Zadanie: PIZ
Dostawca pizzy

Olimpiada Informatyczna

XXIV OI, etap III, dzień próbny. Plik źródłowy `piz.*` Dostępna pamięć: 64 MB. *10.04.2017*

Bajtogród jest malowniczym miastem, w którym znajduje się n skrzyżowań połączonych siecią $n-1$ dwukierunkowych dróg. Przy każdym skrzyżowaniu znajduje się dom, a w jednym z nich mieści się pizzeria Bajtazara. Wszyscy mieszkańcy Bajtogrodu uwielbiają pizzę, więc każdego ranka Bajtazar wypieka $n-1$ pizz i rozwozi je po całym mieście – dokładnie po jednej do każdego domu (poza swoim).

Jako że nikt nie lubi zimnej pizzy, Bajtazar wyposażył swój samochód w supernowoczesny podgrzewacz. Niestety, jest on także superenergochłonny, więc Bajtazar chciałby go używać jak najkrócej. Postępuje on więc następująco: pakuje do samochodu kilka pizz, włącza podgrzewacz i rozwozi pizze do niektórych domów. W momencie dostarczenia ostatniej z nich, wyłącza podgrzewacz i wraca do pizzerii. Bajtazar jest skłonny wykonać *co najwyżej* k takich kursów. Zastanawia się teraz, jaki jest minimalny czas pracy podgrzewacza podczas rozwożenia wszystkich pizz.

Czas pracy podgrzewacza podczas postojów (gdy Bajtazar zanosi pizzę pod drzwi) jest pomijalny.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie dodatnie liczby całkowite n oraz k oddzielone pojedynczym odstępem, oznaczające liczbę skrzyżowań w Bajtogrodzie i maksymalną liczbę kursów, które jest gotów wykonać Bajtazar. Skrzyżowania numerujemy liczbami od 1 do n; pizzeria znajduje się przy skrzyżowaniu numer 1.

Kolejne $n-1$ wierszy zawiera opis sieci drogowej: i-ty z tych wierszy zawiera trzy dodatnie liczby całkowite a_i, b_i i c_i ($a_i, b_i \leq n$, $a_i \neq b_i$) pooddzielane pojedynczymi odstępami, oznaczające, że istnieje dwukierunkowa droga łącząca skrzyżowania o numerach a_i i b_i, której przejechanie w jedną stronę wymaga c_i minut. Sieć drogowa jest tak dobrana, że korzystając z niej, z każdego skrzyżowania można dotrzeć do każdego innego, niekoniecznie bezpośrednio.

Wyjście

W pierwszym i jedynym wierszu standardowego wyjścia należy wypisać jedną liczbę całkowitą, oznaczającą minimalny czas (w minutach), przez jaki podgrzewacz musi być włączony, by Bajtazar mógł rozwieźć wszystkie pizze.

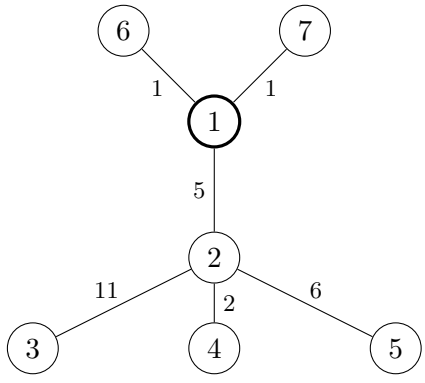
Przykład

Dla danych wejściowych:

```
7 3
1 2 5
2 3 11
2 4 2
5 2 6
1 6 1
7 1 1
```

poprawnym wynikiem jest:

```
34
```



Wyjaśnienie do przykładu: Bajtazar wykona trzy kursy: $1 \to 2 \to 4 \to 2 \to 5 \rightsquigarrow 1$ (czas przejazdu z włączonym podgrzewaczem to 15 minut), $1 \to 2 \to 3 \rightsquigarrow 1$ (16 minut) oraz $1 \to 6 \to 1 \to 7 \rightsquigarrow 1$ (3 minuty).

Testy „ocen":

- `1ocen:` $n = 15$, $k = 3$. Jest to małe, pełne drzewo binarne, w którym czas przejazdu drogami prowadzącymi do liści wynosi 6 minut, a czas przejazdu pozostałymi drogami wynosi 1 minutę.
- `2ocen:` $n = 2000$, $k = 100$. Z pizzerii da się dojechać do wszystkich skrzyżowań bezpośrednio. Duże, losowe czasy przejazdu.

