목차

문제		의도한 난이도	출제자
A	연세여 사랑한다	Beginner	김태윤 ystaeyoon113
B	Prime Arrangement	Easy	김태윤 ystaeyoon113
C	싫은데요	Easy	최우현 starwh03
D	북극곰은 괄호를 찢어	Easy	이협 dlguq0107
E	건너 아는 사이	Medium	김건우 coconut99
F	비밀의 레시피	Medium	이협 dlguq0107
G	Lost Edge	Medium	이승재 pom0319
H	Yonsei Formula 1	Medium	김태윤 ystaeyoon113
I	동아리 박람회	Medium	김건우 coconut99
J	기러기 토마토 스위스 인도인 별똥별	Hard	이승재 pom0319
K	Darkest Dungeon	Hard	이국렬 lky7674
L	안아줘요	Hard	최우현 starwh03
M	Parity Constraint Maximum Flow	Challenging	이국렬 lky7674

A. 연세여 사랑한다

`string`, `implementation`
출제진 의도 – **Beginner**

- ✓ 제출 60번, 정답 49명 (정답률 81.667%)
- ✓ 처음 푼 사람: **김종민(luciaholic)**, 1분
- ✓ 출제자: ystaeyoon113

A. 연세여 사랑한다

- 문자은 점퓨터에서 char 자료형의 정수입니다. 이는 알파벳 대문자도 마찬가지입니다.
- 두 알파벳 s_i와 s_j의 거리 차이를 $|s_i - s_j|$로 구할 수 있습니다.
- 훈규의 시작 위치로부터 I까지의 거리에, I부터 시작해서 "ILOVEYONSEI"를 이동하는 거리를 더하면 됩니다.

B. Prime Arrangement

`math, combinatorics`

출제진 의도 – **Easy**

- 제출 59번, 정답 23명 (정답률 40.678%)
- 처음 푼 사람: **곽재혁 (dreami63)**, 8분
- 출제자: 김태윤 (ystaeyoon113)

B. Prime Arrangement

- 주어진 소수들은 모두 서로 다르기 때문에, 어떻게 소수를 배치해도 각 행의 가중치는 서로 다릅니다.
- 그래서 무작위로 배열에 소수를 배치한 후, P_i 조건에 맞도록 각 행의 순서를 정해준다고 생각하면 됩니다.
- 직접 숫자를 배치할 필요는 없고, 소수를 각 칸에 배치하는 $(R \times C)!$ 개의 전체 경우의 수에서, P_i 조건에 맞도록 배치하는 배치는 $(R \times C)!/R!$ 개만 존재하게 됩니다.
- 이는 $R+1$ 부터 $R \times C$ 까지 모두 곱하는 식으로 계산할 수 있습니다.
- $(A \times B)\ mod\ P = (A\ mod\ P) \times (B\ mod\ P)\ mod\ P$ 를 이용해서 곱할때마다 계속 $mod\ P$ 를 취해주면 됩니다.
- 중간 계산 값이 커질 수 있어서 64bits 이상의 정수 자료형(ex. C++의 long long, Java의 long)을 사용해야 합니다.

C. 싫은데요

`two pointer`
출제진 의도 – **Easy**

- 제출 159번, 정답 38명 (정답률 23.899%)
- 처음 푼 사람: **김종민(luciaholic)**, 8분
- 출제자: 최우현(starwh03)

C. 싫은데요



- ✓ 아주 간단하고 함정도 없는 투포인터 문제입니다.
- ✓ 현재 햄스터가 막고 있는 연속된 구멍에 대한 시작점과 끝점을 관리하면 됩니다.
- ✓ 햄스터의 부피보다 작다면 끝점을 오른쪽으로 움직이고, 크다면 시작점을 오른쪽으로 움직이면서 등장하는 최댓값을 저장해 두면 답을 얻을 수 있습니다.

