

Бинарный поиск. Вещественный бинарный поиск

Лесников Юрий, seagest

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

```
1 public static int binarySearch(int[] array, int target) {  
2     int left = 0;  
3     int right = array.length - 1;  
4  
5     while (left <= right) {  
6         int mid = left + (right - left) / 2;  
7         if (array[mid] == target) {return mid;}  
8         if (array[mid] < target) {left = mid + 1;}  
9         else {right = mid - 1;}  
10    }  
11    return -1;  
12 }
```

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

```
1 public static double sqrtBinarySearch(double n) {  
2     double epsilon = Math.pow(10, -6);  
3     double left = 0;  
4     double right = n + 1;  
5     while (right - left > epsilon) {  
6         double mid = (left + right) / 2;  
7         if (mid * mid < n) {left = mid;}  
8         else {right = mid;}  
9     }  
10    return (left + right) / 2;  
11 }
```

2.1 Анализ сложности вещественного бинарного поиска

Общая постановка задачи. Пусть функция $f: [a, b] \rightarrow \mathbb{R}$ строго монотонна, $\varepsilon > 0$, $y \in \mathbb{R}$. Требуется найти $f^{-1}(y)$ с точностью ε .

Будем делать аналогичный дискретному случаю поиск, но будем останавливаться, если границы поиска находятся на расстоянии ε друг от друга.

- Начальный промежуток: $[a, b]$ (положим $[b - a] = n$)
- На каждой итерации промежуток 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:

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