`3ocen:` $n = 50\,000$, $k = 1000$. Z pizzerii można dojechać bezpośrednio do dwóch skrzyżowań. Z jednego z nich można dojechać do wszystkich pozostałych. Wszystkie czasy są równe 1.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów. We wszystkich testach zachodzi $n \geq 2$, $k \geq 1$ oraz $1 \leq c_i \leq 1\,000\,000$.

Podzadanie	**Warunki**	**Liczba punktów**
1	$n, k \leq 10$	12
2	$n, k \leq 2000$	24
3	$n, k \leq 100\,000$ i $n \cdot k \leq 4\,000\,000$	28
4	$n, k \leq 100\,000$	36

Zadanie: MID
Midas

XXIV OI, etap III, dzień pierwszy. Plik źródłowy `mid.*` Dostępna pamięć: 256 MB. *11.04.2017*

Bajtazar – nadworny inżynier króla Midasa – zbudował na jego polecenie labirynt. W teorii komnaty labiryntu miały zawierać ciekawe eksponaty do zwiedzania, ale w praktyce labirynt służy głównie do zarabiania ogromnej liczby dukatów, które zasilają i tak już pokaźny królewski skarbiec.

Labirynt składa się z komnat i łączących je korytarzy. Jedna z komnat jest oznaczona jako wejście do labiryntu. W każdej komnacie droga rozwidla się i z komnaty można wyjść jednym z dwóch korytarzy – lewym lub prawym (przy czym niektóre z korytarzy w rozwidleniach mogą być zablokowane – nie da się nimi przejść dalej). Korytarze tylko się rozwidlają, a nigdy się nie łączą. Odwiedzający labirynt dostaje urządzenie pobierające opłaty za każde przejście korytarzem. Opłata zależy od dotychczas pobranej liczby dukatów i od tego, czy odwiedzający wybierze lewy czy prawy korytarz. Jeśli wybierze lewy, to zapłaci za przejście tego korytarza dokładnie tyle samo, ile do tej pory wynosiła suma wszystkich uiszczonych przez niego opłat. Jeśli pójdzie w prawo, to zapłaci o jednego dukata więcej niż to, ile do tej pory zapłacił w sumie.

Jednym z powodów tak zawiłego sposobu pobierania opłat za zwiedzanie labiryntu była chęć ukrycia rzeczywistego kosztu zwiedzania. Było to jednak kłopotliwe dla zwiedzających. Bajtazar postanowił choć trochę im pomóc i zatrudnił Cię do przygotowania programu, który będzie odpowiadał na następujące zapytania: jeśli odwiedzający posiada liczbę dukatów dokładnie taką, jaka jest niezbędna do przejścia z komnaty wejściowej do komnaty x, to czy taka liczba dukatów pozwoli mu na przejście z komnaty wejściowej do komnaty y?

Wejście

Pierwszy wiersz standardowego wejścia zawiera jedną dodatnią liczbę całkowitą n oznaczającą liczbę komnat labiryntu. Komnaty numerujemy liczbami od 1 do n. Wejście do labiryntu znajduje się w komnacie numer 1. Kolejne n wierszy opisuje labirynt; w i-tym z tych wierszy znajdują się dwie liczby całkowite l_i, r_i ($0 \leq l_i, r_i \leq n$) oddzielone pojedynczym odstępem. Jeśli $l_i > 0$, to liczba ta oznacza, że wychodząc z komnaty o numerze i lewym korytarzem, dostaniemy się do komnaty o numerze l_i. Jeśli $l_i = 0$, to wyjście lewego korytarza z komnaty numer i jest zablokowane. Analogicznie liczba r_i opisuje prawy korytarz wychodzący z komnaty numer i. Rozpoczynając wędrówkę w komnacie numer 1, da się dojść do każdej z pozostałych komnat.

Następny wiersz zawiera jedną dodatnią liczbę całkowitą z oznaczającą liczbę zapytań do rozważenia. Kolejne z wierszy opisują zapytania; i-ty z tych wierszy zawiera dwie liczby całkowite x_i, y_i ($1 \leq x_i, y_i \leq n$) oddzielone pojedynczym odstępem. Oznacza to zapytanie „Czy liczba dukatów niezbędna i wystarczająca do odwiedzenia komnaty numer x_i wystarczy do odwiedzenia komnaty numer y_i?”.

Wyjście

Na standardowe wyjście należy wypisać dokładnie z wierszy: i-ty z nich ma zawierać jedno słowo `TAK` lub `NIE` w zależności od tego, czy odpowiedź na i-te zapytanie z wejścia jest twierdząca.

