D. Decorações de Jardim

Nome do problema	Decorações de jardim
Tempo Limite	7 seconds
Limite de memória	1 gigabyte

Todos os dias, quando vai para a escola e volta para casa, Detje caminha por uma rua com N casas, numeradas de 0 a $N-1$. Atualmente a casa i é habitada pela pessoa i. Para mudar de cenário, os moradores decidiram trocar de casa entre si. A pessoa que irá se mudar para a casa i é a pessoa a_i (que atualmente mora na casa a_i).

Cada casa tem uma estátua de pássaro no jardim. As estátuas têm dois estados possíveis: ter as asas *abertas* (como se o pássaro estivesse voando) ou *fechadas* (como se estivesse no chão).

Os moradores têm preferências muito fortes sobre a aparência de suas estátuas de pássaros. Atualmente, o pássaro na frente da casa i está no estado preferencial do morador i. Os moradores recusam-se a mudar-se para uma casa a menos que a sua estátua esteja colocada na sua configuração preferida. Detje quer ajudá-los a organizar as estátuas de pássaros para que possam se mudar.

Para isso, ela faz o seguinte: sempre que caminha pela rua (seja no caminho para a escola ou voltando para casa), ela observa os pássaros pelos quais passa e possivelmente ajusta algumas das estátuas (abrindo ou fechando suas asas). Como seus dias na escola e em casa são muito ocupados, ela não se lembra do estado dos pássaros que viu nas caminhadas anteriores.

Felizmente, ela anotou em uma lista $a_0, a_1, ..., a_{N-1}$, então ela sabe qual morador está se mudando para onde.

Ajude Detje a criar uma estratégia que lhe diga quais pássaros manipular para ajustar as estátuas ao gosto dos moradores. Ela pode caminhar pela rua no máximo 60 vezes, mas para obter uma pontuação mais alta, ela deve caminhar menos vezes pela rua.

Implementação

Este é um problema de execução múltipla, o que significa que seu programa será executado várias vezes.

Em cada execução, você deve primeiro ler uma linha com dois inteiros w e N, o índice do passeio e o número de casas. Na primeira execução do seu programa $w = 0$, na segunda $w = 1$ e assim por diante (mais detalhes são explicados abaixo).

Na segunda linha de entrada existem N inteiros $a_0, a_1, ..., a_{N-1}$, o que significa que a pessoa que irá se mudar para a casa i está atualmente morando na casa a_i. Os a_is formam uma *permutação*: ou seja, cada número de 0 a $N - 1$ aparece exatamente uma vez na lista de a_is. Observe que um morador pode optar por não se mudar; isto é, $a_i = i$ é permitido.

Os moradores só trocam de casa uma vez. Isso significa que para cada caso de teste fixo, o valor de N e a lista de a_is serão os mesmos para todas as execuções do seu programa.

Primeira Execuçao.

Para a primeira execução do seu programa, $w = 0$. Nesta execução, você deve simplesmente gerar um único inteiro W ($0 \leq W \leq 60$), o número de vezes que você deseja que Detje passe pelas casas. Seu programa deve então sair. Depois disso, seu programa será executado novamente mais W vezes.

Execuções subsequentes.

Na próxima execução do seu programa, $w = 1$; no seguinte $w = 2$; e assim por diante até a execução final ser $w = W$.

Depois de ler w, N e $a_0, a_1, ..., a_{N-1}$, Detje começa a caminhar pela rua.

- Se w for ímpar, Detje caminha de sua casa até a escola e passará pelas casas na ordem $0, 1, ..., N - 1$.

  Seu programa agora deve ler uma linha com b_0, seja 0 (fechado) ou 1 (aberto), o estado atual da estátua na frente da casa 0. Depois de ler b_0, você deve gerar uma linha com 0 ou 1, o novo valor para o qual deseja definir b_0.

  Então seu programa deverá ler uma linha com b_1, o estado da estátua na frente da casa 1; e mostre o novo valor de b_1. Isto continua para cada uma das N casas. Depois de passar pela casa final (ou seja, você lê e escreve b_{N-1}), seu programa deve sair.

  Observe que seu programa só pode ler o próximo valor b_{i+1} depois de você ter escrito o novo valor de b_i.

- Se w for par, Detje caminha da escola até sua casa e passará pelas casas na ordem inversa $N - 1, N - 2, ...0$.

  O processo é o mesmo de quando w é ímpar, exceto que você começa lendo e escrevendo b_{N-1}, depois b_{N-2}, e assim por diante até b_0.

Quando $w = 1$, os valores de entrada $b_0, b_1, ..., b_{N-1}$ são o estado original das estátuas de pássaros. (que também são os estados preferidos dos moradores). Quando $w > 1$, os valores de entrada $b_0, b_1, ..., b_{N-1}$ para o seu programa serão os definidos pela execução anterior do seu programa.

Observe que seu programa só pode ler o próximo valor b_{i+1} depois de você ter escrito o valor de b_i.

- Se w for par, Detje caminha da escola até sua casa e passará pelas casas na ordem inversa $N-1, N-2, ...0$. Ou seja, você começa lendo e escrevendo b_{N-1}, depois b_{N-2}, e assim por diante até b_0.

