



عروسک مکانیکی

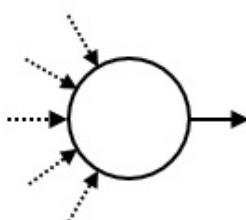
یک عروسک مکانیکی، عروسکی است که دنباله‌ی مشخصی از حرکات را به طور خودکار تکرار می‌کند. در ژاپن، عروسک‌های مکانیکی زیادی از زمان قدیم تولید شده است.

حرکات یک عروسک مکانیکی توسط یک مدار کنترل می‌شود که از **قطعه‌های مختلفی** ایجاد شده است. قطعه‌ها توسط لوله به هم وصل شده‌اند. هر قطعه یک یا دو **خروجی** و تعداد دلخواهی (شاید صفر) **ورویدی** دارد. هر لوله، خروجی یک قطعه را به ورودی همان قطعه و یا ورودی قطعه‌ی دیگری متصل می‌کند. به هر ورودی و خروجی دقیقاً یک لوله متصل است.

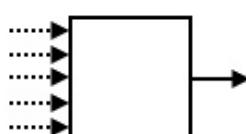
برای توضیح اینکه عروسک چگونه حرکت می‌کند، فرض کنید یک توپ بر روی یکی از قطعه‌ها قرار گرفته است. این توپ داخل مدار حرکت می‌کند. در هر گام از حرکتش، از یکی از خروجی‌های قطعه‌ای که بر روی آن قرار دارد خارج شده، درون لوله‌ای که به این خروجی متصل است حرکت کرده و وارد قطعه‌ای که در انتهای این لوله است می‌شود.

سه نوع قطعه وجود دارد: **مبدا**, **محرك** و **سوئیچ**. دقیقاً یک مبدأ، M محرك و S سوئیچ در مدار وجود دارد (S می‌تواند صفر باشد). شما باید مقدار S را تعیین کنید. هر قطعه یک شماره سریال یکتا دارد.

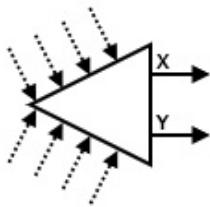
در ابتدا توپ بر روی قطعه‌ی مبدأ قرار می‌گیرد. این قطعه یک خروجی دارد. شماره سریال این قطعه ۰ است.



یک محرك باعث می‌شود وقتی توپ به آن وارد می‌شود، عروسک حرکت خاصی انجام دهد. هر محرك یک خروجی دارد. شماره سریال محرك‌ها از ۱ تا M است.



هر سوئیچ دو خروجی دارد که آن‌ها را 'X' و 'Y' می‌نامیم. **وضعیت هر سوئیچ** 'X' یا 'Y' است. پس از این‌که توپ وارد یک سوئیچ می‌شود، آن را از طریق خروجی‌ای که مطابق با وضعیت کنونی آن است ترک می‌کند. پس از آن، سوئیچ به وضعیت دیگر تغییر وضعیت می‌دهد. در ابتدا وضعیت تمامی سوئیچ‌ها 'X' است. شماره سریال سوئیچ‌ها از ۱ تا S است.



به شما مقدار M (تعداد سوئیچ‌ها) داده می‌شود. همچنین به شما یک دنباله‌ی A به طول N داده می‌شود که هر کدام از اعضایش شماره‌ی سریال یکی از سوئیچ‌ها است. هر سوئیچ ممکن است هر تعداد بار (شاید صفر) در A ظاهر شود. وظیفه‌ی شما این است که مداری بسازید که شرایط زیر را برقرار کند:

- توب پس از چند گام به مبدأ برگردد.
- وقتی توب برای اولین بار به مبدأ برمی‌گردد، وضعیت همه‌ی سوئیچ‌ها 'X' باشد.
- توب پس از وارد شدن به دقیقاً N حرکت برای اولین بار به مبدأ برگردد. شماره سریال این سوئیچ‌ها به ترتیب ورود توب به آن‌ها A_0, A_1, \dots, A_{N-1} باشد.
- فرض کنید P جمع تعداد تغییر وضعیت‌های سوئیچ‌ها قبل از اینکه توب برای اولین بار به مبدأ برگردد باشد. مقدار P نباید بیشتر از 20 000 000 شود.

در عین حال شما نمی‌خواهید از تعداد زیادی سوئیچ استفاده کنید.

جزئیات پیاده‌سازی

شما باید تابع زیر را پیاده‌سازی کنید.

```
create_circuit(int M, int[] A)
```

- M : تعداد حرکت‌ها.
- A : آرایه‌ای به طول N که دنباله‌ی ترتیب ورود توب به حرکت‌ها را نشان می‌دهد.
- این تابع دقیقاً یکبار فراخوانی می‌شود.
- توجه کنید که مقدار N همان طول آرایه‌ی A است و می‌توان آن را به همان صورت که در نکات پیاده‌سازی گفته شده است به دست آورد.

