Бинарный поиск. Вещественный бинарный поиск

ceagest

1 Бинарный поиск

Определение 1.1. *Бинарный поиск — это алгоритм поиска элемента в отсортированном массиве, который делит поисковый интервал пополам, и продолжает поиск в нужной половине.*

Бинарный поиск на Java

```java
public static int binarySearch(int[] array, int target) {
    int left = 0;
    int right = array.length - 1;

    while (left <= right) {
        int mid = left + (right - left) / 2;
        if (array[mid] == target) {return mid;}
        if (array[mid] < target) {left = mid + 1;}
        else {right = mid - 1;}
    }
    return -1;
}
```

1.1 Анализ сложности бинарного поиска

Рекуррентное соотношение. Бинарный поиск можно описать рекуррентным соотношением:

$$T(n) = T\left(\frac{n}{2}\right) + \Theta(1)$$

Применим мастер теорему:

- $a = 1$ (одна подзадача)
- $b = 2$ (данные делятся пополам)
- $c = 0$ ($\Theta(1)$ время на объединение)

Итого, $a = b^c = 1 \implies$ из мастер теоремы:

$$T(n) = \Theta(n^c \log n) = \Theta(\log n)$$

2 Вещественный бинарный поиск

Идея алгоритма. Применяется для поиска корней уравнений или решения задач с вещественными числами, где требуется найти значение с заданной точностью.

Пример вычисления $\sqrt{n}$ с точностью ε при помощи алгоритма вещественного бинарного поиска на Java

```java
public static double sqrtBinarySearch(double n) {
    double epsilon = Math.pow(10, -6);
    double left = 0;
    double right = n + 1;
    while (right - left > epsilon) {
        double mid = (left + right) / 2;
        if (mid * mid < n) {left = mid;}
        else {right = mid;}
    }
    return (left + right) / 2;
}
```

2.1 Анализ сложности вещественного бинарного поиска

- Начальный интервал: $[0, n]$ (длина n)
- На каждой итерации интервал уменьшается в 2 раза
- Требуемая точность: ε
- Количество итераций k, где $\frac{n}{2^k} \leq \varepsilon \implies k \geq \log_2 \frac{n}{\varepsilon}$

Тогда получаем, что:

$$T(n) = \mathcal{O}\left(\log \frac{n}{\varepsilon}\right)$$