یه حالت جدول طور

- محدودیت زمان: ۱.۵ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

یک جدول n در m داریم که در بعضی از خانه‌های جدول دیوار وجود دارد و بقیه‌ی خانه‌ها خالی هستند. دو خانه از جدول به هم متصل هستند اگر جفتشان خالی بوده و یک ضلع مشترک داشته باشند. دو خانه از جدول به هم مسیر دارند اگر دنباله‌ای از خانه‌ها وجود داشته باشد که شروع و پایان این دنباله دو خانه‌ی اولیه بوده و هر دو خانه‌ی متوالی این دنباله بهم متصل باشند. جدول به شما داده شده؛ از شما q پرسش پرسیده می‌شود که آیا دو خانه از جدول به هم مسیر دارند یا خیر.

ورودی

خط اول شامل دو عدد n و m است که با فاصله از هم جدا شده‌اند. در n خط بعدی رشته‌ای به طول m آمده است که اگر در rامین خط حرف cام برابر با `.` باشد، یعنی خانه‌ی سطر rام و ستون cام جدول خالی است و اگرنه برابر با `#` باشد، یعنی در خانه‌ی سطر rام و ستون cام جدول دیوار وجود دارد. سپس در خط بعد q آمده و در q خط بعدی هر کدام چهار عدد $r1$ و $c1$ و $r2$ و $c2$ با فاصله از هم آمده که یعنی آیا خانه‌ی سطر $r1$ام و ستون $c1$ام جدول به خانه‌ی سطر $r2$ام و ستون $c2$ام جدول مسیر دارد یا نه.

$$1 \leq n, m \leq 500$$

$$1 \leq q \leq 10^5$$

خروجی

برنامه شما باید در q خط جواب هر پرسش را چاپ کند. (اگر مسیر دارند `YES` و اگر ندارند `NO`).

مثال

ورودی نمونه

```
4 4
.#.#
.#.#
.#..
###.
3
3 1 1 1
1 1 4 4
4 4 3 3
```

خروجی نمونه

```
YES
NO
YES
```

توضیح اضافه شود.

بهرس

- محدودیت زمان: ۱.۵ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

یک گراف همبند ساده‌ی n راسی و m یالی داریم. می‌خواهیم کمترین تعداد یال را نگه داشته و بقیه یال‌ها را حذف کنیم به صورتی که فاصله‌ی رئوس از راس ۱ تغییر نکند. برنامه‌ای بنویسید که این کار را انجام دهد.

ورودی

خط اول ورودی شامل دو عدد طبیعی n و m می‌شود که با فاصله از هم آمده‌اند. در m خط بعدی، در هر خط مشخصات یک یال به صورت دو عدد v و u که نشان دهنده‌ی رئوس دو سر یال هستند، با فاصله از هم آمده است.

$$1 \leq n, m \leq 10^5$$

$$1 \leq v \neq u \leq n$$

تضمین می‌شود گراف ورودی ساده و همبند باشد.

خروجی

در خط اول k، تعداد یال‌های حفظ شده چاپ شود و در خط بعدی k عدد، اندیس‌های هر یال‌های حفظ شده با فصله از هم چاپ شود. همه‌ی جواب‌های درست پذیرفته می‌شود.

مثال

ورودی نمونه

```
3 3
```

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

خروجی نمونه

```
2
1 2
```

توضیح اضافه شود.

اولین تشریحی

یک جدول درهم تشکیل شده از $m = 11$ خانه (*slot*) را در نظر بگیرید و فرض کنید که کلیدها (که مقادیری صحیح و نامنفی‌اند) به وسیله تابع درهم ساز (*hash function*) زیر به خانه های جدول *map* شده اند:

```
int h1(int key)
{
    int x = (key + 7) * (key + 7);
    x = x / 16;
    x = x + key;
    x = x % 11;
    return x;
}
```

یک راه مقابله با مشکل *collision* در جداول درهم پیاده سازی `open addressing` است.فرض کنید برای پیاده سازی `open addressing` بخواهیم از کاوش درجه دو استفاده کنیم و از تابع کاوش (*probe function*) زیر استفاده کنیم:

$$\frac{x^2 + x}{2}$$

در این صورت مکان هر کلید با تابع زیر مشخص می‌شود:

$$h(k, i) = (h1(k) + \frac{i^2 + i}{2})\ mod\ m$$

کلیدهای داده شده به ترتیب داده شده وارد جدول درهم کنید و اطلاعات خواسته شده در جدول زیر را کامل کنید و در آخر مشخص کنید در جدول نهایی داخل هر *slot* چه کلیدی قرار گرفته است.

key	home slot	probe sequence
43		

key	home slot	probe sequence
23		
1		
0		
15		
31		
4		
7		
11		
3		

دومین تشریحی

فرض کنید یک جدول درهم را پیاده سازی می‌کنیم و می‌خواهیم بین `open addressing` و `chaining` یکی را برای پیاده سازی جدول انتخاب کنیم. برای دو استراتژی از یک *hash function* و *hash table size* برابر استفاده می‌کنیم. به سوالات زیر پاسخ دهید:

الف) تصادف اولیه (*primary collision*) زمانی رخ می‌دهد که دو کلید به یک *slot* توسط تابع درهم ساز نگاشته (*map*) شوند. (در واقع *hash function* یک مقدار یکسان به ازای دو ورودی مختلف می‌دهد.) به ازای کدام یک از دو روش پیاده سازی تعداد *primary collision* کمتری داریم؟

ب) یکی از اشکالات استفاده از `chaining` و `linear search` ، یک لیست پیوندی برای پیدا کردن یک *record* است که هزینه آن زیاد است. آیا استفاده از `open addressing` و کاوش خطی جستجو را بهینه‌تر می‌کند؟ (در صورتی که بدانیم تعداد کم ولی قابل توجه‌ای *collision* داریم.)

ج) اگر از کاوش درجه دو استفاده کنیم چطور؟

د) به طور کلی چه زمانی از `open addressing` و چه زمانی از `chaining` استفاده می‌کنیم؟