D. 북극곰은 괄호를 찢어

`greedy`
출제진 의도 – **Easy**

- 제출 93번, 정답 30명 (정답률 33.333%)
- 처음 푼 사람: **박경욱(sk091204091204)**, 20분
- 출제자: 이협(dlguq0107)

D. 북극곰은 괄호를 찢어



- ✓ 단순한 그리디 문제입니다.
- ✓ O,X 를 놓을 때마다 (와) 의 한 쌍이 동시에 생겨납니다.
- ✓ 괄호를 빨리 닫는 것이 최적입니다.
- ✓ (을 +1,) 을 -1이라 두고 순서대로 더했을 때 절댓값의 최대가 정답이 됩니다.
- ✓ 만약 괄호를 모두 더한 결과가 0이 아니라면 불가능한 경우이므로 -1을 출력하시면 됩니다.

E. 건너 아는 사이

`number_theory, sieve`
출제진 의도 – **Medium**

- ✓ 제출 93번, 정답 28명 (정답률 30.108%)
- ✓ 처음 푼 사람: **한의엽(yup0927)**, 56분
- ✓ 출제자: 김건우(coconut99)

E. 건너 아는 사이

- 간선의 가중치가 주어진 그래프에서 최소 스패닝 트리를 찾는 문제입니다.
- 그러나 알려진 최소 스패닝 트리 알고리즘을 사용하기에는 N 의 제한이 큽니다.

E. 건너 아는 사이



- ✓ Prim 알고리즘의 원리를 사용합시다. 1번 정점으로만 이루어진 트리에서 시작합니다.
- ✓ 간선의 가중치 중 가장 작은 값은 2입니다. 1번과 2번을 연결하고, 2보다 큰 2의 배수들은 모두 2와 연결합니다.
- ✓ 이렇게 2의 배수들을 모두 연결하면 가중치 2로 연결할 수 있는 모든 점을 연결하게 됩니다.

E. 건너 아는 사이



- 다음으로 작은 가중치의 값은 3입니다. 마찬가지로 1번과 3번을 연결하고, 아직 연결되지 않은 3의 배수들을 모두 3과 연결합니다.
- 가중치 4인 간선을 갖는 정점들은 모두 가중치 2로 연결되었습니다.
- 이렇게 N보다 작거나 같은 모든 소수 p에 대해 1과 p를 연결하고, 연결되지 않은 p의 배수들을 모두 p에 연결하면 됩니다.
- 가중치의 합은 2부터 N까지 자연수의 "1이 아닌 가장 작은 약수"의 합입니다.
- 에라토스테네스의 체를 이용하면 $O(N \log \log N)$에 구현할 수 있습니다.

F. 비밀의 레시피

`math, ad-hoc`
출제진 의도 – **Medium**

- ✓ 제출 65번, 정답 8명 (정답률 12.308%)
- ✓ 처음 푼 사람: **김종민(luciaholic)**, 56분
- ✓ 출제자: 이협(dlguq0107)

F. 비밀의 레시피

- N 차 다항식이 있을 때 계수는 $N+1$ 개 입니다.
- 일반적으로 k 개의 미지수로 이루어진 식 k 개가 있을때, 연립방정식을 풀어 미지수를 구할 수 있습니다.
- 하지만 문제 조건에서 한번의 질문으로 모든 계수를 구할 수 있는 방법이 존재합니다.
- 계수가 10^9 미만이고 음이 아닌 정수이므로 10^9 을 식에 대입하여 나온 수의 각 9자리가 계수를 뜻하게 됩니다.
- 숫자가 최대 10^{1000} 까지 나올 수 있기 때문에 문자열 형식으로 입력을 받거나 BigInteger를 사용하여 계산하면 됩니다.

G. Lost Edge

`bfs`, `priority_queue`

출제진 의도 – **Medium**

- 제출 50번, 정답 18명 (정답률 36.000%)
- 처음 푼 사람: **정용진(playsworld16)**, 49분
- 출제자: 이승재(pom0319)

G. Lost Edge



- 맵을 탐색하면서 각 위치의 몬스터를 잡기 위한 최소 레벨을 구하며 잡는 문제입니다.
- 가장 요구레벨이 낮은 몬스터부터 잡으며 움직이기 위해 priority queue가 사용됩니다.

