2024 상반기 전남대학교 PIMM 알고리즘 파티

공식 해설

문제		의도한 난이도	출제자
A	시계탑	Bronze	고민규jjkmk1013
B	전주 듣고 노래 맞히기	Bronze	이윤수lys9546
C	수열 회전과 쿼리	Silver	최정환jh01533
D	육각타일미로 탈출기	Silver	이윤수lys9546
E	전역 역전	Gold	박종현belline0124
F	힘세고 강한 아침	Gold	김근성onsbtyd
G	Move or Block!	Gold	최정환jh01533
H	산림의 수호자	Gold	정영도0do
I	세 수 XOR과 쿼리	Platinum	최정환jh01533

A. 시계탑

`implementation`

출제진 의도 — Bronze

- 제출 219회, 정답 180명 (정답률 82.648%)
- 처음 푼 참가자: **ychangseok**, 1분
- 출제자: 고민규`jjkmk1013`

A. 시계탑

- 분침의 속도가 변하는, 바뀐 시계탑의 30분을 기점으로 식을 나누어 해결할 수 있습니다.
- 올바른 시각의 분 $f(x)$에 대해서, 바뀐 시계탑의 분침이 가리키는 분 x로 식을 세우면 아래와 같습니다.
- $$f(x) = \begin{cases} \frac{1}{2}x & (x \leq 30) \\ \frac{3}{2}x + 15 & (x > 30) \end{cases}$$
- 바뀐 시계탑의 시간에 대해 0분부터 1씩 더해가며 실제 시간을 계산하는 시뮬레이션도 가능합니다.

B. 전주 듣고 노래 맞히기

`implementation, brute_force`

출제진 의도 — Bronze

- 제출 192회, 정답 158명 (정답률 83.333%)
- 처음 푼 참가자: **nflight11**, 2분
- 출제자: 이윤수`lys9546`

B. 전주 듣고 노래 맞히기

- 노래의 제목과 노래에서 등장하는 첫 세개의 음이름을 `map`등의 자료구조에 저장하여 M개의 쿼리들을 처리 할 수 있습니다.
- 첫 세 음이 동일한 노래가 두 개 이상이거나 존재하지 않을 때를 유의해서 구현합니다.

C. 수열 회전과 쿼리

`prefix_sum, implementation`

출제진 의도 — Silver

- 제출 288회, 정답 74명 (정답률 29.514%)
- 처음 푼 참가자: **mj1000j**, 3분
- 출제자: 최정환jh01533

C. 수열 회전과 쿼리

- 수열을 오른쪽으로 k만큼 회전시키는 방법을 생각해봅니다.
- 단순히 덱 자료구조를 이용해서 실제로 마지막 요소를 빼서 앞쪽에 넣는 작업을 k번 반복하게 된다면 시간복잡도 $O(NQ)$로 시간초과가 나게 됩니다.
- 실제로 회전시키지 않고 회전시킨 것과 같은 효과를 얻을 수 있는 방법이 있을까요?

C. 수열 회전과 쿼리

- 수열을 원수열이라고 생각해봅시다. 원수열이 회전했을 때 모든 원소들의 이동한 거리는 동일하고 원소 간 좌우 관계가 항상 일정하게 유지됩니다.
- 즉, 회전하지 않은 상태의 수열과 한 원소가 이동한 거리만 알고 있으면 현재 수열을 알 수 있습니다.
- 이동한 거리를 D라 합시다. (처음에 $D = 0$)
- 1번 쿼리가 들어오면 오른쪽으로 k만큼 회전하기 때문에 $D += k$를 한 것과 같습니다.

C. 수열 회전과 쿼리

- 2번 쿼리가 들어오면 왼쪽으로 k만큼 회전하기 때문에 $D -= k$를 한 것과 같습니다.

C. 수열 회전과 쿼리

- 위 정보를 이용해서 3번 쿼리를 처리해봅시다.
- 입력으로 들어오는 a,b를 D를 이용해서 다시 계산합니다. 편의상 0-based(0부터 인덱스를 매김)를 사용합니다.
- $x = (a - 1 - D) \bmod N$ ($\bmod$는 나머지 연산, 음수 $\bmod$ 처리에 주의)
- $y = (b - 1 - D) \bmod N$

(직접 원수열을 그려서 어떻게 동작하는지 확인해보시면 좋습니다.)