Quando $w = 1$, os valores de entrada $b_0, b_1, ..., b_{N-1}$ são o estado original das estátuas de pássaros. Quando $w > 1$, os valores de entrada $b_0, b_1, ..., b_{N-1}$ para o seu programa serão os definidos pela execução anterior do seu programa.

No fim, após a execução final do seu programa, o valor de b_i deve ser igual ao valor original de b_{a_i} para todos os i, caso contrário você receberá o veredicto Resposta errada.

Detalhes

Se a *soma* dos tempos de execução das $W + 1$ execuções separadas do seu programa exceder o limite de tempo, sua submissão será julgada como Tempo Limite Excedido.

Certifique-se de liberar a saída padrão após imprimir cada linha, caso contrário seu programa poderá ser julgado como Tempo Limite Excedido. Em Python, isso acontece automaticamente desde que você use `input()` para ler as linhas. Em C++, `cout << endl;` libera além de imprimir uma nova linha; se estiver usando printf, use `fflush(stdout)`.

Restrições e pontuação

- $2 \leq N \leq 500$.
- Você pode usar no máximo $W \leq 60$ rodadas.

Sua solução será testada em um conjunto de grupos de teste, cada um valendo um número de pontos. Cada grupo de teste contém um conjunto de casos de teste. Para obter os pontos para um grupo de teste, você precisa resolver todos os casos de teste no grupo de teste.

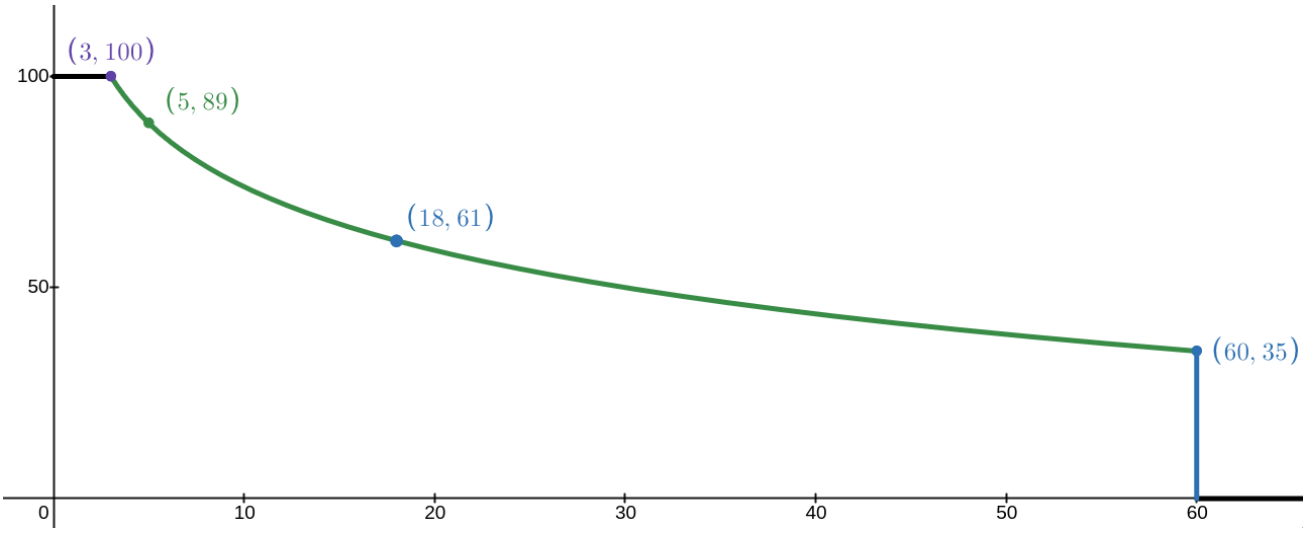
Grupo	Pontuação máxima	Limites
1	10	$N = 2$
2	24	$N \leq 15$
3	9	$a_i = N - 1 - i$
4	13	$a_i = (i + 1) \bmod N$
5	13	$a_i = (i - 1) \bmod N$
6	31	Sem restrições adicionais

Para cada grupo de teste que seu programa resolver corretamente, você receberá uma pontuação baseada na seguinte fórmula:

$$\text{score} = S_g \cdot \left(1 - \tfrac{1}{2} \log_{10}(\max(W_g, 3)/3)\right),$$

onde S_g é a pontuação máxima para o grupo de teste e W_g é o valor máximo de W usado para qualquer caso de teste no grupo de teste. Sua pontuação para cada grupo de teste será arredondada para o número inteiro mais próximo.

O gráfico abaixo mostra a quantidade de pontos, em função de W, que seu programa obterá se resolver todos os grupos de teste com o mesmo valor de W. Em particular, para obter uma pontuação de 100 pontos neste problema, você precisa resolver todos os casos de teste com $W \leq 3$.



Ferramenta de teste

Para facilitar o teste da sua solução, disponibilizamos uma ferramenta simples que você pode baixar. Consulte “anexos” na parte inferior da página de problemas do Kattis. A ferramenta é de

uso opcional. Observe que o programa oficial de avaliação do Kattis é diferente da ferramenta de teste.

