호에 매치되는 짝을 찾고, 서로 짝인 괄호들의 타입이 다르면 우선순위에 따라 이들을 바꿔줍니다.

N이 작기 때문에 단순한 방법으로도 괄호의 짝을 찾을 수도 있지만, 스택을 이용하여 깔끔하게 구현할 수도 있습니다.

- 괄호 순열을 앞에서부터 차례대로 보면서, 여는 괄호가 나타나면 스택에 추가합니다.
- 닫힌 괄호가 나타난다면, 스택의 맨 위에 있는 괄호를 꺼내어 매치시킵니다.

J : 에어컨을 끈다고 전력난이 해결될까? (출제 : LIBe)

다음과 같이틀릴 경우를 예상하고 냈습니다.

1. Yes, No 찍기, new line 안쓰기
2. http://ideone.com/SWMKf7 초기화 안한 경우
3. http://ideone.com/zwAIUk 입력받다가 NO가 되는 경우
   입력을 멈춰 그 뒤의 결과가 이상하게 나오는 경우

이런 실수를 더이상 하지 않을려면 모다?

J : 에어컨을 끈다고 전력난이 해결될까? (출제 : LIBe)

다음과 같이틀릴 경우를 예상하고 냈습니다.

1. Yes, No 찍기, new line 안쓰기
2. http://ideone.com/SWMKf7 초기화 안한 경우
3. http://ideone.com/zwAIUk 입력받다가 NO가 되는 경우
   입력을 멈춰 그 뒤의 결과가 이상하게 나오는 경우

이런 실수를 더이상 하지 않을려면 모다?

JMBook *두번 사면 두번 안해요.*

K : 영국 아일랜드 여행 (출제 : altertain)

저는 지금 영국에 있습니다. 사실 더블린은 이미 갔다왔지요. 제일 좋았던 여행지는 골웨이라는 아일랜드 서쪽 도시입니다.
여러분, 골웨이 꼭 가세요. 두 번 가세요.
-런던에서 출제자가.

K : 영국 아일랜드 여행 (출제 : altertain)

마지막 정점으로부터 모든 정점으로의 최단 거리를 Dijkstra's algorithm 등을 통해 계산해 내고 나면, 그 이후로는 시작점으로부터 확률 테이블을 채워 나갈 수 있습니다.

- 확률 테이블을 채울 때에는 *아이스크림을 사 먹는 경우*를 일단 배제하고 각 정점마다 어떤 다음 정점을 구할 수 있는지 확률을 계산해 둘 수 있으며,
- 이를 통해 출발점에서부터 확률을 뿌려 나가면 됩니다.
- 이 때 최단 거리가 작아지는 쪽으로만 이동할 수 있으므로, 최단 거리가 먼 정점부터 차례대로 고려하면 계산 순서를 위반하지 않게 됩니다.

Credits : Sponsor

KAKAO

장소 제공 : 한양대학교 (Thanks to. 이준엽)

Credits (문제 준비)

- **문제 풀 관리** : 류원하
- **문제 셀렉션** : 구종만, 김진현, 김진호, 류원하, 이도경, 정현환, 최종욱
- **문제 풀 리뷰** : 구종만, 김진현, 김진호, 류원하, 유원석, 이도경, 정현환, 최종욱
- **문제 번역 및 수정** : 구종만, 김진현, 이후연
- **문제 및 솔루션 조판** : 최종욱

Credits (대회 진행)

- **PC^2 컨테스트 설정** : 류원하
- **대회 진행 스태프** : Showbe Sun, 김상일, 송용주, 유영정, 이재훈, 이준서, 정슬아, 정재헌, 최백준

Credits (문제 준비/번역/검증)

- A번 **ALCHEMY** — 최종욱, 구종만, 김진호, 류원하, 박성진, 유원석, 이준성, 이후연, 정현환
- B번 **BUNKER** — 이준성, 구종만, 김진현, 김진호, 류원하, 박성진, 윤형석, 이도경, 이준성, 이후연, 최종욱
- C번 **BRAVEDUCK** — 박성진, 류원하, 이준성, 이태윤, 정현환, 최종욱, Showbe Sun
- D번 **KAKAONEWHIRE** — 정주영, 구종만, 류원하, 박성진, 유원석, 이도경, 이준성, 정현환, 최종욱
- E번 **ANNIETIBBER** — 김진호, 김진현, 류원하, 박성진, 이준성, 이후연, 정현환, 최종욱
- F번 **GANGSI** — 윤형석, 김진호, 류원하, 박성진, 유원석, 이준성, 정현환

Credits (문제 준비/번역/검증)

- G번 **NECKLACE** — 김진호, 구종만, 김진현, 류원하, 박성진, 윤형석, 이준성, 이후연, 최종욱
- H번 **INVENTORY** — 류원하, 구종만, 김진현, 박성진, 이도경, 이준성, 이후연, 최종욱
- I번 **FIXPAREN** — 김진현, 구종만, 류원하, 박성진, 유원석, 이준성, 이후연, 최종욱
- J번 **HOTSUMMER** — 정현환, 류원하, 박성진, 이도경, 이준성, 최종욱
- K번 **UKEIRETRIP** — 이태윤, 구종만, 김진호, 류원하, 박성진, 이준성, 정현환, 최종욱