Homework #3

Due: 11/28

HW3

이번 과제는 Graph, DP 관련 5개의 코딩 문제로 구성되어 있습니다. **(Total 100 points)**
자동 채점으로 점수를 집계하기에, 사소한 실수로 정답이 오답 처리되는 안타까운 경우가 잦습니다.
아래 주의사항들과, 별도 첨부 파일 **CodingAssignmentGuidelines.pdf** 에 명시되어 있는
주의사항들을 꼭 숙지해주세요.

Submission Format

모든 제출 파일은 **학번_문제번호.py** 로 작성해주시고, 하나의 **HW3_학번.zip** 파일로 묶어서 제출해 주세요. 코드는 Python(.py extension)으로 작성하시기 바랍니다.

파일 1: 학번_1.py
파일 2: 학번_2.py
파일 3: 학번_3.py
파일 4: 학번_4.py
파일 5: 학번_5.py

→ **HW3_학번.zip** 압축 후 제출

(**파일명이 다르면 채점이 안됩니다!**)

Recursion Limit

Python 에서는 재귀호출의 default depth 를 1,000 번으로 제한하고 있습니다. 이로 인해, 재귀호출 횟수를 초과하여 에러가 발생할 수 있습니다. 본인의 코드를 재귀함수로 구현한다면, sys 모듈을 이용해 아래와 같이 depth 를 1,000,000 으로 넉넉하게 설정해주세요.

```
import sys
sys.setrecursionlimit(10**6)
```

Note That

HW3 는, 사용 권장한 sys 모듈 외에는 다른 어떠한 import 도 허용하지 않습니다. 이는 특정 자료구조에서 오는 시간적 이점을 최소화하고, 독립적인 알고리즘 설계를 장려하기 위한 목적입니다. 텍스트 검사로 **sys 외의 다른 import 가 발견되면 해당 문제는 0 점 처리**할 예정입니다.

1. **달구의 회로 실습 (20 points)**

디지털 논리 회로의 과제로 독창적인 회로를 설계해보는 과제가 나왔다.

달구는 기본 연산 회로 두 개에 특별 회로 하나를 추가해서 계산기를 만들었다.

계산기를 작동시키면 화면에는 1 이 표시되고, 화면은 **최대 5 자리의 정수만 표현**할 수 있다.

각 버튼은 아래와 같은 기능을 가지고 있다.

Mul: 현재까지의 계산 결과에 2 를 곱한다.

Div: 현재까지의 계산 결과를 3 으로 나누고, 정수가 아니면 소수점 이하를 버린다.

Special: 현재까지의 계산 결과를 뒤집는다. (1 → 1 / 120 → 21 / 9806 → 6089)

달구는 이 계산기를 이용하여 정확히 숫자 N 을 만들고자 한다. **계산기의 세 버튼을 사용하여 숫자 N 을 만들 수 있는지 판별하고, 가능하다면 필요한 최소 조작 횟수를 계산하는 프로그램을 작성하시오.**

Input Format

첫째 줄에 달구가 만들고자 하는 정수 N 이 주어진다. ($1 \le N \le 99{,}999$)

Output Format

첫째 줄에 N 을 만들기 위해 계산기 버튼을 누른 최소 횟수를 출력한다.

만약 N 을 만들 수 없다면 -1 을 출력한다.

Sample Input

```
3
```

Sample Output

```
7
```

그래프 탐색 알고리즘 추천 / timeout per a testcase = 1.64s

2. 달구의 심화 학습 (20 points)

과제 족보를 열심히 살피던 달구는 같은 문제가 없어서 실망했다. 그래서 올해는 스스로 문제를 해결하기로 결심했다. 실력을 늘리기 위해 달구는 알고리즘 문제집을 꺼내 들었고, 다음과 같은 문제를 발견했다.

> 양의 정수 N 이 주어졌을 때, N 을 1, 3, 5 의 합으로 표현하는 방법의 수를 구하라.

달구는 이 문제를 쉽게 해결했고, 더 복잡한 문제에 도전하고 싶어졌다.

그래서 문제를 다음과 같이 변형하였다.

> **양의 정수 N 이 주어졌을 때, N 을 1, 3, 5 의 합으로 표현하는 방법 중에서, 연속해서 같은 숫자가 오는 경우가 없는 표현 방법의 수를 구하라.**

예를 들어, N=6 일 때, (1+5)와 (5+1)은 가능하지만, (3+3)은 연속해서 같은 숫자가 두 번 나오므로 불가능하다.

Input Format

> 첫째 줄에 양의 정수 N 이 주어진다. ($1 \le N \le 100,000$)

Output Format

> 첫째 줄에 (연속해서 같은 숫자가 없고, N 을 1,3,5 의 합으로 표현)하는 방법의 수를 출력한다.
>
> 만약 N 을 표현할 수 없다면 -1 을 출력한다.

