UCPC 2020 예선 풀이

Official Solutions

by

전국 대학생 프로그래밍 대회 동아리 연합

UCPC 2020

2020년 7월 25일

문제		의도한 난이도	출제자
A	수학은 비대면강의입니다	**Easy**	shiftpsh
B	길이 문자열	**Hard**	sait2000
C	삼항 연산자	**Medium**	functionx
D	ㄷㄷㄷㅈ	**Easy**	kdh9949
E	감자 농장	**Hard**	evenharder
F	전투 시뮬레이션	**Medium**	ckdgus2482
G	루머	**Medium**	QuqqU
H	사과나무	**Easy**	evenharder
I	인버스 ㄷㄷㄷㅈ	**Hard**	16silver $^{+2}$
J	역학 조사	**Medium**	functionx

A. 수학은 비대면강의입니다

`bruteforcing`, `math`

출제진 의도 – **Easy**

- 제출 618회, 정답 272팀 (정답률 45.63%)
- 출제자 – `shiftpsh`

올해도 '수학은' 시리즈가 돌아왔습니다!

연립방정식을 푸려면...

- ✓ 가감법, 대입법
- ✓ $\begin{bmatrix} a & b \\ d & e \end{bmatrix}^{-1}$ 을 계산해서 어쩌고 저쩌고...

같은 방법들이 생각나겠지만, 우리에겐 컴퓨터가 있기 때문에 괜찮습니다.

A. 수학은 비대면강의입니다

문제의 정답인 x, y는 모두 -999 이상 999 이하의 정수임이 보장된다고 했으므로,

- ✓ $-999 \leq x \leq 999$와 $-999 \leq y \leq 999$인 x와 y에 대해 가능한 모든 경우를 시도했을 때
- ✓ $ax + by = c$와 $dx + ey = f$ 모두를 만족하는 경우가 있다면

그 경우가 문제의 정답이 됩니다.

가감법으로 계산해도 정답이나, a, b, c, d 중에 0 이 존재하는 케이스가 있었기 때문에 GCD를 구하는 과정에서 많은 팀이 해당 케이스에서 런타임 에러를 받았습니다.

역행렬을 계산해 푼 팀도 있었습니다. 이 경우엔 문제에서 답이 하나만 존재함을 보장했고, 계수와 답에 대한 범위 조건이 있었기 때문에 $\det \begin{bmatrix} a & b \\ d & e \end{bmatrix} \neq 0$ 이었습니다. 따라서 $(x, y) = \left(\frac{ce - bf}{ae - bd}, \frac{cd - af}{bd - ae} \right)$ 를 문제 없이 계산할 수 있습니다.

B. 길이 문자열

`number_theory`
출제진 의도 – **Hard**

- 제출 542회, 정답 16팀 (정답률 3.14%)
- 출제자 – `sait2000`

B. 길이 문자열

- 양의 정수 N 의 십진법 표기의 길이를 v 라고 하면 길이 문자열의 마지막 $v+1$ 글자는 '-' + (N 의 십진법 표기) 입니다.
- 마지막 $v+1$ 글자를 제외한 앞의 $N-v-1$ 글자는 또 다시 길이 문자열입니다.

`(1-3-5-7-10)-13`

B. 길이 문자열

길이가 v 자리 수인 길이 문자열에 대해서 답이 주기 $v+1$ 로 나타납니다.

- ✓ `...1000-1005-1010-1015-1020-102...`
- ✓ `...-1001-1006-1011-1016-1021-10...`
- ✓ `...7-1002-1007-1012-1017-1022-1...`
- ✓ `...98-1003-1008-1013-1018-1023-...`
- ✓ `...999-1004-1009-1014-1019-1024...`

주기성에 의해서 $a \times 10^b \geq 21$ 일 때 답이 될 수 있는 문자열은 다음의 3가지 뿐입니다.

- ✓ `-2-4-6-8-11-14-17...`
- ✓ `1-3-5-7-9-12-15-1...`
- ✓ `1-3-5-7-10-13-16-...`

편의상 앞으로 각 경우를 8, 9, 10이라고 부릅니다.