برنامه شما باید برای پاسخگویی تابع زیر را فراخوانی کند.

```
answer(int[] C, int[] X, int[] Y)
```

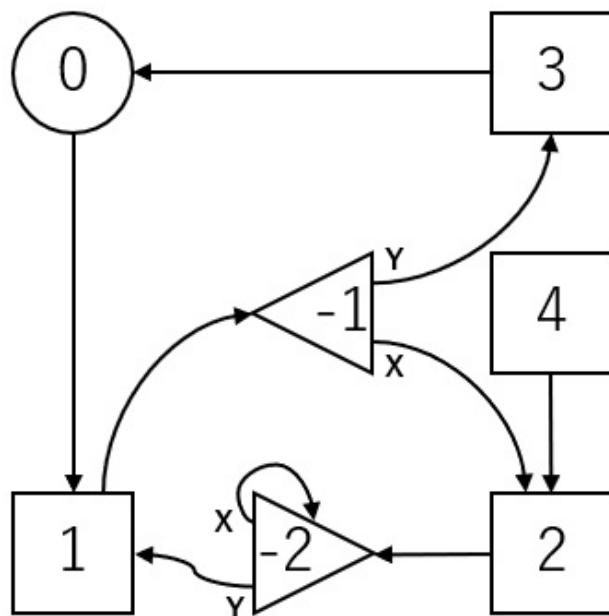
- C : آرایه‌ای به طول $1 + M$. خروجی قطعه‌ی $i^{\text{ام}}$ ($0 \leq i \leq M$) به قطعه‌ی $C[i]$ متصل است.
- X, Y : آرایه‌هایی با طول برابر. طول این آرایه‌ها یعنی S همان تعداد سوئیچ‌ها است. برای سوئیچ $j^{\text{ام}}$ خروجی 'X' آن به قطعه‌ی $[1 - j]X$ و خروجی 'Y' آن به قطعه‌ی $[1 - j]Y$ متصل است.
- تمامی اعضای C, X و Y باید عدد صحیحی بین $-S$ و M باشند.
- S باید حداقل 400 000 باشد.
- این تابع باید دقیقاً یکبار فراخوانی شود.
- مدار مشخص شده با C, X و Y باید تمامی شرط‌های صورت مسئله را برقرار کند.

اگر هر کدام از شرط‌های بالا برقرار نشود، برنامه‌ی شما به صورت **Wrong Answer** داوری می‌شود. در غیر این

صورت برنامه‌ی شما به صورت Accepted داوری می‌شود و نمره‌ی شما بر اساس S محاسبه می‌شود (زیرمسئله‌ها را نگاه کنید).

مثال

فرض کنید $A = [1, 2, 1, 3]$ و $N = 4, M = 4$. ارزیاب، تابع $\text{create_circuit}(4, [1, 2, 1, 3])$ را فراخوانی می‌کند.



شكل بالا مداری را نشان می‌دهد که با فراخوانی $\text{create_circuit}(4, [1, 2, 1, 3])$ ساخته شده است. اعداد داخل شکل، شماره‌ی سریال‌های قطعه‌ها هستند.

دو سوئیچ استفاده شده است. بنابراین $S = 2$.

در ابتدا وضعیت سوئیچ‌های 1-X و 2-Y هر دو 'X' است.

توب به صورت زیر حرکت می‌کند:

$$0 \rightarrow 1 \rightarrow -1 \xrightarrow{X} 2 \rightarrow -2 \xrightarrow{X} -2 \xrightarrow{Y} 1 \rightarrow -1 \xrightarrow{Y} 3 \rightarrow 0$$

- وقتی توب اولین بار به سوئیچ 1-X وارد می‌شود، وضعیت آن 'X' است. بنابراین توب به محرک 2 می‌رود. سپس وضعیت سوئیچ 1-X به 'Y' تغییر می‌کند.
- وقتی توب برای بار دوم به سوئیچ 1-X وارد می‌شود، وضعیت آن 'Y' است. بنابراین توب به محرک 3 می‌رود. پس از آن وضعیت سوئیچ 1-X به 'X' تغییر پیدا می‌کند.

وقتی توب اولین بار به قطعه‌ی مبدأ برمی‌گردد، به محرک‌های 1, 2, 1, 3 وارد شده بوده است. وضعیت‌های سوئیچ‌های 1-X و 2-Y هر دو 'X' است. مقدار P برابر با 4 است. بنابراین مدار توصیف شده، تمامی شرایط را برقرار کرده است.

