

2024 하반기 전남대학교 PIMM 알고리즘 파티

Solutions Commentary Editorial

전남대학교 게임개발동아리 PIMM 알고리즘 연구회

This Contest is operated by

고민규 jj_kmk1013

김근성 onsbytd

송혜근 invrtd_h

박종현 belline0124

이윤수 lys9546

정영도 odo

최정환 jh01533

dongwook7

jwpassion1

kiwiyou

ksoosung77

lycoris1600

measurezero

mujigae

tony9402

utilforever

with ❤.

대회 순위상, 특별상 및 추첨상

- 자세한 특별상과 추첨상 선정 과정은 에디토리얼 리포지토리를 참조해주십시오.
- 부정행위를 한 것이 확실한 참가자가 제외되었습니다.
- 같은 특별상 조건을 여러 명이 달성한 경우, 추첨을 통해 선정되었습니다.

순위상

1. bnb2011
5 AC, -137

2. zhxpdustmqdyd
5AC, -270

3. sehujeong
5AC, -274

특별상

- 「알로록 달로록」 — patata22
가장 다양한 언어로 제출했다.
- 「두돈반이 덜컹거려서 눈을 떴어」 — xiaowuc1
가장 먼저 <훈련병의 편지>를 제출했다.
- 「09.22 패치노트」 이제 가을 시즌이 업데이트됩니다. — fermion5
가장 먼저 <더워!> 문제를 해결했다.
- 「필요한 만큼은 보여줬다」 — seawon0808
한 번만 제출한 사람들 중 가장 먼저 제출했다.
- 「잠시 소란이 있었지만 대회를 다시 진행하겠습니다」 — zhxpdustmqdyd
두 번 이상 제출한 사람들 중 연이은 두 제출의 간격이 가장 길었다.
- 「저희에게 시간과 예산을 더 주신다면…」 — alsduddml7
가장 마지막으로 제출했다.

추첨상

• 스타벅스 아이스 카페 아메리카노 T

john_cowart, zhxpdustmqdyd, ili, bumsoo0515, areian13, choisang0826, golazcc83, kangkh0906, isekaijoucho, hyperbolic

• 메가커피 아이스 아메리카노

bnb2011, alsduddml7, shs0911, bic0893, qkrusthf04, pedalcircle, newbieku, jlib245, soh2254, pda0503

• 베스킨라빈스 싱글레귤러

kes0716, a891, dbgusdn012, uno950114, riroan, mulebanga, rrkcl77, sehujeong, cinador, dabbler1

참가 기념 solved.ac 프로필 배경 / 뱃지

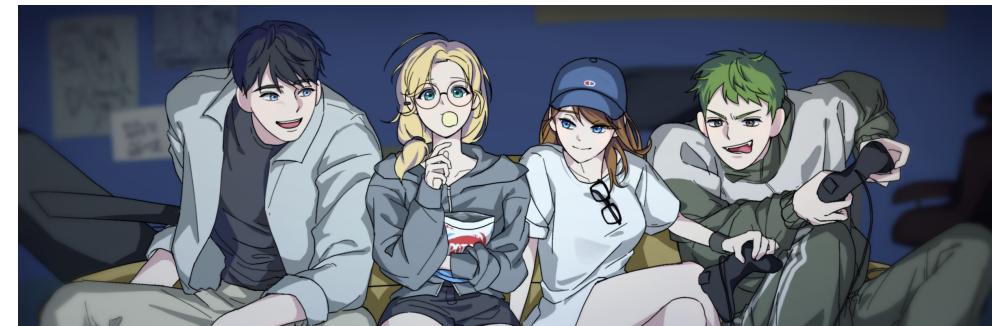
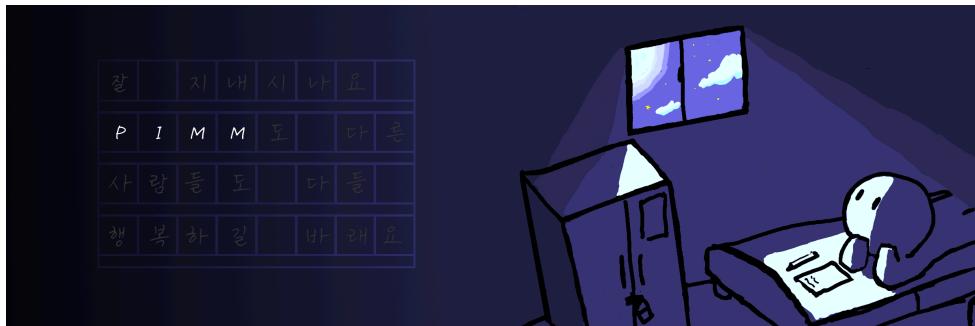
- 부정행위를 한 것이 확실한 제출은 획득 조건 평가에서 제외됩니다.
- 자세한 사항은 솔브드와의 협의 내용에 따라 변경될 수 있습니다.



