



# XLI НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА

## Национален кръг

Велико Търново, 7 - 10 март 2025 г.

Група АВ, 9-12 клас, Ден 1

### Задача АВ12. Pointsort

7.5 сек. 256 MB

Алиса отново се оказва в Огледалния свят и отново се налага да сортира разни неща, които Хъмпти-Дъмпти е изтървал. Този път тя трябва да сортира  $N$   $K$ -измерни точки (да знае реда им по всяко от измеренията поотделно). За целта тя може да разглежда двойка точки и наведнъж да разбира коя е по-голяма и коя по-малка по всяко от измеренията. Тя иска да минимизира броя сравнения, които ще ползва. Отново обаче е без идеи и моли Вас за помощ.

По-формално,  $N$ -те точки са номерирани от 0 до  $N - 1$ . Всяка точка  $i$  има  $K$  на брой целочислени координати  $X_{i,d}$  за  $0 \leq d < K$ . Също така знаем, че за всяко измерение  $d$  координатите на точките по това измерение формират пермутация. Т.е. редицата  $X_{0,d}, \dots, X_{N-1,d}$  е пермутация на числата на числата от 0 до  $N - 1$ . Гарантирано е, че тези пермутации са произволно избрани (с еднаква вероятност за всяка възможна пермутация) независимо една от друга.

Алиса може да сравни две точки  $i$  и  $j$  по всяко измерение наведнъж. Т.е. тя избира двете точки и като отговор получава списък от  $K$  булеви резултата:  $d$ -тият резултат е единица, ако  $X_{i,d} < X_{j,d}$ , а иначе е нула. Целта на Алиса е чрез такива сравнения да разбере координатите на всички точки по всички измерения, т.е. стойностите на  $X_{i,d}$  за всяко  $i$  и  $d$ .

### Детайли по имплементацията

Вие трябва да имплементирате функцията `pointsort`:

```
std::vector<std::vector<int>> pointSort(int n, int k);
```

Тя ще бъде извикана  $T$  пъти с едни и същи  $N$  и  $K$  всеки път. Всяко извикване представлява независим събтест. Функцията трябва да върне стойностите на точките, индексирани първо по точки, а после по измерения.

Вашата програма може да вика функцията `compare` на журито:

```
std::vector<bool> compare(int i, int j);
```

Тя може да бъде извиквана произволна бройка пъти. Избраните  $i$  и  $j$  трябва да са различни един от друг валидни индекси на точки. Функцията връща резултатите на  $K$ -те сравнения по всяко от измеренията.

Вашата програма трябва да включва хедър файла `pointsort.h`, не трябва да има `main` функция и не трябва да чете от стандартния вход или да пише на стандартния изход.

### Ограничения

- $N = 10^4$
- $2 \leq K \leq 5$
- $6 \leq T \leq 15$



# ХЛІ НАЦИОНАЛНА ОЛИМПИАДА ПО ИНФОРМАТИКА

## Национален кръг

Велико Търново, 7 - 10 март 2025 г.

Група АВ, 9-12 клас, Ден 1

### Локално тестване

Предоставени са Ви хедър файл и локален грейдър, които да компилирате с програмата си. Първо се въвеждат  $T$ ,  $N$  и  $K$ . След това се въвежда 1, ако искате локалния грейдър да генерира произволни събтестове, или 0, ако искате да въведете свои събтестове. След това се въвеждат данните за всеки от  $T$ -те събтеста. Ако сме в режим 1, се въвежда само по едно число: сийдът за произволния генератор. Ако сме в режим 0, се въвеждат по  $N$  реда от по  $K$  числа: координатите на точките. Програмата ще изведе средния брой направени сравнения на събтест или съобщение за грешка при проблем.

### Оценяване

Всеки от четирите теста на задачата се оценява поотделно. Резултатът Ви на даден тест (частта от точките за теста, която ще получите) се определя по следния начин:

Ако програмата Ви направи невалидно сравнение или върне грешен отговор, резултатът Ви е 0. Иначе, нека  $Q$  е средният брой сравнения, които програмата Ви прави на събтест (т.е. средния брой сред  $T$ -те изпълнения). Също, нека  $Q^*$  да е желания брой сравнения. Ако,  $Q \leq Q^*$ , резултатът Ви е 1. Иначе, дефинираме релативния брой допълнителни сравнения  $E$  като:

$$E = Q/Q^* - 1$$

Тогава резултатът Ви  $S$  е:

$$S = \max(1 - 3.75 \times \min(E, 0.15) - \max(E - 0.15, 0), 0.1)$$

Дадени са някои примерни стойности на  $S$  като функция на  $E$ :

$E$	$S$
0.00	100.00%
0.05	81.25%
0.10	62.50%
0.15	43.75%
0.20	38.75%
0.30	28.75%
0.40	18.75%
0.50	10.00%

### Тестове

Тест	Точки	$K$	$T$	$Q^*$
1	25	2	15	187500
2	35	3	10	267000
3	25	4	8	350800
4	15	5	6	434500