CPC 2022 div1 풀이

Official Solutions

중앙대학교 알고리즘 학회 ChAOS

Div 1	문제명	난이도
A	근무 지옥에 빠진 푸앙이 (Large)	Easy
B	86 – 에이티식스 – 1	Medium
C	마트료시카 박스 I	Medium
D	배수로	Hard
E	푸앙이와 계단 수열	Hard
F	푸앙이와 코인	Hard
G	푸앙이와 레벨업	Challenging
H	마트료시카 박스 III	Challenging

A. 근무 지옥에 빠진 푸앙이 (Large)

`map, string, trie`
출제진 의도– **Easy**

✅ 출제자: `smmaker118`

A. 근무 지옥에 빠진 푸앙이(Large)

- ✓ N의 최대 값이 5,000이므로 근무표에 존재하는 이름의 수는 5,000 X 7 X 4 로 최대 140,000개 입니다.
- ✓ 근무자의 수는 제시된 것처럼 최대 5,000명 입니다.(이때 근무자 수를 M이라고 해봅시다.)
- ✓ 근무자의 이름의 길이는 제시된 것처럼 최대 20입니다.

A. 근무 지옥에 빠진 푸앙이(Large)

- ✓ Small 버전의 풀이로 푼다면 O(NM)이 700,000,000으로 시간초과가 나게 됩니다.
- ✓ 근무자 이름을 탐색할때 map, trie 등을 사용하면 logM시간에 탐색할 수 있습니다.
- ✓ 따라서 총 시간복잡도 O(NlogM)에 해결할 수 있습니다.

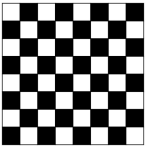
B. 86 — 에이티식스 — 1

`backtracking, bruteforcing`

출제진 의도 – **Medium**

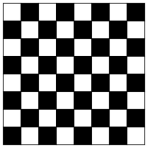
✅ 출제자: `BothEarRim`

B. 86–에이티식스–1



- ✓ 체스판처럼 검은색, 흰색 칸으로 영역을 나눠 생각해보면
  신에이 노우젠은 같은 색의 영역만 갈 수 있습니다.
- ✓ 레기온을 해치우는 경우는 같은 색의 영역에 있는 레기온들 뿐입니다.
- ✓ 따라서, 모든 레기온이 신에이 노우젠과 같은 색의 영역에만 있는지 검사하면 됩니다.

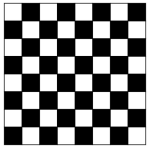
B. 86–에이티식스–1



✓ 특정 좌표가 어느 색의 영역인지 판단하는 방법은 간단하게 좌표의 행+열 값이 홀/짝인지로 구분하면 됩니다.

✓ 같은 영역의 좌표간 최소 이동 시간은 $\max(|\ y_1 - y_2\ |, |\ x_1 - x_2\ |)$ 입니다.

B. 86-에이티식스-1



- ✓ 모든 레기온을 해치우는 최소 시간을 구하는 방법은 여러가지 있습니다.
- ✓ 백트래킹으로 방문할 레기온 순서를 바꿔가며 최단 시간을 찾습니다. Python이라면 permutations를 써도 됩니다.
- ✓ TSP와 비슷하게 bit DP로 푸는 방법도 있으며 더 빠릅니다.

C. 마트료시카 박스 I

`ad_hoc, bfs, graphs, graph_traversal`
출제진 의도 – **Medium**

✅ 출제자: wapas

C. 마트료시카 박스 I

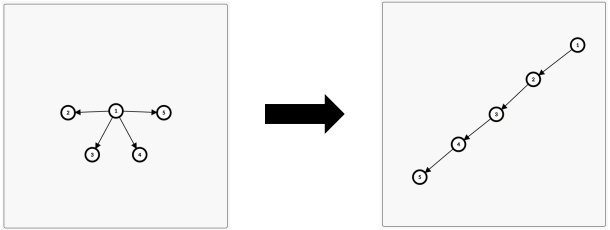
- ✓ 문제 상황을 그래프 용어로 치환하면 아래와 같습니다.
  - ✓ 설계도 = 트리 그래프
  - ✓ 박스 = 노드
  - ✓ 메인 박스 = 루트 노드
  - ✓ 서브 박스 = 자식 노드
- ✓ ‘상자 포함 관계가 유지되었다’의 의미는 **수정 전 트리에서 모든 노드의 조상-후손 관계가 수정 후 트리에서도 유지되었다**를 의미합니다.

