ICPC Sinchon
Winter Camp 2021
Final Exam

Official Solutions

문제			의도한 난이도	출제자	
A	초급 A	영단어 암기는 괴로워	Silver	김도은숙명여대	whaeun25
B	초급 B 중급 A	그렇고 그런 사이	Silver	정기웅서강대	QuqqU
C	초급 C	겹치는 건 싫어	Silver	김도현홍익대	swoon
D	초급 D	숫자 할리갈리 게임	Silver	김도은숙명여대	whaeun25
E	중급 B	걷는 건 귀찮아	Gold	김주현서강대	woonikim
F	초급 E	트리의 기둥과 가지	Gold	백성익홍익대	906bc
G	초급 F	메이플스토리	Gold	이상원서강대	gumgood
H	초급 G	얼음 미로	Gold	백성익홍익대	906bc

문제			의도한 난이도	출제자	
I	초급 H	Degree Bounded-Minimum Spanning Tree	Gold	이국렬연세대	lky7674
J	중급 C	중간	Gold	이국렬연세대	lky7674
K	중급 D	우주 정거장	Platinum	이국렬연세대	lky7674
L	중급 E	혹 떼러 갔다 혹 붙여 온다	Platinum	정연두홍익대	Green55
M	중급 F	약수 의식	Platinum	정연두홍익대	Green55
N	중급 G	마스크펑크 2077	Platinum	김주현서강대	woonikim
O	중급 H	카드모래성	Platinum	김주현서강대	woonikim
P	중급 I	Parity Constraint-Perfect Matching	Diamond	이국렬연세대	lky7674

A. 영단어 암기는 괴로워

출제자 : whaeun25(김도은, Algos)

- 입력받을 시 해당하는 단어가 외울 단어에 해당하는 m보다 긴 것임을 확인함과 동시에 각 단어의 빈도수를 저장합니다.
- 우선순위 사항을 순서대로 반영될 수 있도록 하여 정렬합니다.
- $O(N^2)$으로 정렬할시 시간초과가 납니다.

B.그렇고 그런 사이

출제자 : QuqqU(정기웅, Sogang ICPC Team)

- 시간이 아주 널널합니다.(시간이 4.242인 이유는 사4이2 라서...)
- $O(N^2)$, $O(N\log N)$, $O(N)$... 등 적당한 시간복잡도로 어떻게든 풀면 됩니다!
- 여기서는 $O(N^2)$ 풀이법과 $O(N)$ 풀이법을 소개하겠습니다.

- 다음과 같은 순열을 생각해 봅시다.

$$\{ 1, 2, 3, 4, 5 \}$$

- (아무거나) 그렇고 그런 사이가 아닌, 인접한 두 원소를 바꿔봅니다.

$$\{ 1, 3, 2, 4, 5 \}$$

- 그러면 정확히 **그렇고 그런 사이**가 하나 증가합니다!
- 이는 두 원소를 기준으로,

  왼쪽 및 오른쪽 원소들과의 대소관계가 바뀌지 않기 때문입니다.

- **그렇고 그런 사이**가 아닌 인접한 두 원소를 바꾸면,

  정확히 한 개의 **그렇고 그런 사이**가 증가하는 것을 관찰할 수 있습니다.

- 이것을 2중 for문을 한 번 돌리면서,

  K번 swap한 순열을 출력하면 $O(N^2)$로 AC를 받을 수 있습니다.

- 다음의 성질을 이용하면 *O(N)*만에 풀 수 있습니다.
- 가장 큰 수를 가장 왼쪽 칸에 넣으면,

  항상 모든 나머지 수와 **그렇고 그런 사이**가 됩니다.

  [5, 3, 2, 1, 4]

- 가장 작은 수를 가장 왼쪽 칸에 넣으면,

  항상 모든 나머지 수와 **그렇고 그런 사이**가 아닙니다.

  [1, 2, 5, 3, 4]

예제)

- 현재 [1, 2, 3, 4, 5]를 가지고 있을 때,

  이 수를 적절히 배치해서 **그렇고 그런 사이**를 6개 만들어 봅시다.

- 먼저 가지고 있는 수 중 가장 큰 수를 내려 놓습니다.

  그러면, 뒤에 순서가 어떻든 **그렇고 그런 사이**가 4개 만들어 집니다.

  앞으로 **그렇고 그런 사이**를 2개만 더 만들면 됩니다.

[5, *, *, *, *]

- 현재 가지고 있는 수 : [1, 2, 3, 4] / 더 만들어야 하는 **그렇고 그런 사이** : 2개

- 내가 가지고 있는 수 중, 가장 큰 수는 4입니다.

  4를 넣게 되면 **그렇고 그런 사이**가 3개가 더 만들어 집니다.

  [5, 4, *, *, *]

- 가장 큰 수를 놓을 수 없을 땐, 가지고 있는 수 중 가장 작은 수를 놓습니다.

  이렇게 수를 놓으면, 절대 **그렇고 그런 사이**가 더 만들어지지 않습니다.

  [5, 1, *, *, *]

- 현재 가지고 있는 수 : [2, 3, 4] / 더 만들어야 하는 **그렇고 그런 사이** : 2개

- 현재 가지고 있는 수 중 가장 큰 수는 4입니다.

  4를 놓게 되면, **그렇고 그런 사이**가 2개 더 만들어집니다. 앞으로 0개를 더 만들면 됩니다.

  [5, 1, 4, *, *]

- *if 가장 큰 수 놓기 가능? -> 가장 큰 수 놓는다*

  else -> 가장 작은 수 놓는다

  이것을 계속 반복하면 *O(N)*에 정답 순열을 구축할 수 있습니다.

  [5, 1, 4, 2, 3]

이게 왜 될까요?

- 이 문제와 동치인 문제를 생각해 봅시다.
- 1~(N-1) 까지 자연수의 합으로 K$_{(K \le N(N-1)/2)}$ 를 항상 만들 수 있을까요?
- 예를 들어 N = 5, K = 6라면,

  6 = 4 + 2 로 만들 수 있습니다.
- 항상 가능하다는 것은, 수학적 귀납법으로 어렵지 않게 증명할 수 있습니다.(여백이 부족해 생략합니다)

- 랜덤을 적절하게 사용하면 높은 확률으로 AC를 받을 수 있습니다.(Gratus99님 감사합니다)

C.겹치는 건 싫어

출제자 : swoon(김도현, HI-ARC)

- 0으로 초기화된 크기가 10만인 배열을 만듭니다.
- 투 포인터를 사용하여 right를 늘려가며 현재 나온 수를 카운팅해줍니다.
- 방금 추가된 수의 카운팅이 K개 초과인지 검사합니다.
- K개 초과라면 left를 늘리고 빠진 수를 카운팅에서 빼줍니다.
- K개 초과가 아니라면 아니라면 길이를 +1해준다.
- 가장 길었던 길이가 답입니다.