Para usar a ferramenta, crie um arquivo de entrada, como “sample1.in”, que deve começar com um número N seguido por uma linha com N números especificando a permutação, e outra linha com N bits (0 ou 1) especificando os estados iniciais dos pássaros. Por exemplo:

```
6
1 2 0 4 3 5
1 1 0 0 1 0
```

Para Python , use `solution.py` (normalmente roda como `pypy3 solution.py`):

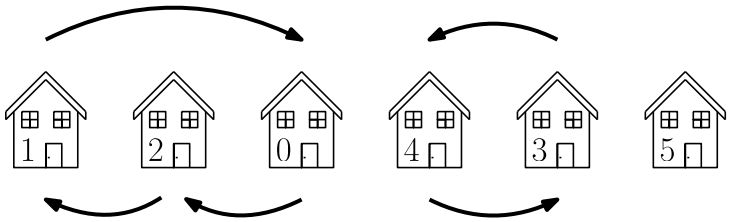
```
python3 testing_tool.py pypy3 solution.py < sample1.in
```

Para C++ , primeiro complete com (e.g. com `g++ -g -O2 -std=gnu++20 -static solution.cpp -o solution.out`) e rode:

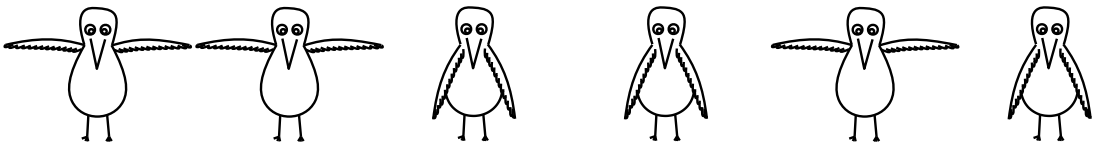
```
python3 testing_tool.py ./solution.out < sample1.in
```

Exemplo

Na amostra, temos a seguinte permutação de pessoas nas casas:

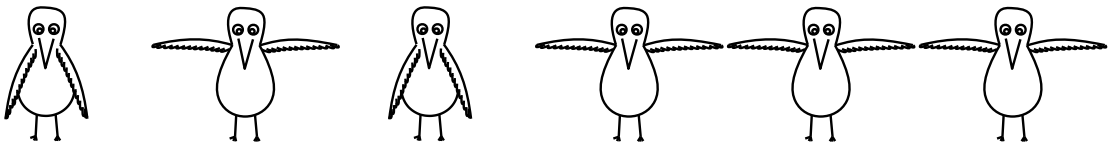


Na primeira vez que o programa de exemplo é executado (com $w = 0$), ele gera $W = 2$, o que significa que Detje caminhará pela rua duas vezes (e o programa será executado mais duas vezes). Antes da primeira caminhada, os pássaros nos jardins ficam assim:



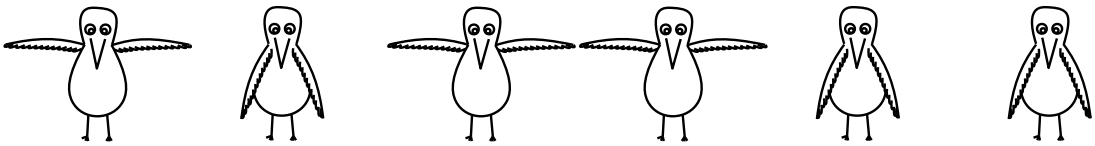
Então o programa é executado com $w = 1$, indicando a primeira caminhada de Detje. Ela passa pelos pássaros um por um, começando pela esquerda, e possivelmente muda seu estado. O programa de exemplo deve exibir o estado do i^{o} pássaro antes de vermos o $(i + 1)^{\text{o}}$ pássaro.

Depois que Detje chegou à escola, o estado dos pássaros ficou assim:



Na execução final do programa (com $w = 2$), Detje volta da escola para casa. Lembre-se que neste caso ela irá percorrer os pássaros da direita para a esquerda e processá-los nesta ordem! Isso significa que ela precisa determinar o estado do i^{o} pássaro antes de ver o $(i-1)^{\text{o}}$ pássaro.

Depois que ela termina sua caminhada, os pássaros ficam assim:



Na verdade, esta é a configuração correta. Por exemplo, a estátua do pássaro 3 (ou seja, a quarta da esquerda) está aberta (agora $b_3 = 1$), o que está correto, pois a pessoa 4 estará se mudando para lá ($a_3 = 4$) e ela originalmente tinha uma estátua de pássaro aberta (originalmente $b_4 = 1$).

saída do gerador	sua saída
0 6	
1 2 0 4 3 5	
	2

saída do gerador	sua saída
1 6	
1 2 0 4 3 5	
1	
	0
1	
	1
0	
	0
0	
	1
1	
	1
0	
	1

saída do gerador	sua saída
2 6	
1 2 0 4 3 5	
1	
	0
1	
	0
1	
	1
0	
	1
1	
	0
0	
	1