B. ქალაქების გაცვლა

Time limit	2 s
Memory limit	512 MB

აღწერა

ინდონეზიაში N ქალაქია, გადანომრილი 0_დან $N-1$_მდე. ქალაქებს აკავშირებს M ცალი ორმხრივი გზა, გადანომრილი 0_დან $M-1$_მდე. თითოეული გზა ორ განსხვავებულ ქალაქს აკავშირებს. i-ური გზა აკავშირებს $U[i]$-ურ და $V[i]$-ურ ქალაქებს და მოითხოვს $W[i]$ ერთეულ გაზს მანქანით გავლისას. ქალაქები ისეა დაკავშირებული, რომ ნებისმიერ ორ ქალაქს შორის მგზავრობაა შესაძლებელი.

შემდგომი Q დღიდან თითოეულს, ქალაქების რომელიღაც წყვილს სურს პოლიტიკური ურთიერთობის ჩამოყალიბება. კერძოდ, j-ურ დღეს, $X[j]$-ურ ქალაქს სურს $Y[j]$-ურ ქალაქთან პოლიტიკური ურთიერთობის ჩამოყალიბება. ამის გასაკეთებლად, $X[j]$-ურმა ქალაქმა უნდა გაგზავნოს წარმომადგენელი $Y[j]$-ურ ქალაქში მანქანით. ანალოგიურად, $Y[j]$-ურმა ქალაქმაც უნდა გაგზავნოს წარმომადგენელი $X[j]$-ურ ქალაქში.

დაჯახების თავიდან ასაცილებლად ეს მანქანები ერთმანეთს არ უნდა შეხვდნენ. კერძოდ, ეს ორი მანქანა არ უნდა იმყოფებოდეს ერთი და იგივე ქალაქში ერთდროულად დროის არცერთ მომენტში და მათ არ უნდა გაიარონ ერთი და იგივე გზა საპირისპირო მიმართულებით ერთდროილად. დამატებით, მანქანებმა დაწყებული გზა უნდა დაასრულონ და დანიშნულების ქალაქამდე მივიდნენ(ანუ ქალაქებს შორის შუა გზაზე ვერ მოტრიალდებიან). მანქანებს აქვთ უფლება გაიარონ ერთი და იგივე ქალაქში ან გზაზე რამდენჯერმე. მათ ასევე შეუძლიათ მოიცადონ ნებისმიერ ქალაქში დროის ნებისმიერ მომენტში.

რადგან მაღალი საწვავის ტევადობის მანქანები ძვირია, ქალაქებს სურთ აირჩიონ მანქანების მარშრუტები ისე, რომ მაქსიმალური საწვავის ტევადობა იყოს მინიმალური. თითოეულ ქალაქში არის გაზგასამართი სადგური, ამიტომ საჭირო ტევადობა არის მარშრუტის გზებს შორის არსებული მაქსიმალური მოთხოვნის ტოლი.

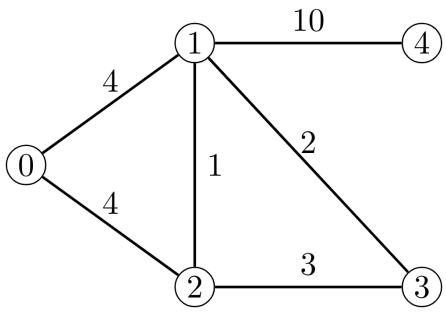
ამოცანა

თქვენ უნდა მოახდინოთ `init` და `getMinimumFuelCapacity` ფუნქციების იმპლემენტაცია.

- `init(N, M, U, V, W)` - ეს ფუნქცია გამოიძახება გრადერის მიერ ზუსტად ერთხელ `getMinimumFuelCapacity` ფუნქციის ყოველი გამოძახების წინ.
  - N: მთელი რიცხვი, რომელიც წარმოადგენს ქალაქების რაოდენობას..
  - M: მთელი რიცხვი, რომელიც წარმოადგენს გზების რაოდენობას..
  - $U: M$ მთელი რიცხვებისაგან შედგენილი მასივი, რომელიც აღწერს გზების ერთ ბოლოს.
  - $V: M$ მთელი რიცხვებისაგან შედგენილი მასივი, რომელიც აღწერს გზების მეორე ბოლოს.
  - $W: M$ მთელი რიცხვებისაგან შედგენილი მასივი, რომელიც აღწერს შესაბამის გზაზე გაზის ხარჯს.
- `getMinimumFuelCapacity(X, Y)` - ეს ფუნქცია გამოიძახება გრადერის მიერ Q-ჯერ.
  - X: მთელი რიცხვი, რომელიც წარმოადგენს პირველ ქალაქს.
  - Y: მთელი რიცხვი, რომელიც წარმოადგენს მეორე ქალაქს.
  - ამ ფუნქციამ უნდა დააბრუნოს ინტეჯერი, რომელიც წარმოადგენს მინიმალურ შესაძლო მაქსიმუმს ორი მანქანის ტევადობებისა, ისე, რომ X-ური ქალაქის წარმომადგენელს შეუძლია წავიდეს Y-ურ ქალაქში და Y-ური ქალაქის წარმომადგენელს შეუძლია წავიდეს X-ურ ქალაქში მოცემული პირობების დაცვით, ან დააბრუნოს -1 თუ ეს შეუძლებელია.