D.숫자 할리갈리 게임

출제자 : whaeun25(김도은, Algos)

- 입력으로부터 도도의 덱과 수연의 덱을 만듭니다.
- 한 차례 진행할 때마다 덱에 있는 카드를 자신의 그라운드에 내려놓고 종을 치는 상황인지 혹은 게임이 종료되는지 확인합니다.
- 종을 치게 될 경우 그라운드에 있는 카드 더미를 자신의 덱으로 가져옵니다.
- 게임을 M번 진행한 후 덱에 더 많은 카드를 가진 사람을 출력합니다.

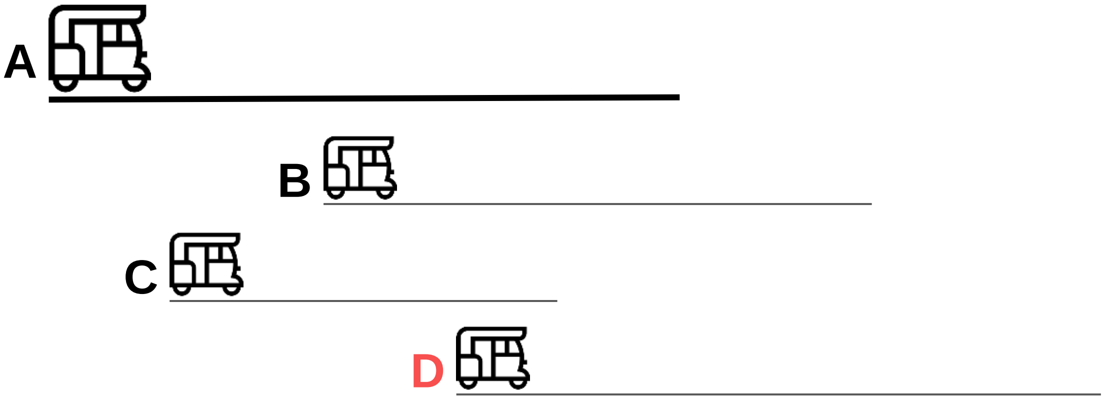
- 게임이 종료되는 조건과 종을 치는 조건에 유의하며 구현하면 됩니다.
- 게임이 종료되는 조건:
  - 두 플레이어의 덱 중 하나가 비게 되는 경우
  - m번 게임을 진행한 경우
- 각 플레이어가 종을 치는 조건:
  - 수연이가 종을 치는 조건: 비어있는 그라운드가 없고 그라운드 맨 위 카드의 합이 5가되는 순간
  - 도도가 종을 치는 조건: 그라운드 맨 위 카드로 숫자 5가 나오는 순간

E. 걷는 건 귀찮아

출제자 : woonikim(김주현, Sogang ICPC Team)

- 일직선 상의 인력거들을 최소한 환승해서 목적지에 도달해봅시다.
  - DP Method : $O(N^2)$를 Segment Tree와 같은 자료 구조를 이용하여 O(NlogN)으로 최적화?
  - Greedy Method : 현솔이는 매 환승마다 멀리 도달할 수 있는 인력거로 갈아타는 것이 최적입니다.
- 매 환승마다 최대한 멀리 갈 수 있는 인력거로 갈아타봅시다.
  - 멀리 갈수록 더 많은 인력거들을 확인할 수 있고, 더 멀리 갈 수 있는 가능성이 높아집니다.

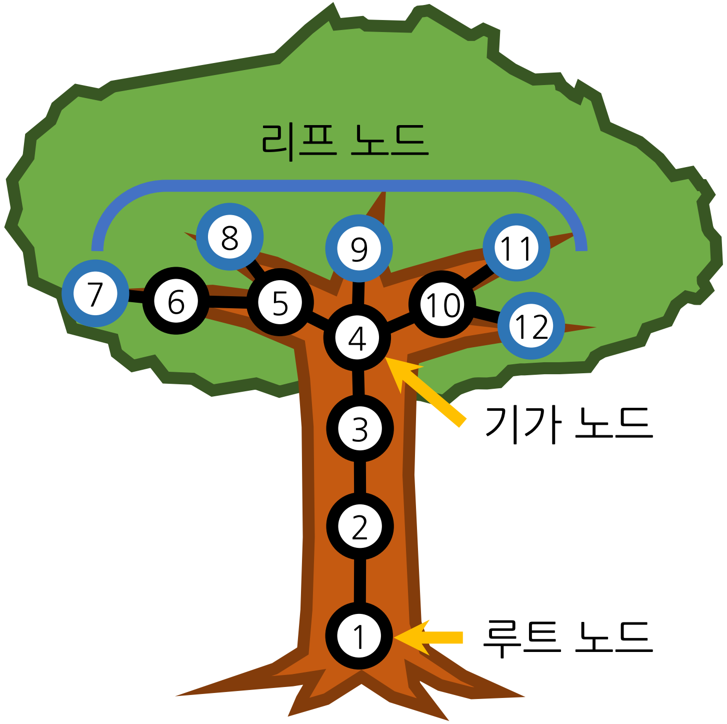
- 현재 현솔이가 타고 있는 인력거 A의 이동 범위가 다음과 같다고 가정해봅시다.
- 현솔이가 갈아탈 수 있는 인력거 B, C, D 중 어느 것으로 갈아타는 것이 좋을까요?
    - 인력거 C는 갈아탈 필요가 없습니다.
    - 인력거 B와 D 중 더 멀리 갈 수 있는 인력거 D로 갈아타는 것이 최적입니다.



- 갈아탈 수 있는 인력거 중, 가장 멀리 갈 수 있는 인력거로 갈아타면 됩니다. (M에 도달하지 못하면 -1)
- 이미 정렬되어 있기 때문에, O(N) 또는 O(M) sweeping으로 해결할 수 있습니다.

F.트리의 기둥과 가지

출제자 : 906bc(백성익, HI-ARC)