C. 마트료시카 박스 I

- ✓ i번 노드의 레벨 (루트 노드로부터 거리)을 v_i 라고 정의합시다.
- ✓ 조상-후손 관계를 유지하려면 수정 전 트리에서 서로 다른 두 임의 노드 i번, j번에 대해 $v_i > v_j$ 이었다면, 수정 후에서도 $v_i > v_j$ 를 유지하면 됩니다.
- ✓ 한편, 위 조건만 유지한다면 노드의 위치는 마음대로 이동이 가능합니다.

C. 마트료시카 박스 I

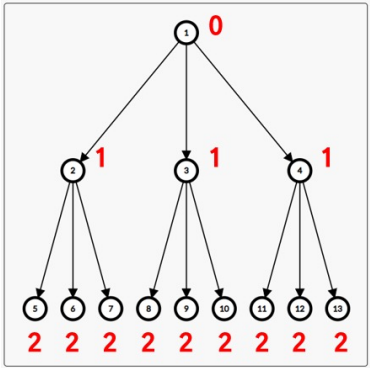
- ✓ 조건을 지키면서 노드를 재배치를 하는 방법은 여러가지 있습니다.
- ✓ 그 중에서 일자 모양의 트리를 만들어 재배치하는 방법이 있습니다.

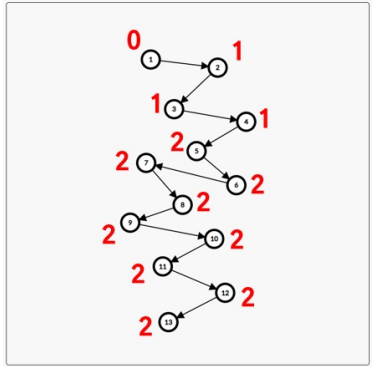


C. 마트료시카 박스 I

- ✓ 수정 후 트리를 추가 노드 없이 일자 모양의 트리로 만들었을 때, 루트 노드를 1번, 단말 노드를 N번이라고 정의합시다.
- ✓ 이때 수정 후 트리는 서로 다른 두 임의 노드 i번, j번에 대해 $i \geq j$일 때 $v_i \geq v_j$를 만족해야 합니다.
- ✓ 즉, 수정 후 트리를 일자 모양 트리로 만들었을 때, 수정 전 노드의 레벨이 비내림차순으로 구성되어야 합니다.

C. 마트료시카 박스 I


1
0
2
1
3
1
4
1
5
6
7
8
9
10
11
12
13
2
2
2
2
2
2
2
2
2


0
1
1
1
2
2
2
2
2
2
2
2
2
2
2
1
2
3
4
5
6
7
8
9
10
11
12
13

C. 마트료시카 박스 I

- ✓ 수정 전 트리 노드의 레벨이 비내림차순으로 배치하기 위해 BFS를 활용합니다.
- ✓ 수정 전 트리의 루트 노드에서 BFS를 시작합니다. 그 후 방문한 순서대로 일자 트리를 구성하면 트리 노드의 레벨이 비내림차순으로 수정 후 트리를 만들 수 있습니다.

C. 마트료시카 박스 I

✓ 그 외 재귀를 활용한 전위 순회 풀이법, 큐를 활용한 위상 정렬 풀이법도 존재합니다.