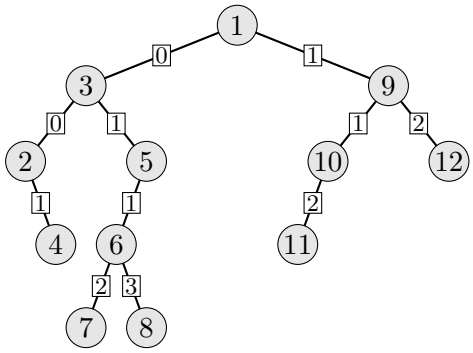
Przykład

Dla danych wejściowych:

```
12
3 9
0 4
2 5
0 0
6 0
7 8
0 0
0 0
10 12
11 0
0 0
0 0
6
11 8
8 6
11 7
1 2
4 10
3 3
```

poprawnym wynikiem jest:

```
NIE
TAK
TAK
TAK
NIE
TAK
```



Wyjaśnienie. Rysunek przedstawia labirynt z powyższego przykładu: komnaty są oznaczone kółkami, a liczba dukatów wymagana do przejścia danego korytarza zapisana jest w kwadraciku. Przechodzenie rysunku od góry do dołu i kolejno wybór lewej lub prawej gałęzi odpowiadają przechodzeniu labiryntu i wybraniu lewego lub prawego korytarza. Przykładowo, za odwiedzenie komnaty numer 11 trzeba zapłacić $1+1+2=4$ dukaty; jest to wartość niewystarczająca, aby odwiedzić komnatę numer 8 – w tym celu potrzeba $0+1+1+3=5$ dukatów (odpowiada temu pierwsze zapytanie z przykładu).

Testy „ocen”:

- `1ocen:` $n=7$ komnat tworzących pełne drzewo binarne, $z=49$ zapytań o wszystkie możliwe pary komnat;
- `2ocen:` $n=2^{19}-1$ komnat tworzących pełne drzewo binarne, $z=n+1$ zapytań o pary sąsiednich liści (o każdą parę liści są dwa zapytania – jedno na `TAK` i jedno na `NIE`);
- `3ocen:` $n=1\,000\,000$; ścieżka o długości $\frac{n}{2}$ idąca tylko w lewo; od każdej komnaty na tej ścieżce odchodzi jeden korytarz w prawo; $z=10$ losowych zapytań.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Warunki	Liczba punktów
1	$n \leq 50$, $z \leq 10$	15
2	$n \leq 1000$, $z \leq 10$	9
3	$n \leq 1000$, $z \leq 1\,000\,000$	14
4	$n \leq 1\,000\,000$, $z \leq 10$	11
5	$n \leq 1\,000\,000$, $z \leq 1\,000\,000$	51

Zadanie: OCE
Oceny

Olimpiada Informatyczna

XXIV OI, etap III, dzień pierwszy. Plik źródłowy `oce.*` **Dostępna pamięć: 256 MB.** *11.04.2017*

Nauczyciel wychowania fizycznego ma w swojej klasie n uczniów. Zbliża się koniec roku szkolnego, więc najwyższy czas wystawić wszystkim oceny końcowe. Nauczyciel przypisał każdemu z uczniów liczbę a_i z zakresu od 1 do n, która jest tym większa, im większe są umiejętności sportowe ucznia. Każdy uczeń ma przypisaną inną wartość a_i. Na początku zajęć uczniowie ustawiają się w szeregu. Nauczyciel wystawia oceny po kolei, zaczynając od tego po lewej, a kończąc na tym po prawej. Skala ocen zaczyna się na 1, a kończy na n, więc jest aż n różnych ocen!

Nauczyciel chciałby, aby wystawione przez niego oceny spełniały następujące wymagania:

- Dla każdych dwóch uczniów v i u, jeśli v ma większe umiejętności sportowe niż u, to v nie może dostać gorszej oceny niż u. Byłoby to ewidentnie niesprawiedliwe.
- Dla każdych dwóch uczniów v i u, jeśli v stoi na prawo od u, przez co zostanie rozważony później, to v nie może dostać gorszej oceny niż u. Byłoby mu wtedy bardzo przykro.
- Biorąc pod uwagę powyższe warunki, nauczyciel chce wystawić możliwie najwięcej różnych ocen.

Na każdych zajęciach uczniowie są ustawieni w szeregu w pewnej kolejności. Nauczyciel jest przyzwyczajony do stałej kolejności uczniów w szeregu, jednak na ich prośbę zgodził się, aby pomiędzy każdymi kolejnymi zajęciami dwóch uczniów zamieniało się miejscami. Nauczyciel nie zdecydował jeszcze, na których zajęciach wystawi wszystkim oceny. Pomóż mu i napisz program, który dla każdych zajęć obliczy, ile różnych ocen będzie mógł na nich wystawić, jeśli wystawianie ocen przeprowadzi właśnie na nich.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie dodatnie liczby całkowite n i z oznaczające liczbę uczniów i liczbę zajęć, które pozostały do końca roku.