나 더 이상 못 풀어

© 정희수@PIMM

2024 하반기 전남대학교 PIMM 알고리즘 파티에서 한 문제 이상 해결



훈련병의 편지

© 김근성@PIMM

2024 하반기 전남대학교 PIMM 알고리즘 파티에서 한 문제 이상 해결

시즌 두 번째 켄왕

© 정희수@PIMM

2024 하반기 전남대학교 PIMM 알고리즘 파티에서 두 문제 이상 해결

문제	의도한 난이도	출제자
A 더블 팰린드롬	Easy	이윤수 lys9546
B 균성아 일하자	Normal	고민규 jjkmk1013, 이윤수 lys9546
C 나무가 되고 싶다	Normal	송혜근 invrtd_h
D 더워!	Hard	고민규 jjkmk1013
E 도서 검색 프로그램	Hard	정영도 odo
F 훈련병의 편지	Challenging	김근성 onsbyd
G 피라미드 게임	Challenging	최정환 jh01533

A. 더블 팰린드롬

#combinatorics, #string

출제진 의도 – **Easy**

- 출제자: 이윤수 lys9546
- 제출 317회, 정답 156건
정답률: 50.789%
- 최초 해결: ychangseok, +1분

A. 더블 팰린드롬

- ✓ 더블팰린드롬 현상을 일으키는 문자열을 보면 X_1, X_2 가 팰린드롬인 경우에만 성립함을 알 수 있습니다.
- ✓ 따라서, 주어진 문자열 중 팰린드롬인 문자열의 개수를 세서 만들 수 있는 더블팰린드롬 현상을 $N * (N - 1)/2$ 로 세주면 답을 구할 수 있습니다.

B. 근성아 일하자

#data_structures, #implementation

출제진 의도 – Normal

- 출제자: 고민규^{jjkmk1013}, 이윤수^{lys9546}
- 제출 274회, 정답 79건
정답률: 29.562%
- 최초 해결: p_ce1052, +6분

B. 근성아 일하자

- ✓ 1번 쿼리가 들어올 때마다 쓰레기를 저장하여 2번 쿼리가 들어올 때 정렬한 후 현재 위치를 기준으로 좌우의 가장 가까운 쓰레기를 반복해서 찾아야 합니다.
- ✓ 이를 효율적으로 처리하기 위해서 두 개의 스택을 이용하여 현재 위치보다 작은 그룹 큰 그룹으로 나눈 후, 반복해서 이동시켜 이동 거리를 구할 수 있습니다.
- ✓ 그 외에 set과 lower_bound를 이용한 풀이, 투 포인터를 이용한 풀이 등이 있습니다.

C. 나무가 되고 싶다

#bfs, #dfs, #hash_set

출제진 의도 – Normal

- 출제자: 송혜근^{invrtd_h}
- 제출 151회, 정답 31건
정답률: 21.192%
- 최초 해결: chmpro, +8분

C. 나무가 되고 싶다

다양한 풀이가 존재하는 문제입니다.

