Zadanie: WYS
Wysoce poszukiwana permutacja

polish

ONTAK, dzień 3. Dostępna pamięć: 256 MB. Limit czasu: 3 s. *04.07.2024*

Poszukujesz pewnej permutacji p, czyli ustawienia ciągu $(0, 1, \ldots, N-1)$ w pewnej kolejności $(p[0], p[1], \ldots, p[N-1])$. Możesz dowiedzieć się czegoś o niej tylko na jeden sposób: dzielisz wszystkie elementy $\{0, 1, \ldots, N-1\}$ na pewne zbiory $A_1, \ldots, A_k$ – liczbę k i podział możesz wybrać wedle uznania – i dostajesz w odpowiedzi zbiory $p(A_1), p(A_2), \ldots, p(A_k)$, jednak w innej, dowolnej kolejności. (Zbiór $p(A)$ to zbiór wszystkich wartości $p(x)$ dla $x \in A$. Zauważ, że skoro p jest permutacją, musi mieć zawsze taką samą liczność, co A.)

Interakcja

To jest zadanie interaktywne. Twój program powinien komunikować się z programem sprawdzającym, zadawać mu pytania (opisanego powyżej rodzaju), odczytywać odpowiedzi, a na końcu podać permutację p. Komunikacja następuje poprzez standardowe wejście i wyjście, w następujący sposób:

- Najpierw wczytujesz ze standardowego wejścia trzy liczby całkowite T (numer podzadania), Q (liczba zapytań, którą Twój program powinien wykonać, aby dostać maksymalną liczbę punktów) oraz N (liczność zbioru, na którym działa permutacja).
- Potem Twój program powinien zadawać pytania wypisując na standardowe wejście kolejno następujące wiersze:
  ? k

  s_1 $A_1[1]$ ... $A_1[s_1]$

  $\vdots$

  s_k $A_k[1]$ ... $A_k[s_k]$

  Jest to zapytanie o podział $\{0, \ldots, N-1\}$ na k zbiorów $A_1, A_2, \ldots, A_k$, gdzie zbiór numer j liczy sobie s_j elementów oznaczonych $A_j[1], A_j[2], \ldots, A_j[s_j]$. Zbiory A_j nie mogą mieć wspólnych elementów, a w sumie muszą zawierać wszystkie elementy $\{0, \ldots, N-1\}$ (w szczególności suma wszystkich s_j musi wynosić n).
- Na takie pytanie sprawdzarka odpowiada w następującym formacie:

  t_1 $B_1[1]$ ... $B_1[t_1]$

  $\vdots$

  t_k $B_k[1]$ ... $B_k[t_k]$,

  Jest to podział tego samego zbioru $\{0, \ldots N-1\}$ na $B_1 \cup B_2 \cup \ldots \cup B_k$. Zbiory $B_1, B_2, \ldots, B_k$ to $p(A_1), p(A_2), \ldots, p(A_k)$, ale niekoniecznie w tej samej kolejności.
- Po zadaniu wszystkich zapytań należy podać odpowiedź w formacie:

  ! $P[0]$ ... $P[N-1]$

Po każdym zapytaniu i po każdej odpowiedzi należy wywołać polecenie `cout.flush()`, lub `fflush(stdout)` jeżeli używasz `printf`. Należy przestrzegać podanego wyżej formatu wejścia i wyjścia, łącznie z białymi znakami i podziałem na wiersze – w przeciwnym przypadku program może zakończyć się werdyktem *Przekroczenie limitu czasu* zamiast *Błędna odpowiedź*.

Przykładowa interakcja:

Interaktor	Program	Wyjaśnienie
0 100 3		$N = 3, Q = 100$
	? 2 1 0 2 1 2	zapytanie o $\{A_1, A_2\} = \{\{0\}, \{1, 2\}\}$
1 2 2 0 1		możemy wywnioskować, że $P[0] = 2$
	? 2 1 1 2 0 2	
1 0 2 1 2		
	! 2 0 1	odpowiedź to $\{2, 0, 1\}$

Testy przykładowe:

`0a:` $N = 3$, $P = \{2, 0, 1\}$

`0b:` $N = 5$, $P = \{0, 3, 2, 1, 4\}$

`0c:` $N = 100$, $P = \{99, 98, \ldots 1, 0\}$

Ocenianie

W każdym podzadaniu można zadać maksymalnie 100 zapytań. Jeżeli podzadanie warte jest S punktów, Q jest liczbą zapytać podaną na wejściu, natomiast Twój program zada (maksymalnie wśród wszystkich testów tego podzadania) R pytań, otrzyma on $S \cdot \frac{log(Q+1)}{log(R+1)}$ punktów. (W szczególności 100% punktów za Q zapytań, a 50% punktów za $Q^2 + 2Q$ zapytań.)

Podzadanie	Ograniczenia	Punkty
1	$3 \leq N \leq 6$, $Q = 100$	8
2	$3 \leq N \leq 100$, $Q = 100$	9
3	$3 \leq N \leq 100$, $Q = 3$	17
4	$N = 4\,095$, $Q = 2$	18
5	$100 < N \leq 5\,000$, $Q = 3$	23
6	$N = 4\,098$, $Q = 2$	12
7	$100 < N \leq 5\,000$, $Q = 2$	13

Eksperymenty

W zakładce *Pliki* możesz pobrać przykładową sprawdzarkę. Wywołuje się ją komendą

```
python3 run.py 1 wyssoc [rozwiazanie] [test]
```

przy czym plik `wyssoc.cpp` oraz rozwiązanie muszą być skompilowane.