W drugim wierszu znajduje się ciąg n liczb całkowitych $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq n)$ oznaczających umiejętności sportowe kolejnych uczniów, według kolejności, w jakiej ustawili się w szeregu na pierwszych zajęciach. Liczby te są parami różne.

W następnych $z-1$ wierszach znajdują się opisy zamian. W i-tym z tych wierszy znajdują się dwie liczby całkowite p_i i q_i $(1 \leq p_i < q_i \leq n)$ oznaczające, że uczniowie, którzy na i-tych zajęciach stali w szeregu na miejscach p_i i q_i, zamienią się miejscami przed następnymi zajęciami.

Wyjście

Na standardowe wyjście należy wypisać dokładnie z wierszy. Liczba z i-tego wiersza powinna oznaczać maksymalną liczbę różnych ocen, które mógłby wystawić nauczyciel podczas i-tych zajęć.

Przykład

Dla danych wejściowych:	poprawnym wynikiem jest:

`3 4`
`1 2 3`
`1 3`
`1 2`
`2 3`
	`3`
	`1`
	`1`
	`2`

Wyjaśnienie do przykładu: Na pierwszych zajęciach uczniowie stoją w kolejności $1, 2, 3$, więc każdy uczeń może dostać inną ocenę. Na drugich stoją w kolejności $3, 2, 1$, a na trzecich w kolejności $2, 3, 1$. W obu tych przypadkach uczeń 1 nie może dostać lepszej oceny niż ktokolwiek inny, ponieważ ma najmniejsze umiejętności sportowe, ani gorszej, ponieważ jest oceniany jako ostatni. Na ostatnich zajęciach uczniowie stoją w kolejności $2, 1, 3$. Wówczas uczeń 3 może dostać lepszą ocenę, a uczniowie 1 i 2 muszą dostać tę samą ocenę.

Testy „ocen”:

`1ocen:` mały test poprawnościowy – na pierwszych zajęciach wszyscy uczniowie na pozycjach nieparzystych mają większe umiejętności od wszystkich na parzystych, na ostatnich stoją w kolejności rosnącej według umiejętności.

`2ocen:` $n = 2000$, $z = n/2 + 1$, $a_i = i + 1$ dla i nieparzystych, $a_i = i - 1$ dla i parzystych, $p_i = 2i - 1$, $q_i = 2i$ (zamieniają się kolejne pary uczniów); liczba ocen wzrasta od $n/2$ do n;

`3ocen:` $n = z = 300\,000$, $a_i = i$ (uczniowie uporządkowani rosnąco względem umiejętności), $p_i = i$, $q_i = i-1$ (uczeń n zamienia się kolejno ze wszystkimi); liczba ocen spada od n do 1.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	**Warunki**	**Liczba punktów**
1	$n, z \leq 2000$	24
2	$n \leq 2000$, $z \leq 300\,000$	8
3	$n, z \leq 100\,000$	30
4	$n \leq 1\,000\,000$, $z \leq 300\,000$, nauczyciel nigdy nie może wystawić więcej niż 15 różnych ocen	10
5	$n \leq 1\,000\,000$, $z \leq 300\,000$, uczniowie zamieniają się tylko z najbliższym sąsiadem z lewej lub z prawej	20
6	$n \leq 1\,000\,000$, $z \leq 300\,000$	8

Zadanie: ZAP
Zapiekanki

Olimpiada Informatyczna

XXIV OI, etap III, dzień pierwszy. Plik źródłowy `zap.*` Dostępna pamięć: 256 MB. *11.04.2017*

Bajtazar prowadzi budkę, w której sprzedaje zapiekanki. Jego produkty znane są w całej Bajtocji z doskonałej jakości i niepowtarzalnego smaku. Przez lata właściciel interesu ustabilizował swoją pozycję na rynku oraz zyskał stałe grono klientów.

Codziennie do budki przychodzi k klientów, i-ty w chwili t_i. Każdy z nich zamawia dokładnie jedną zapiekankę. Bajtazar dba o swoich klientów, dlatego najchętniej od razu wydałby im ich zamówienia, jednak piekarnik ma swoje ograniczenia. W piekarniku można na raz upiec co najwyżej z zapiekanek, co zajmuje czas d, i nie można w tym czasie otwierać piekarnika. Bajtazar chce, aby sumaryczny czas czekania przez klientów na realizację ich zamówień był jak najkrótszy. Może on zacząć piec zapiekankę, zanim klient się po nią zgłosi, ale musi skończyć ją piec, kiedy klient już przyjdzie – nikt nie lubi jeść zimnych zapiekanek. Bajtazar przychodzi do swojej budki w chwili 0.