- ✓ 가장 단순한 풀이는 1번 노드부터 DFS를 시작하는 것입니다.
 - ✓ V에 속하는 노드에는 방문하지 않습니다.
 - ✓ DFS를 돌면서 방문한 노드의 번호가 2^{60} 이상이 되면, 그 노드의 후손 노드는 모두 V 내의 어떤 노드도 조상으로 갖지 않는 노드가 됩니다. 따라서 탐색을 종료하고 -1을 출력하면 됩니다.
 - ✓ 방문한 노드의 번호가 2^{60} 미만이라면, 기존 DFS와 똑같이 동작합니다.

C. 나무가 되고 싶다

- ✓ DFS 풀이가 시간 안에 작동함을 증명하려면, 문제에서 요구하는 집합이 유한집합이라면 그 크기는 항상 $N - 1$ 이하임을 수학적 귀납법 등으로 증명하면 됩니다.
- ✓ 문제에서 요구하는 집합의 크기가 K 이면 그 집합과 인접한 노드의 개수가 $K + 1$ 임을 먼저 수학적 귀납법으로 증명합니다. 그리고 그 집합과 인접한 노드들의 집합은 V 의 부분집합임을 증명하면 됩니다.
- ✓ 그래프 탐색만 잘하면 되므로 BFS 등의 다른 탐색 방법을 사용해도 됩니다.
그러나 BFS를 사용할 때는 노드의 번호를, set을 사용하여 관리해야 하고, 방문한 노드의 개수가 N 을 넘어서면 바로 탐색을 종료하는 등의 전략이 필요합니다.

D. 더워!

#bfs, #graph_traversal

출제진 의도 — Hard

- 출제자: 고민규 jjkmk1013
- 제출 63회, 정답 35건
정답률: 55.556%
- 최초 해결: fermion5, +17분

D. 더워!

- ✓ 격자의 제한이 $100 * 100$, 민규의 불쾌함의 범위가 100 이하이므로 아래와 같이 상태를 저장하는 배열을 주어진 메모리 내에서 만들 수 있습니다.
- ✓ `visited[i][j][k]`: i 행 j 열에 도달하여 민규의 불쾌함이 k 일때의 최단 거리
- ✓ 민규의 집 위치인 S 의 좌표를 (x_s, y_s) , 약속 장소인 E 의 좌표를 (x_e, y_e) 라 했을 때, `visited[xs][ys]=0`으로 BFS를 시작하여 방문된 `visited[xe][ye]` 중 최솟값을 찾으면 문제를 해결할 수 있습니다. 만약 `visited[xe][ye]` 중 방문된 점이 없으면 -1 을 출력합니다.
- ✓ 지문에서 주어진 대로, 이동 → 불쾌함 변화 → 도착 판단의 순서를 잘 지켜 구현해야 하는 문제입니다.

E. 도서 검색 프로그램

#implementation, #parsing, #stack, #string

출제진 의도 – Hard

- 출제자: 정영도^{0do}
- 제출 12회, 정답 5건
정답률: 41.667%
- 최초 해결: bnb2011, +66분

E. 도서 검색 프로그램

도서 검색 프로그램은 주어진 지문의 이해가 어렵고 양이 많은 문제입니다.

특히, 문제를 풀기 위해서는 지문에서 주어진 배커스-나우르 표기법에 대한 이해가 필요합니다. 배커스-나우르 표기법에 대해서는 지문 및 연결된 위키백과 링크에서 충분히 설명하고 있으므로 이번 에디토리얼에서 따로 이에 대해 부연 설명을 하진 않습니다.

E. 도서 검색 프로그램

- ✓ 문제에서 주어지는 모든 `<book>`에 대해 `<search>`를 수행해야 하므로, 이에 대한 기본적인 시간복잡도는 $O(N * M * \text{len}(<\text{search}>))$ 가 됩니다.
`<method>`에 등장하는 기호의 시간복잡도를 적용하는 것으로, 문제의 최대 시간복잡도를 확인할 수 있습니다.
- ✓ `<AND>`, `<OR>`, `<NOT>`을 하나로 묶어 논리 메소드라 하겠습니다.
논리 메소드를 $O(\text{len}(<\text{search}>))$ 에 계산하는 알고리즘을 작성하게 되면 시간 초과가 발생합니다. 특히, 문제의 `<methods>`가 재귀적으로 등장할 수 있음을 고려하면 프로그램의 동작 시간이 지수 단위로 늘어날 수 있습니다.
그렇기 때문에 논리 메소드를 $O(1)$ 에 동작시키는 알고리즘을 작성하는 게 이 문제의 핵심이 되겠습니다.