- ✓ 주기성으로부터 $10^{v-1} - v$ 부터 10^{v-1} 까지의 답만 알고 있으면 더 큰 수에 대한 답을 알아낼 수 있습니다.
- ✓ $10^{v-1} - v$ 부터 10^{v-1} 까지의 답을 이미 알고 있다고 가정하면 쿼리를 오프라인으로 처리할 수 있으므로 $a \times 10^b$ 이 v 자리수인 테스트 케이스에 대해서 $a \times 10^b \mod (v+1)$ 을 계산해서 답을 알아냅니다.

두 자리수에 대해서는 8부터 10까지의 답이 8, 9, 10입니다.
세 자리수 이상에 대해서는 $10^{v-1} - v$ 부터 10^{v-1} 까지의 답으로부터 $10^v - (v+1)$ 부터 10^v 까지의 답을 계산합니다.

✓ $10^v - (v+1)$ 부터 $10^v - 1$ 까지의 답은 $10^{v-1} - v$ 부터 10^{v-1} 까지의 답을 왼쪽으로 $9 \times 10^{v-1} - 1$ 번 시프트한 형태가 됩니다.

수	97	98	99	100
답	10	8	9	9

수	996	997	998	999
답	9	10	8	9

- 10^v 의 답은 $10^v - 1$ 의 답과 같습니다.

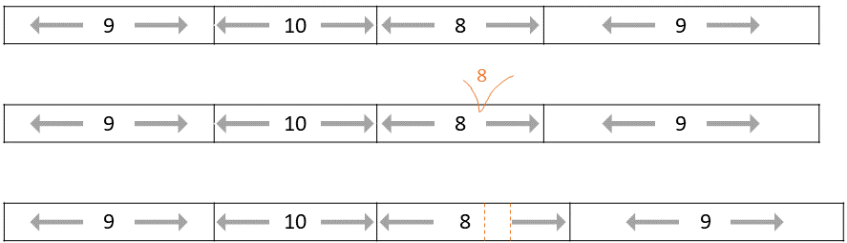
```
...-987-991-995-999
...-987-991-995-1000
```

수	996	997	998	999	1000
답	9	10	8	9	9

B. 길이 문자열

claim: 한 주기의 답은 항상 `/8+9+(10)+/`을 시프트한 형태가 됩니다!!

- ✓ 두 자리수의 경우 주기가 8, 9, 10을 시프트한 형태입니다.
- ✓ 10^v 의 답이 추가될 때 항상 추가되는 위치에 인접한 수와 같은 답이 추가됩니다.
- ✓ 이 때 원래 수가 속해있는 구간의 길이가 1이 늘어납니다.



한 주기 내에서 8, 9, 10이 연속해서 등장하는 3개의 구간 각각에 대해 시작하는 위치만을 계산하는 방식으로 구현할 수 있습니다.

C. 삼항 연산자

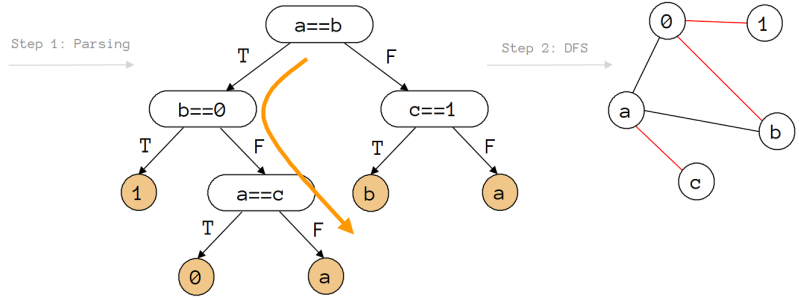
`parsing`, `dfs`

출제진 의도 – **Medium**

- 제출 180회, 정답 25팀 (정답률 13.89%)
- 출제자 – `functionx`

C. 삼항 연산자

```
a==b?b==0?1:a==c?
        0:a:c==1?b:a
```



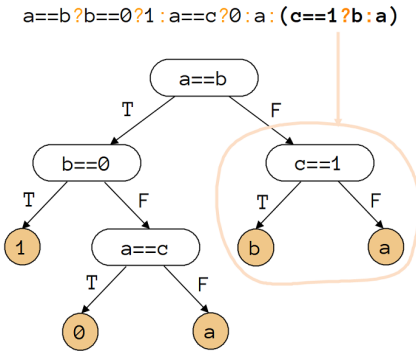
- ✓ Part 1. 식 파싱
- ✓ Part 2. 경우의 수 계산

```
a==b?b==0?1:a==c?0:a:c==1?b:a
```