G. Lost Edge

- 플레이어의 시작지점으로부터 탐색을 하며 priority queue에 (최소 사냥 레벨, 좌표)를 넣습니다.
- 이때 최소 사냥 레벨이란 각종 몬스터를 잡으며 해당 지점에 도달해야 하므로 (현재 좌표의 최소 사냥 레벨)과 (새 좌표의 몬스터 레벨 + 1) 중 더 큰 값이 됩니다.
- priority queue에서 원소를 꺼낼 때는 이미 사냥한 몹이라면 무시하고, 아니라면 사냥 후 주변 칸을 탐색하면 됩니다. 만약 레벨이 낮아 사냥할 수없다면, 루프에서 탈출합니다.
- 레이드 장소의 경우 탐색 도중 나올 경우 도달이 가능한 것이기 때문에, 도달 가능함을 저장하면 됩니다.

H. Yonsei Formula 1

`dp`, `binary_search`, `math`

출제진 의도 – **Medium**

- ✓ 제출 12번, 정답 4명 (정답률 33.333%)
- ✓ 처음 푼 사람: **김종민(luciaholic)**, 127분
- ✓ 출제자: 김태윤(ystaeyoon113)

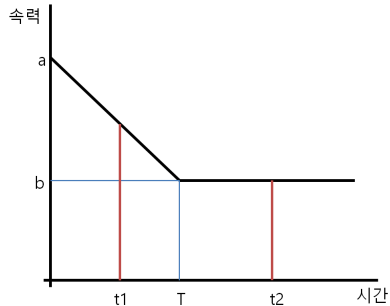
H. Yonsei Formula 1



- ✓ 풀이보다 구현이 더러운 문제입니다. ㅠㅠ
- ✓ 항상 출발 지점에서 타이어를 가는 순간 몇 바퀴를 달릴지를 정하고 달린다고 생각해 봅시다.
- ✓ 현재 상황을 (몇번째 타이어를 쓰고 있는지, 몇 바퀴가 남았는지) 로 분석할 수 있습니다.
- ✓ 제한이 작으므로 동적 계획법으로 분석할 수 있습니다.
- ✓ 현재 출발 지점에서 i 번째 타이어를 달았고, j 바퀴가 남았을 때 해당 타이어로 x 바퀴 $x = 0, 1, \cdots, j$ 바퀴를 주행할 수 있습니다.
- ✓ 해당 타이어로 x 바퀴를 주행하고 나면, 이제는 $i + 1$ 번째 타이어로 $j - x$ 바퀴가 남은 상황입니다.
- ✓ $dp(i, j) = min\{dp(i + 1, j - x) + t(i, x)\}\ (x = 0, 1, \cdots, j)$
- ✓ $t(i, x)$ 는 i 번째 타이어로 x 바퀴를 달릴 때 걸리는 시간입니다.

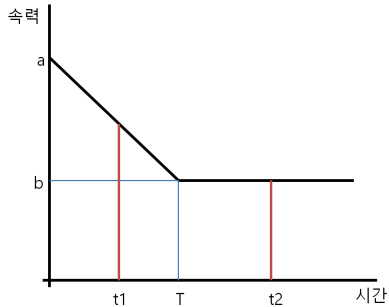
H. Yonsei Formula 1

- ✓ $t(i, x)$ 를 구하는 방법을 알아 봅시다.
- ✓ 레이싱카의 속력이 a_i 부터 b_i 까지는 감소하다가, b_i 부터는 일정하므로, 두 구간을 나누어서 분석해야 합니다.



- ✓ 먼저 T 까지 움직였을 때 거리를 D 라고 하면, D 는 $O(1)$ 에 쉽게 구할 수 있습니다.

H. Yonsei Formula 1



- ✓ $D \leq xL$ 이면, 일차함수이므로 처음으로 xL 를 넘기는 $t(i,x)$ 를 $O(1)$ 에 구할 수 있습니다.
- ✓ $D < xL$ 이면, t 에 관한 이차식을 해결해야 합니다. 속력이 항상 양수이므로, 이분탐색을 통해 처음으로 xL 를 넘기는 $t(i,x)$ 를 구할 수 있습니다.
- ✓ Time Complexity : $O(NM^2 + NM \log A) \rightarrow t(i,x)$ 를 미리 전처리 해놨을 경우