- 입력으로부터 트리를 만들고,

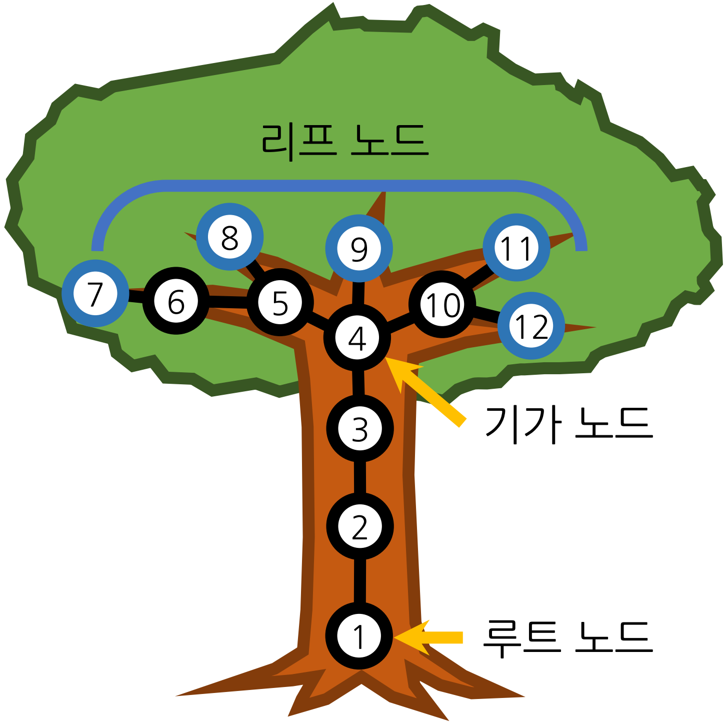
  루트 노드에서부터 기가 노드까지,

  기가 노드에서부터 리프 노드까지

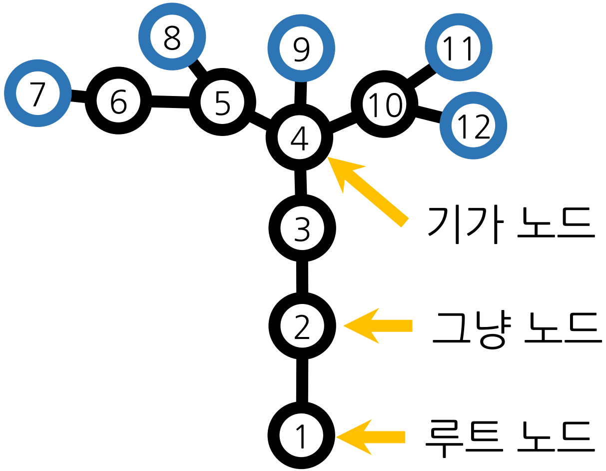
  순회를 하면 되는 문제입니다.

- 기가 노드를 판별하는 것이

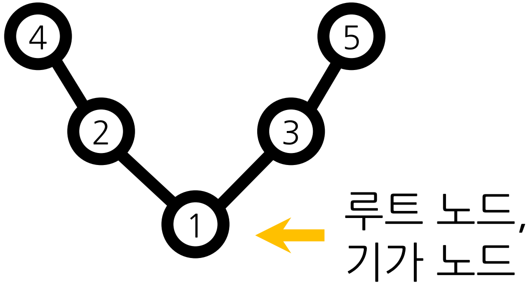
  핵심입니다.



- 순회 이후 최초로 진출차수가 2 이상인 노드가 기가 노드임이 명료합니다.
- 하지만 입력이 Undirected graph로 주어지므로 차수만 이용할 수 있습니다.



- 기가 노드(4)의 차수는 4입니다.
- 그냥 노드(2)의 차수는 2입니다.
- 순회 이후 최초로 차수가 3 이상인 노드가 기가 노드인 것처럼 보입니다.
- 하지만 다음 슬라이드와 같은 반례가 있습니다.



- 1번 노드는 기가 노드이지만,

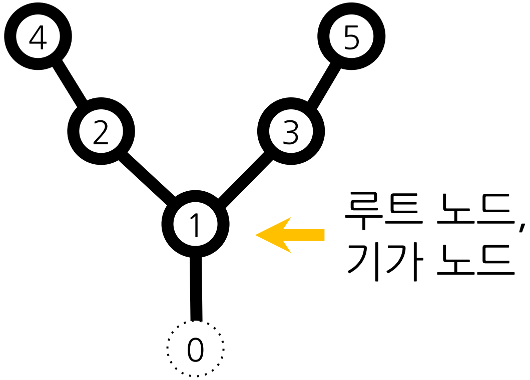
  차수 ≥ 3 만으로 기가 노드를 판별한다

  면

  1번 노드의 차수가 2이므로

  왼쪽 트리에서는 기가 노드를

  찾지 못합니다.



- 방법 1.

  이미 방문한 0번 가상 노드를 만들고,

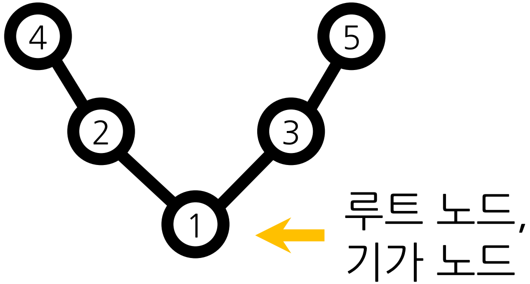
  이를 루트 노드에 연결합니다.

- 왼쪽 트리의 루트 노드의

  차수는 3이므로

  차수 ≥ 3 만으로 기가 노드를 판별할 수

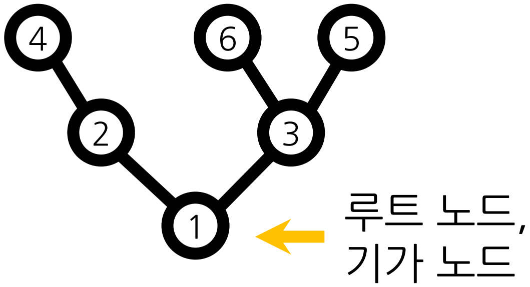
  있습니다.

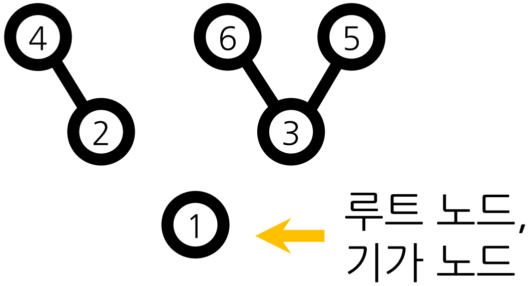


- 방법 2.

  차수 ≥ 3 - (현재 노드 == 루트 노드) 로

  기가 노드를 판별합니다.





- 방법 3.

  차수 ≥ 2 로 기가 노드를 판별하되,

  방문한 노드에 연결된 간선들을 제거합니다.

- 이외에도 기가 노드를 판별하는 많은 방법이 있습니다.
- 시간복잡도 O(N) 트리 순회를 하면서,

  기가 노드 전까지는 기둥의 길이를 갱신하고,

  기가 노드 후로는 가지의 길이를 갱신하면 됩니다.

G.메이플스토리

출제자 : gumgood(이상원, Sogang ICPC Team)

- **입장에 필요한 최소 경험치**가 0인 사냥터 중 하나에서 사냥을 시작합니다.
- 현재 경험치보다 **입장에 필요한 최소 경험치**가 낮거나 같은 사냥터로 이동할 수 있습니다.
- 사냥을 시작하고 매 분마다 다른 사냥터로 이동할지 말지 결정합니다.
- T분 동안 얻을 수 있는 최대 경험치를 구해야 합니다.

- 이 문제는 Dynamic Programming으로 해결할 수 있습니다.
  - dp(i, j) := i초간 사냥하고 j번 사냥터에서 사냥이 끝날 때 얻을 수 있는 최대 경험치
- dp(i, j) $\geq c_k$를 만족하는 사냥터에 대해 다음과 같이 전이할 수 있습니다.
  - k번 사냥터로 옮기기로 한 경우, dp(i + t[j][k], k)를 업데이트 할 수 있습니다.
  - 사냥터를 옮기지 않고 계속 사냥하는 경우, dp(i + 1, j)를 업데이트할 수 있습니다.