D. 배수로

`data structures, disjoint set`

출제진 의도 – **Hard**

✅ 출제자: saywoo

D. 배수로

- ✓ 다음에 해당하는 도시들을 하나의 그룹이라고 하겠습니다.
  - ✓ 공사를 해서 배수로 용량이 공유되는 도시들
  - ✓ 다른 도시들과 배수로 용량이 공유되지 않는 하나의 도시
- ✓ 처음에는 N개의 도시 모두 각각의 그룹에 속해있습니다.
- ✓ 이 상태에서 홍수가 날 도시의 개수를 전처리합니다.

D. 배수로

- ✓ 이후에는 쿼리를 처리합니다
  - ✓ 1번 쿼리가 주어질 때: 입력 받은 두 도시의 그룹이 다르다면 두 도시가 속한 그룹을 하나로 합치고, 그 그룹이 홍수가 날지 확인하여 전처리 하였던 홍수가 발생할 도시의 개수를 업데이트합니다.
  - ✓ 2번 쿼리가 주어질 때: 전처리 하였던 홍수가 발생할 도시의 개수를 출력합니다.
- ✓ 그룹을 합치는 연산을 유니온 파인드로 구현하면 시간 안에 문제를 해결할 수 있습니다.

E. 푸앙이와 계단 수열

dp

출제진 의도 – **Hard**

✅ 출제자: dkvltmxhf

E. 푸앙이와 계단 수열

- ✓ 수열을 양쪽에서 전부 삭제 가능한 것과, 한쪽에서만 삭제 가능한 것은 동치임을 알 수 있습니다.
- ✓ 따라서 왼쪽에서만 삭제를 한다고 가정하고 문제를 해결 가능합니다.

E. 푸앙이와 계단 수열

- ✓ 수열의 첫번째 원소부터 N번째 원소까지 삭제했을 때의 최솟값을 알아야합니다.
- ✓ 해당 테이블은 최적 부분 구조를 만족하므로 Dynamic Programming으로 해결 가능합니다.
- ✓ 계단 수열을 삭제하지 않았을 때

$$DP[i] = \min(dp[i-1], dp[i-2], dp[i-3]) + 1$$

라는 식을 얻을 수 있으며 $A_{i-k} \dots A_i$ 가 계단 수열을 만족할 때 다음 식이 성립합니다.

$$DP[i] = \min(dp[i-1], dp[i-2], dp[i-3], dp[i-k]) + 1$$

E. 푸앙이와 계단 수열

- ✓ 길이가 K인 계단 수열을 매번 반복분으로 확인하는 것은 많은 시간이 걸립니다.
- ✓ 따라서 연속된 계단 수열을 누적합을 이용해 관리할 수 있습니다.
- ✓ 하나의 DP 테이블을 채우는데 $O(1)$ 의 시간이 걸리므로 모든 DP 테이블을 채우는데 $O(N)$ 이 소요됩니다

F. 푸앙이와 코인

`dp, prefix_sum`

출제진 의도 – **Hard**

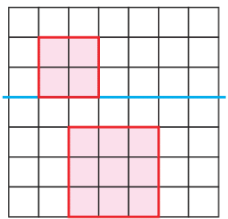
✅ 출제자: `halin`

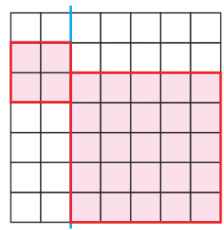
F. 푸앙이와 코인

✓ 한번 그물을 쳐서 코인을 얻는 방법은 $\sum i^2 \approx N^3/3$ 이므로, 두 번 그물을 치는 모든 방법을 확인하는 것은 $O(N^6)$으로, $N = 400$ 일 때 시간 초과가 날 것입니다.

F. 푸앙이와 코인

- ✓ 다음과 같이 문제의 조건을 만족시키는 빨간 경계로 표시된 두개의 그물을 찾았다고 가정해봅시다.
- ✓ 두 그물은 각각 파란색 경계선으로 구분되는 두 개의 영역에서 찾을 수 있는 최선입니다.
- ✓ 또한 가능한 모든 경우는 가로선 또는 세로선의 경계선으로 두 개의 그물을 구분할 수 있습니다.