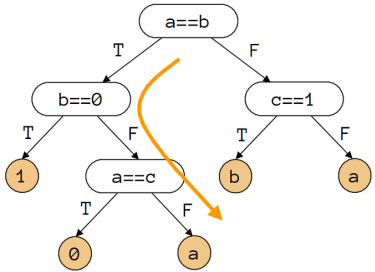
```
a==b?b==0?1:a==c?0:a:(c==1?b:a)
```

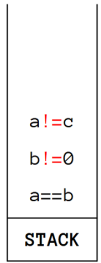
- 일단 ?, : 를 기준으로 식들을 나눕니다.
- 마지막 ?을 계산하는 방법은 언제나 유일합니다.

C. 삼항 연산자



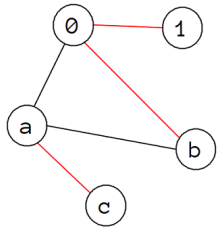
- 식을 뒤에서부터 읽으면서 계산 트리를 만듭니다.
- 식1 ? 식2 : 식3 이 나오면 식1 → 식2, 식1 → 식3 간선을 긋고 식1 만 남깁니다.





- 계산 트리에서 DFS를 하여 리프 노드까지 탐색합니다.
- 리프 노드까지 가면서 지나친 명제들은 스택으로 관리합니다.

C. 삼항 연산자



- 이제 모은 명제들을 가지고 경우의 수를 셉니다.
- 0, 1과 N 개의 변수들이 노드입니다.
- 두 노드의 값이 같으면 검은 간선으로, 다르면 빨간 간선으로 연결합니다.
- 리프 노드와 0을 검은 간선으로, 0과 1을 빨간 간선으로 연결합니다.

- 그래프 탐색을 하여 모순이 없는지 판별합니다.
- 모순이 없다면, 경우의 수는 $2^{(\text{연결 요소의 수}-1)}$ 입니다.

D. ㄷㄷㄷㅈ

`tree`

출제진 의도 – **Easy**

- ✓ 제출 1,372회, 정답 189팀 (정답률 14.07%)
- ✓ 출제자 – kdh9949

D. ㄷㄷㄷㅈ

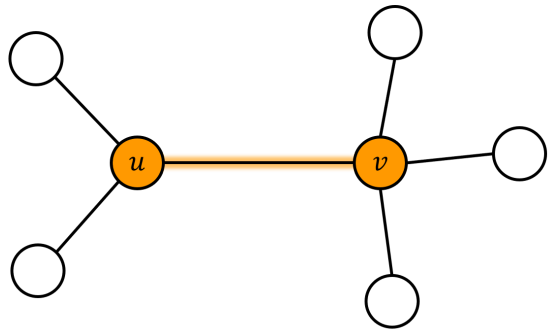
- 트리에서 'ㄷ'의 개수와 'ㅈ'의 개수를 각각 세서 비교하면 문제를 풀 수 있습니다.
- 모든 가능한 정점 4개 집합을 다 잡아서 세 보는 방식은 느릴 뿐더러, 구현도 생각보다 힘듭니다.
- 각각을 빨리 셀 수 있는 방법에 대해 생각해 봅시다.

1. ㄷ 세기

- ✓ 'ㄷ' 모양은 간선 3개가 일렬로 있는 형태이니, 가운데 간선을 고정해 봅시다.
- ✓ 정점 u와 v를 잇는 간선이 가운데 간선인 'ㄷ' 모양의 개수를 구해 봅시다.
- ✓ u에서 뻗어나가는 간선 하나와 v에서 뻗어나가는 간선 하나를 각각 고르면 'ㄷ' 모양이 됩니다. 단, u - v 간선을 또 고르면 안 됩니다.
- ✓ 정점 x의 차수 (x에 연결된 간선의 수)를 d_x라 하면, 트리의 각 간선 u - v에 대해 $(d_u - 1)(d_v - 1)$를 모두 더한 값이 'ㄷ'의 개수가 됩니다.

D. ㄷㄷㄷㅈ

아래 그림에서 u — v 간선이 가운데 간선인 'ㄷ' 모양은 총 6 개 있습니다.



