საგზაო ხარჯები

იაპონიაში ქალაქები ერთმანეთთან დაკავშირებულია გზატკეცილთა ქსელით. ეს ქსელი შეიცავს N ქალაქსა და M გზატკეცილს. ყოველი გზატკეცილი აკავშირებს ორ განსხვავებულ ქალაქს. ქალაქების არცერთ წყვილს არ აერთებს ერთზე მეტი გზატკეცილი. ქალაქები გადანომრილია 0-დან $(N-1)$-მდე, ხოლო გზატკეცილები გადანომრილია 0-დან $(M-1)$-მდე. ნებისმიერ გზატკეცილზე მოძრაობა შესაძლებელია ორივე მიმართულებით. გზატკეცილთა საშუალებით შესაძლებელია ნებისმიერი ქალაქიდან ნებისმიერ სხვა ქალაქამდე მისვლა.

თითოეულ გზატკეცილზე გავლისათვის დაწესებულია გადასახადი. გადასახადი კონკრეტული გზატკეცილისათვის დამოკიდებულია **მოძრაობის პირობებზე** , რომელიც შეიძლება იყოს **მსუბუქი** ან **მძიმე**. თუ მოძრაობის პირობები მსუბუქია, მაშინ გადასახადი უტოლდება A იენს (იაპონური ფული). თუ მოძრაობის პირობები მძიმეა, მაშინ გადასახადის სიდიდე B იენია. გარანტირებულია, რომ $A < B$. მიაქციეთ ყურადღება, რომ A-ს და B-ს მნიშვნელობები თქვენთვის ცნობილია.

თქვენ გაქვთ მანქანა, რომელიც მოძრაობის მოცემული პირობებისათვის გამოითვლის უმცირეს მინიმალურ ხარჯს ქალაქთა ნებისმიერ S და T $(S \neq T)$ წყვილისათვის, ერთ-ერთი მათგანიდან მეორემდე მისასვლელად.

თუმცა მანქანა უბრალოდ პროტოტიპია. S-ის და T-ს მნიშვნელობები მასში დაფიქსირებულია (ანუ ჩაკოდირებულია მანქანაში), მაგრამ უცნობია თქვენთვის. თქვენი ამოცანაა განსაზღვროთ S და T. ამისათვის თქვენ უნდა დაგეგმოთ და მიაწოდოთ მანქანას მოძრაობის პირობების გარკვეული რაოდენობა და მის მიერ თითოეული შემთხვევისათვის დაბრუნებული მინიმალური ჯამური დანახარჯი გამოიყენოთ S-ის და T-ს გამოსაცნობად. ცხადია, თქვენ არ გსურთ, მანქანის ძალიან ბევრჯერ გამოყენება.

იმპლემენტაციის დეტალები

თქვენ იმპლემენტაცია უნდა გაუკეთოთ შემდეგ ფუნქციას:

```
find_pair(int N, int[] U, int[] V, int A, int B)
```

- `N`: ქალაქების რაოდენობა.
- `U` და `V`: M სიგრძის მასივები, სადაც M არის ქალაქთა დამაკავშირებელი გზატკეცილების რაოდენობა. ყოველი i-სათვის $(0 \leq i \leq M-1)$, i ნომრის მქონე გზატკეცილი აერთებს `U[i]` და `V[i]` ქალაქებს.

- A: გადასახადი იმ გზატკეცილებზე, სადაც მოძრაობის მსუბუქი პირობებია.
- B: გადასახადი იმ გზატკეცილებზე, სადაც მოძრაობის მძიმე პირობებია.
- ეს ფუნქცია გამოიძახება ზუსტად ერთხელ თითოეული ტესტისათვის.

- მიაქციეთ ყურადღება, რომ M მასივების სიგრძეა და მისი მნიშვნელობის მიღება შეიძლება ისე, როგორც ეს იმპლემენტაციით არის განსაზღვრული.

  find_pair ფუნქციამ შეიძლება გამოიძახოს შემდეგი ფუნქცია:

```
int64 ask(int[] w)
```