E. 도서 검색 프로그램

- ✓ 논리 메소드를 연산자로 간주할 때, <search> 또한 연산자가 피연산자의 앞쪽에 나타나는 “전위 표기식”으로 간주할 수 있습니다.
전위 표기식을 빠르게 처리하기 위해서는 스택 자료구조를 사용해야 합니다. 문자열 <search> 자체를 처리하는 스택을 만드는 것으로 논리 메소드를 $O(1)$ 안에 처리할 수 있습니다.
- ✓ <nameCorrect>와 <authorCorrect>는 $O(1)$ 안에 처리할 수 있습니다.
<tagCorrect>는 집합(set) 자료구조를 사용하는 것으로 $O(1)$ 안에 처리할 수 있어 집합 자료형을 사용하여 문제를 푸는 것이 권장됩니다. <nameInclude>는 $O(\text{len}(<\text{str}>))$ 안에 처리할 수 있습니다.
따라서, 문제의 최대 시간복잡도는 $O(N * M * \text{len}(<\text{search}>) * \text{len}(<\text{str}>))$ 로 나타나며, <search> 안에 들어가는 <method>의 개수는 <search>의 길이보다 작기에 실제 연산량은 시간복잡도 식에 최대 크기의 상수를 대입한 것보다 더 작습니다.

F. 훈련병의 편지

#bitset, #dp, #string, #z

출제진 의도 – Challenging

- 출제자: 김근성^{onsbtyd}
- 제출 27회, 정답 0건
정답률: 0.000%
- 최초 해결: -

F. 훈련병의 편지

- ✓ 본 문제의 내용을 요약하면 “ N 개의 편지지에서 정해지지 않은 M 개의 편지지를 제거한 후 내용을 모두 합쳤을 때 항상 특정한 이름이 등장하는지 판단해라”입니다.
- ✓ 만약 N 과 M 이 작다면 조합을 사용해서 실제로 M 개의 편지지를 제거한 후 편지의 내용을 합치고, 각 이름이 등장하는지 판단할 수 있습니다.
- ✓ 하지만, 이 문제에서는 N, M 는 450이기에 최대 $\binom{450}{225}$ 가지 조합이 발생합니다. 따라서 실제로 제거한 후 합쳐보는 방법은 사용할 수 없습니다.

F. 훈련병의 편지

- ✓ 제한에 의해 이름의 최대 길이는 항상 편지지 중 가장 짧은 길이보다 짧습니다. 따라서 이름은 최대 두 개의 편지지에만 나눠 등장할 수 있습니다.
- ✓ 그렇기에 이름 등장 여부를 확인하기 위해 모든 조합을 고려한 후 편지지를 합칠 필요가 없습니다.
- ✓ 지금 보고 있는 편지지가 i 번일 때 이 편지지로 인해 이름이 만들어지는지 확인하려면, $1 \sim (i - 1)$ 번 편지지를 i 번 편지지와 합쳐보며 이름이 등장하는지 확인하면 됩니다.
- ✓ 쉽게 판단할 수 있어 보이지만 지금 조건에서는 $1 \sim (i - 1)$ 번 사이에 어떤 편지지가 누락되었는지, 이미 이름이 만들어져 있는지 등의 상태를 모두 고려해야 합니다. 이는 역시나 매우 긴 시간, 큰 공간이 문제가 됩니다.