- 문제의 정답은 dp(T, 1), dp(T, 2), …, dp(T, N) 중 가장 큰 값이 됩니다.
- 시간 복잡도 $O(TN^2)$에 해결할 수 있습니다.
  - DP 상태가 O(TN)개 존재합니다.
  - 각 상태마다 O(N)개의 상태로 전이합니다.

H. 얼음 미로

출제자 : 906bc(백성익, HI-ARC)

- 격자 구조 위에서 최단 경로를 구하는 문제입니다.
- 정해는 다익스트라입니다.

- 크게 두 가지 접근법이 있습니다.
- 1. 사전 연결) 모든 바위의 상하좌우로 다른 바위에 부딪힐 때까지 선형탐색

  바위, 출구에 도달하면 출발 정점과 도착 정점을 연결합니다.

  간선을 모두 연결하고 나면 시작 정점에서 최단 경로 탐색을 합니다.
- 2. 그때그때 연결) 시작 정점에서 먼저 최단 경로 탐색을 합니다.

  바위에 도달하면 다른 바위나 출구로 이동할 수 있는지 탐색합니다.

- 정점의 수는 최대 N^2, 한쪽으로 부딪힐 때까지 이동하는 것은 $O(N)$입니다.
- $O((N^2 \log N^2) \times N)$에 해결할 수 있습니다.

I. Degree Bounded Minimum Spanning Tree

출제자 : lky7674(이국렬, 모르고리즘)

- NP-Complete 문제입니다. 즉, 알려진 다항 시간 알고리즘이 존재하지 않습니다.
- 따라서, 정해는 Brute-Force 알고리즘입니다.
- 다만, $O(M \times 2^M)$ 알고리즘은 시간 초과가 발생합니다.

- 주어진 M개의 간선들 중 (N-1) 개를 선택합니다.
- 그래프 탐색(DFS, BFS)를 통해서 선택한 간선이 트리를 구성할 수 있는지 확인할 수 있습니다.
- 총 시간 복잡도는 $O(N \times {}_{M}C_{N-1})$입니다.
- 최대 계산량은 $10 \times {}_{27}C_{9} = 46,868,250$로 시간 내에 풀 수 있습니다.

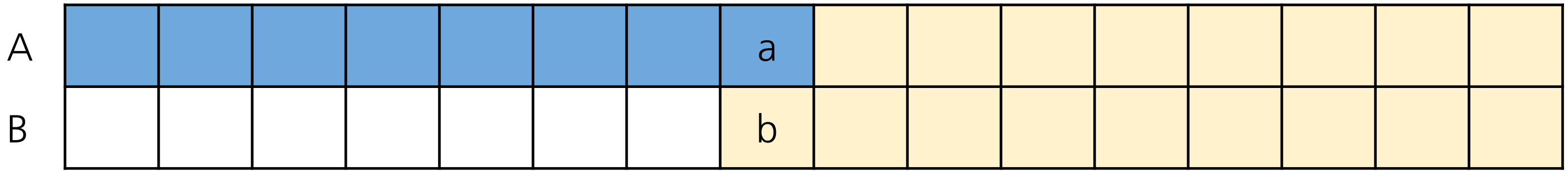
- Union-Find로도 트리인지 확인할 수 있습니다.
- 총 시간 복잡도는 $O(N \log N \times {}_{M}C_{N-1})$이지만, N의 크기가 작아서 그래프 탐색 방법에 비해서 시간이 빠르게 나옵니다.

J.중간

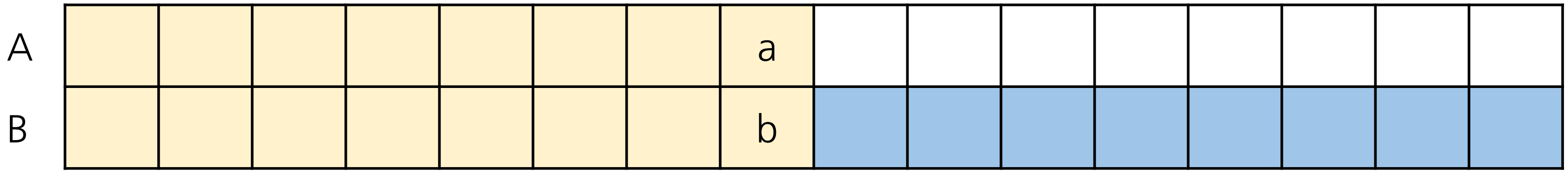
출제자 : lky7674(이국렬, 모르고리즘)

A								a								
B								b								

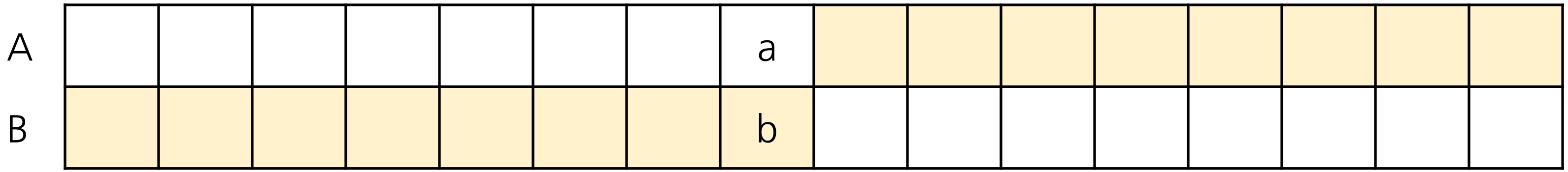
- 수열의 길이 : N
- a = A[N/2], b = B[N/2].
- a < b라고 가정합니다.



- A[1] ~ A[N/2] 이상인 수가 최소 (N+1)개가 됩니다.



- B[N/2 + 1] ~ B[N] 이하인 수가 최소 N개가 됩니다.



- A[N/2 + 1] ~ A[N], B[1] ~ B[N/2]에서 중간값을 재귀적으로 찾으면 됩니다.
- 호출 횟수 : $2 \log N + 2 \leq 40$.

- 주의점
  - flush() 이전에 개행을 출력 안 하시면 **시간초과**가 나옵니다.
  - 다른 인터랙티브 문제 푸실 때 참고 부탁드립니다.

K.우주 정거장

출제자 : lky7674(이국렬, 모르고리즘)

- 개구리 점프(BOJ 17619)를 2차원으로 확장 시킨 문제입니다.
- 마찬가지로 Union-Find로 해결할 수 있습니다.