- w მასივის სიგრძე უნდა იყოს M. w აღწერს მოძრაობის პირობებს.
- ყოველი i-სათვის ($0 \leq i \leq M-1$), w[i] გვაძლევს მოძრაობის პირობებს i ნომრის მქონე გზატკეცილისათვის. w[i]-ის მნიშვნელობა შეიძლება იყოს 0 ან 1.
  - w[i] = 0 აღნიშნავს, რომ მოძრაობის პირობები i-ური გზატკეცილისათვის მსუბუქია.
  - w[i] = 1 აღნიშნავს, რომ მოძრაობის პირობები i-ური გზატკეცილისათვის მძიმეა.
- ფუნქცია აბრუნებს უმცირეს ჯამურ დანახარჯს, რომელიც საჭიროა S და T ქალაქებს შორის მგზავრობისას, w მასივით განსაზღვრული მოძრაობის პირობებისათვის.
- ფუნქციის გამოძახება შესაძლებელია არაუმეტეს 100-ჯერ (თითოეული ტესტისათვის).

find_pair ფუნქციამ პასუხის გამოსატანად უნდა გამოიძახოს შემდეგი ფუნქცია:

```
answer(int s, int t)
```

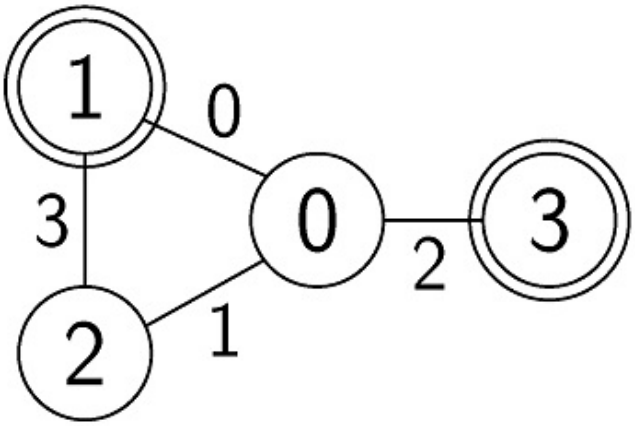
- s და t უნდა გვაძლევდეს S და T წყვილს (თანმიმდევრობას მნიშვნელობა არ აქვს).
- ეს ფუნქცია გამოძახებულ უნდა იქნას ზუსტად ერთხელ.

თუ ზემოთ ნახსენები ყველა პირობა არ იქნება დაკმაყოფილებული, თქვენი პროგრამა შეფასდება, როგორც **Wrong Answer**. სხვა შემთხვევაში, თქვენი პროგრამა შეფასდება, როგორც **Accepted** და ქულები გამოითვლება ask ფუნქციის გამოძახების რაოდენობათა მიხედვით (იხილე ქვეამოცანები).

მაგალითი

ვთქვათ, $N = 4$, $M = 4$, $U = [0, 0, 0, 1]$, $V = [1, 2, 3, 2]$, $A = 1$, $B = 3$, $S = 1$, და $T = 3$.

გრადერი გამოიძახებს find_pair(4, [0, 0, 0, 1], [1, 2, 3, 2], 1, 3).



ზემოთ ნაჩვენებ ნახაზზე i ნომრის მქონე წიბო შეესაბამება i ნომრის მქონე გზატკეცილს.

ქვემოთ მოყვანილია `ask` ფუნქციის მიერ დაბრუნებული მნიშვნელობები ზოგიერთი შეკითხვისათვის:

Call	Return
`ask([0, 0, 0, 0])`	2
`ask([0, 1, 1, 0])`	4
`ask([1, 0, 1, 0])`	5
`ask([1, 1, 1, 1])`	6

როცა ვიძახებთ ფუნქციას `ask([0, 0, 0, 0])`, მოძრაობის პირობები თითოეული გზატკეცილისათვის მსუბუქია და ამიტომ გადასახადი თითოეულ გზატკეცილზე 1-ს უტოლდება. ყველაზე იაფი მარშრუტი $S = 1$-დან $T = 3$-მდე არის $1 \to 0 \to 3$. მისი ჯამური დანახარჯი 2-ის ტოლია. მაშასადამე, ფუნქცია დააბრუნებს 2-ს.

სწორი პასუხისათვის `find_pair` ფუნქციამ უნდა დააბრუნოს `answer(1, 3)` ან `answer(3, 1)`.

მიბმულ დაარქივებულ პაკეტში არსებული sample-01-in.txt ფაილი შეესაბამება მოცემულ მაგალითს. შეტანის სხვა მაგალითებიც ამავე პაკეტშია მოცემული.