Ile wynosi sumaryczny czas oczekiwania przez klientów na zapiekanki, przy założeniu, że Bajtazar wie, w których momentach przyjdą poszczególni klienci, i wybierze optymalną strategię pieczenia?

Wejście

Pierwszy wiersz standardowego wejścia zawiera trzy dodatnie liczby całkowite k, z i d oznaczające liczbę klientów, pojemność piekarnika i czas pieczenia. W drugim wierszu znajduje się ciąg k liczb całkowitych $t_1, t_2, \ldots, t_k$ ($0 \leq t_1 \leq t_2 \leq \ldots \leq t_k$); liczba t_i oznacza chwilę, w której do budki przyjdzie i-ty klient.

Wyjście

Na standardowe wyjście należy wypisać jedną liczbę całkowitą – sumaryczny czas oczekiwania klientów, przy optymalnej strategii pieczenia zapiekanek.

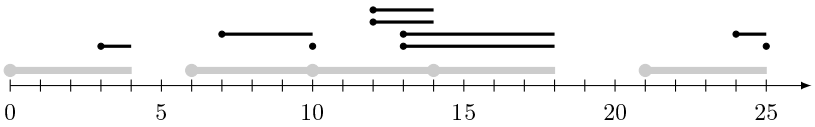
Przykład

Dla danych wejściowych:

```
9 2 4
3 7 10 12 12 13 13 24 25
```

poprawnym wynikiem jest:

```
19
```



Wyjaśnienie do przykładu: W optymalnej strategii pieczenia (przedstawionej na rysunku) Bajtazar uruchamia piekarnik w chwilach 0, 6, 10, 14 i 21. Podczas pierwszego uruchomienia piecze tylko jedną zapiekankę, potem po dwie. Czasy pieczenia zilustrowane są szarym kolorem, a czasy oczekiwania przez klientów – czarnym.

Testy „ocen":

`1ocen:` $k = z = 10$, $d = 1$, wszyscy klienci przychodzą w chwili 0;

`2ocen:` $k = 2000$, $z = 5$, $d = 200$, klienci przychodzą w odstępach większych niż d;

`3ocen:` $k = 3000$, $z = 7$, $d = 1\,000\,000$, połowa klientów przychodzi w chwili 0, połowa w chwilach $t_{\frac{k}{2}+i} = i$.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Warunki	Liczba punktów
1	$z \leq k \leq 200$; $d \leq 200$; $t_k \leq 10\,000$	20
2	$z \leq k \leq 200$; $d, t_k \leq 1\,000\,000$	30
3	$z \leq k \leq 3000$; $d, t_k \leq 1\,000\,000$	50

Zadanie: KUC
Kucharz

Olimpiada Informatyczna

XXIV OI, etap III, dzień drugi. Plik źródłowy `kuc.*` **Dostępna pamięć: 8 MB.** *12.04.2017*

Kucharz Bajtazar pracuje w restauracji i ma n zamówień czekających na przygotowanie. Każde zamówienie zapisane jest na karteczce, a wszystkie karteczki są nadziane na szpikulec. Bajtazar musi wykonywać zamówienia od góry, ponieważ może ściągać karteczki tylko z góry szpikulca. Przyrządzanie dań jest czasochłonne i kucharz chciałby wykonać wszystkie zamówienia możliwie jak najszybciej. Nie musi jednak wykonywać ich pojedynczo. Załóżmy, że na szpikulcu zostało k karteczek. Oto co może zrobić:

- Wziąć jedno zamówienie z góry i przyrządzić je w czasie `jedno`(k).
- Jeśli $k > 1$, to może wziąć dwa zamówienia z góry i przyrządzić je w czasie `dwa`(k).
- Jeśli $k > 1$, to może wziąć $\lfloor k/2 \rfloor$ zamówień z góry i wykonać je w czasie `polowa`(k).

Dodatkowo, Bajtazar ma pewien poziom energii, początkowo równy e. Operacje trzeciego rodzaju, tzn. te biorące połowę zamówień, bardzo go męczą i obniżają jego poziom energii o 1. Jest jednak mistrzem w operacjach pierwszego rodzaju (czyli w obsługiwaniu jednego zamówienia) i każda taka operacja podnosi jego poziom energii o 1. Poziom energii kucharza nie może nigdy spaść poniżej 0. Bajtazara nie interesuje jego końcowy poziom energii; chciałby jedynie jak najszybciej przygotować wszystkie zamówienia.

Napisz program komunikujący się z biblioteką dającą wszystkie potrzebne informacje na temat stanu Bajtazara i jego kuchni, który znajdzie najmniejszy możliwy czas, w jakim Bajtazar będzie w stanie przygotować wszystkie zamówienia. W tym zadaniu zwróć szczególną uwagę na limit dostępnej pamięci.