C. 수열 회전과 쿼리

- 이렇게 되면 회전하지 않은 상태의 수열에서 x인덱스부터 y인덱스까지의 합을 구하는 문제가 되고 이는 누적합 테크닉을 이용해서 구할 수 있습니다.
- $y < x$일 경우 (x부터 $(N-1)$까지의 구간합) + (0부터 y까지의 구간합)으로 처리해야 하는 것을 주의합시다.
- 총 시간복잡도는 $O(N+Q)$입니다.

D. 육각타일미로 탈출기

`graph, bfs`

출제진 의도 — Silver

- 제출 88회, 정답 60명 (정답률 69.318%)
- 처음 푼 참가자: **kyo20111**, 11분
- 출제자: 이윤수`lys9546`

D. 육각타일미로 탈출기

- 육각형으로 이루어진 인접한 타일들을 홀수 행과 짝수 행을 각각 다르게 처리하여 그래프를 모델링 할 수 있습니다.
- 그 후, 시작점 $(0, 0)$ 에서 $(N-1, M-1)$ 까지 BFS를 통해 최단경로 타일의 개수를 구할 수 있습니다.

E. 전역 역전

`dp, knapsack`

출제진 의도 — Gold

- 제출 66회, 정답 25명 (정답률 37.879%)
- 처음 푼 참가자: **dabbler1**, 25분
- 출제자: 박종현belline0124

E. 전역 역전

- 주어진 세 개의 쿼리 모두 영도의 전역을 상대적으로 앞당깁니다.
- 본 문제에서는 종현과 영도의 상대적인 전역일 차이를 물어보므로, 세 개 쿼리를 영도의 전역을 앞당기는 것처럼 처리할 수 있습니다.

E. 전역 역전

- 조기 전역: D만큼 영도의 전역을 앞당깁니다.
- 군기교육대: D만큼 영도의 전역을 앞당깁니다.
- 임기제 부사관: $M \times 30$ 만큼 영도의 전역을 앞당깁니다.

E. 전역 역전

- 쿼리의 대상을 한 명으로 통일한다면 0-1 냅색으로 이 문제를 해결할 수 있습니다.
- 둘 간의 상대적인 전역일 차이를 처리해야 하므로, 종현의 전역을 뒤로 미루는 것처럼 처리할 수 있습니다.

F. 힘세고 강한 아침

`floyd_warshall, dijkstra`

출제진 의도 — Gold

- 제출 75회, 정답 22명 (정답률 29.333%)
- 처음 푼 참가자: **kyo20111**, 21분
- 출제자: 김근성`onsbtyd`

F. 힘세고 강한 아침

- 이 문제의 지문을 요약하면 "S번 노드에서 E번 노드로 갈 때, K번 노드를 거치지 않고 가는 최소비용을 Q번 구하여라" 입니다.
- 이런 최단 경로를 구하는 알고리즘에는 대표적으로 벨만포드, 플로이드 워셜, 다익스트라가 있습니다.

F. 힘세고 강한 아침

- **벨만포드**
  - $O(N^3M)$의 시간복잡도로 동작합니다. 주어진 제한에서 대략 100억으로 시간초과가 발생합니다.
- **플로이드 워셜**
  - $O(N^4)$ 의 시간복잡도로 동작합니다. 주어진 제한에서 대략 1억으로 제한시간 내에 통과합니다.

F. 힘세고 강한 아침

- **힙 다익스트라**
  - $O(N^2(N+M)\log N)$ 의 시간복잡도로 동작합니다. 주어진 제한에서 대략 7억으로 시간초과가 발생합니다.
- **N^2 다익스트라**
  - $O(N^4)$ 의 시간복잡도로 동작합니다. 주어진 제한에서 대략 1억으로 제한시간 내에 통과합니다.

F. 힘세고 강한 아침

- 의도된 풀이는 $O(N^2)$ 다익스트라를 N^2회, 혹은 $O(N^3)$ 플로이드 워셜을 N회 동작시키는 것입니다.
- **다익스트라**
  - `D[N][N][N]` 형식으로 배열을 선언합니다.
  - i번 노드에서 시작해 j번 노드를 방문하지 않는다면 `D[j][i][1~N]`을 채우는 방식으로 다익스트라를 N^2 번 미리 돌려 계산해 둡니다.

