Билет 3. Бинарный поиск. Вещественный бинпоиск.

Определение

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

Целочисленный бинарный поиск

Анализ

Рекуррентное соотношение:

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

Применение мастер-теоремы:

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

Сравнение: a vs b^c

$$a = 1, \quad b^c = 2^0 = 1 \quad \Rightarrow \quad a = b^c$$

Случай 2 мастер-теоремы:

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

```java
class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    int[] arr = new int[5];
    for (int i = 0; i < 5; i++) {
      arr[i] = sc.nextInt();
    }
    int res = BinarySearch.binarySearch(arr,  target: 0);
    System.out.println(res);
  }
}

class BinarySearch {  1 usage
  public static int binarySearch(int[] arr, int target) {
    int left = 0;
    int right = arr.length - 1;

    while (left <= right) {
      int mid = left + (right - left) / 2;

      if (arr[mid] == target) {
        return mid; // Элемент найден
      } else if (arr[mid] < target) {
        left = mid + 1; // Ищем в правой половине
      } else {
        right = mid - 1; // Ищем в левой половине
      }
    }

    return -1;
  }
}
```


Рис. 1: Бинпоиск на Java

Вещественный бинарный поиск

Определение

Вещественный бинарный поиск применяется для поиска корней уравнений или решения задач с вещественными числами, где требуется найти значение с заданной точностью.

Анализ

Анализ сложности:

- **Начальный интервал:** $[0, n]$ (длина $= n$)
- **Точность:** ε
- **Количество итераций:** k, где $\dfrac{n}{2^k} \leq \varepsilon$

Решаем неравенство:

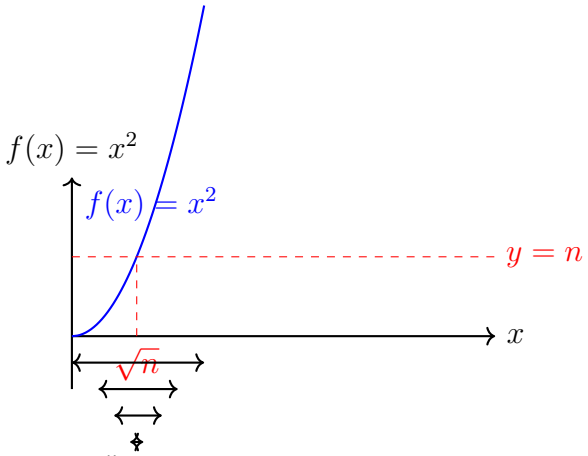
$$\frac{n}{2^k} \leq \varepsilon \quad \Rightarrow \quad 2^k \geq \frac{n}{\varepsilon} \quad \Rightarrow \quad k \geq \log_2 \left(\frac{n}{\varepsilon}\right)$$

Итоговая сложность:

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

```java
class Main {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    double n = sc.nextDouble();
    double res = SquareRootBinarySearch.sqrt(n,  epsilon: 1e-10);
    System.out.println(res);
  }
}

class SquareRootBinarySearch {  1 usage

  public static double sqrt(double n, double epsilon) {  1 usage

    // Особые случаи
    if (n == 0 || n == 1) {
      return n;
    }

    double left, right;

    left = 0; right = n;

    // Бинарный поиск
    while (right - left > epsilon) {
      double mid =  (right + left) / 2;
      if (mid*mid < n) {
        left = mid;
      } else {
        right = mid;
      }
    }
    return (left + right) / 2;
  }
}
```

Рис. 2: Вещ. бинпоиск на Java



Финальный интервал $< \varepsilon$

Сравнение подходов

Параметр	Целочисленный	Вещественный
Область поиска	Дискретные значения	Непрерывный интервал
Критерий остановки	left $>$ right	right - left $< \varepsilon$
Возвращаемое значение	Индекс элемента	Приближённое значение
Сложность	$O(\log n)$	$O(\log \frac{n}{\varepsilon})$

Применение мастер-теоремы

Анализ

Для бинарного поиска имеем:

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

Раскрываем рекуррентность:

$$\begin{aligned} T(n) &= \Theta(1) + T\left(\frac{n}{2}\right) \\ &= \Theta(1) + \Theta(1) + T\left(\frac{n}{4}\right) \\ &= \Theta(1) + \Theta(1) + \cdots + T(1) \\ &= \Theta(1) \times (\text{количество итераций}) \end{aligned}$$

Количество итераций: $n/2^k = 1 \Rightarrow k = \log_2 n$
Итог: $T(n) = \Theta(1) \times \Theta(\log n) = \Theta(\log n)$