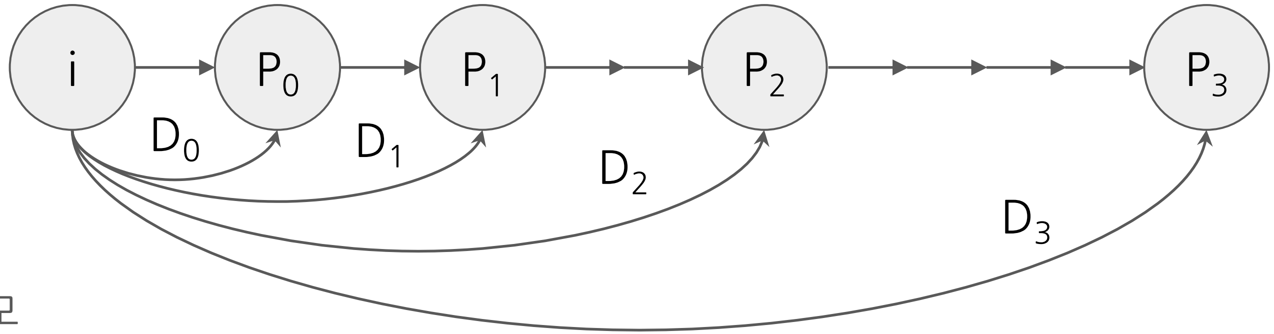
- 각 끝점들을 x좌표 순으로 정렬해서 겹치는 선분끼리 Union을 해줍니다.
- 마찬가지로 y좌표 순으로 정렬해서 Union을 해줍니다.
- 각 쿼리별로 같은 집합에 속해있는지 확인하면 됩니다.
- 시간 복잡도 : O(N log N + Q log N)

L. 훅 떼러 갔다 훅 붙여 온다

출제자 : Green55 (정연두, HI-ARC)

- "특이한 방법"으로 ad-hoc을 하기 때문에, 온라인 쿼리만 가능합니다.
  - 온라인 쿼리 : 쿼리가 나오는 즉시 답을 출력한다.
  - 오프라인 쿼리 : 쿼리 목록을 다 입력 받고, 최종 상태의 트리를 엿본다.
- 하지만, 일단은 최종 상태의 트리를 알고 있다고 가정해봅시다.
- query를 어떻게 처리할까요?

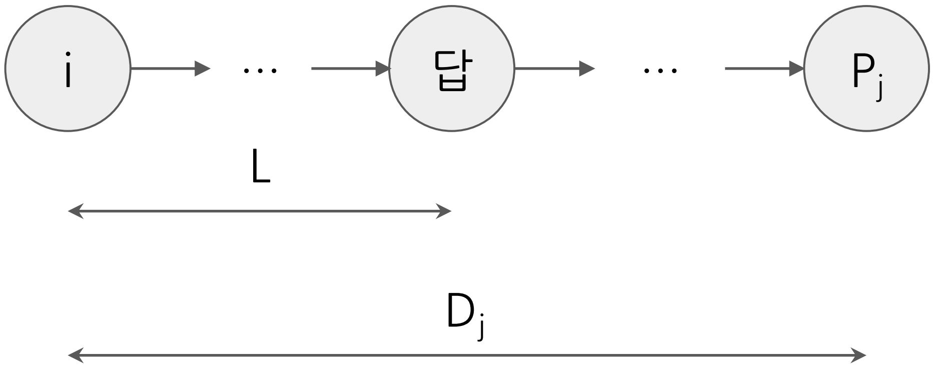
- 스파스 테이블
  - P[i][j] = i번 노드의 2^j 번째 부모
  - D[i][j] = i번 노드에서 2^j 번째 부모까지의 거리의 합



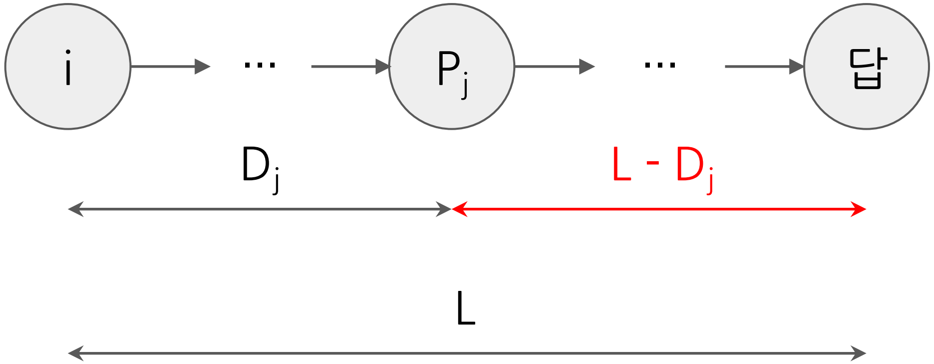
P_j : 2^j 번째 부모

D_j : 2^j 번째 부모까지의 거리

- query : i번 노드에서 거리 L만큼 올라갔을 때 나오는 노드는?
- P_{17}, P_{16}, ⋯, P_0로 올라가도 되는지 판별합니다. (Binary Lifting)
  - $D_j > L$: P_j 로 올라가지 않습니다.



- $D_j \le L$: P_j 로 올라가고, 앞으로 $L - D_j$ 만큼 더 올라가야 합니다.
- 즉, $i := P_j$ / $L := L - D_j$



- 시간 복잡도 : O(logN)

- 속도는 느리지만 좀 더 생각하기는 쉬울수도 있는 방법
  - 어떤 노드의 k번째 부모까지의 거리를 구하는 것은 전형적인 문제
  - “k번째 부모까지의 거리가 L 이상인가?”

    k로 이분탐색을 진행 -> 질문을 만족하는 최소 k를 구하면 됩니다.
  - 시간 복잡도 : $O(\log^2 N)$

- ad-hoc : i의 부모(P[i][0]) 와 부모까지의 거리 (D[i][0])가 주어집니다.
- P[i][1], P[i][2], … 와 D[i][1], D[i][2], … 를 만들기 위해서 필요한 값은
  - P[i][j] = P[P[i][j-1]][j-1] / D[i][j] = D[i][j-1] + D[P[i][j-1]][j-1]
- 매 ad-hoc 마다, P[i]와 D[i]를 “즉석에서” 만듭니다.
- i의 조상들의 P[]와 D[]는 이미 완성되어 있으므로 문제 없습니다.
- 시간 복잡도 : O(logN)

M. 약수 의식

출제자 : Green55 (정연두, HI-ARC)

문제 요약 :

- (N-1)장의 카드를 뒤집어서 나온 수를 순서대로 $X_1, X_2, \cdots, X_{N-1}$ 이라고 합시다.
- 남은 한 장의 카드에 적힌 수가 M이면,

  X = "$X_1X_2\cdots X_{N-1}$" 에 대해 X%M = 0 일 확률은?

  ("$X_1X_2\cdots X_{N-1}$"는 N-1개의 숫자를 순서대로 읽어 만든 정수)

- 마지막에 남는 카드 M을, 그냥 처음에 정해버립시다.
  - 그냥 마지막까지 그 카드를 뽑지 않고 아껴둔다고 생각하면 됩니다.
- 일단 M을 고정시켰다고 해봅시다.
- 남은 (N-1)장의 카드를 뒤집어, M이 X의 약수가 될 확률은 어떻게 구할까요?