فایل sample-01-in.txt موجود در بسته‌ی فشرده‌ی پیوست مربوط به همین مثال است. مثال‌های دیگری نیز

در این بسته وجود دارند.

محدودیت‌ها

- $1 \leq M \leq 100\,000$
- $1 \leq N \leq 200\,000$
- $(0 \leq k \leq N - 1) \quad 1 \leq A_k \leq M$

زیرمسئله‌ها

نمره و محدودیت‌های هر مورد آزمون به صورت زیر است:

1. (۲ نمره) به ازای هر i ($1 \leq i \leq M$), عدد صحیح i حداکثر یک بار در دنباله‌ی A_0, A_1, \dots, A_{N-1} ظاهر شده است.
2. (۴ نمره) به ازای هر i ($1 \leq i \leq M$), عدد صحیح i حداکثر دو بار در دنباله‌ی A_0, A_1, \dots, A_{N-1} ظاهر شده است.
3. (۱۰ نمره) به ازای هر i ($1 \leq i \leq M$), عدد صحیح i حداکثر ۴ بار در دنباله‌ی A_0, A_1, \dots, A_{N-1} ظاهر شده است.
4. (۱۰ نمره) $N = 16$
5. (۱۸ نمره) $M = 1$
6. (۵۶ نمره) بدون محدودیت اضافی

به ازای هر مورد آزمون، اگر برنامه‌ی شما به صورت **Accepted** داوری شود، نمره شما با توجه به مقدار S محاسبه می‌شود.

- اگر $S \leq N + \log_2 N$, شما نمره کامل را به ازای این مورد آزمون به دست می‌آورید.
- برای هر مورد آزمون در زیرمسئله‌های ۵ و ۶ اگر $N + \log_2 N < S \leq 2N$, شما نمره‌ی جزئی دریافت می‌کنید. این نمره برای یک مورد آزمون برابر $0.5 + 0.4 \times \left(\frac{2N - S}{N - \log_2 N} \right)^2$ ضرب در نمره‌ی مربوط به زیرمسئله‌ی آن است.
- در غیر این صورت نمره‌ی شما ۰ است.

توجه کنید که نمره‌ی شما برای هر زیرمسئله برابر با کمترین نمره‌ی دریافتی در موارد آزمون آن زیرمسئله است.

ارزیاب نمونه

ارزیاب نمونه ورودی را از ورودی استاندارد در قالب زیر می‌خواند.

- خط ۱: $N \ M$
- خط ۲: $A_{N-1} \dots A_1 \ A_0$

ارزیاب نمونه سه خروجی تولید می‌کند.

ابتدا ارزیاب نمونه جواب شما را در فایلی به نام `out.txt` در قالب زیر چاپ می‌کند.

• خط $S : 1$

• خط $C[i] : (0 \leq i \leq M) 2 + i$

• خط $X[j - 1] Y[j - 1] : (1 \leq j \leq S) 2 + M + j$

سپس ارزیاب نمونه حرکات توب را شبیه‌سازی می‌کند و شماره سریال‌های قطعه‌هایی که توب به آن‌ها وارد شده است را به ترتیب در فایلی به نام `log.txt` چاپ می‌کند.

در آخر ارزیاب نمونه نتیجه‌ی جواب شما را در خروجی استاندارد چاپ می‌کند.

• اگر برنامه‌ی شما به صورت **Accepted** داوری شود، ارزیاب S و P را در قالب P چاپ می‌کند.

• اگر برنامه‌ی شما به صورت **Wrong Answer** داوری شود، ارزیاب `MSG` را چاپ می‌کند. معانی `MSG` در زیر آمده است:

◦ تابع `answer` دقیقاً یک بار فراخوانی نشده است.

◦ طول C برابر $M + 1$ نیست یا طول X و Y متفاوت است.

◦ S over 400 000 switches

◦ عضوی در C , X یا Y وجود دارد که کوچکتر از S – یا بزرگتر از M است.

◦ over 20 000 000 inversions: توب پس از 20 000 000 بار تغییر وضعیت سوئیچ‌ها به قطعه‌ی مبدأ برگشته است.

◦ 'Y' state: هنگامی که توب برای اولین بار به قطعه‌ی مبدأ برگشته سوئیچی در وضعیت 'Y' وجود دارد.

◦ wrong motion: حرکت‌هایی که باعث حرکات شده‌اند با دنباله A تفاوت دارند.

توجه کنید در صورتی که برنامه‌ی شما نتیجه‌ی `Wrong Answer` را دریافت کند، ارزیاب نمونه `out.txt` یا `log.txt` را نمی‌سازد.