# Задача А5. Поиск значения в отсортированной матрице

Дана квадратная матрица $A$ размера $N \times N$, заполненная целыми числами, а также целое число $key$. Матрица $A$ характеризуется тем, что:

- значения в строках отсортированы по возрастанию, а
- значения в столбцах отсортированы по убыванию.

Предполагается, что заданная матрица заполнена *уникальными* значениям. Например, таким условиям отвечает следующая квадратная матрица:

$$\begin{pmatrix} 11 & 12 & 18 \\ 8 & 9 & 10 \\ 5 & 6 & 7 \end{pmatrix}$$

**1. Разработайте линейный по времени алгоритм поиска значения key в заданной матрице A. Представьте описание алгоритма любым удобным способом — псевдокод, блок-схема, программный код на C++ и пр.**

```cpp
#include <vector>

bool findKey(const std::vector<std::vector<int>>& A, int key) {
  int N = A.size();
  int i = 0;
  int j = 0;

  while (i < N && j < N) {
    if (A[i][j] == key) {
      return true;
    } else if (A[i][j] > key) {
      ++i;
    } else {
      ++j;
    }
  }
  return false;
}
```

1. Если текущий элемент равен key, возвращаем true.
2. Если текущий элемент больше key, перемещаемся вниз по строкам (i++), так как перемещение вниз приводит к уменьшению значений в столбце.

3. Если текущий элемент меньше key, перемещаемся вправо по столбцам (j++), так как перемещение вправо приводит к увеличению значений в строке.

**2. Выполните анализ временной сложности разработанного алгоритма, составив точное выражение функции временной сложности T(N). Докажите, что T(N) = O(N) в соответствии с определением асимптотической верхней границы.**

## Анализ временной сложности алгоритма:

Количество шагов: Индекс i: может увеличиться от 0 до N, то есть не более чем на N единиц. Индекс j: аналогично может увеличиться от 0 до N, то есть не более чем на N единиц. Максимальное общее количество увеличений индексов i и j составляет N+N=2N. Это происходит, когда мы перемещаемся по диагонали матрицы от верхнего левого угла к правому нижнему.

Функция временной сложности: $T(N) = 2N$.

### Доказательство, что $T(N) = O(N)$:

По определению «О-большое» ($O(N)$):
∃ такая константа $c > 0$ и число $N_0$, что $\forall\ N \geq N_0 : T(N) \leq cN$.

В нашем случае:
$T(N) = 2N$
Выберем $c = 2$ и $N_0 = 1$
Тогда для всех $N \geq 1$ выполняется $T(N) = 2N \leq 2N$
Следовательно, $T(N) = O(N)$.

### Вывод:

Разработанный алгоритм имеет линейную временную сложность относительно размера матрицы $N$, что подтверждается как практическим анализом количества операций, так и формальным доказательством согласно определению асимптотической верхней границы.