〈비트마스크 DP〉

- **상태 정의**
  - dp[①][②] = “현재 상태에서, 약수 의식이 성공할 확률은?”
- **각 상태에 대해 관리하는 정보**

① (N-1)장의 카드를 각각 뒤집었는지의 여부 (비트필드로 관리)

② 현재까지 k장의 카드를 뽑았다면, (“$X_1X_2\cdots X_k$” % M)의 값

(k = ①에서 켜져있는 비트의 수)

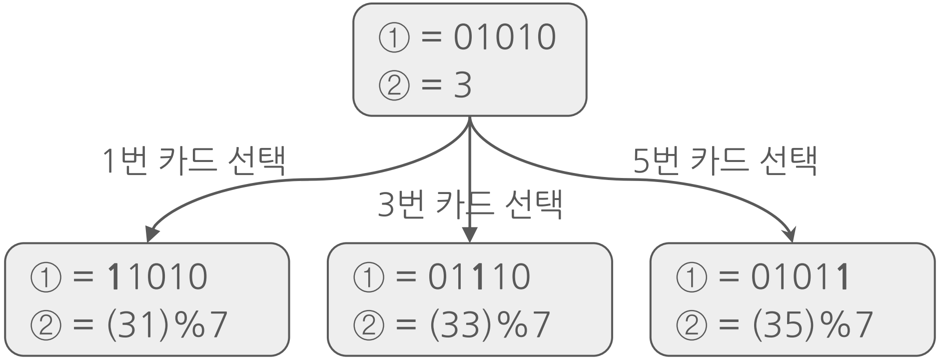
- **기저 사례**
  - (N-1)장의 카드를 모두 뽑았으면 (①의 모든 비트가 1이면) 약수 의식 끝!
    - (② = X)가 M으로 나누어 떨어지면 성공 -> 1.0
    - 그렇지 않으면 실패 -> 0.0

상태 전이

- 뒤집지 않은 카드 중, 하나를 골라 뒤집습니다. 뒤집은 후의 상태는?
- “새로운 ①” = “기존 ①”에서 고른 카드에 해당하는 비트를 켭니다.
- “새로운 ②”
  - “$X_1X_2\cdots X_k$” % M -> “$X_1X_2\cdots X_kX_{k+1}$” % M
  - “새로운 ②” = (“기존 ②” × 10 + (새로 고른 카드의 값)) % M
- 따라서, 현재 상태에서 의식이 성공할 확률은

  (각 카드를 뒤집었을 때 성공 확률의 합) / (남은 카드의 개수)

뽑을 N-1장의 카드 : [1, 2, 3, 4, 5]
M : 7



- **예시**
  - 아직 뒤집지 않은 카드가 1번, 3번, 5번이고
  - 각 카드를 뒤집었을 때 의식이 성공할 확률이 0.2, 0.4, 0.6면
  - 모든 카드를 뒤집을 확률은 동일하므로, 현재 상태에서 의식이 성공할 확률은

    (0.2 + 0.4 + 0.6) / 3

- **시간복잡도**
  - "마지막 카드"를 1~9 중 어떤 것으로 할지 정하고 -> $O(M)$
  - DP의 상태의 수 -> $O(2^{N-1} \times M)$
  - 각 상태에서, 전이 가능한 상태의 수 -> $O(N)$
  - $O(N \times M^2 \times 2^{N-1})$
    - $N = 16, M = 9$ -> 42,467,328
    - $O(N^2 \times M \times 2^N)$ 정도로 짜도 충분히 시간 내에 작동합니다.

N. 마스크펑크 2077

출제자 : woonikim (김주현, Sogang ICPC Team)

- 각 UPDATE 쿼리마다, 집집마다의 이동 거리에 변동이 생깁니다.
- 각 CALL 쿼리마다, 해당 위치의 집에서 얻을 수 있는 가장 싼 마스크의 가격을 출력해야 합니다.

- UPDATE x t : x번째 집과 x+1번째 집 사이의 이동 시간을 t분으로 갱신
  - 단순히 갱신하는 것이기 때문에, 그렇게 어렵지 않습니다.
- CALL x m : x번째 집이 m분 이내에 얻을 수 있는 가장 싼 마스크의 가격을 출력
  - D(i, j) : i번째 집과 j번째 집 사이의 이동 시간 ($i \le j$)
  - L : $D(x, l) \le m$ 을 만족하는 l 중 가장 작은 값
  - R : $D(x, r) \le m$을 만족하는 r 중 가장 큰 값
  - [L, R]에 속하는 집들의 마스크 중 가장 싼 마스크의 가격을 구하면 됩니다.

- $D(i, j) = D(i, i+1) + D(i+1, i+2) + \cdots + D(j-2, j-1) + D(j-1, j)$
  - UPDATE 쿼리마다 $D(x, x+1)$의 값이 변하기 때문에, 전처리를 할 수 없습니다.
  - $D(i, j)$는 구간합 Segment Tree를 이용하여 구할 수 있습니다.
- x번째 집에 대해서, L과 R을 구하는 방법 ($D(x,0) = D(x, N+1) = INF$라고 가정)
  - L : $\cdots > D(x, L-1) > m \geq D(x, L) > D(x, L+1) > \cdots > D(x, x-1) > D(x, x) = 0$
  - R : $0 = D(x, x) < D(x, x+1) < \cdots < D(x, R-1) < D(x, R) \leq m < D(x, R+1) < \ldots$
  - 위와 같이 $D(i, j)$는 단조 증가의 성질을 가지고 있기 때문에, Binary Search를 이용하여 L과 R을 구할 수 있습니다.