F. 훈련병의 편지

- ✓ 이를 해결하기 위해 문제를 좀 더 간단하게 표현하겠습니다.
- ✓ “ N 개의 편지지 중 어떤 M 개의 편지지를 제거해도 편지지의 내용을 합친 문자열에 이름이 들어간다.”
- ✓ 이는 다음 문장과 동치입니다.
“ N 개의 편지지 중 어떤 $N - M$ 개의 편지지를 골라도 편지지의 내용을 합친 문자열에 이름이 들어간다.”
- ✓ 또, $N - M$ 개의 편지지를 골랐을 때 이름이 들어가므로 자연스럽게 $N - M$ 개 이상의 편지를 골랐을 때도 이름이 들어갑니다.

F. 훈련병의 편지

- ✓ 그렇기에 최종적으로 다음 문장과 같이 변형할 수 있습니다.
“편지지의 내용을 합친 문자열에 이름이 들어가지 않도록 편지지를 고르면 $N - M$ 개 이상을 고를 수 없다”

이렇게 되면 이제 편지지에 이름이 들어가지 않으면서 고를 수 있는 최대 편지지 개수를 구하는 문제로 바뀌게 됩니다.

- ✓ 이름이 생기는 경우는 지문에도 나와 있듯 두 가지입니다.
 1. 이름이 포함된다.
 2. 순서가 역전되지 않도록 두 편지지를 합쳤을 때 이름이 생긴다.
- ✓ 그래서 이 경우에 맞게 두 가지를 고려하면 됩니다.

F. 훈련병의 편지

- 1번 경우에 해당하는 편지지는 답에 포함될 수 없습니다. 그렇기에 계산하지 않습니다.
 - 2번 경우는 $i < j$ 일때 i 번째 편지지와 j 번째 편지지를 합쳐서 이름이 생기지 않는다면, i 번째 편지지 다음에 j 번째 편지지를 선택해 놓을 수 있습니다.
- ✓ 따라서 j 번째 편지지를 골랐을 때 편지지의 최대 개수는 i 번째에서의 편지지 최대 개수 +1입니다.

이러한 계산은 DP를 활용해 처리할 수 있습니다.

```
for j in 1 until N:  
    if j번째 편지지에 이름이 포함되지 않음 then  
        dp[j] = 1  
        for i in 1 until (j - 1):  
            if i번째 편지지와 j번째 편지지를 합쳤을 때 이름이 포함되지 않음 then  
                dp[j] = max(dp[i] + 1, dp[j])
```

F. 훈련병의 편지

- ✓ 이 과정에서 계산되는 $dp[1 \sim n]$ 중 최대 값이 $M - N$ 미만이라면 답이 Yes, 아니라면 No입니다.
- ✓ 이 과정을 모든 이름에 대해 반복합니다.

이 과정까지의 시간복잡도입니다:

- ✓ $O(N^2 * (\max(|s_i|) + \max(|k_i|)) * K)$
 $= O(\text{두 편지 고르기} * \text{이름 등장 판별} * \text{이름 개수})$
- ✓ 주어진 제한에서 최대 820억이 발생할 수 있으므로, 더 최적화해야 합니다.

F. 훈련병의 편지

- ✓ 편지 2개를 고르고 두 편지를 합쳐 이름의 등장 여부를 판단하는 것이 너무 느립니다.
 - ✓ 따라서 각 편지에서 이름의 등장 가능성을 확인해야 합니다.
 - ✓ 또 그것을 구한 이후 두 편지를 합쳐서 이름이 등장하는지 빠르게 확인해야 합니다.
- ⇒ 이름이 두 편지에 나눠 등장하는 경우를 확인하겠습니다.

F. 훈련병의 편지

geunseong 기준

- ✓ [geunseong/] [geunseon/g] [geunseo/ng] ... [g/eunsong] [/ geunseong]
- ✓ 예를 들어 앞쪽 편지지에 geunseo, 뒤쪽 편지지에 ng 가 등장하면, 두 편지지가 합쳐질 때 geunseong이 나타납니다.
- ✓ 앞쪽 편지지에서 geunseong, geunseon, …, ge, g, 공백, 즉 이름의 접두사들이 등장하는지 구해두고,
뒤쪽 편지지에서 공백, g, ng, …, eunseong, geunseong, 즉 이름의 접미사들이 등장하는지 구해두면 이를 합쳐 이름의 등장 여부를 빠르게 구할 수 있습니다.