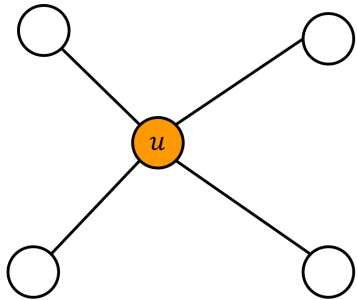
$$(d_u - 1)(d_v - 1) = (3 - 1)(4 - 1) = 6$$

2. ㅈ 세기

- ‘ㅈ’ 모양은 정점 하나를 중심으로 나머지 정점들이 붙어있는 형태이니, 이번에는 가운데 정점을 고정해 봅시다.
- 정점 u 가 가운데 정점인 ‘ㅈ’ 모양의 개수를 구해 봅시다.
- u 와 인접한 서로 다른 정점 3 개를 고르면 ‘ㅈ’ 모양이 됩니다.
- 각 정점에 대해 $\binom{d_u}{3} = \frac{d_u(d_u-1)(d_u-2)}{6}$ 를 모두 더한 값이 ‘ㅈ’의 개수가 됩니다.

D. ㄷㄷㄷㅈ

아래 그림에서 정점 u가 가운데 정점인 'ㅈ' 모양은 총 4개 있습니다.



$$\binom{d_u}{3} = \binom{4}{3} = \frac{4!}{3!\,1!} = 4$$

- 경우의 수를 계산 및 저장할 때는 64-bit 자료형 (`long long` 등) 을 사용해야 함에 유의하세요.
- ‘ㄷ’의 개수가 ‘ㅈ’의 개수의 3 배보다 큰지 작은지 같은지 비교할 때는 `D == 3 * G` 와 같이 정수로만 이루어진, 나눗셈이 들어가지 않은 식을 쓰는 것이 좋습니다. 다른 많은 문제에도 똑같이 적용되는 팁입니다.

E. 감자 농장

`prefix_sum`

출제진 의도 – **Hard**

- 제출 383회, 정답 35팀 (정답률 9.14%)
- 출제자 – `evenharder`

출제진의 예상보다 훨씬 많이 시도되고 또 풀린 문제입니다.

- 시뮬레이션을 통해 이하를 한 칸씩 이동시켜볼 수 있습니다. 시간복잡도는 $\mathcal{O}(QN^2)$ 입니다.
- 조금 더 빨리 해볼 수 없을까요? 두 포인터 기법을 이용해 이하가 도달한 가장 왼쪽 칸과 오른쪽 칸의 위치를 저장하고, 이 포인터들만 움직이면 $\mathcal{O}(QN)$ 에 해결할 수 있습니다.
- 약간 더 나아가면, 빈 칸을 하나로 묶어서 생각해볼 수 있습니다. 그래도 시간 복잡도는 변하지 않지만... 뭘 알아낼 수 있을까요?

E. 감자 농장

| 를 감자라 하고, 각 칸에서 움직이기 시작한 후 처음으로 방향을 전환한 후 칸을 방문한 횟수는 다음과 같이 나타납니다. <>는 해당 칸에서 시작했다는 뜻입니다.

```
 0 | 1 | 2 | 3 | 4 | 5
<1>
 1   <3>   2
 1    3   <5>   4    2
           2   <4>   3    1
                     <2>   1
                           <0>
```

(a) 감자가 5개 있는 농장

```
 0 | 1 | 2 | 3 | 4 | 5 | 6
<1>
 1   <3>   2
 1    3   <5>   4    2
           2    4   <6>   5    3    1
                     2   <4>   3    1
                               <2>   1
                                     <0>
```

(b) 감자가 6개 있는 농장

- 자연스럽게 위치에 따른 줍는 감자의 개수를 구할 수 있습니다.
- 각각의 간격을 d_i 라 하면, 탈출 시간 또한 $\sum d_i$ 와 $\sum i \cdot d_i$ 의 부분합 등을 통해 계산할 수 있습니다.
- 이하가 처음에 몇 번째 감자 사이에 있는지는 전처리를 통해 계산할 수 있으므로, 돌이 없는 농장에선 전처리 $\mathcal{O}(N)$ 이후 쿼리당 $\mathcal{O}(1)$ 에 구할 수 있습니다.

돌이 있을 때는 어떻게 해야 할까요?