Komunikacja

Aby użyć biblioteki, należy wpisać na początku programu:

- **C/C++**: `#include "ckuclib.h"`
- **Pascal**: `uses pkuclib;`

Biblioteka udostępnia następujące funkcje i procedury:

- `dajn`, `daje`
  Pierwsza funkcja daje w wyniku liczbę całkowitą n, oznaczającą liczbę zamówień do przygotowania, zaś druga – liczbę całkowitą e, oznaczającą początkowy poziom energii Bajtazara.
  - **C/C++:**
    ```
    int dajn();
    int daje();
    ```
  - **Pascal:**
    ```
    function dajn: LongInt;
    function daje: LongInt;
    ```
- `jedno`(k), `dwa`(k), `polowa`(k)
  Funkcje dają w wyniku czas przyrządzania zamówień, gdy na szpikulcu jest aktualnie k karteczek, a Bajtazar postanowi zrealizować, odpowiednio, jedno, dwa lub $\lfloor k/2 \rfloor$ zamówień. Dla $k = 1$ wartości funkcji `dwa`(k) i `polowa`(k) są nieistotne. Czas jest liczbą całkowitą z przedziału od 1 do 10^7.
  - **C/C++:**
    ```
    int jedno(int k);
    int dwa(int k);
    int polowa(int k);
    ```
  - **Pascal:**
    ```
    function jedno(k: LongInt): LongInt;
    function dwa(k: LongInt): LongInt;
    function polowa(k: LongInt): LongInt;
    ```
- `odpowiedz`($wynik$)
  Odpowiada bibliotece, że $wynik$ to najmniejszy możliwy czas, w którym Bajtazar może przygotować wszystkie zamówienia. Wywołanie tej funkcji **kończy działanie Twojego programu**.
  - **C/C++:**
    ```
    void odpowiedz(int wynik);
    ```
  - **Pascal:**
    ```
    procedure odpowiedz(wynik: LongInt);
    ```

Twój program **nie może** czytać żadnych danych (ani ze standardowego wejścia, ani z plików). **Nie może** również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`) – pamiętaj jednak, że zużywa to cenny czas. Bibliotekę można pytać wielokrotnie o dane wartości.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Warunki	Liczba punktów
1	$n, e \leq 1000$	12
2	$n, e \leq 50\,000$	8
3	$n, e \leq 1\,000\,000$	80

Przykładowy przebieg programu

C/C++	Pascal	Wynik
`n = dajn();`	`n := dajn;`	3
`e = daje();`	`e := daje;`	1
`p[3] = polowa(3);`	`p[3] := polowa(3);`	1
`p[2] = polowa(2);`	`p[2] := polowa(2);`	4
`j[2] = jedno(2);`	`j[2] := jedno(2);`	2
`j[1] = jedno(1);`	`j[1] := jedno(1);`	5
`d[2] = dwa(2);`	`d[2] := dwa(2);`	6
`odpowiedz(7);`	`odpowiedz(7);`	—

Powyższy przebieg programu jest poprawny pod względem formalnym, choć niekoniecznie daje poprawny wynik. Z podanych informacji wnioskujemy, że wykonanie operacji `polowa` i `dwa` (o koszcie $p[3] + d[2] = 7$) jest lepsze od wykonania jednej operacji `polowa` i dwóch operacji `jedno` (o koszcie $p[3] + j[2] + j[1] = 8$). Wiemy też, że nie opłaca się wykonywać operacji `jedno` przy trzech zamówieniach (bo $p[3] = 1$, a $j[3] \geq 1$). Wykonanie dwóch operacji `polowa` nie jest możliwe ze względu na to, że $e = 1$. Ale bez znajomości kosztu operacji `dwa(3)` nie możemy stwierdzić, czy wykonanie kolejno operacji `dwa` i `jedno` nie dałoby lepszego wyniku.

Eksperymenty

W katalogu `dlazaw` dostępna jest przykładowa biblioteka, która pozwoli Ci przetestować poprawność formalną rozwiązania. Biblioteka wczytuje informacje ze standardowego wejścia w następującym formacie:

- w pierwszym wierszu dwie liczby całkowite n i e;
- w drugim wierszu n liczb całkowitych z przedziału $[1, 10^7]$ – wartości funkcji `jedno` dla kolejnych liczb $k = 1, \ldots, n$;
- w trzecim i czwartym wierszu wartości dla funkcji `dwa` i `polowa` (w takim samym formacie); jako pierwsza musi pojawić się wartość dla $k = 1$, ale jest ona nieistotna.