F. 힘세고 강한 아침

- **플로이드 워셜**:
  - `D[N][N][N]` 형식으로 N층의 $N \times N$ 배열을 선언합니다.
  - 플로이드 워셜 루프의 가장 바깥 `k` 루프가 k번 노드로 들어오는 경로와 k번 노드에서 나가는 경로를 연결한다는 점을 이용합니다.
  - `D[1~k-1][1~N][1~N], D[k+1 ~ N][1~N][1~N]`, 즉 k번째 층을 제외한 나머지에서 갱신시켜줍니다.

F. 힘세고 강한 아침

위의 과정을 계산해두고 쿼리마다 답하면 됩니다. 총 시간복잡도는 $O(N^4 + Q)$ 입니다.

- 핵심

1. 1억회 연산은 1초 내에 돌아갈 수 있다.
2. 완전 그래프에서는 힙 다익스트라가 N^2 다익스트라보다 느리다.
3. 플로이드의 가장 바깥 k 루프는 들어오는 경로와 나가는 경로를 연결한다.

G. Move or Block!

`game_theory`

출제진 의도 — Gold

- 제출 74회, 정답 9명 (정답률 12.162%)
- 처음 푼 참가자: **fermion5**, 49분
- 출제자: 최정환`jh01533`

G. Move or Block!

- 편의상 움직이는 행동을 move, 벽을 세우는 행동을 block이라 표현하고, 게임보드의 밖에는 벽이 세워져 있다고 가정합니다.
- 우선 자명한 케이스를 먼저 처리해봅시다.
- `~~XOX~~` – 후턴승(입력으로 주어지지 않습니다.)
- `~~XO.~~`, `~~.OX~~` – 선턴승
- 이 다음 설명은 위 케이스를 고려하지 않습니다.

G. Move or Block!

이 문제를 풀기 위해서는 몇 가지 관찰들을 잘 조합해야 합니다.

1. 게임을 끝낼 수 있거나 어쩔 수 없는 경우를 제외하고, 말의 인접한 칸을 block하거나 벽이 있는 곳으로 move를 하면 안됩니다.
    - 바로 패배로 이어지는 수입니다.
    - 다른 관찰들은 이 규칙이 지켜진다는 전제 하에 성립합니다.

G. Move or Block!

2. 필승법을 가지고 있는 플레이어는 모든 턴에 block을 하는 것이 최선입니다.
   - 만약에 move만이 최선이라고 가정해봅시다.

     다음 턴인 사람이 다시 원래 자리로 move한다면 해당 패턴이 반복되고 게임이 영원히 끝나지 않습니다. 즉, 이길 수 없습니다.
   - 따라서 필승법을 가지고 있다면 모든 턴에 block을 하는 것이 최선이라는 것을 알 수 있습니다.

G. Move or Block!

3. ~~x.o.x~~로 말이 막혀 있을 때
    - 남아 있는 .의 개수가 홀수면 선턴승, 짝수면 후턴승입니다.
4. 게임이 끝난다면 3.의 케이스를 반드시 거쳐갑니다.
    - 1.에 의해서 항상 성립합니다.
5. 현재 3.경우가 아니라면 1.을 만족하면서 .개수의 기우성을 자신의 턴이 끝나고 원하는 대로 만들 수 있습니다.
    - 현재의 기우성을 유지하고 싶으면 move 아니라면 block하면 됩니다.

G. Move or Block!

- 플레이어들은 3.의 경우를 만들면서 .의 개수를 짝수로 만드는 수를 두면 이깁니다.
- 3.을 만들 때는 반드시 block을 해야함으로 승리하려면 자신의 턴에 .의 개수가 홀수여야 합니다
- 하지만 현재 시점에서 3.이 아닐 때 5.에 근거해서

  .의 개수가 짝수인 상황을 항상 상대방에게 넘겨줄 수 있습니다.

  4.에 따라 이는 상대방이 절대로 이길 수 없게 만들 수 있음을 뜻합니다.

G. Move or Block!

즉, 게임의 첫 번째 턴에 3.을 만들지 못하면 항상 무승부가 나옵니다.

따라서 전체를 5가지 케이스로 나눌 수 있습니다.

- ~~XOX~~: 후턴승 (입력으로 주어지지 않아 처리하지 않아도 무방합니다.)
- ~~XO.~~, ~~.OX~~: 선턴승
- ~~X.O.X~~: .의 개수가 홀수면 선턴승, 짝수면 후턴승
- ~~X.O..~~, ~~..O.X~~: .의 개수가 홀수면 선턴승, 짝수면 무승부
- ~~..O..~~: 무승부

H. 산림의 수호자

`tree, graph_theory, graph_traversal, greedy`

출제진 의도 — Gold

- 제출 16회, 정답 4명 (정답률 25%)
- 처음 푼 참가자: **dabbler1**, 117분
- 출제자: 정영도`0do`

H. 산림의 수호자

이 문제의 출력으로 요구되는 것은 복제하기 사용 횟수의 최댓값입니다.