I. 동아리 박람회

`constructive`, `ad-hob`

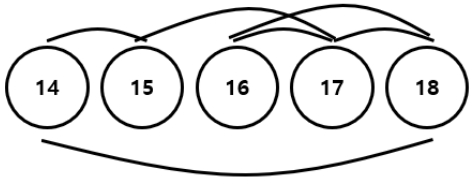
출제진 의도 – **Medium**

- ✓ 제출 6번, 정답 4명 (정답률 66.667%)
- ✓ 처음 푼 사람: **한의엽(yup0927)**, 150분
- ✓ 출제자: 김건우(coconut99)

NAVER D2 START LINK

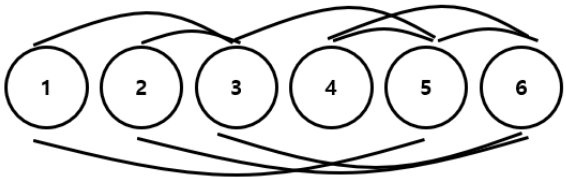
- ✓ 그래프에서 각 정점을 한 번씩만 방문하고 시작점으로 돌아오는 문제입니다.
- ✓ bitwise AND 연산값이 0이 아닌 두 정점은 모두 간선을 갖습니다.
- ✓ 한 정점을 두 번 방문할 수 없으므로, 어떤 정점으로 들어오는 간선과 나가는 간선은 달라야 합니다.
- ✓ 따라서 모든 정점은 2개 이상의 간선을 가져야 합니다.

- 조건을 만족하는 경로가 존재하지 않는 경우를 찾아보겠습니다.



- 2^k 번 정점은 $2^k+1, 2^k+2, \cdots, 2^{k+1}-1$ 번 정점과 모두 연결됩니다.
- 그러나 $1, 2, \cdots, 2^k-1$ 번 정점과는 연결되지 않습니다.
- $N=2^k, N=2^k+1$ 이라면 2^k 번 정점은 각각 0개, 1개만의 간선을 갖게 됩니다.
- 따라서 이때 조건을 만족하는 경로는 존재하지 않습니다.

- 이제 K 값에 따른 경로 존재 여부를 보겠습니다.
- 1 번 정점에서 가장 가까운 두 정점은 $3, 5$ 입니다.



- 따라서, K 가 4 보다 작다면 1 번과 5 번 정점이 연결될 수 없고, 1 번 정점은 2개보다 적은 간선을 갖게 됩니다.



- 정리하면, 경로가 존재하지 않는 경우는 $N = 2^k$ 이거나 $N = 2^k + 1$ 일 때, 또는 $K < 4$ 일 때입니다.
- 다른 경우에서 경로가 존재함은 간단히 증명할 수 있습니다.
- 이제 경로가 존재할 때 최단경로의 길이를 찾아보겠습니다.
- 가장 이상적인 방법은 방향을 바꾸지 않고 1 번부터 N 번까지 간 다음, 방향을 바꾸지 않고 돌아오는 것입니다.
- 그러나 2^k 번 정점과 연결된 정점은 모두 번호가 2^k 보다 크기 때문에 2^k 번에서 반드시 방향을 바꾸어야 합니다.

- 방향을 바꾸면 역행한 거리의 2배만큼 경로의 길이가 늘어납니다.
- 역행의 횟수의 최솟값은 N보다 작은 2의 거듭제곱수의 갯수로 정해져 있습니다.
- $2^k+1 \to 2^k \to 2^k+4$와 같은 방법으로 이동하면 역행 거리를 1로 최소화할 수 있습니다.
- 따라서 최단 거리는 $(N-1+k)\times 2$입니다. 이때 k는 $2^k < N < 2^{k+1}$를 만족합니다.
- 4로 나눈 나머지를 적절히 이용하면 한 번에 이동하는 거리가 최대 4인 경로를 만들 수 있습니다.