მაგალითი

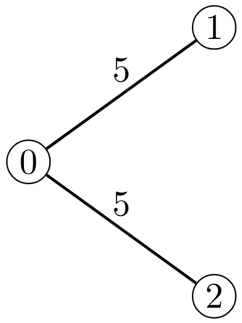
პირველ მაგალითში, $N = 5$, $M = 6$, $U = [0, 0, 1, 1, 1, 2]$, $V = [1, 2, 2, 3, 4, 3]$, $W = [4, 4, 1, 2, 10, 3]$, $Q = 3$, $X = [1, 2, 0]$, $Y = [2, 4, 1]$. მაგალითი ილუსტრირებულია ნახატზე:



`გრადერი თავიდან გამოიძახებს init(5, 6, [0, 0, 1, 1, 1, 2], [1, 2, 2, 3, 4, 3], [4, 4, 1, 2, 10, 3])`. ამის შემდეგ გრადერი გამოიძახებს:

- `getMinimumFuelCapacity(1, 2)`. ჯერ, მანქანას პირველი ქალაქიდან შეუძლია წავიდეს მესამე ქალაქში. შემდეგ, მანქანას მეორე ქალაქიდან შეუძლია წავიდეს პირველ ქალაქში და მანქანა მესამე ქალაქიდან წავიდესმეორე ქალაქში. შესაბამისად, მაქსიმალური საჭირო ტევადობა ამ ორი მანქანის არის 3 ერთეული (საჭირო მესამედან ეორე ქალაქში ჩასასვლელად.) არ არსებობს მარშრუტი რომელსაც ნაკლები ტევადობა სჭირდება, ანუ ფუნქციამ უნდა დააბრუნოს 3.
- `getMinimumFuelCapacity(2, 4)`. ნებისმიერ მანქანას, რომელიც შედის ან გამოდის მეოთხე ქალაქში დასჭირდება 10 ერთეული საწვავის ტევადობა, ანუ ფუნქციამ უნდა დააბრუნოს 10.
- `getMinimumFuelCapacity(0, 1)`. ფუნქციამ უნდა დააბრუნოს 4.

მეორე მაგალითში, $N = 3$, $M = 2$, $U = [0, 0]$, $V = [1, 2]$, $W = [5, 5]$, $Q = 1$, $X = [1]$, $Y = [2]$. მაგალითი გამოსახულია სურათზე:



`გრადერი გამოიძახებს init(3, 2, [0, 0], [1, 2], [5, 5])`. ამის შემდეგ გრადერი გამოიძახებს:

- `getMinimumFuelCapacity(1, 2)`. შეუძლებელია მანქანა პირველი ქალაქიდან ჩავიდეს მეორეში, ისე, რომ მეორე მანქანას არ შეხვდეს გზაში, ამიტომ -1.

შეზღუდვები

- $2 \leq N \leq 100\,000$.
- $N - 1 \leq M \leq 200\,000$.
- $0 \leq U[i] < V[i] < N$.
- ქალაქთა ყოველ წყვილისათვის არსებობს არაუმეტეს ერთი გზისა.

- გზების საშუალებით შესაძლებელია ნებისმიერი ქალაქიდან ნებისმიერ ქალაქში მისვლა.
- $1 \leq W[i] \leq 10^9$.
- $1 \leq Q \leq 200\,000$.
- $0 \leq X[j] < Y[j] < N$.

ქვეამოცანა 1 (6 ქულა)

- ყოველი ქალაქი წარმოადგენს ბოლო წერტილს არაუმეტეს ორი გზისთვის.

ქვეამოცანა 2 (7 ქულა)

- $M = N - 1$.
- $U[i] = 0$.

ქვეამოცანა 3 (17 ქულა)

- $Q \leq 5$.
- $N \leq 1\,000$.
- $M \leq 2\,000$.

ქვეამოცანა 4 (20 ქულა)

- $Q \leq 5$.

ქვეამოცანა 5 (23 ქულა)

- $M = N - 1$.

ქვეამოცანა 6 (27 ქულა

- დამატებითი შეზღუდვების გარეშე.

სანიმუშო გრადერი

სანიმუშო გრადერი კითხულობს შესატან მონაცემებს შემდეგი ფორმატით:

```
N M
U[0] V[0] W[0]
U[1] V[1] W[1]
.
.
.
U[M-1] V[M-1] W[M-1]
Q
X[0] Y[0]
X[1] Y[1]
.
.
.
X[Q-1] Y[Q-1]
```

თითოეული `getMinimumFuelCapacity` გამოძახებისთვის, სანიმუშო გრადერი დაბეჭდავს ფუნქციის მიერ დაბრუნებულ მნიშვნელობას.