შეზღუდვები

- $2 \leq N \leq 90\,000$
- $1 \leq M \leq 130\,000$
- $1 \leq A < B \leq 1\,000\,000\,000$
- ყოველი $0 \leq i \leq M - 1$
  - $0 \leq U[i] \leq N - 1$
  - $0 \leq V[i] \leq N - 1$

    - $U[i] \neq V[i]$
- $(U[i], V[i]) \neq (U[j], V[j])$ and $(U[i], V[i]) \neq (V[j], U[j])$ $(0 \leq i < j \leq M - 1)$
- თქვენ შეგიძლიათ გზატკეცილთა საშუალებით ნებისმიერი ქალაქიდან ნებისმიერ სხვა ქალაქამდე მისვლა.
- $0 \leq S \leq N - 1$
- $0 \leq T \leq N - 1$
- $S \neq T$

ამ ამოცანაში გრადერი არ არის ადაპტიური. ეს ნიშნავს, რომ S-ის და T-ს მნიშვნელობები დაფიქსირებულია გრადერის მიერ მისი მუშაობის დაწყებისთანავე და ეს მნიშვნელობები დამოკიდებული არ არიან თქვენის პროგრამის მიერ დასმულ შეკითხვებზე.

ქვეამოცანები

1. (5 ქულა) S ან T-ს მნიშვნელობებიდან ერთ-ერთი არის 0, $N \leq 100$, $M = N - 1$
2. (7 ქულა) S ან T-ს მნიშვნელობებიდან ერთ-ერთი არის 0, $M = N - 1$
3. (6 ქულა) $M = N - 1$, $U[i] = i$, $V[i] = i + 1$ $(0 \leq i \leq M - 1)$
4. (33 ქულა) $M = N - 1$
5. (18 ქულა) $A = 1$, $B = 2$
6. (31 ქულა) დამატებითი შეზღუდვების გარეშე.

თუკი თქვენმა პროგრამამ მიიღო შეფასება **Accepted** და `ask` ფუნქცია გამოიძახა X-ჯერ, მაშინ თქვენი პროგრამა მიიღებს P ქულას, რომელიც ქვეამოცანებთან დამოკიდებულებაში, გამოითვლება შემდეგნაირად:

- ქვეამოცანა 1. $P = 5$.
- ქვეამოცანა 2. თუ $X \leq 60$, $P = 7$. სხვა შემთხვევაში $P = 0$.
- ქვეამოცანა 3. თუ $X \leq 60$, $P = 6$. სხვა შემთხვევაში $P = 0$.
- ქვეამოცანა 4. თუ $X \leq 60$, $P = 33$. სხვა შემთხვევაში $P = 0$.
- ქვეამოცანა 5. თუ $X \leq 52$, $P = 18$. სხვა შემთხვევაში $P = 0$.
- ქვეამოცანა 6.
    - If $X \leq 50$, $P = 31$.
    - If $51 \leq X \leq 52$, $P = 21$.
    - If $53 \leq X$, $P = 0$.

მიაქციეთ ყურადღება, რომ თქვენი ქულა ქვეამოცანაში ამ ქვეამოცანაში შემავალ ტესტებს შორის მინიმალური ქულის ტოლია.

სანიმუშო გრადერი

სანიმუშო გრადერი კითხულობს შესატან მონაცემებს შემდეგი ფორმატით:

-სტრიქონი 1: N M A B S T
-სტრიქონი $2 + i$ $(0 \leq i \leq M - 1)$: $U[i]$ $V[i]$

თუ თქვენი პროგრამა შეფასდა როგორც **Accepted**, სანიმუშო გრადერი გამოიტანს `Accepted: q`, სადაც `q` არის `ask` ფუნქციის გამოძახებათა რაოდენობა.

თუ თქვენი პროგრამა შეფასდა როგორც **Wrong Answer**, სანიმუშო გრადერი გამოიტანს `Wrong Answer: MSG`, სადაც `MSG` არის ერთ-ერთი შემთხვევებიდან:

- `answered not exactly once`: ფუნქცია `answer` არ იქნა გამოძახებული ზუსტად ერთხელ.
- `w is invalid`: `w` მასივის სიგრძე, რომელიც მოცემულია `ask` ფუნქციით, არ არის M ან `w[i]`-ის მნიშვნელობა არ არის 0 ან 1, რომელიმე i-სთვის ($0 \leq i \leq M-1$).
- `more than 100 calls to ask`: `ask` ფუნქციის გამოძახებათა რაოდენობამ გადააჭარბა 100-ს.
- `{s, t} is wrong`: `answer` ფუნქცია გამოძახებულ იქნა არასწორი `s` და `t` წყვილისათვის.