J. 기러기 토마토 스위스 인도인 별똥별

`greedy`, `prefix_sum`

출제진 의도 – **Hard**

- ✓ 제출 4번, 정답 0명 (정답률 0.000%)
- ✓ 처음 푼 사람: **XXX**, XXX분
- ✓ 출제자: 이승재(pom0319)

NAVER D2 START LINK

J. 기러기 토마토 스위스 인도인 별똥별



- 이차원 비트 팰린드롬을 만족하려면, 위와 같이 기존 행렬을 $(N/2) \times (M/2)$ 크기의 4개의 사분면으로 분리했을 때 대칭적으로 존재하는 비트끼리 같은 값을 가지고 있으면 됩니다.
- 단순히 $K \times K$ 크기의 정사각형을 한쪽 끝부터 움직여가며 팰린드롬 상태가 아닐 때마다 팰린드롬 상태로 만들기 위해 비트를 바꾸어야 하는 사분면을 골라 비트를 전환하는 방식으로도 답을 얻을 수 있습니다.
- 하지만 이와 같이 할 경우 시간복잡도가 $O(NMK^2)$ 으로, 시간 초과가 나오게 됩니다.

- ✓ 우선 팰린드롬의 성질을 이용해 계산이 필요한 영역을 하나의 $(N/2) \times (M/2)$ 크기의 행렬로 만들어봅시다.
- ✓ 대칭적으로 존재하는 비트를 1, 2, 3, 4 사분면에 위치한 것 순서대로 적으면, 해당 위치의 $state$ 를 4 종류로 구분할 수 있습니다.
  - 모든 사분면의 비트가 같은 경우 $(0000, 1111)$ 는 0 번
  - 1,4 사분면과 2,3 사분면 끼리 같은 경우 $(0110, 1001)$ 를 1 번
  - 1,3 사분면과 2,4 사분면 끼리 같은 경우 $(0101, 1010)$ 를 2 번
  - 1,2 사분면과 3,4 사분면 끼리 같은 경우 $(0011, 1100)$ 를 3 번

- ✓ $action\ i$를 4사분면과 i사분면의 비트를 동시에 전환하는 행동이라 정의하면 $state\ i$에서 $action\ j$를 했을 때 $state\ (i\ xor\ j)$로 상태가 전이됩니다.
- ✓ 위 규칙을 통해 하나의 행렬에서 $K \times K$ 크기의 사각형에 $action\ i$를 취하는 행동을 통해 모든 칸의 $state$를 0으로 만드는 문제로 축소할 수 있습니다.



- ✓ 1행 1열부터 해당 점이 $K \times K$ 사각형의 왼쪽 위 점이 되도록 탐색을 진행한다고 합시다. 이때 해당 점의 값은 초기값에서 해당 점을 오른쪽 아래에 두는 $K \times K$ 사각형 내의 점들에서의 연산만큼 바뀌게 됩니다.
- ✓ 각 점에서의 연산을 저장하고, 2D prefix sum을 통해 관리하면 $O(NM)$ 안에 문제를 풀 수 있습니다.

K. Darkest Dungeon

`tree`, `graph_traversal`

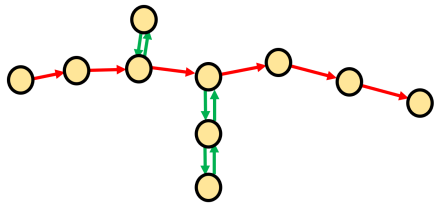
출제진 의도 – **Hard**

- ✓ 제출 8번, 정답 1명 (정답률 12.500%)
- ✓ 처음 푼 사람: **곽재혁 (dreami63)**, 277분
- ✓ 출제자: 이국렬 (lky7674)

K. Darkest Dungeon

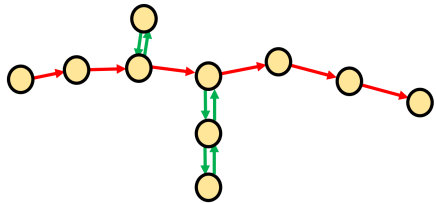
- ✓ 트리 상에서 최대 K 번을 움직여서 최대한 많은 서로 다른 정점을 방문하려고 합니다.
- ✓ K 가 트리의 지름보다 작거나 같다면 그냥 트리의 지름 위에서 움직이면 됩니다.
- ✓ 이제 K 가 트리의 지름보다 큰 경우는 조금 더 생각을 해야 합니다.