Sample Input

```
6
```

Sample Output

```
2
```

DP 추천 / timeout per a testcase = 4.12s

3. **보노의 도주를 도와라 (20 points)**

달구의 절친 보노는 누명을 썼고, 현풍에서 경찰에게 쫓기고 있다.

현풍을 node 와 edge 로 구성된 **undirected weighted graph** 로 매핑할 수 있다고 가정하자.

현풍은 총 n 개의 node 와 e 개의 edge 로 이루어져 있으며, 각 node 는 1 부터 n 까지의 번호로 식별된다. 보노는 node 1 에서 시작하여 node n 으로 가야 한다.

(제약 조건) 그러나, 경찰은 그 중 k 개의 node 에 순찰차를 배치하여 보노가 그 node 를 통과하는 것을 방지하고 있다. 순찰차가 배치된 node 를 통과하는 것은 불가능하다. 단, node 1 과 node n 에는 순찰차가 없다고 가정한다.

각 edge 는 양방향으로 두 node 를 연결하며, 임의의 두 node 사이 edge 는 0 개 또는 1 개이다. edge 의 weight 는 해당 edge 를 통과하는데 걸리는 시간을 의미한다.

달구는 node 1 에서 node n 까지 순찰차를 피해 도달할 수 있는 가장 빠른 경로의 시간을 출력하는 프로그램을 작성하여 보노의 도주를 도우려 한다.

Input Format

첫째 줄에는 node 의 개수 n 과 edge 의 개수 e, 순찰차의 개수 k 가 공백으로 구분되어서 주어진다.

각 node 의 이름은 1 부터 n 까지의 정수로 매핑한다. ($2 \le n \le 1{,}000$, $0 \le e \le \frac{n*(n-1)}{2}$, $0 \le k \le n-2$)

그 아래 e 개의 줄에는 edge 의 정보가 담겨있는데, a b w 가 공백으로 구분되어서 주어진다.

e 개의 a b w 쌍은 distinct 하다. (a < b, w = edge [node a - node b]의 weight) ($1 \le w \le 100$)

다음 k 개의 줄에 걸쳐 순찰차가 배치된 node 번호 p 가 주어진다. ($1 \le p \le n$, $p \ne 1$, $p \ne n$)

정리하면, **총 (1+e+k)개 줄**이 input 으로 들어간다. (Input 파일에 있는 모든 숫자들은 정수다.)

Output Format

첫째 줄에 보노가 node 1 에서 node n 까지 순찰차를 피해 도달할 수 있는 가장 빠른 경로의 시간을 출력한다. 만약 보노가 node n 에 도달할 수 없다면 -1 을 출력한다.

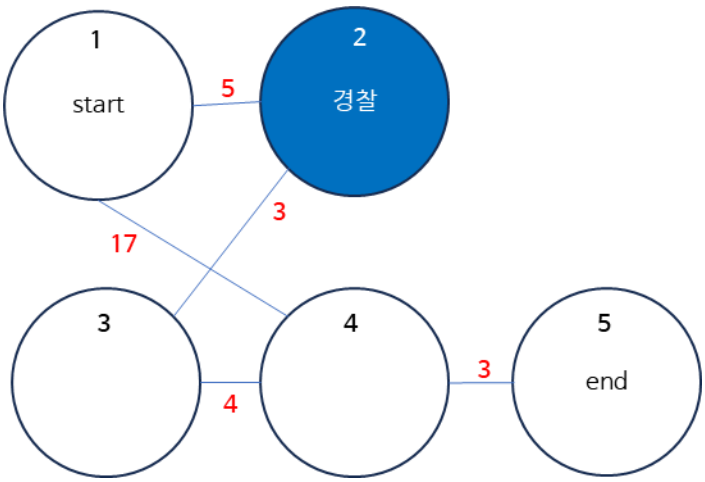
Sample Input

```
5 5 1
1 2 5
2 3 3
3 4 4
1 4 17
4 5 3
2
```

Sample Output

```
20
```

이해를 돕기 위해, 위 sample 은 아래와 같이 그려볼 수 있다.



그래프 탐색 알고리즘 추천 / timeout per a testcase = 5.80s

4. **숲의 먹이사슬 (20 points)**

3 번 프로그램은 최단경로의 소요시간만 출력한다. 길을 모르는 보노는 결국 잡혔다…

다행히 보노의 누명은 풀렸지만 인생 덧없음을 느꼈고, 귀농하여 자신만의 목장을 꾸리려 한다.

자본금이 부족했던 보노는 인적이 드문 숲으로 들어가 야생 양을 모아오려고 한다.

숲은 **바이너리 트리**로 표현할 수 있다. 숲에는 양, 늑대, 호랑이가 먹이사슬을 이루어 살고 있다. 트리의 각 노드는 **한 마리의 양** 또는 **한 마리의 늑대** 또는 **한 마리의 호랑이**가 살고 있다.