- ✓ 돌 사이에 있으면 탈출할 수 없으므로, 사이의 감자 개수만 세면 됩니다. 전처리로 가능합니다.
- ✓ 그러므로 돌이 동쪽에만 있는 경우와 서쪽에만 있는 경우만 고려하면 됩니다.

E. 감자 농장

돌을 #라 표기하겠습니다.

```
 0 | 1 | 2 | 3 | 4 # 5
<1>
 1  <3>  2
 1   3  <5>  4   2
 1   3   5  <7>  6
 1   3   5   7  <9>
```

(a) 감자가 5개 있는 농장

```
0 # 1 | 2 | 3 | 4 | 5 | 6
   <10> 9   7   5   3   1
    6  <8>  7   5   3   1
    2   4  <6>  5   3   1
            2  <4>  3   1
                   <2>  1
                        <0>
```

(b) 감자가 6개 있는 농장

방문 횟수는 출구쪽에서는 벽이 없는 것마냥 증가하며, 벽 건너편에서는 도달 불가능하기에 0이 됩니다.

돌을 처리하는 방법도 다양합니다.

- ✓ 돌 사이에 있는지 아닌지는 전처리를 통해 판별 가능합니다.
- ✓ 돌이 있어도 여러 종류의 부분합을 이용하여 탈출 시간을 계산할 수 있습니다. 돌의 위치와 탈출 방향, 감자 개수의 홀짝성 등을 따져서 만들어줄 수 있습니다.
- ✓ 돌을 간격 0 짜리 빈 칸이 10^6 개 있는 환경으로 생각해볼 수 있습니다.
  - 돌이 없는 농장의 풀이를 그대로 쓰면 탈출 시간을 구할 수 있습니다.
  - 가장 좌측/우측의 돌만 바꾸면 되므로, 칸 수는 최대 3×10^6 언저리가 됩니다.
  - 수확한 감자의 개수는 따로 구해줄 수 있습니다.

결론적으로 돌이 있어도 전처리 $\mathcal{O}(N)$, 쿼리당 $\mathcal{O}(1)$ 에 구할 수 있습니다.

- 검수 과정에서 segment tree를 이용해 농장을 갱신 가능한 $\mathcal{O}((N+Q)\lg N)$ 풀이도 발견되었으며, 이를 통해 해결한 팀도 있었습니다.
- 쿼리가 몇 번째 감자 사이에 있는지를 이분 탐색으로 구한 팀이 많았습니다.

F. 전투 시뮬레이션

`dijkstra`

출제진 의도 – **Medium**

- 제출 241회, 정답 21팀 (정답률 9.129%)
- 출제자: `ckdgus2482`

- **그래프 탐색**이지만 **문해력**을 요하는 문제입니다.
- DFS 탐색을 기반으로 적절한 휴리스틱 알고리즘과 가지치기를 통해 빠른 탐색 알고리즘을 만들 수 있지만 알고리즘의 정당성을 보장하기가 어렵고 구현 난이도가 높은 풀이입니다.

- 보다 간단한 풀이는 **다익스트라** 알고리즘을 이용한 풀이입니다. 다익스트라는 출발지로부터 경로의 비용이 적게 드는 정점부터 차례로 방문합니다. 따라서 이동력을 초과하는 순간 그 이후의 정점들은 탐색하지 않더라도 도달할 수 없는 정점이라는 것을 알 수 있습니다.
- 이 문제에서는 이동 가능한 모든 지형이 최소 1 의 험준도를 가지며, 유닛의 최대 이동력은 20 입니다.
- 따라서 최악의 경우라도 택시거리가 20 이하인 정점들만 방문하게 됩니다. 마찬가지로 다익스트라 알고리즘을 위해 최단거리를 무한대로 초기화하는 전처리 또한 택시거리 이내의 정점들만 해주면 됩니다.

문제를 잘 읽고 구현시 유의할 점

- 약진 명령의 목적지가 이동불가 지형이거나 다른 유닛 (세력에 관계 없이 모든 유닛을 의미) 이 존재한다면 수행 불가능한 명령입니다.
- 약진 중에는 같은 세력의 유닛을 통과하여 지나갈 수 있습니다.
- 약진 중에 다른 세력의 유닛과 인접하는 순간 교전이 발생하므로 약진 목적지가 아닌 위치에서 교전이 발생하는 명령은 수행 불가능한 명령입니다.
- 교전은 이미 같은 세력의 유닛이 위치하고 있는 정점에서도 새로 발생할 수 있습니다. 이는 대회중 질의응답을 통해 공개된 내용입니다. 즉 교전 발생 여부를 확인할 때 상대 유닛이 이미 교전중인지 확인할 필요는 없습니다.