K. Darkest Dungeon



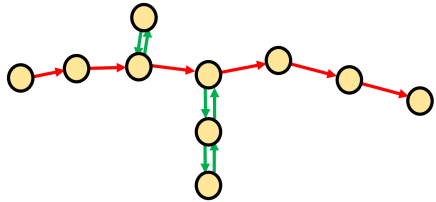
- ✓ K 가 트리의 지름보다 큰 경우는 다음과 같이 움직이면 됩니다.
  - 우선 트리의 지름의 정점을 모두 방문하는 경로를 하나 구합니다.
  - 총 이동 횟수가 K 를 넘지 않을 때까지 다음 과정을 반복합니다. 경로와 이웃한 방문하지 않은 정점이 있는 경우, 해당 정점을 경로 중간에 추가합니다. 경로에 정점을 하나씩 추가할 때마다 이동 횟수가 2 씩 증가합니다.

K. Darkest Dungeon



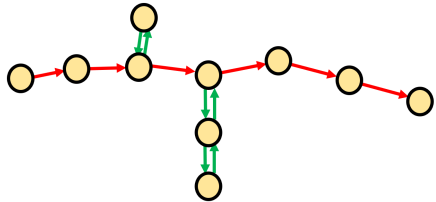
- 해당 방법을 통해서 구한 경로가 서로 다른 정점을 가장 많이 방문하는 최적의 경로인 이유를 간단히 언급하고자 합니다.
  - 한 간선을 3번 이상을 사용하지 않는 최적의 경로는 반드시 존재합니다. 이는 주어진 그래프가 트리이기 때문입니다.
  - 이에 따라 K 가 트리의 지름보다 큰 경우에는 리프에서 출발해서 다른 리프로 도착하는 최적의 경로는 반드시 존재합니다.

K. Darkest Dungeon



- ✓ 해당 방법을 통해서 구한 경로가 서로 다른 정점을 가장 많이 방문하는 최적의 경로인 이유를 간단히 언급하면 다음과 같습니다.
  - 따라서 한 간선을 최대 2번만 방문하고, 출발 지점과 도착 지점이 트리의 리프 노드인 경로만 고려할 것입니다.
  - 이러한 경로 상에서 단 한 번만 사용한 간선들의 집합은 단순 경로의 형태를 띕니다. (그림의 빨간색 간선) 이 단순 경로의 길이가 길수록 더 많은 정점들을 방문함을 알 수 있습니다.

K. Darkest Dungeon



- ✓ subtask 1은 트리의 지름은 $O(N^2)$에 찾으면, subtask 2는 트리의 지름을 $O(N)$ 또는 $O(N \log N)$에 찾으면 점수를 얻을 수 있습니다.
- ✓ 구현이 좀 많이 귀찮습니다. 화이팅!

L. 안아줘요

`dp, segment_tree`
출제진 의도 – **Hard**

- ✓ 제출 24번, 정답 1명 (정답률 4.167%)
- ✓ 처음 푼 사람: **박경욱(sk091204091204)**, 297분
- ✓ 출제자: 최우현(starwh03)

- ✓ 문제에서 날다람쥐가 i 번 휴식점에서 j 번 휴식점으로 이동하기 위한 조건은 $Y_i > Y_i$, $|X_i - X_j| \geq |Y_i - Y_j|$ 두 가지이고, 앞 조건에 따라 뒤 조건은 $|X_i - X_j| \geq Y_i - Y_j$ 가 됩니다.
- ✓ 여기서 $|X_i - X_j| \geq Y_i - Y_j$ 는 $X_i - X_j \geq -Y_j$ or $-(X_i - X_i) \geq Y_i - Y_j$ 로 표현할 수 있고, 이는 다시 $X_i - Y_i \geq X_j - Y_j$ or $X_i + Y_i \leq X_j + Y_j$ 로 바꿀 수 있습니다.
- ✓ 이 두가지를 관리하는 세그먼트 트리를 하나씩 만들어 관리하는 방식으로 전체 문제를 해결할 수 있다.