보노는 **루트 노드에서 출발**한다. **루트 노드에는 항상 양**이 살고 있다. 양이 사는 노드를 방문할 때, 그 양은 보노의 뒤로 일렬로 줄지어 합류한다. 하지만, 육식동물이 속한 노드에 방문하면, 아래의 이벤트가 발생한다.

> 늑대: 줄의 맨 뒤에 합류한다. 이 때, (줄에 속한) 양의 수보다 늑대의 수가 같거나 더 많아지면, 다른 늑대들과 협동하여 바로 모든 양을 잡아먹는다.
>
> 호랑이: 겨울잠에 들어 깨지 않는다. 즉, 줄에 합류하지 않는다.
>
> 호랑이는 숲의 우두머리라서, 숲에는 항상 정확히 한 마리의 호랑이가 존재하고, 호랑이가 존재하는 노드는 항상 leaf 노드다.
>
> 아무도 없는 노드(한번 방문한 노드)에 재방문하는 것은 제한이 없다.

보노의 목표는 1) **양이 늑대에게 잡아먹히는 일은 절대 없도록 하면서,** 2) **최대한 많은 수의 양을 모아서 루트 노드로 돌아오는 것**이다.

바이너리 트리의 루트 노드에서 출발한 보노가 위의 두 조건을 모두 만족하며 모을 수 있는 양은 최대 몇 마리인지 구하는 프로그램을 작성하시오.

Input Format

첫째 줄에는 트리의 노드 개수 n 이 주어진다. (1 ≤ n ≤ 100,000, 루트 노드 포함)

각 노드의 이름은 1 부터 n 까지의 정수로 매핑한다. 노드 1 은 루트 노드다.

둘째 줄에는 트리의 각 노드에 있는 동물의 종류를 나타내는 숫자 n 개가 노드 순서대로 공백으로 구분되어 주어진다. 숫자는 "0" (양), "1" (늑대), "2" (호랑이) 중 하나이다.

루트 노드에는 항상 양이 있으므로 둘째 줄의 첫 번째 숫자는 항상 “0”이다.

호랑이는 항상 정확히 한 마리 존재하므로 둘째 줄에서 "2"는 단 한 번 나타난다.

셋째 줄부터는 **n-1** 개의 줄에 걸쳐서 트리의 간선 정보가 주어진다. 각 줄에는 부모 노드 번호와 자식 노드 번호가 공백으로 구분되어 주어진다. (1 ≤ 부모 노드 번호, 자식 노드 번호 ≤ n)

정리하면, **총 (n+1)개 줄**이 input 으로 들어간다. (Input 파일에 있는 모든 숫자들은 정수다.)

Output Format

첫째 줄에 보노가 모을 수 있는 양의 최대 수를 출력한다.

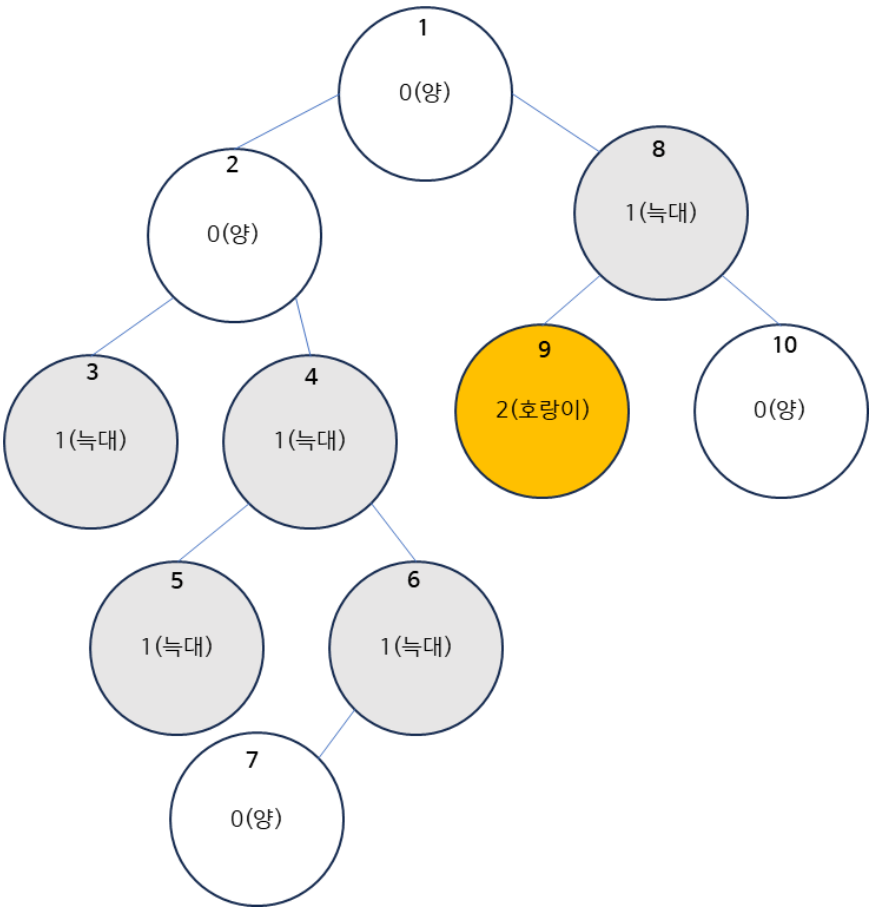
Sample Input

```
10
0 0 1 1 1 1 0 1 2 0
1 2
2 3
2 4
4 5
4 6
6 7
1 8
8 9
8 10
```