F. 훈련병의 편지

- ✓ 이렇게 각 길이에 해당하는 접두사, 접미사를 구하는 데에는 Z 알고리즘을 사용할 수 있습니다.
- ✓ 임의의 이름 name과 편지지 letter가 있을 때
 $target = name + \$ + letter$ 로 두고, 이 target으로 Z를 돌리면 Z에 name과 일치하는 접두사의 최대 길이가 저장됩니다.
- ✓ $1 \leq i \leq |name|$ 일 때, $z[|target|-i]$ 에 i 가 저장되어 있다면
name의 앞에서부터 i 글자가 letter의 끝에 등장한다는 뜻이 됩니다.

F. 훈련병의 편지

- ✓ 마찬가지로 접미사 또한 구할 수 있습니다.
- ✓ $\text{target} = \text{letter} + \$ + \text{name}$ 로 두었을 때, 이 target 으로 Z를 돌리면 Z에 letter 와 일치하는 접두사의 최대 길이가 저장됩니다.
- ✓ $1 \leq i \leq |\text{name}|$ 일 때, $z[|\text{target}| - i]$ 에 i 가 저장되어 있다면 name 의 뒤에서부터 i 글자가 letter 의 시작에 등장한다는 뜻이 됩니다.

- ✓ $\text{prefix}[i]$ 에 i 길이의 접두사가 있는지, $\text{suffix}[i]$ 에 i 길이의 접미사가 있는지 저장합니다.
- ✓ 이렇게 접두사, 접미사를 구하더라도 조합을 확인하는데 $|\text{name}|$ 만큼 확인해야 합니다. 좀 더 빠른 처리가 필요합니다.

F. 훈련병의 편지

접두사와 접미사를 비트셋에 저장하면 비트셋끼리 AND 연산을 사용해 한 번에 비교할 수 있습니다.

- ✓ 비트셋 AND를 위해 접미사가 저장되는 방식을 조금 변경합니다.
- ✓ prefix[i]와 매칭되는 것은 suffix[|name| - i]입니다.
 - ✓ 같은 자리의 비트끼리 연산 되는 비트 연산의 특성을 고려하여 접미사를 저장할 때부터 suffix[i]에 |name| - i 길이의 접미사가 있는지 저장한다면, prefix[i]와 suffix[i]의 매칭을 비교하면 되고, !(suffix & prefix).any() 와 같은 비트셋끼리의 연산 한 번으로 이름의 존재 여부를 알 수 있습니다.
- ✓ 추가로 접두사를 구하는 과정에서 만약 $\max(z[i])$ 가 $|name|$ 라면, 해당 편지지에 이름이 온전하게 등장함을 의미합니다. 이 경우, 이후 DP에서 이름이 포함된 편지지를 빠르게 건너뛰기 위해서 skip[i] 등의 플래그로 표시합니다.

F. 훈련병의 편지

이제 더 빠르게 이름의 등장 여부를 확인할 수 있게 되었습니다.

- ✓ 앞서 이야기한 DP를 비트셋과 skip을 이용하여 수행합니다.
- ✓ 이러한 과정이 이름마다 반복되므로, 시간복잡도는 총 $O(KN^2 \max(s_i)/64)$ 입니다.

F. 훈련병의 편지

- ✓ 정해는 Z이지만, 다양한 풀이가 존재할 수 있는 문제입니다.
- ✓ 파이썬은 이 풀이로는 시간초과가 발생합니다.

G. 피라미드 게임

#game_theory, #sprague_grundy

출제진 의도 – Challenging

- 출제자: 최정환 jh01533
- 제출 18회, 정답 3건
정답률: 16.667%
- 최초 해결: fermion5, +81분

게임의 단순화

- ✓ $1 \dots K - 1 (K \neq 1)$ 층은 답에 영향을 주지 않기 때문에 빙간으로 생각하는 것이 편합니다.
- ✓ 먼저 피라미드를 다루기 익숙한 그래프로 생각해 봅시다.