- ✓ $Y_i > Y_j$ 조건을 따르기 위해 Y 좌표가 작은 점부터 확인한다.
- ✓ 두 세그먼트 트리에 대해서 범위 최댓값 쿼리를 이용하면 $O(\log N)$ 시간에 현재 점에서 시작할 경우의 답을 얻을 수 있다.
- ✓ 이후 세그먼트 트리에 현재 점을 업데이트한다. 이때 걸리는 시간은 $O(\log N)$ 이다.
- ✓ 이 과정을 모든 점에 대해 반복할 경우 $O(1)$ 시간에 임의의 점에서 시작할 때의 답을 구할 수 있다.
- ✓ 따라서 $O(N \log N + Q)$ 시간에 문제를 해결할 수 있다.

M. Parity Constraint Maximum Flow

`flow, circulation`

출제진 의도 – **Challenge**

- ✓ 제출 0번, 정답 0명 (정답률 00.000%)
- ✓ 처음 푼 사람: **XX**, XX분
- ✓ 출제자: 이국렬 (lky7674)



- ✓ 연세대학교 안형찬 교수님 알고리즘 분석 수업 때 배우는 circulation과 관련된 문제를 하나 출제했습니다.
- ✓ circulation 관련 문제들이 최소 다이아 티어인 만큼 상당히 어려운 문제입니다.
- ✓ 이번 풀이 슬라이드에서는 circulation에 대한 기본적인 개념을 알고 있다고 가정하겠습니다.



- ✓ subtask 1은 간단합니다.
- ✓ 모든 간선의 parity constraint는 짝수입니다. 따라서 capacity를 절반으로 나누어서 maximum flow를 찾으면 됩니다.
- ✓ 제한이 꽤 큽니다. Edmond-Karp Algorithm은 시간 초과가 발생하기에 Dinic Algorithm이나 더 효율적인 알고리즘을 통해서 해결하셔야 합니다.



- ✓ subtask 2부터는 좀 많이 복잡합니다. 가장 거슬리는 부분은 홀수 constraint 간선들입니다.
- ✓ 따라서 홀수 constraint 간선을 제거하는 방향으로 문제를 해결하도록 하겠습니다.
- ✓ 홀수 constraint 간선들에 미리 flow를 1씩 흘려주면, 이후 flow를 2씩 흘려줄 수 있습니다.
  이런 방식으로 maximum flow를 찾을 것입니다.

K. Darkest Dungeon



- ✓ 그러나 홀수 constraint 간선들에 미리 flow를 1씩 흘려주면, 각 정점들의 demand가 0이 아닐 수 있습니다.
- ✓ source와 sink 정점이 아닌 demand가 홀수인 정점이 있다면, flow를 2 단위로 흘리면서 demand에 맞는 flow를 만드는 것은 불가능하오니 -1을 출력하면 됩니다.
- ✓ sink에서 source로 capacity가 무한대인 간선을 만들면 circulation 형태로 바꿀 수 있습니다. 이후 변화된 demand를 만족하는 circulation이 존재하지 않는다면 -1을 출력하면 됩니다.
- ✓ 주의할 점은 source와 sink의 demand가 홀수가 될 수 있는데, 이때는 maximum flow의 크기가 홀수인 경우입니다. 불가능한 경우가 아니므로 sink에서 source로 가는 간선의 constraint를 홀수로 설정한 뒤에 circulation을 찾으면 됩니다.

K. Darkest Dungeon



- ✓ lr-flow 찾는 것과 마찬가지로 demand에 맞는 circulation을 찾은 뒤에 그냥 최대한 flow를 흘려주시면 됩니다.
- ✓ 이렇게 maximum flow를 찾으면, Dinic Algorithm을 사용하면 시간 복잡도 $O(N^2M)$에, FIFO를 이용한 Push-relabel Algorithm을 사용하면 시간 복잡도 $O(N^3)$에 문제를 해결할 수 있습니다.
- ✓ 다른 방법으로 maximum flow를 찾을 수 있는데, sink에서 source로의 간선의 upper bound에 대한 이분 탐색을 통해서 풀 수 있습니다. 다만 Dinic Algorithm을 사용해도 시간 복잡도가 $O(N^2M\log(MAX_FLOW))$이 되므로 subtask 1과 3에서는 시간초과가 발생합니다.
- ✓ subtask 2의 제한은 상당히 작기에 Edmond-Karp로도 통과함이 보장되어 있습니다. 다만 Edmond-Karp 알고리즘을 사용한다면 subtask 1과 3에서 시간 초과를 받게 됩니다.