F. 푸앙이와 코인

- ✓ 따라서 $(0, 0)$ 칸과 (i, j) 칸을 꼭짓점으로 하는 영역 내에서 한 번의 그물을 쳐서 얻을 수 있는 최대의 코인을 저장하는 배열 A와, $(N-1, N-1)$칸과 (i, j) 칸을 꼭짓점으로 하는 영역 내에서 한 번의 그물을 쳐서 얻을 수 있는 최대의 코인을 저장하는 배열 B를 각각 만들면, 가로와 세로 경계선 총 $2(N-1)$ 번의 검사를 거쳐 최대 코인을 찾을 수 있습니다.
- ✓ 그러한 배열은 크기가 1^2 부터 $(N-1)^2$까지의 가능한 모든 그물에 대해 각 그물을 쳤을 때 얻을 수 있는 코인의 개수를 오른쪽 아래 꼭짓점과 왼쪽 위 꼭짓점의 좌표에 해당하는 배열 A, B의 위치에 저장하고, A, B 각각에 누적 최댓값을 적용해 얻을 수 있습니다.

F. 푸앙이와 코인

- ✓ 각 그물을 쳤을 때 얻을 수 있는 코인의 개수는 처음 격자의 정보에 2차원 누적합을 이용해 빠르게 얻을 수 있습니다.
- ✓ 따라서 총 시간복잡도는 $O(N^3)$ 에 해결할 수 있습니다.

G. 푸앙이와 레벨업

`parametric_search`, `sliding_window`

출제진 의도 – **Challenging**

✅ 출제자: `dkvltmxhf`

G. 푸앙이와 레벨업

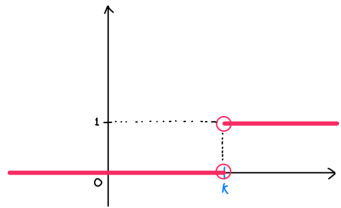
- ✓ 단순 연산으로 K를 최적화 시키기엔 굉장히 어렵습니다.
- ✓ 따라서 이를 결정 문제로 바꾸어 만약, K가 고정되어있다면, 푸앙이는 레벨업을 할 수 있는가? 라는 질문을 생각해볼 수 있습니다.

G. 푸앙이와 레벨업

✓ 발도술 범위 K가 주어졌을 때 얻을 수 있는 경험치를 sum이라 할때 F(k)는 다음과 같이 정의할 수 있습니다.

$$F(k) = \begin{cases} 1 \; if \; sum \geq r \\ 0 \; if \; sum < r \end{cases}$$

✓ K에 따른 sum이 단조증가성을 보임을 쉽게 알 수 있으므로 F(k)의 형태는 다음과 같습니다.



G. 푸앙이와 레벨업

- ✓ 따라서 우리는 이분 탐색을 이용하여 F(k)가 1이되는 정수 지점을 logN 시간에 찾을 수 있습니다.
- ✓ 이제 K가 주어졌을 때, sum을 계산하는 방법을 찾아야합니다.
- ✓ 모든 지점에서 $K \times K$ 그리드의 최댓값을 반복문으로 구한다면, $O(N^2K^2)$이 소요됩니다.
- ✓ 중요한 사실은, 임의의 점 (i,j)에서 살펴볼 영역과, 점(i,j+1)에서 살펴볼 영역이 크게 차이 나지 않는 다는것 입니다.

G. 푸앙이와 레벨업

- ✓ 만약, 1차원 k영역에서의 최댓값을 모두 찾을 수 있다면, 이는 2차원으로도 적용 가능합니다.
- ✓ 다음은 k=2일때, 기존 배열의 모든 행에 대해 최댓값을 적용한 결과입니다. 각 색의 영역은 오른쪽의 하나의 수에 대응됩니다.