- 또 다른 괜찮은 풀이 방법이 있습니다. 01-BFS 알고리즘을 응용한 풀이로 5개의 큐를 만들고 각각의 큐를 탐색중인 경로의 비용으로부터 추가적으로 필요한 코스트별로 그룹화하여 관리하는 방법입니다.
- 험준도가 1부터 4까지 4종류이므로 4그룹이 필요하고 현재 탐색중인 노드와 같은 비용을 가지는 노드들의 그룹 때문에 1그룹이 필요합니다.
- 이렇게 하면 일반적인 BFS탐색과 같이 상수시간 복잡도로 큐에서 최소 비용 노드를 가져올 수 있습니다. 최종 목적지에 도달 가능한지만 관심이 있으므로 다익스트라와 같이 탐색 대상 노드들의 최단거리를 저장할 필요가 없습니다.

G. 루머

`bfs`

출제진 의도 – **Medium**

- 제출 817회, 정답 209팀 (정답률 26.93%)
- 출제자 : QuqqU

시간 제한 10초! 메모리 제한 1GB! 최대 입력 파일 10MB!
아주 무자비한 문제처럼 보이지만, 간단한 문제입니다

$O(n^2)$ 보다 빠른 시간으로 어떻게든 풀면 됩니다!

여기서는 $O(n)$ 풀이법을 하나 소개하겠습니다.

먼저 BFS의 정의를 살짝 바꾸어 봅니다.

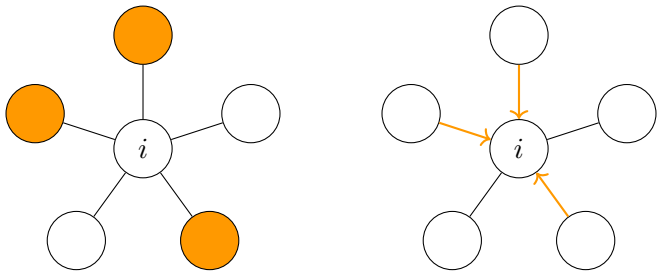
- ✓ Ordinary BFS : 정점 i에 대해 1회 발견 → 정점 i 추후 방문 (Queue에 정점 i 삽입)
- ✓ Modified BFS : 정점 i에 대해 $*$회 발견 → 정점 i 추후 방문 (Queue에 정점 i 삽입)

이제 Modified BFS를 문제에 적용합니다.

G. 루머

아래의 두 명제는 동치입니다.

- ✓ 정점 i의 주변 정점 절반 이상이 루머를 믿는다
- ✓ Modified BFS 탐색 중 정점 i를 인접 간선을 통해 C_i번 이상 발견했다 ($C_i = \left\lceil \frac{\deg(i)}{2} \right\rceil$)



G. 루머

Modified BFS에서 C_i 번 발견한 정점을 Queue에 삽입 ($C_i = \left\lceil \frac{\deg(i)}{2} \right\rceil$)

Queue에 삽입한 정점은?

✓ 주변 정점들이 절반 이상 루머를 믿는 정점

계속 그래프 탐색 (Modified BFS) 을 진행하다,
탐색이 끝나고도 방문 하지 않은 정점은 **-1**을 출력하면 됩니다.

H. 사과나무

greedy

출제진 의도 – **Easy**

- 제출 1,166회, 정답 217팀 (정답률 18.87%)
- 출제자 – evenharder

물뿌리개로 만들 수 있는 배치는 '높이의 합이 3의 배수 ($3x$) 이고, 각 높이를 2로 나눈 몫의 합이 x 이상인 배치'와 동치입니다.

- ✓ ($\Rightarrow$) 수학적 귀납법을 이용하면 증명할 수 있습니다.
- ✓ ($\Leftarrow$) 높이를 2로 나눈 몫의 합이 x 이상이므로, 2짜리 물뿌리개를 x번 사용 가능합니다. 이후, 남은 높이만큼 x번 1짜리 물뿌리개를 사용하면 원하는 배치를 만들 수 있습니다.

