(PL) Sortowanie XORem

EJOI 2020, dzień 2
5.09.2020

Kod zadania: **xorsort**
Limit czasu: **1 s**
Limit pamięci: **512 MB**

Dana jest liczba naturalna S oraz tablica A indeksowana od 1 składająca się z N nieujemnych liczb całkowitych. Możesz wykonywać operację polegającą na wybraniu pozycji i ($1 \le i \le N$) oraz jednego z jej sąsiadów j ($1 \le j \le N$, $j = i - 1$ lub $j = i + 1$) i zamiany A_i na $(A_i \oplus A_j)$, gdzie $\oplus$ to operacja bitowa XOR. Definicję możesz przeczytać na końcu tej treści zadania.

Twoim celem jest sprawienie, aby po wykonaniu pewnej liczby zmian elementy w tablicy A były posortowane.

- Jeśli $S = 1$, to uzyskana tablica ma być ściśle rosnąca (tzn. $A_i < A_{i+1}$ dla każdego $1 \le i < N$).
- Jeśli $S = 2$, to uzyskana tablica ma być niemalejąca (tzn. $A_i \le A_{i+1}$ dla każdego $1 \le i < N$).

Znajdź dowolny ciąg operacji, który osiąga cel. Nie musisz minimalizować liczby operacji. Musisz jedynie zmieścić się w limicie $40\,000$ operacji.

Wejście

W pierwszym wierszu wejścia znajdują się dwie liczby naturalne N oraz S. W drguim (ostatnim) wierszu wejścia znajduje się N nieujemnych liczb całkowitych A_i (elementy tablicy).

Wyjście

W pierwszym wierszu wyjścia powinna się znaleźć jedna nieujemna liczba całkowita K ($0 \le K \le 40\,000$) – liczba wykonanych operacji. W kolejnych K wierszach powinny się znajdować po dwie liczby opisujące wykonywane operacje w kolejności chronologicznej: pierwsza liczba określa pozycję i elementu zamienianego, a druga liczba określa pozycję j sąsiada zaangażowanego w operację (zgodnie z opisem powyżej).

Ograniczenia

- $S \in \{1, 2\}$
- $2 \le N \le 1\,000$
- $0 \le A_i < 2^{20}$

Podzadania

Zestaw testów dzieli się na podane poniżej podzadania.

Dodatkowe ograniczenia	Liczba punktów
$2 \le N \le 150$, $S = 1$, wszystkie elementy A są parami różne	25
$2 \le N \le 200$, $S = 1$, wszystkie elementy A są parami różne	35
$2 \le N \le 1\,000$, $S = 2$	40

Przykład

Wejście:

```
5 1
3 2 8 4 1
```

Wyjście:

```
3
1 2
4 3
5 4
```

Wyjaśnienie do przykładu: [3, 2, 8, 4, 1] → [**1**, 2, 8, 4, 1] → [1, 2, 8, **12**, 1] → [1, 2, 8, 12, **13**]

Wejście:

```
5 2
4 4 2 0 1
```

Wyjście:

```
3
3 2
4 3
5 4
```

Wyjaśnienie do przykładu: [4, 4, 2, 0, 1] → [4, 4, **6**, 0, 1] → [4, 4, 6, **6**, 1] → [4, 4, 6, 6, **7**]

Uwagi

Wykonując operację XOR pomiędzy bitami a i b, wynikowy bit jest 0 jeśli $a = b$ oraz 1 w przeciwnym przypadku.

Wykonując operację XOR pomiędzy liczbami a i b, wyniki dla kolejnych bitów obliczane są niezależnie:

- $75 \oplus 29 = 86$
- $1001011_{(2)} \oplus 0011101_{(2)} = 1010110_{(2)}$

W C/C++/Java możesz użyć operatora ^ aby obliczyć XOR.