Sample Output

```
3
```

이해를 돕기 위해, 위 sample 은 아래와 같이 그려볼 수 있다.



그래프 탐색 알고리즘 추천 / timeout per a testcase = 3.39s

5. 양들의 이동 (20 points)

보노는 양 목장을 운영하게 되었다. 그런데 최근 양들 사이에 전염병이 돌기 시작했다. 보노는 전염병의 확산을 막기 위해 N 개의 우리에 있는 양들을 관리해야 한다. 각 우리에는 여러 마리의 양이 있을 수 있고, 양들은 우리 간에 이동이 가능하다. 최근 한 우리에서 시작된 전염병이 다른 우리로 퍼지고 있고, 전염병이 더 확산되기 전에 하루라도 빨리 건강한 양들을 특정 "안전 우리"로 이동시켜야 한다. 이 "안전 우리"는 전염병으로부터 보호된 상태로 갖춰진 특수 설비를 갖춘 우리로 정확히 하나 존재한다. 보노의 목표는 모든 건강한 양들을 "안전 우리"로 가능한 한 빨리 이동시키는 것이다. 양들의 이동은 아래 조건을 만족한다.

1. 전염병에 걸린 우리의 양들은 다른 우리로 이동할 수 없다.
2. 전염병이 없는 양들만 "안전 우리"로 이동할 수 있다.
3. 우리 간 이동에는 일정한 시간이 소요된다.

“안전 우리”로 이동할 수 있는 우리들은 정해져 있다. 해당 우리들에 속한 건강한 양들이 “안전 우리”까지 이동하는데 걸리는 최소 시간의 합을 구하는 프로그램을 작성하시오.

Input Format

첫째 줄에는 우리의 수 N($1 \le N \le 100$)과 이동 가능한 우리 쌍의 수 M($1 \le M \le N(N-1)/2$)이 공백으로 구분되어 주어진다. 이 때, 각 우리의 번호는 1 부터 N 까지 부여된다.

다음 M 개의 줄에 걸쳐, 각 줄에 세 정수 A, B, T 가 주어진다. 이는 우리 A 에서 우리 B 로 양을 이동시키는 데 T 시간이 걸린다는 것을 의미한다 ($1 \le A, B \le N$, $1 \le T \le 100$).

다음 줄에는 전염병에 걸린 우리의 수 K($0 \le K < N$)가 주어진다.

그 다음 줄에는 그 우리들의 번호가 공백으로 구분되어 주어진다.

그 다음 줄에는 "안전 우리"의 번호가 주어진다.

Output Format

"안전 우리"로 이동할 수 있는 모든 건강한 우리 출신 양들을 “안전 우리”로 이동시키는 데 필요한 최소 시간의 합을 출력한다.

만약 어떠한 양도 "안전 우리"로 이동시킬 수 없다면 -1 을 출력한다.

이 때, “안전 우리”에 처음부터 속한 양들은 고려하지 않는다.

Sample Input

```
5 6
1 2 5
2 3 2
3 4 3
4 5 4
5 1 1
1 3 3
2
3 5
1
```

Sample Output

```
5
```

위 예시에서, 우리 3 과 5 는 전염병으로 인해 이동이 불가능하다. "안전 우리"는 1 로 지정됐다. 그렇기 때문에, 우리 1 에 이미 있는 양들은 이동할 필요가 없어, 고려 대상에서 제외한다. 우리 2 에서 우리 1 로의 최소 이동 시간은 5 이고다. 우리 4 에서 우리 1 로 직접적인 경로는 없고, 우리 4 -> 우리 5 -> 우리 1 로 갈 수 있으나 우리 5 가 전염병에 걸렸기 때문에 이 경로는 사용할 수 없다. 결과적으로 안전 우리로 이동할 수 있는 우리들의 최소 이동 시간의 합은 5 가 된다.

(DP + 그래프 탐색 알고리즘) 추천 / timeout per a testcase = 2.53s