- 복제하기는 지나가는 모든 정점에서 실행하는 것이 가장 효율적임을 알 수 있고, 기다리기 또한 사용하지 않는 것이 가장 효율적임을 알 수 있습니다.
- 다시 말해 “근성이 가능한 많은 정점을 지나가는 경로로 이동했을 때, 방문한 정점의 개수는 몇 개인가?” 로 단순화할 수 있습니다.

H. 산림의 수호자

$O(N)$ 풀이

- 한 개의 간선을 3회 이상 이용하는 것은 비효율적인 행동입니다.
- 두 개의 정점 사이를 불필요하게 왔다갔다 하는 행동입니다.
- 즉, 모든 간선은 2회 이하로만 이용해야 합니다.

H. 산림의 수호자

- 2회 이용한 간선의 수가 한 개 늘어날 때마다 방문한 정점의 수가 한 개 늘어납니다.
- 동일하게, 1회 이용한 간선의 수가 한 개 늘어날 때에도 방문한 정점의 수가 한 개 늘어납니다.
- 2회 이용한 간선과 1회 이용한 간선의 합이 최대일 때 최적의 해가 보장됩니다.
- 근성이 이동하는 임의의 경로를 만든다고 가정했을때, 1회 이용한 간선의 개수는 b번 정점으로부터 경로의 목적지가 되는 정점 사이 간선의 개수와 동일합니다.

H. 산림의 수호자

- “근성이 a번 정점이 있는 방향으로 최대한 이동했을 때 최적의 답이 보장된다” 는 것을 연역적으로 도출할 수 있습니다.
- 최적 경로는 한 개 이상입니다.
- 최적 경로의 목적지가 되는 트리 위 임의의 정점을 c번 정점이라고 가정하겠습니다.

H. 산림의 수호자

- a번 정점을 트리의 루트 노드로 정할 때 c번 정점이 b번 정점의 자손 노드라면
- 근성이 최대한 불길 방향으로 이동한 후 c번 정점으로 이동하는 게 최적 경로 중 하나가 됨이 보장됩니다.
- 1회 이용한 간선 개수는 변하지 않은 채, 2회 이용한 간선 개수가 최대화되기 때문입니다.

H. 산림의 수호자

- a번 정점을 트리의 루트 노드로 정할 때, c번 정점이 b번 정점의 자손 노드가 아니라면
- b번 정점에서 c번 정점으로 도달하기 위해 근성은 a번 정점이 있는 방향으로 이동하여야만 합니다.
- 근성이 위치한 정점의 자손 노드가 c번 정점이 될 때까지 이동했다면
- 근성이 최대한 불길 방향으로 이동한 후 c번 정점으로 이동하는 게 최적 경로 중 하나가 됨이 보장됩니다.

H. 산림의 수호자

$O(N)$의 시간복잡도에 동작하는 풀이를 도출할 수 있습니다.

1. a번 정점과 b번 정점의 중간점을 구합니다.
    - 이 중간점은 b번 정점에서 a번 정점 방향으로 최대한 이동했을 때, 근성이 서있을 수 있는 마지막 정점입니다.
    - 이 정점을 d번 정점이라고 가정합니다.
2. 트리 상의 임의의 정점을 선택한 후, 이 정점을 c번 정점이라고 가정합니다.

H. 산림의 수호자

3. c번 정점이 목적지일 때의 최적 경로와, 최적 경로에서의 지나간 정점 개수를 구합니다.
    - [(b번 정점에서부터 d번 정점 사이의 정점 개수) + (d번 정점에서부터 c번 정점 사이의 정점 개수) – (앞선 두 경로의 겹치는 부분의 정점 개수)] = 지나간 정점 개수
4. 모든 정점에 대하여 $2.$ ~ $3.$번 과정을 반복합니다. 모든 정점에 대해 $3.$번 과정에서 나온 정점 개수의 최대값이 문제의 정답이 됩니다.

- $3.$번 과정을 수행하기 위해 필요한 값들은 $4.$번 과정을 수행하기 전, 미리 $O(N)$의 시간에 계산해둘 수 있습니다.