Przykładowe wejście dla biblioteki znajduje się w pliku `kuc0.in`. Po wywołaniu procedury `odpowiedz`, biblioteka wypisuje na standardowe wyjście informację o udzielonej odpowiedzi.

W tym samym katalogu znajdują się przykładowe rozwiązania `kuc.c`, `kuc.cpp` i `kuc.pas` korzystające z biblioteki. Rozwiązania te nie są poprawne i zawsze wykonują operację `polowa`, poza ostatnim wywołaniem, w którym wykonują operację `jedno`.

Do kompilacji rozwiązania wraz z biblioteką służą polecenia:

- **C**: `gcc -O2 -static ckuclib.c kuc.c -lm -std=gnu99`
- **C++**: `g++ -O2 -static ckuclib.c kuc.cpp -lm -std=c++11`
- **Pascal**: `ppc386 -O2 -XS -Xt kuc.pas`

Plik z rozwiązaniem i biblioteka powinny znajdować się w tym samym katalogu.

Zadanie: ROZ
Rozdroża parzystości

Olimpiada Informatyczna

XXIV OI, etap III, dzień drugi. Plik źródłowy `roz.*` Dostępna pamięć: 256 MB. *12.04.2017*

Bajtazar zarządza nowo powstającym księstwem w królestwie Bajtocji. Księstwo to składa się z n miast, ale jest dopiero na etapie tworzenia infrastruktury, zatem nie są w nim jeszcze wybudowane żadne drogi. Bajtazar przygotował projekt budowy m dwukierunkowych dróg, z których każda łączyłaby dwa spośród miast. Gdyby plan został zrealizowany w całości, dałoby się dojechać z każdego miasta do każdego innego. Brak doświadczenia Bajtazara w projektowaniu dał jednak znać o sobie – drogi nie są wytyczone zbyt ekonomicznie i każda kolejna z nich jest trudniejsza do zbudowania niż wszystkie poprzednie razem wzięte. Bajtazar oszacował, że koszt budowy i-tej drogi to 2^i bajtalarów.

Niestety, w międzyczasie w Bajtocji pojawiła się nowa moda – obywatele mają teraz obsesję na punkcie parzystości. Mieszkańcy niektórych miast wierzą, że parzystość jest symbolem dopasowania, harmonii i spokoju, zatem nalegają, aby liczba dróg wychodzących z ich miast była parzysta. Za to mieszkańcy innych miast sądzą, że nieparzystość jest symbolem samodzielności, indywidualności i zaradności, więc wymagają, aby liczba dróg wychodzących z ich miast była nieparzysta.

Bajtazar musi stworzyć nowy plan, wybierając tylko część spośród zaprojektowanych dróg tak, aby spełnić wymagania mieszkańców wszystkich miast. Oczywiście zależy mu na tym, aby sumaryczny koszt był możliwie mały. Bajtockie prawo o zamówieniach publicznych każe jednak czasem odrzucić $k-1$ najtańszych ofert, w związku z czym Bajtazar sposobi się, aby wybudować k-tą najtańszą sieć dróg spośród spełniających wymagania Bajtoczan. Zwróć uwagę na to, że Bajtoczan przestało interesować, aby wybudowanymi drogami dało się dojechać z ich miasta do każdego innego miasta. Mimo że oryginalny projekt Bajtazara miał tę własność, może ona być nie do pogodzenia z wymaganiami dotyczącymi parzystości.

Żeby tego było jeszcze mało, parlament Bajtocji często przegłosowuje nową ustawę o zamówieniach publicznych, zmieniając wartość liczby k, a mieszkańcy miast często zmieniają swoje przekonania i przestają wielbić liczby parzyste, a zaczynają lubować się w liczbach nieparzystych, lub na odwrót. Bajtazar musi dostosowywać swoje plany do tych wszystkich zmian. Pomóż Bajtazarowi okazać swoją hojność, sprawność w rządzeniu i łatwość nadążania za zmianami humorów mieszkańców jego księstwa, podpowiadając mu plany sieci dróg, które powinien wybudować!

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite n i m ($1 \leq n, m \leq 500\,000$) oznaczające odpowiednio liczbę miast oraz liczbę potencjalnych dróg możliwych do wybudowania w księstwie Bajtazara.

Dalej następuje m wierszy opisujących zaprojektowane drogi. W i-tym z tych wierszy znajdują się dwie liczby całkowite a_i, b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$) oznaczające, że jest możliwe wybudowanie dwukierunkowej drogi łączącej miasta a_i oraz b_i. Jej budowa kosztowałaby 2^i bajtalarów. Żadna para miast nie pojawi się na wejściu więcej niż raz.