그러므로 N개의 높이를 입력받은 후 위 조건을 만족하면 "YES"를, 아니면 "NO"를 출력하면 됩니다.

I. 인버스 ㄷㄷㄷㅈ

`constructive`

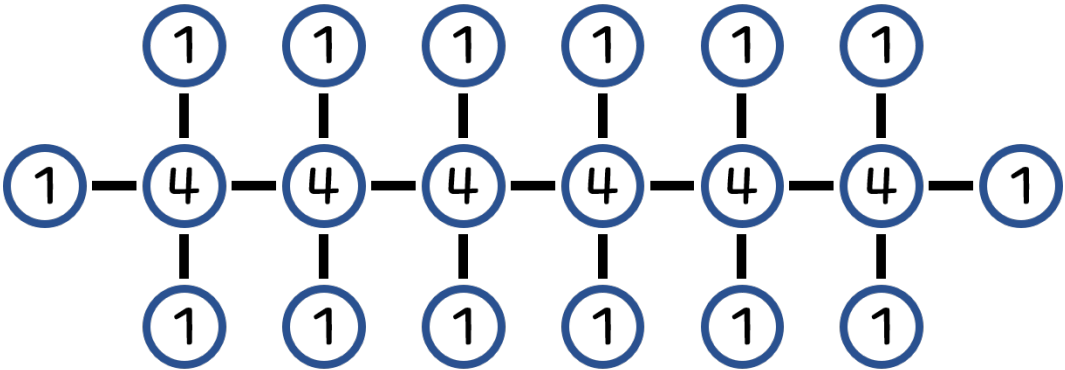
출제진 의도 – **Hard**

- 제출 78, 정답 24팀 (정답률 30.77%)
- 출제자: `16silver`(preparation) + `tncks0121`, `kdh9949`(idea)

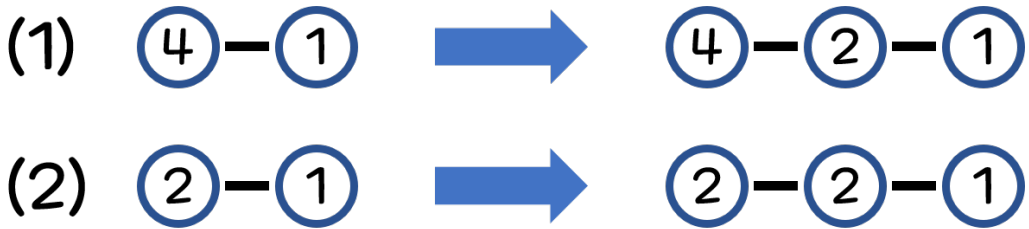
- 어떤 방식으로든 DUDUDUNGA-트리를 만들면 됩니다!
- 출제진 쪽에서도 다양한 형태의 트리가 나왔습니다.

  한 가지 풀이 (by 16silver)

- 차수가 4인 정점 x개가 일렬로 연결된 그래프를 생각합니다. (나머지 정점들의 차수는 모두 1)



- 이 그래프에 차수가 2인 정점들을 추가합니다. 추가 방법은 다음의 두 가지만을 사용합니다.



- 처음 그래프의 정점의 개수는 $3x + 2$개, $3ㅈ - ㄷ$ 의 값은 $3x + 9$입니다.
- (1)은 $3ㅈ - ㄷ$을 3 줄이며, (2)는 $3ㅈ - ㄷ$을 1 줄입니다.
- (2)를 사용하기 위해서는 (1)을 최소 한 번은 써야 합니다.
- (1)만 사용하여 정점을 추가하면 $x + 3$개의 정점을 추가, 최종 정점 개수는 $4x + 5$입니다.
- (1)을 한 번만 사용하면 최종 정점 개수는 $6x + 9$입니다.
- $4x + 5$와 $6x + 9$ 사이에 있는 모든 홀수 케이스를 이 방법으로 해결할 수 있습니다.
- x를 잘 조절하면 9 이상의 모든 홀수 $2k + 1$에 대해 정점이 $2k + 1$개인 DUDUDUNGA-트리를 만들 수 있습니다!