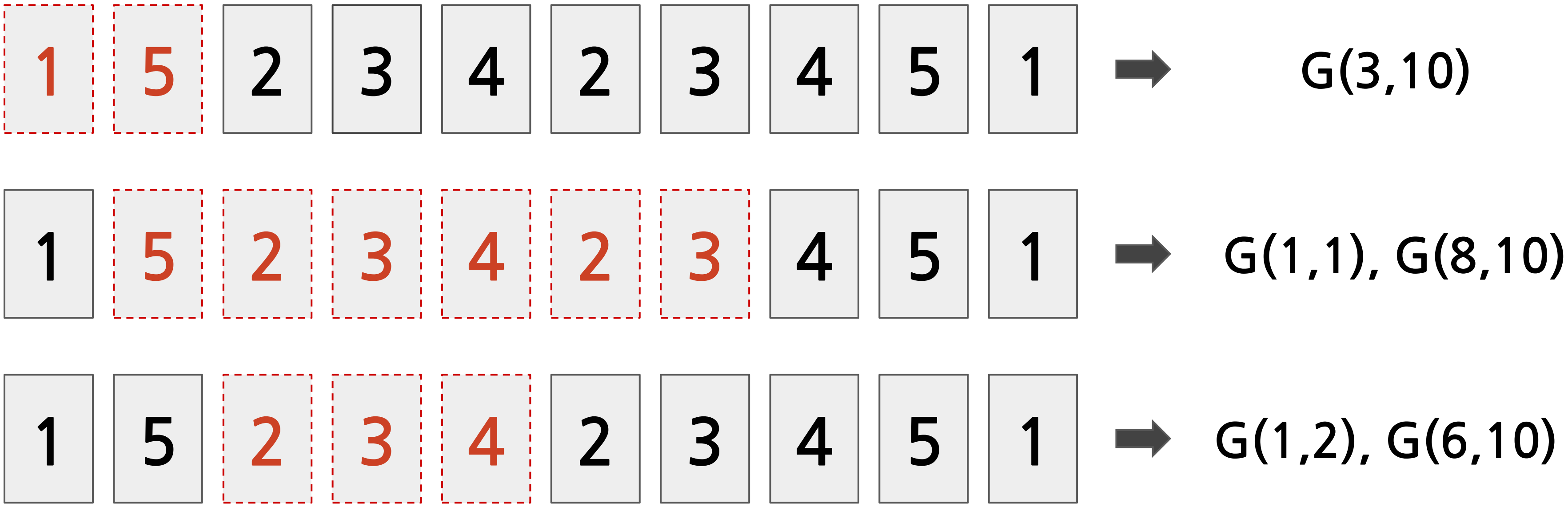
- 따라서, 두 개의 Segment Tree를 각각 관리하면 해결할 수 있습니다.
  - 집과 집 사이의 이동 거리 : 구간합 Segment Tree
  - 마스크 가격 : 최솟값 Segment Tree
- UPDATE 쿼리 : $O(\log(N))$, CALL 쿼리 : $O(\log^2(N))$
- 총 시간 복잡도 : $O(Q\log^2(N))$
- 구현이 다소 까다롭지만, Sqrt Decomposition을 사용하여 $O(QN^{0.5})$에 해결할 수도 있습니다.

O. 카드 모래성

출제자 : woonikim (김주현, Sogang ICPC Team)

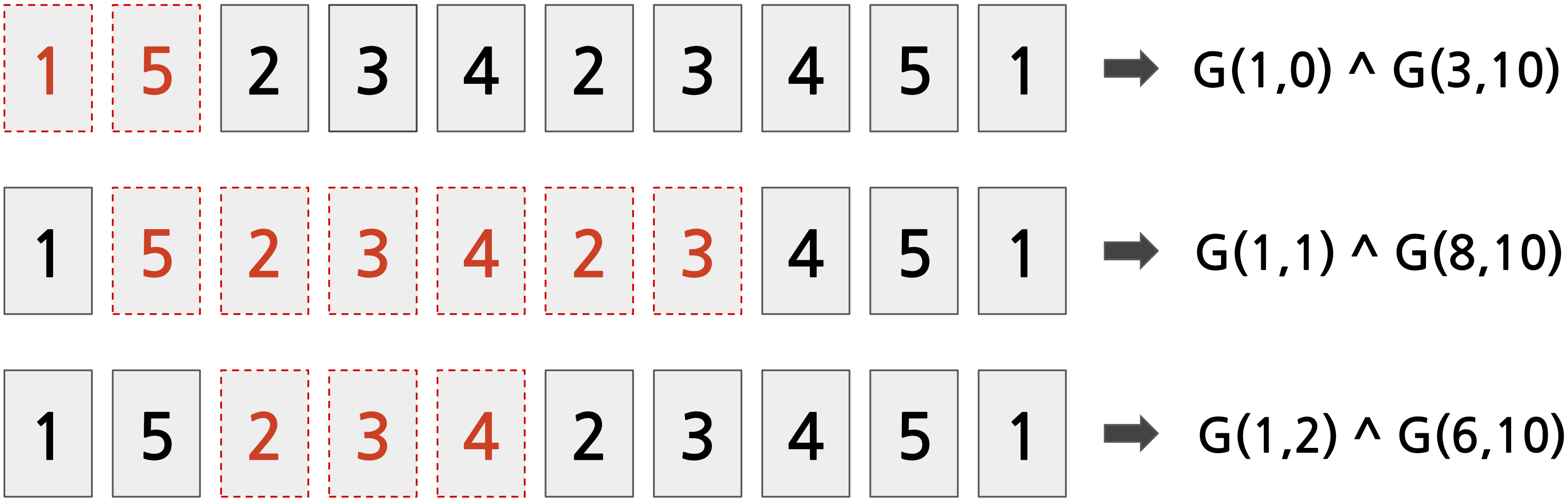
- 주현이가 승리하기 위해, 가장 먼저 선택해야 하는 카드 슬롯의 번호는?
  - 주현이와 세중이의 행동 집합이 동일합니다.
  - 두 명의 게임 플레이어는 동일한 목표를 가지고 있습니다.
  - 게임에서의 모든 정보는 두 플레이어에게 공개됩니다.
  - 게임 이론을 적용해봅시다.

- G(L, R) : L~R번 카드슬롯이 전부 선택 가능한 상태에서의 그런디 수
  - 다음 행동 집합이 어떻게 되는지 확인해봅시다.

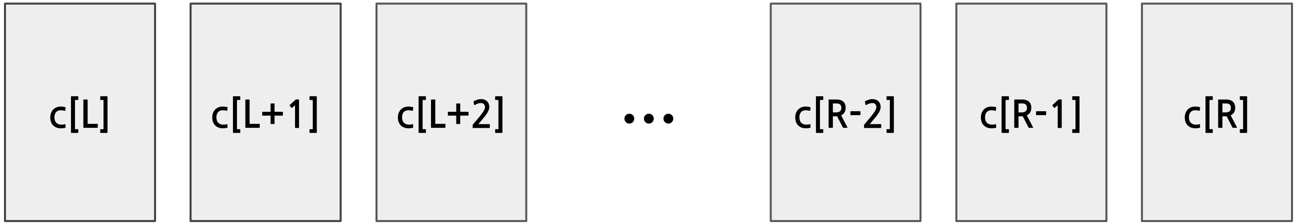


0. 카드 모래성

- G(L, R) : L~R번 카드슬롯이 전부 선택 가능한 상태에서의 그런디 수
  - 한 플레이어의 턴이 종료되면, 게임은 두 가지 상태의 게임으로 나뉘어진다고 볼 수 있습니다.
  - 따라서, 다음 상태의 grundy number는 두 sub-game의 grundy number를 XOR하여 구할 수 있습니다.
  - 선택할 수 있는 카드 슬롯이 없거나 L > R인 경우 다음 상태가 존재하지 않으므로, G(L, R) = 0 입니다.