I. 세 수 XOR과 쿼리

`bitmask, lazyprop`

출제진 의도 — Platinum

- 제출 16회, 정답 1명 (정답률 12.5%)
- 처음 푼 참가자: **kyo20111**, 63분
- 출제자: 최정환jh01533

I. 세 수 XOR과 쿼리

1번 쿼리를 먼저 처리해봅시다.

- A_i가 작기 때문에 어떤 수를 만들 수 있는 모든 세 수의 쌍을 전처리할 수 있습니다.
- x를 만들 수 있는 세 쌍의 경우의 수는 715개로, 구간 $[l, r]$에 존재하는 수의 개수만 안다면 1번 쿼리를 처리할 수 있어 보입니다.
- 세그먼트 트리를 이용하면 구간의 수의 개수를 구할 수 있지만 $O(Q64 \log N + 715Qc)$으로 시간 안에 돌아가기는 무리가 있어 보입니다.

I. 세 수 XOR과 쿼리

- ‘x를 만들 수 있는 세 쌍의 경우의 수’를 잘 살펴봅시다.
- 세 쌍 중에 3개의 수가 같은 경우는 (x, x, x) 꼴이고 2개의 수가 같은 경우는 (y, y, x) 꼴입니다.
- 이는 x가 3개 이상 존재하는가? 어떤 수가 2개 이상 존재하고 x가 존재하는가? 로 단순화 할 수 있습니다.
- 따라서 해당 형태를 따로 처리해준다면 수의 개수가 아닌 수의 집합만 구하면 되고 이는 비트 집합을 이용한 세그먼트 트리로 빠르게 구할 수 있습니다.

I. 세 수 XOR과 쿼리

- 위 정보를 토대로 비트 집합을 이용해서 세그먼트 트리를 구성해봅시다.
- `unsigned long long`(또는 `bitset`) 자료형을 이용해서 i번째 비트에 현재 구간에 i라는 수가 존재하는가?를 저장합니다.
- 노드를 합치는 연산은 두 노드의 `OR` 연산으로 빠르게 계산할 수 있습니다.
- 이와 같이 구간의 수 집합을 $O(\log N)$으로 빠르게 구했습니다.

I. 세 수 XOR과 쿼리

- (x, x, x)꼴과 (y, y, x)꼴을 처리하는 방법은 후술하고, 이를 제외한 651개의 경우를 확인해야 합니다.
- 연산량이 많아보여도 구간의 수 집합이 비트이기 때문에 단순한 & 연산 한번으로 경우 하나를 처리할 수 있어 빠른 속도에 처리할 수 있습니다.
- 651개의 경우의 수를 다 확인하지 않고, 집합에 포함된 숫자만 확인한다면 XOR의 특성에 의해 연산량이 2배 이상 줄어듭니다.

I. 세 수 XOR과 쿼리

2번 쿼리를 처리해봅시다.

- 느리게 갱신되는 세그먼트 트리로 처리할 수 있습니다. 이 역시 비트연산으로 고속화 할 수 있습니다.
- 노드에 $+x \bmod 64$를 하는 것은 비트를 오른쪽으로 x칸 회전시키는 것과 동일합니다.
- `node = ((node << (64-x)) | (node >> x)) & (1<<64-1` 64비트 자료형을 사용한다면 없어도 무방함, 오버플로우에 주의)

I. 세 수 XOR과 쿼리

- ‘x가 3개이상 존재하는가? 어떤 수가 2개 이상 존재하고 x가 존재 하는가’ 를 처리 하는 방법은 여러가지가 있습니다.
  - 노드에 2개 이상 존재하는 수 집합, 3개 이상 존재하는 수 집합을 추가로 넣습니다. 이 역시 비트 연산으로 간단하게 처리 가능합니다.
  - 3개 이상 존재하는 수 집합은 넣지 않아도 비둘기 집의 원리로 처리 가능합니다.

I. 세 수 XOR과 쿼리

- 전체 시간복잡도는 $O(N \log N + Q \log N + 651Q)$입니다.
- $651Q$ 부분이 커보이지만 앞서 서술한 것처럼 &연산 한 번에 처리가 가능한 간단한 연산이고, 구간 안에 있는 수들만 탐색하면 XOR의 특성에 따라 연산량이 절반으로 줄어듭니다. 실제로 별다른 최적화 없이도 0.1초 정도에 처리가 가능합니다.