빙간을 제외한 각 칸을 하나의 노드로 생각하고 고를 수 있는 부분피라미드의 꼭대기에서 부분피라미드의 빙간을 제외한 나머지 칸에 간선을 그어줍니다.

해당 그래프는 특수한 형태의 DAG로, 어떤 노드 X 를 하나 골라 X 에 적힌 수 x 를 y 만큼 줄이고 $x \oplus (x - y)$ 만큼 자식노드에 XOR 해주는 게임으로 바꿀 수 있습니다.

G. 피라미드 게임

풀이

- ✓ 각 노드에 아래의 규칙을 따라 AB 인덱싱을 해줍니다.

$\text{index}[X] = \{\text{노드 } X\text{의 자식노드 } Y_i\text{에 있는 } \text{index}[Y_i] = A\text{의 개수가 짝수면 } A, \text{ 아니면 } B\}$

- ✓ 이렇게 하면 다음이 성립합니다.

$\text{index}[X] = A$ 인 노드에 적힌 수들을 모두 XOR한 값이 0이면 후공, 아니면 선공이 게임에서 필승법을 갖는다.

G. 피라미드 게임

증명

- ✓ 먼저 A 에 적힌 수들을 모두 XOR 한 값을 g 라 하겠습니다.

게임이 끝나는 시점의 $g = 0$ 입니다.

1. 게임은 항상 끝난다.

- ✓ 그래프가 DAG로 항상 상위층이 줄어드는 행동을 하므로 게임은 항상 끝납니다.
피라미드를 아래에서부터 한 층씩 늘려가면서 보면 어렵지 않게 관찰할 수 있습니다.

G. 피라미드 게임

2. 현재 턴에 $g = 0$ 이라면 어떠한 행동을 하든지 $g \neq 0$ 이 된다.

- ✓ $\text{index}[X] = \{\text{노드 } X\text{의 자식노드 } Y_i\text{에 있는 } \text{ind}[Y_i] = A\text{의 개수가 짝수면 } A, \text{ 아니면 } B\}$ 라는 index 의 정의에 따라 어떠한 부분피라미드를 골라도 해당 부분피라미드에 포함된 A 의 개수는 항상 홀수입니다.

즉 어떤 수 P 를 골라 행동한다고 했을 때 $g \rightarrow g \oplus P$ 가 된다는 것을 알 수 있습니다.

P 는 0이 될 수 없으므로 항상 성립합니다.

3. 현재 턴에 $g \neq 0$ 이라면 $g = 0$ 으로 만들 수 있는 행동이 반드시 하나 이상 존재한다.

- ✓ A 로 인덱싱된 노드에 적힌 수들만 따로 빼어서 봅시다.

어떠한 행동을 했을 때 님버(g)가 변하는 것이 일반적인 님 게임과 완전히 동일합니다.

님 게임의 증명에 따라 A 로 인덱싱된 노드 중에 g 를 0으로 만들 수 있는 행동이 하나 이상 존재한다는 뜻이 되고 해당 명제도 성립합니다.

G. 피라미드 게임

- ✓ 현재 상태가 $g \neq 0$ 이라면 항상 $g = 0$ 으로 만들 수 있고 $g = 0$ 이라면 어떠한 행동을 하든지 $g \neq 0$ 이 됩니다. 게임은 항상 끝나고 게임의 끝에서 $g = 0$ 이므로 $g = 0$ 으로 만들 수 있는 플레이어가 승리하게 됩니다.

구현

- ✓ 간선이 상당히 많기 때문에 일반적으로 그래프를 만드는 방법은 시간, 메모리 초과가 납니다.
- ✓ 그래프가 특이하다는 점을 이용해서 누적합, DP 등을 이용하여 A 의 개수만 세어주면 $O(N^2)$ 에 처리할 수 있습니다.