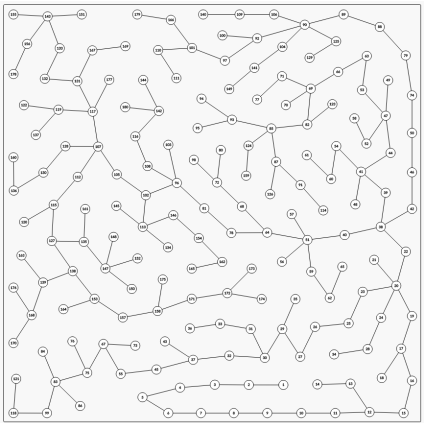
- 짝수인 경우에는 차수가 3인 점 하나를 끝에 연결해놓고 시작하면 됩니다.
- 처음 그래프의 정점의 개수는 $3x+4$개, 3ㅈ $-$ ㄷ 의 값은 $3x+6$입니다.
- (1) 만 사용하여 정점을 추가하면 $x+2$개의 정점을 추가, 최종 정점 개수는 $4x+6$입니다.
- (1) 을 한 번만 사용하면 최종 정점 개수는 $6x+8$입니다.
- x를 잘 조절하면 10 이상의 모든 짝수 $2k$에 대해 정점이 $2k$개인 DUDUDUNGA-트리를 만들 수 있습니다!

- $N = 6, 7, 8$일 때의 트리는 직접 만듭니다.
- 사실 대부분의 풀이에서 N이 작은 경우의 트리를 따로 만들어야 합니다.
- 랜덤으로 트리를 만들었을 때 생각보다 DUDUDUNGA-트리가 잘 나옵니다!
- 랜덤을 잘 만들면 큰 N에 대해서도 답을 잘 냈지만, 랜덤만으로 문제를 푸는 것은 매우 어려웠을 것입니다.

이외에도 기본 형태의 트리를 두고 정점을 k개씩 추가하는 형태의 풀이들이 있었습니다.

I. 인버스 ㄷㄷㄷㅈ

$N = 180$인 DUDUDUNGA-트리 중 하나



J. 역학 조사

`implementation`
출제진 의도 – **Medium**

- 제출 730회, 정답 93팀 (정답률 12.88%)
- 출제자 – `functionx`

J. 역학 조사

- 처음에 한 명이 감염되었다면 일련의 모임을 통해 전염병이 퍼집니다.
- 처음에 i만 감염되었을 때 마지막에 감염된 사람의 정보 $F[i]$를 비트로 생각합니다.

- 처음에 i, j 가 감염되었다면 마지막에 감염된 사람은 $F[i] \vee F[j]$ 입니다.
- 처음에 $i_1, ..., i_N$ 이 감염되었다면 마지막에 감염된 사람은 $F[i_1] \vee ... \vee F[i_N]$ 입니다.

- 초기 감염자가 될 수 없는 사람을 전부 제거합니다.
- 나머지 사람들을 전부 감염시킨 다음 예상대로 나오는지 체크하면 됩니다.

- 초기 감염자가 될 수 없는 사람을 모임 역순으로 보면서 계산합니다.
- 각 모임에 비감염자가 한 명이라도 있다면, 모임 시작 전엔 감염자가 있을 수 없습니다.
- 첫 번째 모임까지 올라오면, 초기 감염자의 후보를 구할 수 있습니다.

- 모임이 전부 끝난 후 감염자인 사람을 '최종 감염자', 아닌 사람을 '최종 비감염자'라 할 때, 초기 감염자 후보들을 전부 초기 감염자라 생각해도 최종 비감염자가 감염자가 되는 일은 없습니다.
- 이를 통해 만약 감염자가 된다면, 최종 비감염자와 초기 감염자를 연결하는 모임의 집합이 있기 때문에 모순이기 때문입니다.
- 그러므로 초기 감염자 선택 유무는 최종 감염자의 감염 여부에만 영향을 미칩니다.

- 때문에 초기 감염자 후보를 전부 초기 감염자라 하고, 모임 순서대로 시뮬레이션 하면 됩니다.
- 감염 과정 특성상, 후보를 전부 선택했는데도 최종 감염자를 전부 감염시킬 수 없으면 더 적은 인원으로도 할 수 없습니다.
- 총 시간복잡도는 $\mathcal{O}\left(N + M + \sum k_i\right)$ 입니다.