W kolejnym wierszu znajduje się n liczb całkowitych $p_1, \ldots, p_n$ oddzielonych pojedynczymi odstępami. Jeżeli $p_i = 0$, to mieszkańcy i-go miasta uwielbiają liczby parzyste, a jeżeli $p_i = 1$, to wierzą oni w magię liczb nieparzystych.

W następnym wierszu znajduje się liczba k ($1 \leq k \leq 10^{18}$) oznaczająca, że Bajtazar rozważa aktualnie k-ty w kolejności najtańszy plan budowy dróg zgodny z poglądami mieszkańców ($k = 1$ oznacza, że Bajtazar rozważa najtańszy z nich). Łatwo zauważyć, że żadne dwa różne plany nie mają takiego samego kosztu.

W kolejnym wierszu następuje liczba całkowita q ($0 \leq q \leq 500\,000$) oznaczająca liczbę zapytań, a po niej q wierszy je opisujących. Opis zapytania składa się z litery c opisującej typ zapytania, a następnie liczby v. Jeżeli $c =$ `M`, to znaczy, że mieszkańcy miasta o numerze v ($1 \leq v \leq n$) zmienili swoje poglądy na przeciwne. Jeżeli $c =$ `K`, to znaczy, że zmieniła się ustawa i Bajtazar od tego momentu chce wybudować v-ty najtańszy z kolei plan budowy dróg ($1 \leq v \leq 10^{18}$). Jeżeli natomiast $c =$ `D`, to znaczy, że Bajtazar prosi Cię o odpowiedź na pytanie, czy w aktualnie rozpatrywanym planie zostanie wybudowana droga o numerze v ($1 \leq v \leq m$), tzn. ta o koszcie 2^v.

Wyjście

W pierwszym wierszu standardowego wyjścia wypisz opis sieci dróg, którą powinien zbudować Bajtazar, zanim mieszkańcy zmienią zdanie – możesz założyć, że taka sieć zawsze będzie istniała. Opis powinien składać się z m liczb oddzielonych pojedynczymi odstępami, przy czym i-ta liczba powinna być równa 1, jeśli i-ta droga

ma zostać wybudowana, a 0, jeśli nie należy jej budować. W kolejnych wierszach, dla każdego zapytania typu `D` powinieneś odpowiedzieć na pytanie Bajtazara. Jeżeli żądany plan dróg nie istnieje – tj. przestał istnieć po którejś ze zmian poglądów Bajtoczan albo planów jest za mało i wszystkie zostaną odrzucone na mocy ustawy – powinieneś wypisać -1. Jeżeli natomiast szukany plan dróg istnieje, to powinieneś wypisać 1, jeżeli wskazana droga powinna zostać wybudowana w aktualnie rozpatrywanym planie, lub 0 w przeciwnym przypadku.

Przykład

Dla danych wejściowych:

```
3 3
1 2
2 3
3 1
1 1 0
1
7
D 1
M 2
D 1
M 3
D 2
K 2
D 2
```

poprawnym wynikiem jest:

```
1 0 0
1
-1
1
0
```

Wyjaśnienie do przykładu: Są trzy miasta i trzy drogi tworzące cykl. Najtańszy plan budowy dróg, w którym liczba dróg wychodzących z miast 1 i 2 jest nieparzysta, a z miasta 3 parzysta, zawiera jedynie drogę 1–2 (o koszcie 2). Nie istnieje plan budowy dróg, w którym jest dokładnie jedno miasto, z którego wychodzi nieparzysta liczba dróg. Istnieją dokładnie dwa plany budowy dróg, w których liczba dróg wychodzących z miast 1 i 3 jest nieparzysta (a z miasta 2 parzysta): tańszy z tych planów zawiera drogi 1–2 i 2–3 (o koszcie $2+4=6$), droższy z nich składa się z drogi 1–3 (o koszcie 8).

Testy „ocen”:

`1ocen:` $n=6$, $m=15$, $q=50$, drogi między każdą parą miast, brak zapytań typu `K`;

`2ocen:` $n=10$, $m=10$, $q=84$, drogi ustawione w cykl, w momencie zadawania zapytania typu `D` są zawsze dokładnie dwa możliwe plany budowy;

`3ocen:` $n=101$, $m=150$, 50 trójkątów, każdy następny ma jeden wierzchołek wspólny z poprzednim, wszystkie wymagania parzyste, $k=2^{50}$, brak zapytań.

Ocenianie

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Warunki	Liczba punktów
1	$k=1$, $q=0$ (pytamy o najtańszy plan; nie ma żadnych zapytań)	32
2	$q=0$ (nie ma żadnych zapytań)	25
3	wszystkie zapytania są typu `M` lub `D`	30
4	brak dodatkowych warunków	13