- G(L, R) : L~R번 카드슬롯이 전부 선택 가능한 상태에서의 그런디 수



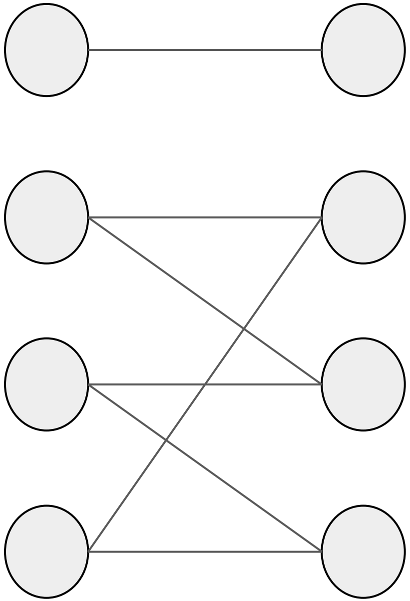
- $G(l, r) = \text{mex}\{\, G(L, i-1) \wedge G(i + c[i] + 1, R) \;\; \forall\, L \le i \le R,\ i + c[i] \le R \,\}$
- 특정 상태에서 선택 가능한 카드슬롯은 최대 N개입니다. 따라서, 다음 상태도 최대 N개입니다.
- 다음 상태가 최대 N개이므로, 모든 상태의 그런디 수는 N 이하라는 것이 보장됩니다.
- 따라서, $O(N^3)$에 모든 상태의 그런디 수를 계산해낼 수 있습니다.
- 위 사실을 알지 못해도 N이 최대 200이므로, 정렬을 이용해서 $O(N^3 \log N)$으로도 가능합니다.

P. Parity Constraint Perfect Matching

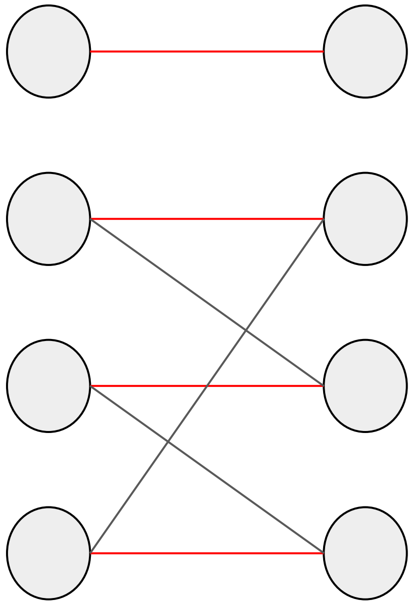
출제자 : lky7674(이국렬, 모르고리즘)

- 세 번째 Parity Constraint 문제입니다.
- 작년 중급 모의고사 문제인 Unique Solution(BOJ 19647)의 원본입니다.
- 작년 연세대회, 모의고사에 출제하려고 했다가 너무 어렵다고 판단해서 이보다 쉬운 문제로 대체했습니다.

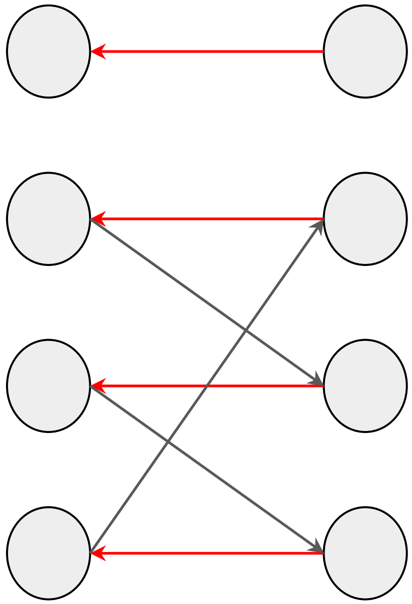
- Unique Solution과 마찬가지로 Residual Graph에서 Cycle을 찾는 문제입니다.



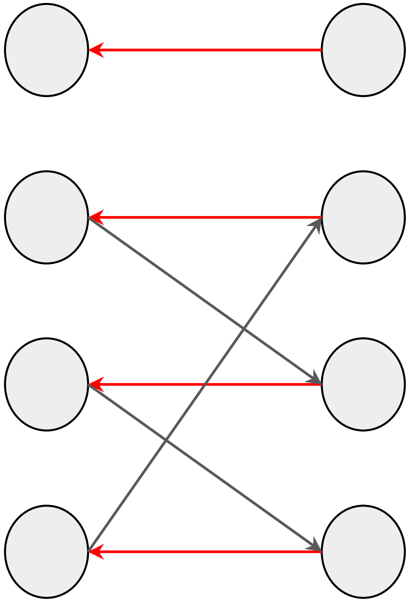
- 임의의 Perfect Matching을 찾습니다.



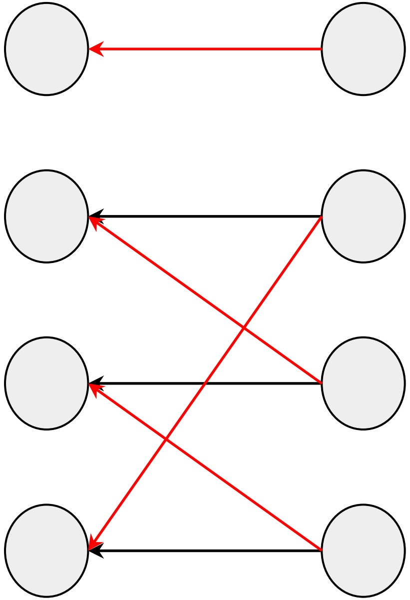
- 임의의 Perfect Matching을 찾습니다.
- Matching에 속하면 역방향, 안 속하면 정방향으로 방향을 정해줍니다.



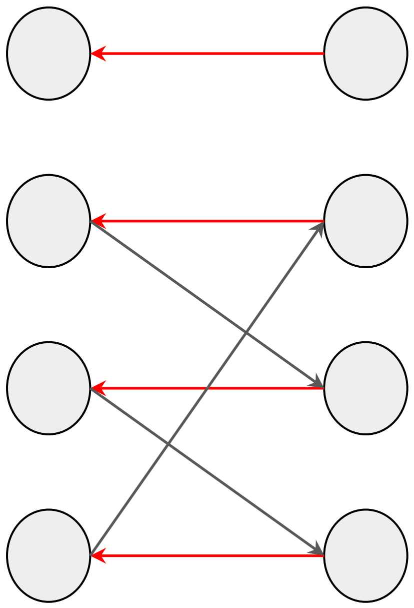
- 홀수면 cost를 1, 짝수면 cost를 0으로 지정합니다.
- odd cycle을 하나 찾습니다.



- odd cycle에 속한 역방향 간선을 matching에서 제외하고, 정방향 간선을 매칭에 추가합니다.
- parity가 다른 matching을 찾을 수 있습니다.



- 각 정점을 시작점으로 해서 BFS로 탐색하는 방식으로 odd cycle을 $O(N^3)$에 구할 수 있습니다.
- 시간 복잡도 : $O(N^3)$



- 추가 풀이)
- 원래 제약 조건은 $N \leq 100{,}000$, $M \leq 200{,}000$ 였으나 완화되었습니다.
- Perfect Matching은 Hopcroft-Karp로 $O(M \sqrt{N})$으로 구할 수 있습니다.
- Odd Cycle은 SCC로 선형 시간에 찾을 수 있습니다.
- 시간 복잡도 : $O(M \sqrt{N} + N)$.