Завдання: WYS
Wysoce poszukiwana permutacja

ukrainian

ONTAK, день 3. Обмеження пам'яті: 256 MB. Ліміт часу: 3 s. *04.07.2024*

Ви шукаєте певну перестановку p, тобто розташування послідовності $(0, 1, \ldots, N-1)$ у певному порядку $(p[0], p[1], \ldots, p[N-1])$. Ви можете дізнатися про неї лише одним способом: поділяєте усі елементи $\{0, 1, \ldots, N-1\}$ на певні множини $A_1, \ldots, A_k$ – кількість k та поділ можете обрати на власний розсуд – і отримуєте у відповідь множини $p(A_1), p(A_2), \ldots, p(A_k)$, але в іншому, довільному порядку. (Множина $p(A)$ це множина всіх значень $p(x)$ для $x \in A$. Зауважте, що оскільки p є перестановкою, вона завжди має таку ж чисельність, як і A.)

Взаємодія

Це інтерактивне завдання. Ваша програма повинна взаємодіяти з перевіряючою програмою, задавати їй питання (описаного вище типу), читати відповіді та в кінці подати перестановку p. Взаємодія відбувається через стандартний вхід і вихід, наступним чином:

- Спочатку зчитуєте зі стандартного входу три цілі числа T (номер підзавдання), Q (кількість запитів, яку ваша програма повинна виконати, щоб отримати максимальну кількість балів) та N (чисельність множини, на якій діє перестановка).
- Потім ваша програма повинна задавати питання, виводячи на стандартний вихід послідовно такі рядки:
  ? k

  s_1 $A_1[1]$ ... $A_1[s_1]$

  ⋮

  s_k $A_k[1]$ ... $A_k[s_k]$

  Це запит про поділ $\{0, \ldots, N-1\}$ на k множин $A_1, A_2, \ldots, A_k$, де множина номер j містить s_j елементів позначених $A_j[1], A_j[2], \ldots, A_j[s_j]$. Множини A_j не можуть мати спільних елементів, а загалом повинні містити всі елементи $\{0, \ldots, N-1\}$ (зокрема, сума всіх s_j має дорівнювати n).
- На таке запитання перевіряюча програма відповідає в наступному форматі:

  t_1 $B_1[1]$ ... $B_1[t_1]$

  ⋮

  t_k $B_k[1]$ ... $B_k[t_k]$,

  Це поділ тієї ж самої множини $\{0, \ldots N-1\}$ на $B_1 \cup B_2 \cup \ldots \cup B_k$. Множини $B_1, B_2, \ldots, B_k$ це $p(A_1), p(A_2), \ldots, p(A_k)$, але не обов'язково в тому ж порядку.
- Після задання всіх запитань слід подати відповідь у форматі:

  ! $P[0]$ ... $P[N-1]$

Після кожного запитання і після кожної відповіді слід викликати команду `cout.flush()`, або `fflush(stdout)` якщо використовуєте `printf`. Слід дотримуватися вказаного вище формату входу та виходу, включно з пробілами та поділом на рядки – інакше програма може завершитися вердиктом *Перевищено ліміт часу* замість *Неправильна відповідь*.

Приклад взаємодії:

Інтерактор	Програма	Пояснення
0 100 3		$N = 3, Q = 100$
	? 2 1 0 2 1 2	запит про $\{A_1, A_2\} = \{\{0\}, \{1, 2\}\}$
1 2 2 0 1		можемо зробити висновок, що $P[0] = 2$
	? 2 1 1 2 0 2	
1 0 2 1 2		
	! 2 0 1	відповідь це $\{2, 0, 1\}$

Приклади тестів:

`0a`: $N = 3$, $P = \{2, 0, 1\}$

`0b`: $N = 5$, $P = \{0, 3, 2, 1, 4\}$

`0c`: $N = 100$, $P = \{99, 98, \ldots 1, 0\}$

Оцінювання

У кожному підзавданні можна задати максимум 100 запитів. Якщо підзавдання вартує S балів, Q це кількість запитів поданих на вході, а ваша програма задасть (максимально серед усіх тестів цього підзавдання) R запитів, вона отримає $S \cdot \frac{log(Q+1)}{log(R+1)}$ балів. (Зокрема 100% балів за Q запитів, а 50% балів за Q^2+2Q запитів.)

Підзавдання	Обмеження	Бали
1	$3 \leq N \leq 6$, $Q = 100$	8
2	$3 \leq N \leq 100$, $Q = 100$	9
3	$3 \leq N \leq 100$, $Q = 3$	27
4	$N = 4\,095$, $Q = 2$	23
5	$100 < N \leq 5\,000$, $Q = 3$	33
6	$N = 4\,098$, $Q = 2$	12
7	$100 < N \leq 5\,000$, $Q = 2$	13

Експерименти

У вкладці *Files* ви можете завантажити приклад перевіряючої програми. Викликається вона командою

```
python3 run.py 1 wyssoc [solution] [test]
```

при цьому файл `wyssoc.cpp` та рішення повинні бути скомпільовані.