5	3	2	1
2	7	1	4
4	8	5	2
1	4	2	4

5	3	2	1
7	7	4	4
8	8	5	2
4	4	4	4

G. 푸앙이와 레벨업

✓ 얻어진 2차원 행렬(행에 대한 max)에 다시 열에 대한 최댓값을 적용할 수 있습니다.

5	3	2	1
7	7	4	4
8	8	5	2
4	4	4	4

7	7	4	4
8	8	5	4
8	8	5	4
4	4	4	4

✓ 이렇게 2번의 1차원 연산을 통해 우리는 KxK 그리드에 대한 최댓값을 얻을 수 있습니다.

G. 푸앙이와 레벨업

- ✓ - 이제 1차원 K영역에 대한 최댓값을 구하기만 하면 해결할 수 있습니다.
- ✓ - 정해는, Monotone queue Technique을 사용하여 해결하는 것입니다.
- ✓ - 단조 큐 테크닉을 이용하면 1차원 최댓값을 $O(N)$ 시간 안에 구할 수 있습니다.
- ✓ - 따라서 모든 연산을 $O(N^2 logN)$ 안에 실행할 수 있고, 이는 시간 내에 해결이 가능합니다.
- ✓ - 추가적으로, 1차원 최댓값을 얻는 데에 우선순위큐를 사용한 $O(N^2 log^2 N)$ 로도 해결할 수 있습니다.

H. 마트료시카 박스 III

`data_structures, lca, sparse_table, tree`
출제진 의도 – **Challenging**

✅ 출제자: wapas

H. 마트료시카 박스 III

- 문제 상황을 그래프 용어로 치환하면 아래와 같습니다.
  - 설계도 = 트리 그래프
  - 박스 = 노드
  - 메인 박스 = 루트 노드
  - 서브 박스 = 자식 노드
- ‘상자 포함 관계가 유지되었다’의 의미는 **수정 전 트리에서 모든 노드의 조상-후손 관계가 수정 후 트리에서도 유지되었다**를 의미합니다.

H. 마트료시카 박스 III

- ✓ K < S이면 0을 출력합니다.
- ✓ 수정 후 트리에서 자식의 개수가 M개 초과이면 0을 출력합니다.
- ✓ i번 노드의 레벨 (루트 노드로부터 거리)을 v_i 라고 정의합시다.
- ✓ 조상-후손 관계를 유지하려면 수정 전 트리에서 서로 다른 두 임의 노드 i번, j번에 대해 $v_i > v_j$ 이었다면, 수정 후에서도 $v_i > v_j$ 를 유지하면 됩니다.

H. 마트료시카 박스 III

- ✓ 수정 전 트리에서 서로 다른 A, B, C 노드가 있을 때, A가 B의 부모 노드, B가 C의 부모 노드라고 합시다.
- ✓ 수정 후 트리에서 A & B의 조상-후손 관계, B & C의 조상-후손 관계를 만족한다면 A & C의 조상-후손 관계도 만족합니다.
- ✓ 따라서 수정 전 트리에서의 부모-자식 관계의 모든 쌍이 수정 후 트리에서 조상-후손 관계를 만족하는지 확인을 하면 됩니다.
- ✓ 즉, 한 쌍이라도 조상-후손 관계를 만족하지 않는다면 0을 출력하고, 모든 쌍이 만족하면 1을 출력하면 됩니다.

H. 마트료시카 박스 III

- ✓ (루트 노드가 존재하는) 트리에서 두 임의 노드 A, B가 조상-후손 관계임을 확인하려면 최소 공통 조상 노드(LCA)를 활용할 수 있습니다.
- ✓ A, B의 LCA가 A 또는 B라면 두 노드는 조상-후손 관계입니다.
- ✓ 입력 범위가 크므로, 한 칸씩 확인하며 LCA를 구하면 시간 복잡도 $O(N * (N + S))$로 시간 초과가 발생합니다.
- ✓ 따라서 Sparse Table 활용하여 LCA를 구하면 $O(N * \log(N + S))$로 제한 시간 안에 모든 LCA를 구할 수 있습니다.